Teoretické otázky neuronových sítí

Jiří Šíma & Roman Neruda

motto:

„Chci, abyste povzbuzeni v srdci a spojeni láskou hluboce pochopili a plně poznali Boží tajemství, jímž je Kristus; v něm jsou skryty všechny poklady moudrosti a poznání.“

apoštol Pavel

Katce & Vladce

Předmluva

V posledních deseti letech opět vzrostl zájem o tzv. (*umělé*) *neuronové sítě*, což jsou velmi zjednodušené matematické modely nervových systémů živých organismů. Jeden směr výzkumu v této oblasti se snaží pochopit a modelovat, jakým způsobem myslíme a jak funguje náš mozek. Na druhé straně tohoto úsilí stojí inženýři, kteří, inspirováni neurofyziologickými poznatky, tyto modely neuronových sítí modifikují, popř. hardwarově realizují, aby je mohli využít pro řešení úloh z umělé inteligence.

Simulace neuronových sítí překvapivě vykazují prvky podobné lidské inteligenci: schopnost učit se a zobecňovat předchozí zkušenosti. Tato jejich vlastnost patrně přitahuje velké množství laiků i odborníků. Avšak aplikace neuronových sítí při řešení reálných problémů ukazují i na jejich omezení. Např. učící proces je časově náročnou záležitostí s nejistým výsledkem. Také základní výzkum v oboru neuronových sítí, přestože existují desítky specializovaných konferencí a časopisů, nepřinesl zásadní pokrok. V literatuře jsou popisovány stovky různých variant modelů neuronových sítí, jejichž oprávněnost a výhody jsou často ilustrovány jen na izolovaných jednoduchých příkladech.

Domníváme se, že tato situace v základním výzkumu neuronových sítí je způsobena nesprávnou metodou použitou při studiu těchto modelů. Modely neuronových sítí jsou totiž matematické povahy, a proto adekvátním nástrojem pro zkoumání jejich vlastností by měla být matematika, popř. teoretická informatika. Z tohoto hlediska výpovědi typu „neuronová síť správně generalizuje“, založené na experimentálních výsledcích, nemají tu sílu a obecnost jako exaktně zformulovaná matematická tvrzení. Na druhou stranu ani teorie nedává přesvědčivou odpověď na otázku, co to znamená, že neuronová síť se chová „inteligentně“.

Není pravdou, že by neexistoval výzkum v teorii neuronových sítí, ale jeho výsledky nejsou dostatečně známy a interpretovány. Tato kniha je skromným pokusem tento nedostatek napravit v česky mluvícím prostředí. Skládá se ze tří částí.

Vzhledem k tomu, že chybí základní učebnice neuronových sítí v češtině, první část knihy obsahuje stručný úvod do této oblasti. Výklad je zde spíše zaměřen na motivace a základní typové modely a není žádným uceleným přehledem modelů neuronových sítí.

V druhé části knihy pohlížíme na neuronovou síť jako na výpočetní model a zkoumáme jeho výpočetní sílu a deskriptivní složitost. Také se zabýváme efektivitou učících algoritmů a složitostí problému učení a generalizace. Hlavním nástrojem studia uvedených otázek je teorie složitosti.

Třetí část knihy zkoumá typy funkcí, které jsou realizovány neuronovými sítěmi. Z tohoto hlediska se zabývá aproximačními vlastnostmi různých modelů neuronových sítí a jejich funkční ekvivalencí. Tato část využívá prostředky funkcionální analýzy.

Předkládaná práce původně vznikla jako učební text ke stejnojmenné dvousemestrální přednášce, kterou jsme měli na matematicko-fyzikální fakultě Univerzity Karlovy v letech 1993-96 pro studenty oboru informatika. Obsahuje relativně nové výsledky, z nichž některé jsou zatím dostupné jen v původních článcích. Ty obohatily látku, kterou jsme čerpali z ojedinělých existujících zahraničních monografií. I když jsme se snažili ji zpřístupnit co nejširšímu okruhu zájemců o neuronové sítě, přesto k plnému pochopení zvláště technických partií pomohou předběžné znalosti z teorie složitosti a funkcionální analýzy.

Chtěli bychom touto cestou poděkovat RNDr. Jiřímu Hořejšovi, CSc., který v nás svými zajímavými přednáškami vzbudil zájem o neuronové sítě. Dále děkujeme RNDr. Věře Kůrkové, CSc. a RNDr. Jiřímu Wiedermannovi, DrSc., kteří nám ukázali, že tento zájem je možné podložit teorií. Uvítali jsme, že se oba ujali nevděčné úlohy recenzentů. Také děkujeme našemu kolegovi RNDr. Petru Savickému, CSc. za rozhovory o české terminologii v oblasti booleovské složitosti a za první připomínky k rukopisu. Předběžnou verzi této knihy podrobně pročetl náš kolega Mgr. Jiří Červenka, kterému jsme vděčni za množství připomínek, které nám pomohly odstranit nejzávažnější nedostatky a chyby. Náš dík patří našim kolegyním a kamarádkám Tereze Vavříkové, která ochotně zrealizovala několik obrázků v této knize na počítači, a Hance Klímové, která nám pomohla při přípravě bibliografie. Také našemu kolegovi a příteli Mgr. Arnoštovi Štědrému děkujeme za pomoc s přípravou některých obrázků a za rady při sázení textu. Práce na této knize byla částečně podporována z grantů GA ČR číslo 201/95/0976 a 201/96/0917. V neposlední řadě jsme vděčni svým manželkám za jejich trpělivost, kterou s námi měly při přípravě rukopisu. Tuto knihu věnujeme právě jim.

Obsah

II Složitost neuronových sítí 131

(Jiří Šíma)

III Aproximace funkcí pomocí neuronových sítí 323

(Roman Neruda)

Část I

Úvod do neuronových sítí

První část knihy je stručným úvodem do neuronových sítí a lze ji chápat jako pozvání k hlubšímu studiu tohoto oboru. Výklad je zde spíše zaměřen na motivace a základní typové modely a není žádným uceleným přehledem modelů neuronových sítí. Připravuje půdu pro ty, kdo se s tímto fenoménem ještě nesetkali, aby mohli docenit následující teoretické části knihy. Ale poslouží i těm, kteří se chtějí seznámit se základními modely, aby je mohli aplikovat v oblasti svého odborného zájmu. Tato část nevyžaduje předběžné hlubší znalosti, obsahuje mnoho obrázků a matematický formalismus je v první kapitole redukován na co nejmenší možnou míru a v následujících kapitolách je použit jen k technickému popisu modelů neuronových sítí. K formalizaci některých z popisovaných modelů se pak vrátíme při jejich teoretické analýze.

První kapitola se snaží popsat a objasnit fenomén neuronových sítí. Obsahuje stručný přehled historie bádání v oblasti neurovýpočtů a objasňuje neurofyziologické motivace, které vedly k matematickému modelu neuronu a neuronové sítě. Ukazuje, že konkrétní model neuronové sítě lze zadat pomocí organizační, aktivní a adaptivní dynamiky, které určují vývoj jednotlivých charakteristik neuronové sítě v čase. Dále představuje neuropočítače založené na těchto modelech jako alternativu ke klasické von neumannovské architektuře počítače a vymezuje vhodné oblasti jejich aplikace.

Druhá kapitola popisuje klasické modely neuronových sítí. Nejprve krátce zmiňuje historicky nejstarší model síť perceptronů. Dále se podrobněji zabývá v praxi nejčastěji používaným modelem vícevrstvé neuronové sítě s učícím algoritmem backpropagation. Popis kromě různých variant tohoto modelu obsahuje i implementační poznámky. Následuje výklad lineárního modelu MADALINE adaptovaného podle Widrowova pravidla. Kapitolu pak uzavírá učící algoritmus pro kaskádové architektury, který v průběhu organizačně–adaptivního režimu přidává k síti nové neurony. Tento model není úplně klasický, avšak je všeobecně známým rozšířením algoritmu backpropagation.

Ve třetí kapitole je výklad zaměřen na modely neuronových sítí, které se využívají jako autoasociativní nebo heteroasociativní paměti. Na příkladu lineární asociativní sítě jsou vysvětleny principy adaptace podle Hebbova zákona. Dalším modelem je známá Hopfieldova síť motivovaná fyzikálními ději, která je představitelem cyklických neuronových sítí. Analogovou verzi této sítě lze použít k heuristickému řešení optimalizačních úloh (např. obchodní cestující). Zavedením parametru teploty do Hopfieldovy sítě podle fyzikální analogie obdržíme stochastický model, tzv. Boltzmannův stroj.

Čtvrtá kapitola se zabývá samoorganizací (tzv. učení bez učitele). Vyložíme tu Grayův algoritmus vektorové kvantizace a jeho neuronovou online variantu nazývanou Kohonenova síť. Dále věnujeme pozornost Kohonenovým samoorganizujícím mapám a různým variantám jejich dodatečného

učení známým jako učící vektorová kvantizace. Na závěr uvedeme model sítě typu counterpropagation.

V páté kapitole se věnujeme dopředným sítím s lokálními jednotkami, které jsou v jistém smyslu komplementární k perceptronům. Nejznámější modelem těchto sítí jsou RBF sítě. Kromě základního modelu si uvedeme semi-lokální gaussovské sítě a zobecněné RBF sítě (GRBF) Poggia a Girosiho.

Poznatky v této části knihy lze najít v libovolné monografii či v přehledových článcích o neuronových sítích. Při její kompozici jsme vycházeli zejména z prací [63, 97, 105, 109, 129, 130, 175, 256].

Kapitola 1

Fenomén neuronových sítí

1.1 Historie neurovýpočtů

Za počátek vzniku oboru neuronových sítí je považována práce Warrena McCullocha a Waltera Pittse z roku 1943 [189], kteří vytvořili velmi jednoduchý matematický model *neuronu*, což je základní buňka nervové soustavy. Číselné hodnoty parametrů v tomto modelu byly převážně bipolární (tj. z množiny $\{-1, 0, 1\}$). Ukázali, že nejjednodušší typy neuronových sítí mohou v principu počítat libovolnou aritmetickou nebo logickou funkci. Ačkoliv nepočítali s možností bezprostředního praktického využití svého modelu, jejich článek měl velký vliv na ostatní badatele. Například zakladatel kybernetiky Norbert Wiener se jím inspiroval při studiu podobnosti činnosti nervové soustavy a systémů výpočetní techniky. Nebo autor amerického projektu elektronických počítačů John von Neumann napsal práce [206, 207], ve kterých navrhoval výzkum počítačů, které by byly inspirovány činností mozku. Tyto návrhy, přestože byly hojně citovány, nepřinesly zpočátku očekávané výsledky.

V roce 1949 napsal Donald Hebb knihu *The Organization of Behavior* [98], ve které navrhl učící pravidlo pro *synapse* neuronů (mezineuronové rozhraní). Toto pravidlo bylo inspirováno myšlenkou, že podmíněné reflexy, které jsou pozorovatelné u všech živočichů, jsou již vlastností jednotlivých neuronů. Hebb se tak snažil vysvětlit některé experimentální výsledky z psychologie. Také jeho práce ovlivnila ostatní vědce, kteří se začali zabývat podobnými otázkami. Avšak 40. a 50. léta zatím ještě nepřinesla zásadní pokrok v oblasti neurovýpočtů. Typickým příkladem výzkumu z tohoto období byla v roce 1951 konstrukce prvního neuropočítače *Snark*, u jehož zrodu stál Marvin Minsky [196]. Snark byl sice úspěšný z technického hlediska, dokonce již automaticky adaptoval *váhy* (míra synaptické propustnosti), ale ve skuteč-

nosti nebyl nikdy využit k řešení nějakého zajímavého praktického problému. Nicméně jeho architektura později inspirovala další konstruktéry neuropočítačů.

V roce 1957 Frank Rosenblatt vynalezl tzv. *perceptron* [238] (viz podkapitolu 2.1), který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Pro tento model navrhl učící algoritmus, o kterém matematicky dokázal, že pro daná tréninková data nalezne po konečném počtu kroků odpovídající váhový vektor parametrů (pokud existuje) nezávisle na jeho počátečním nastavení. Tento výsledek vzbudil velké nadšení. Rosenblatt také napsal jednu z prvních knih o neurovýpočtech *Principles of Neurodynamics* [239].

Na základě tohoto výzkumu Rosenblatt spolu s Charlesem Wightmanem a dalšími [238] sestrojili během let 1957 a 1958 první úspěšný neuropočítač, který nesl jméno *Mark I Perceptron*. Protože původním odborným zájmem Rosenblatta bylo rozpoznávání obrazců, Mark I Perceptron byl navržen pro rozpoznávání znaků. Znak byl promítán na světelnou tabuli, ze které byl snímán polem 20×20 fotovodičů. Intenzita 400 obrazových bodů byla vstupem do neuronové sítě perceptronů, jejímž úkolem bylo klasifikovat, o jaký znak se jedná (např. „A“, „B“ apod.). Mark I Perceptron měl 512 adaptovatelných váhových parametrů, které byly realizovány polem $8 \times 8 \times 8$ potenciometrů. Hodnota odporu u každého potenciometru, která právě odpovídala příslušné váze, byla nastavována automaticky samostatným motorem. Ten byl řízen analogovým obvodem, který implementoval perceptronový učící algoritmus. Jednotlivé perceptrony bylo možné spojit se vstupy libovolným způsobem. Typicky bylo použito náhodné zapojení, aby se ilustrovala schopnost perceptronu učit se požadované vzory bez přesného zapojení drátů v protikladu ke klasickým programovatelným počítačům. Díky úspěšné prezentaci uvedeného neuropočítače se neurovýpočty, které byly alternativou ke klasickým výpočtům realizovaným na von neumannovské architektuře počítače, staly novým předmětem výzkumu. Frank Rosenblatt je proto dodnes některými odborníky považován za zakladatele tohoto nového oboru.

Krátce po objevu perceptronu Bernard Widrow se svými studenty vyvinul další typ neuronového výpočetního prvku, který nazval *ADALINE* (**ADA**ptive **LIN**ear **E**lement) [287] (viz podkapitolu 2.3). Tento model byl vybaven novým výkonným učícím pravidlem, které se až doposud využívá. Widrow se svými studenty demonstroval funkčnost ADALINE na mnoha jednoduchých typových příkladech. Widrow také založil první firmu (*Memistor Corporation*) orientovanou na hardware neuropočítačů, která v první polovině 60. let vyráběla a prodávala neuropočítače a jejich komponenty.

Na přelomu 50. a 60. let dochází k úspěšnému rozvoji neurovýpočtů v oblasti návrhu nových modelů neuronových sítí a jejich implementací. Například Karl Steinbuch vyvinul model binární asociativní sítě nebo Roger

Barron a Lewey Gilstrap založili v roce 1960 první firmu zaměřenou na aplikace neurovýpočtů. Výsledky z uvedeného období jsou shrnuty v knize Nilse Nilssona *Learning Machines* [208] z roku 1965.

Přes nesporné úspěchy dosažené v tomto období se obor neuronových sítí potýkal se dvěma zřejmými problémy. Za prvé, většina badatelů přistupovala k neuronovým sítím z experimentálního hlediska (připomínajícího tak trochu alchymii) a zanedbávala analytický výzkum neuronových modelů. Za druhé, nadšení některých výzkumných pracovníků vedlo k velké publicitě neopodstatněných prohlášení jako například, že za několik málo let bude vyvinut umělý mozek. Tyto skutečnosti diskreditovaly neuronové sítě v očích odborníků z jiných oblastí a odradily vědce a inženýry, kteří se o neurovýpočty zajímali. Navíc se samotný obor neuronových sítí vyčerpal a další pokrok v této oblasti by býval vyžadoval radikálně nové myšlenky a postupy. Nejlepší odborníci oblast neuronových sítí opouštěli a začali se zabývat příbuznými obory umělé inteligence.

Poslední epizodou tohoto období byla kampaň vedená Marvinem Minským a Seymourem Papertem, kteří využili svého vlivu, aby diskreditovali výzkum neuronových sítí, nacházející se v krizi, ve snaze převést finanční zdroje z této oblasti na jiný výzkum v umělé inteligenci. V té době koloval rukopis jejich výzkumné zprávy, která napomáhala tomuto záměru. Uvedený rukopis pak byl v upravené formě publikován v roce 1969 pod názvem *Perceptrons* [197]. V této knize Minsky a Papert využili pro svoji argumentaci známého triviálního faktu, že jeden perceptron nemůže počítat jednoduchou logickou funkci, tzv. vylučovací disjunkci (XOR). Tento problém lze sice vyřešit vytvořením dvouvrstvé sítě se třemi neurony, ale pro vícevrstvý perceptron nebyl v té době znám učící algoritmus. Autoři z toho nesprávně vyvodili, že takový algoritmus vzhledem ke komplikovanosti funkce, kterou vícevrstvá síť počítá, snad ani není možný. Jejich tvrzení bylo všeobecně přejato a považováno za matematicky dokázané. Kampaň Minského a Paperta byla úspěšná, výzkum neuronových sítí již nebyl dále dotován a neurovýpočty byly považovány za neperspektivní.

V dalším období od roku 1967 do 1982 probíhal výzkum neuronových sítí ojediněle a izolovaně, převážně mimo území Spojených států, kde kniha *Perceptrons* měla velký vliv. Většina prací byla publikována např. pod hlavičkou adaptivní zpracování signálů, rozpoznávání obrazců a biologické modelování. Avšak již v počátcích tohoto „tichého“ období se neurovýpočty začali zabývat talentovaní badatelé, mezi nimiž byli např. Shun-Ichi Amari, James Anderson, Kunihiko Fukushima, Stephen Grossberg, Harry Klopf, Teuvo Kohonen a David Willshaw. Tito vědci později přispěli svými objevy k renesanci neuronových sítí.

Počátkem 80. let se badatelé v oblasti neurovýpočtů osmělili a začali podávat vlastní grantové projekty zaměřené na vývoj neuropočítačů a je-

jich aplikace. Zásluhou programového manažera Ira Skurnicka začala v roce 1983 americká grantová agentura *DARPA* (**D**efense **A**dvanced **R**esearch **P**rojects **A**gency) finančně podporovat výzkum neuronových sítí a její příklad v krátké době následovaly další organizace podporující základní i aplikovaný výzkum.

Další zásluhu na renesanci oboru neuronových sítí měl světově uznávaný fyzik John Hopfield, který se v této době začal zabývat neurovýpočty. Své výsledky publikoval v roce 1982 a 1984 ve dvou velmi čtivých článcích [122, 123], kde ukázal souvislost některých modelů neuronových sítí s fyzikálními modely magnetických materiálů. Svými zvanými přednáškami, které měl po celém světě, získal pro neuronové sítě stovky kvalifikovaných vědců, matematiků a technologů.

V roce 1986 publikovali své výsledky badatelé z tzv. „*PDP* skupiny“ (**P**arallel **D**istributed **P**rocessing Group) ve sborníku editovaném Davidem Rumelhartem a Jamesem McClellandem [244]. Zde se objevil článek Rumelharta, Geoffreyho Hintona a Ronalda Williamse [243], kteří v něm popsali učící algoritmus zpětného šíření chyby (*backpropagation*) pro vícevrstvou neuronovou síť (viz podkapitolu 2.2) a vyřešili tak problém, který se Minskému a Papertovi v 60. letech jevil jako nepřekonatelná překážka pro využití a další rozvoj neuronových sítí. Tento algoritmus je dosud nejpoužívanější učící metodou neuronových sítí. Publikováním uvedeného sborníku dosáhl zájem o neuronové sítě svého vrcholu. Nic na tom nemění fakt, že uvedený algoritmus, jak se později ukázalo, byl vlastně znovu objeven, protože byl již znám a publikován některými vědci v „tichém“ období (např. Arthur Bryson a Yu-Chi Ho, 1969 [44]; Paul Werbos, 1974 [286]; David Parker, 1985 [218]).

Známým příkladem, který v počátcích ilustroval praktický význam neuronového učícího algoritmu backpropagation, byl systém *NETtalk* vyvinutý Terrencem Sejnowskim a Charlesem Rosenbergem [249]. Tento systém, vytvořený v krátké době učením neuronové sítě z příkladů, úspěšně konvertoval anglický psaný text na mluvený. Konkuroval tak svému předchůdci, systému *DECtalk* (**D**igital **E**quipment **C**orporation), který obsahoval stovky pravidel vytvářených lingvisty po celá desetiletí.

V roce 1987 se v San Diegu konala první větší konference specializovaná na neuronové sítě (*IEEE International Conference on Neural Networks*), na které bylo 1700 účastníků, a byla založena mezinárodní společnost pro výzkum neuronových sítí *INNS* (**I**nternational **N**eural **N**etwork **S**ociety). O rok později INNS začala vydávat svůj časopis *Neural Networks*. V následujících letech vznikly další specializované časopisy, jako např. *Neural Computation* (1989), *IEEE Transactions on Neural Networks* (1990) a mnoho jiných (např. v Praze vychází od roku 1991 mezinárodní časopis *Neural Network World*). Od roku 1987 mnoho renomovaných univerzit založilo nové výzkumné ústavy

zabývající se neuronovými sítěmi a vyhlásilo výukové programy zaměřené na neurovýpočty. Tento trend pokračuje až dodnes, kdy se zdá, že široký záběr výzkumu a vynaložené investice neodpovídají kvalitě dosažených výsledků. Teprve blízká budoucnost opět prověří životnost oboru neuronových sítí.

1.2 Neurofyziologické motivace

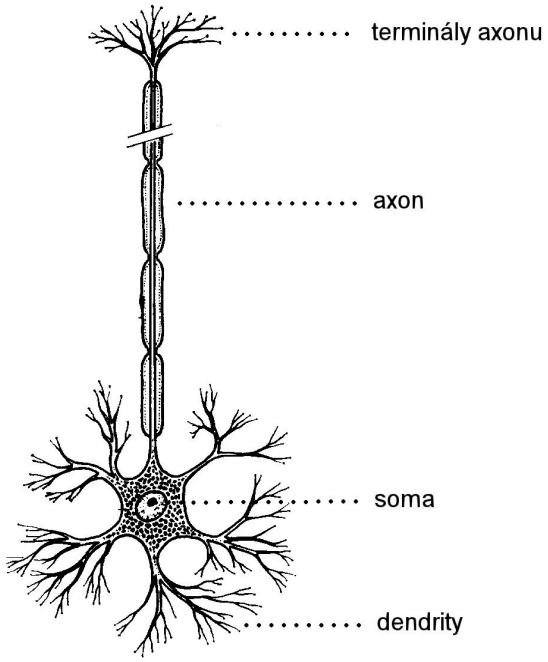
Původním cílem výzkumu neuronových sítí byla snaha pochopit a modelovat, jakým způsobem myslíme a jak funguje lidský mozek. Neurofyziologické poznatky umožnily vytvořit zjednodušené matematické modely, které se dají využít pro neurovýpočty při řešení praktických úloh z umělé inteligence. To znamená, že neurofyziologie zde slouží jen jako zdroj inspirací a navržené modely neuronových sítí jsou již dále rozvíjeny bez ohledu na to, zda modelují lidský mozek. Přesto, je-li to užitečné, lze se k této analogii vracet pro nové inspirace nebo je možné ji využít při popisu vlastností matematického modelu.

Proto je účelné se seznámit se základními poznatky z neurofyziologie v takovém rozsahu, který nám umožní pochopit původní motivace matematických modelů neuronových sítí. Nejsme v tomto oboru žádnými odborníky, ale následující výklad bychom snad mohli obhájit výrokem jistého neurofyziologa, který tvrdí, že současné poznání činnosti lidského mozku je natolik povrchní, že cokoliv o mozku řekneme, může být považováno za pravdivé. Z týchž úst jsme slyšeli, že přístroje (např. EEG), kterými v současnosti studujeme mozek, lze přirovnat k mikrofonu nad rozbouřeným fotbalovým stadiónem, pomocí kterého bychom se rádi dozvěděli, co si říkají dva diváci v zástupu fanoušků. Na druhou stranu nám nejde o vytvoření identické kopie mozku, ale chceme napodobit jeho základní funkce (podobně jako letadla mají s ptáky společné hlavně to, že létají).

Nervová soustava člověka (obecně živých organismů) zprostředkuje vztahy mezi vnějším prostředím a organismem, i mezi jeho částmi, a zajišťuje tak příslušnou reakci na vnější podněty i na vnitřní stavy organismu. Tento proces probíhá šířením vzruchů z jednotlivých čidel, tzv. *receptorů*, které umožňují přijímat mechanické, tepelné, chemické a světelné podněty, směrem k jiným nervovým buňkám, které tyto signály zpracovávají a přivádí k příslušným výkonným orgánům, tzv. *efektorům*. Tyto vzruchy se po *projekčních drahách*, kde dochází k prvnímu předzpracování, kompresi a filtraci informace, dostávají až do *mozkové kůry*, která je nejvyšším řídícím centrem nervového systému. Na povrchu mozku můžeme rozlišit celkem šest primárních vzájemně propojených *projekčních oblastí* odpovídajících přibližně smyslům, ve kterých dochází k paralelnímu zpracování informace. Komplexní

zpracování informace, které je základem pro vědomé řízení činnosti efektorů, probíhá již sekvenčně v tzv. *asociačních oblastech*.

Základním stavebním funkčním prvkem nervové soustavy je nervová buňka, tzv. *neuron*. Jen mozková kůra člověka je tvořena asi 13 až 15 miliardami neuronů, z nichž každý může být spojen s 5000 jinými neurony. Neurony jsou samostatné specializované buňky určené k přenosu, zpracování a uchování informací nutných pro realizaci životních funkcí organismu. Struktura neuronu je schematicky znázorněna na obrázku 1.1. Neuron je přizpůsoben pro



Obr. 1.1: Biologický neuron.

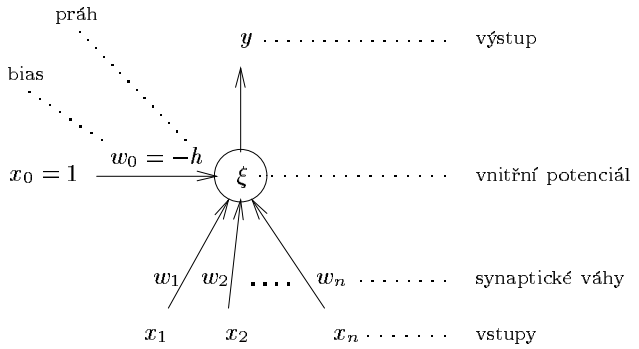
přenos signálů tak, že kromě vlastního těla, tzv. *somatu*, má i vstupní a výstupní přenosové kanály: *dendrity* a *axon*. Z axonu obvykle odbočuje řada větví, tzv. *terminálů*, zakončených blánou, která se převážně stýká s výběžky, tzv. *trny*, dendritů jiných neuronů, jak je naznačeno na obrázku 1.2. K přenosu informace pak slouží unikátní mezineuronové rozhraní, tzv. (chemická) *synapse*. Míra synaptické propustnosti je nositelem všech význačných informací během celého života organismu. Z funkčního hlediska lze synapse roz-

Nervová soustava člověka je velmi složitý systém, který je stále předmětem zkoumání. Uvedené velmi zjednodušené neurofyziologické principy nám však v dostatečné míře poslouží k formulaci matematického modelu neuronové sítě.

1.3 Matematický model neuronové sítě

1.3.1 Formální neuron

Základem matematického modelu neuronové sítě bude *formální neuron*, který získáme přeformulováním zjednodušené funkce neurofyziologického neuronu do matematické řeči. Jeho struktura je schematicky znázorněna na obrázku 1.3. Formální neuron (dále jen neuron) má n obecně reálných *vstupů*

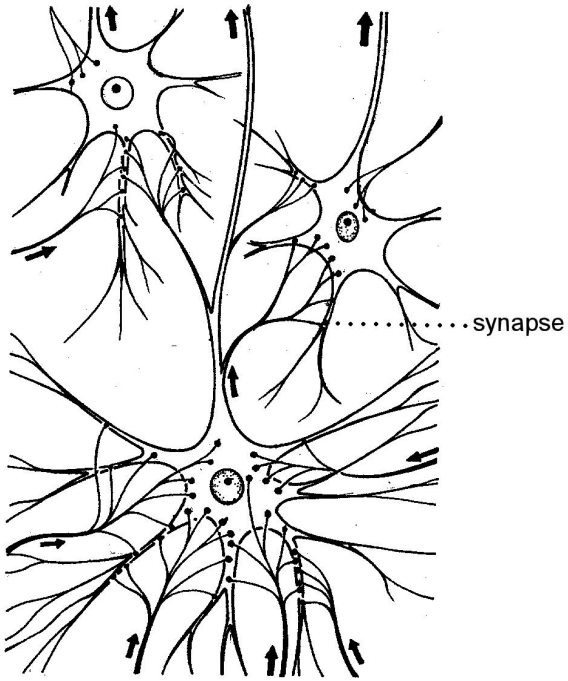


Obr. 1.3: Formální neuron.

$x_1, \ldots, x_n$, které modelují dendrity. Vstupy jsou ohodnoceny odpovídajícími obecně reálnými synaptickými *váhami* $w_1, \ldots, w_n$, které určují jejich propustnost. Ve shodě s neurofyziologickou motivací mohou být synaptické váhy záporné, čímž se vyjadřuje jejich inhibiční charakter. Zvážená suma vstupních hodnot představuje *vnitřní potenciál* neuronu:

$$\xi = \sum_{i=1}^{n} w_i x_i \,. \tag{1.1}$$

Hodnota vnitřního potenciálu ξ po dosažení tzv. *prahové* hodnoty h indukuje *výstup* (*stav*) neuronu y, který modeluje elektrický impuls axonu. Nelineární nárůst výstupní hodnoty $y = \sigma(\xi)$ při dosažení prahové hodnoty potenciálu h



Obr. 1.2: Biologická neuronová síť.

dělit především na tzv. *excitační*, které umožňují rozšíření vzruchu v nervové soustavě, a na tzv. *inhibiční*, které způsobují jeho útlum. Paměťová stopa v nervové soustavě vzniká pravděpodobně právě zakódováním synaptických vazeb na cestě mezi receptorem a efektorem.

Šíření informace je umožněno tím, že soma i axon jsou obaleny membránou, která má schopnost za jistých okolností generovat elektrické impulsy. Tyto impulsy jsou z axonu přenášeny na dendrity jiných neuronů synaptickými branami, které svojí propustností určují intenzitu podráždění dalších neuronů. Takto podrážděné neurony při dosažení určité hraniční meze, tzv. *prahu*, samy generují impuls a zajišťují tak šíření příslušné informace. Po každém průchodu signálu se synaptická propustnost mění, což je předpokladem paměťové schopnosti neuronů. Také propojení neuronů prodělává během života organismu svůj vývoj, v průběhu učení se vytváří nové paměťové stopy nebo při zapomínání se synaptické spoje přerušují.

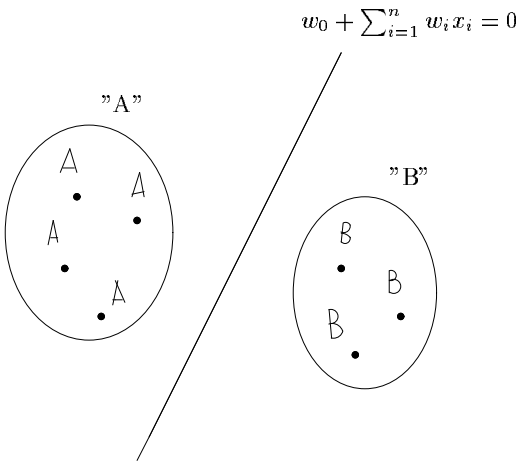
Tato nadrovina dělí vstupní prostor na dva poloprostory. Souřadnice bodů $[x_1^+, \ldots, x_n^+]$, které leží v jednom poloprostoru, splňují následující nerovnost:

$$w_0 + \sum_{i=1}^{n} w_i x_i^+ > 0\,. \tag{1.5}$$

Body $[x_1^-, \ldots, x_n^-]$ z druhého poloprostoru vyhovují nerovnosti s opačným relačním znaménkem:

$$w_0 + \sum_{i=1}^{n} w_i x_i^- < 0\,. \tag{1.6}$$

Synaptické váhy neuronu $w_0, \ldots, w_n$ (včetně biasu) lze chápat jako koeficienty této nadroviny. Je zřejmé, že neuron ve vstupním prostoru klasifikuje, ve kterém ze dvou poloprostorů určených touto nadrovinou leží bod, jehož souřadnice jsou na vstupu, tj. neuron realizuje tzv. *dichotomii* vstupního prostoru. Přesněji řečeno, neuron je *aktivní* (tj. stav neuronu je $y = 1$), jestliže vstupy neuronu splňují podmínku (1.5) nebo (1.4), tj. představují souřadnice bodu, který leží v prvním poloprostoru nebo na nadrovině. V případě, že tento bod leží ve druhém poloprostoru, vstupy neuronu vyhovují podmínce (1.6) a neuron je *pasivní* (tj. stav neuronu je $y = 0$).



Obr. 1.5: Separace obrazů „A“ a „B“ pomocí neuronu.

Význam funkce neuronu ilustrujeme na motivačním (trochu nadneseném) příkladu z oblasti rozpoznávání obrazců. Představme si, že školák z první

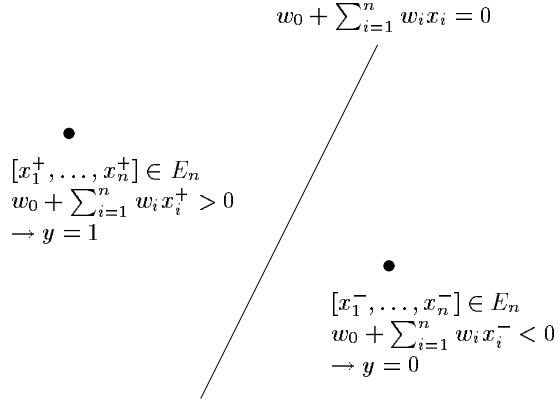
je dán tzv. *aktivační* (*přenosovou*) *funkcí* σ. Nejjednodušším typem aktivační funkce je tzv. *ostrá nelinearita*, která má tvar:

$$\sigma(\xi) = \begin{cases} 1 & \text{jestliže } \xi \geq h \\ 0 & \text{jestliže } \xi < h\,. \end{cases} \tag{1.2}$$

Formální úpravou lze docílit toho, že funkce σ bude mít nulový práh a vlastní práh neuronu se záporným znaménkem budeme chápat jako váhu, tzv. *bias* $w_0 = -h$ dalšího formálního vstupu $x_0 = 1$ s konstantní jednotkovou hodnotou, jak je naznačeno na obrázku 1.3. Matematická formulace funkce neuronu je potom dána vztahem:

$$y = \sigma(\xi) = \begin{cases} 1 & \text{jestliže } \xi \geq 0 \\ 0 & \text{jestliže } \xi < 0 \end{cases}, \quad \text{kde } \xi = \sum_{i=0}^{n} w_i x_i\,. \tag{1.3}$$

K lepšímu pochopení funkce jednoho neuronu nám pomůže geometrická představa načrtnutá na obrázku 1.4. Vstupy neuronu budeme chápat jako



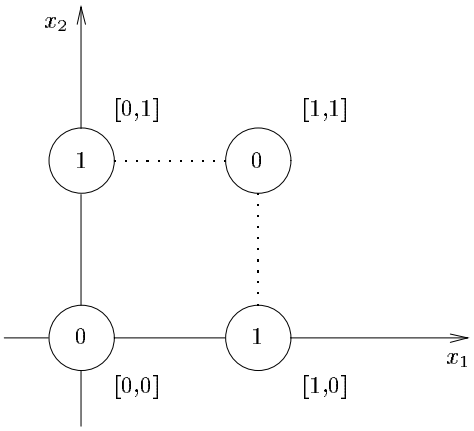
Obr. 1.4: Geometrická interpretace funkce neuronu.

souřadnice bodu v n-rozměrném euklidovském, tzv. *vstupním prostoru* E_n. V tomto prostoru má rovnice nadroviny (např. v E_2 přímka, v E_3 rovina) tvar:

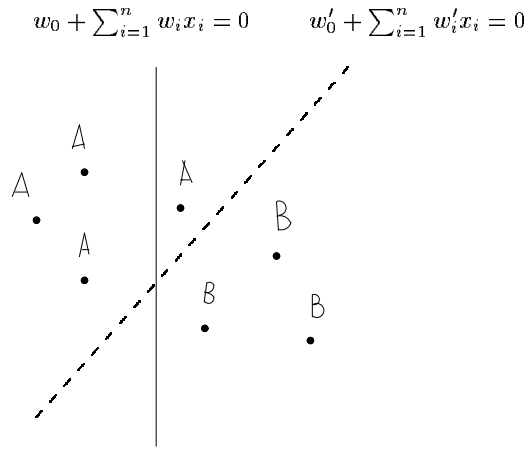
$$w_0 + \sum_{i=1}^{n} w_i x_i = 0\,. \tag{1.4}$$

hoto písmene od obrazů ostatních písmen. Je pravděpodobné, že na začátku školního roku budou váhy neuronu odpovídajícího za dichotomii písmen „A“ a „B“ nastaveny náhodně, protože školák ještě neumí číst. V průběhu učení, kdy mu je předkládáno množství vzorových písmen, se váhy tohoto neuronu budou postupně *adaptovat* tak, aby příslušná nadrovina začala oddělovat shluk bodů odpovídající písmenu „A“ od shluku bodů reprezentující písmeno „B“.

Například si představme, že školák již celkem dobře od sebe rozliší vzorová písmena „A“ a „B“. V této fázi učení se setká s již trochu vypsaným rukopisem své maminky, kde obraz písmene „A“ bude deformován tak, že jej zprvu bude číst jako písmeno „B“. To znamená, že příslušná nadrovina reprezentovaná uvedeným neuronem rozděluje vstupní prostor takovým způsobem, že bod odpovídající deformovanému obrazu písmene „A“ leží nesprávně v poloprostoru příslušejícím písmenu „B“. Při chybném čtení maminka školáka opraví a školák si přizpůsobí příslušné synaptické váhy $w_0, \ldots w_n$ tak, aby se nadrovina natočila a zahrnula nový vzor do správného poloprostoru. Tato situace je znázorněna na obrázku 1.6, kde je nová poloha nadroviny (reprezentovaná váhami $w'_0, \ldots w'_n$) vyznačena přerušovanou čarou. V uvedeném případě je tedy potřeba korekce učitele, kterého představuje maminka. Někdy může být stimulací k učení negativní zkušenost, kdy správné rozlišení objektů je otázkou přežití (např. kuře se musí naučit rozlišit hospodáře, který mu přináší potravu, od dravce, který je chce zahubit). Adaptování vah



Obr. 1.7: Geometrické znázornění funkce XOR.



Obr. 1.6: Adaptace vah neuronu při chybné klasifikaci obrazu „A“.

třídy se učí číst a rozpoznávat písmena. Pro jednoduchost předpokládejme, že za dichotomii písmen „A“ a „B“ je v jeho mozku odpovědný jeden neuron (srovnejte s využitím Rosenblattova neuropočítače Mark I Perceptron v podkapitole 1.1). Obraz jednoho písmene je rozložen na matici $n = k \times k$ bodů. Intenzita těchto obrazových bodů vyjádřená reálnými čísly je vstupem uvedeného neuronu. Tedy každé písmeno v naší geometrické představě odpovídá bodu v n-rozměrném vstupním prostoru neuronu. Je zřejmé, že školák se při svém učení setká s různými výskyty písmene „A“ (např. vzorové písmo v čítance, písmo učitele apod.), jejichž tvar nebude vždy totožný, ale měl by být velmi podobný (vždy se jedná o písmeno „A“). Body ve vstupním prostoru neuronu odpovídající těmto výskytům písmene „A“ by neměly být vzhledem k podobnosti obrazových vzorů od sebe příliš vzdálené (měřeno euklidovskou metrikou). Na druhou stranu bod reprezentující písmeno „B“, jehož obraz se dostatečně liší od obrazu písmene „A“, bude ve vstupním prostoru poněkud vzdálen od bodů odpovídajících písmenu „A“. Jinými slovy body reprezentující písmena „A“ a „B“ vytvoří ve vstupním prostoru dva oddělené shluky. Funkcí uvedeného neuronu je oddělit tyto shluky příslušnou nadrovinou tak, jak je naznačeno na obrázku 1.5.

Na uvedeném příkladě je vidět metodologickou odlišnost reprezentace konceptů v neuronových sítích. Školák si nemusí pamatovat jednotlivé obrazy všech výskytů daného písmene. Stačí, když synaptické váhy příslušného neuronu budou nastaveny tak, že odpovídající nadrovina oddělí obrazy to-

u formálního neuronu modeluje změnu synaptických propustností biologického neuronu a vznik paměťových stop v nervové soustavě živého organismu.

Uvedený motivační příklad měl osvětlit základní principy matematického modelu neuronu. Je zřejmé, že jeden neuron může řešit jen velmi jednoduché úlohy. Typickým příkladem logické funkce, kterou nelze počítat pomocí jednoho neuronu je vylučovací disjunkce XOR (srovnejte s argumentem Minského a Paperta proti perceptronu v podkapitole 1.1). Uvažujme např. jen dva binární vstupy (jejichž hodnoty jsou z množiny $\{0,1\}$) a jeden binární výstup, jehož hodnota je 1, právě když hodnota právě jednoho vstupu je 1 (tj. $XOR(0,0)=0$, $XOR(1,0)=1$, $XOR(0,1)=1$, $XOR(1,1)=0$). Z obrázku 1.7, kde jsou všechny možné vstupy znázorněny ve vstupním prostoru E_2 a ohodnoceny odpovídajícími výstupy, je zřejmé, že neexistuje nadrovina (přímka), která by oddělila body příslušející k výstupu 1 od bodů odpovídajících výstupu 0. Z uvedeného vyplývá, že pro řešení složitějších úloh, je potřeba neurony spojovat do sítě stejně tak, jak je tomu v nervové soustavě člověka.

1.3.2 Neuronová síť

Neuronová síť se skládá z formálních neuronů, které jsou vzájemně propojené tak, že výstup neuronu je vstupem obecně více neuronů podobně, jako terminály axonu biologického neuronu jsou přes synaptické vazby spojeny s dendrity jiných neuronů. Počet neuronů a jejich vzájemné propojení v síti určuje tzv. *architekturu* (*topologii*) neuronové sítě. Z hlediska využití rozlišujeme v síti *vstupní*, *pracovní* (*skryté*, *mezilehlé*) a *výstupní* neurony. Lze zjednodušeně říci, že v neurofyziologické analogii vstupní neurony odpovídají receptorům, výstupní neurony efektorům a propojené pracovní neurony mezi nimi vytváří příslušné dráhy, po kterých se šíří vlastní vzruchy. Tyto dráhy budeme v matematickém modelu nazývat cestami. Šíření a zpracování informace na cestě v síti je umožněno změnou stavů neuronů ležících na této cestě. Stavy všech neuronů v síti určují tzv. *stav* neuronové sítě a synaptické váhy všech spojů představují tzv. *konfiguraci* neuronové sítě.

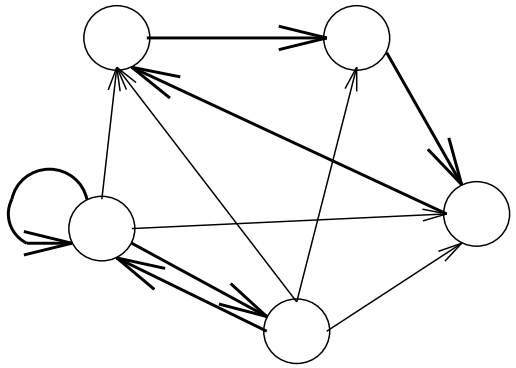
Neuronová síť se v čase vyvíjí, mění se propojení a stav neuronů, adaptují se váhy. V souvislosti se změnou těchto charakteristik v čase je účelné celkovou *dynamiku* neuronové sítě rozdělit do tří dynamik a uvažovat pak tři *režimy* práce sítě: *organizační* (změna topologie), *aktivní* (změna stavu) a *adaptivní* (změna konfigurace). Toto dělení neodpovídá neurofyziologické skutečnosti, protože v nervové soustavě probíhají příslušné změny současně. Uvedené dynamiky neuronové sítě jsou obvykle zadány počátečním stavem a matematickou rovnicí, resp. pravidlem, které určuje vývoj příslušné charakteristiky sítě (topologie, stav, konfigurace) v čase. Změny, které se řídí

těmito zákonitostmi, probíhají v odpovídajících režimech práce neuronové sítě.

Konkretizací jednotlivých dynamik pak obdržíme různé *modely* neuronových sítí vhodné pro řešení určitých tříd úloh. To znamená, že pro specifikaci konkrétního modelu neuronové sítě stačí, když definujeme jeho organizační, aktivní a adaptivní dynamiku. V následujícím výkladu popíšeme obecné principy a různé typy těchto tří dynamik, které jsou základem pro dělení a klasifikaci modelů neuronových sítí. Také zmíníme několik typických příkladů, které se nám budou později hodit při popisu konkrétních modelů neuronových sítí.

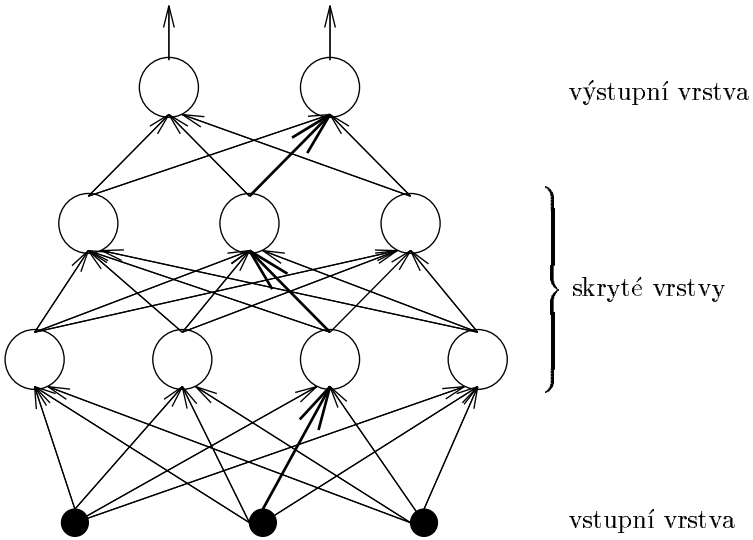
Organizační dynamika

Organizační dynamika specifikuje architekturu sítě a její případnou změnu. Změna topologie se většinou uplatňuje v rámci adaptivního režimu tak, že síť je v případě potřeby rozšířena o další neurony a příslušné spoje. Avšak organizační dynamika převážně předpokládá pevnou architekturu neuronové sítě, která se již nemění.



Obr. 1.8: Příklad cyklické architektury.

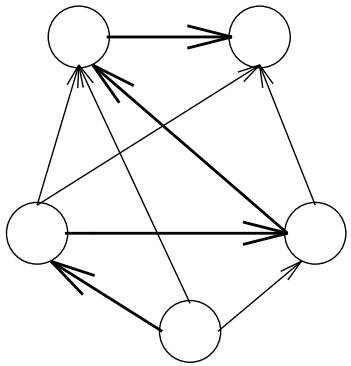
Rozlišujeme v zásadě dva typy architektury: *cyklická* (resp. *rekurentní*) a *acyklická* (resp. *dopředná*) síť. V případě cyklické topologie existuje v síti skupina neuronů, která je zapojena v kruhu (tzv. *cyklus*). To znamená, že v této skupině neuronů je výstup prvního neuronu vstupem druhého neuronu, jehož výstup je opět vstupem třetího neuronu atd., až výstup posledního neuronu v této skupině je vstupem prvního neuronu. Nejjednodušším příkladem cyklu je *zpětná vazba* neuronu, jehož výstup je zároveň jeho vstu-



Obr. 1.10: Příklad architektury vícevrstvé neuronové sítě 3–4–3–2.

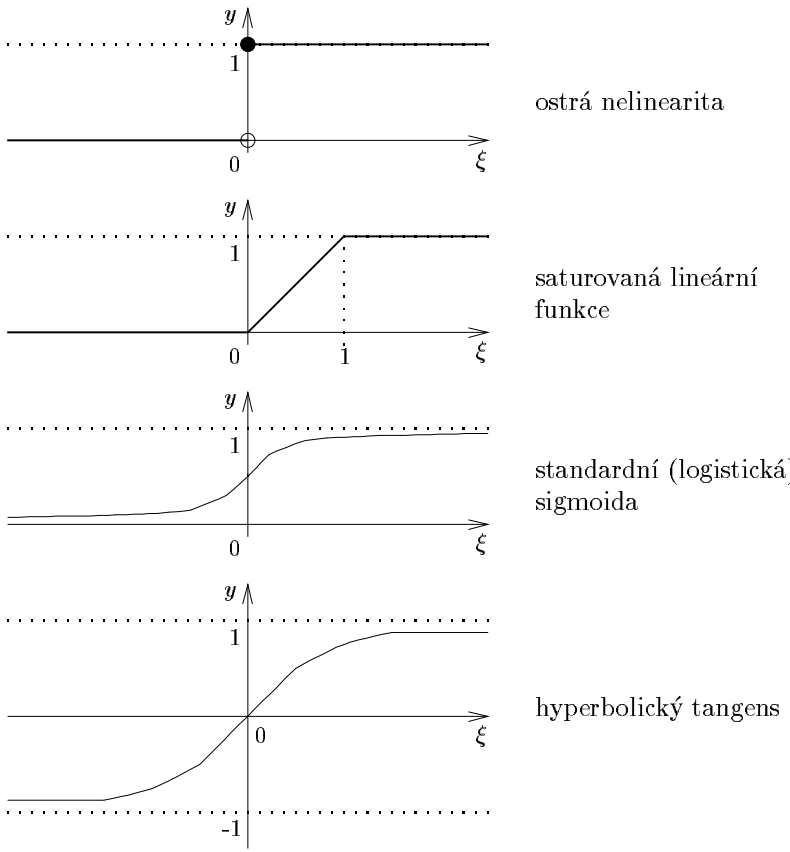
Aktivní dynamika

Aktivní dynamika specifikuje *počáteční stav* sítě a způsob jeho změny v čase při pevné topologii a konfiguraci. V aktivním režimu se na začátku nastaví stavy vstupních neuronů na tzv. *vstup sítě* a zbylé neurony jsou v uvedeném počátečním stavu. Všechny možné vstupy, resp. stavy sítě, tvoří tzv. *vstupní prostor*, resp. *stavový prostor*, neuronové sítě. Po inicializaci stavu sítě probíhá vlastní výpočet. Obecně se uvažuje spojitý vývoj stavu neuronové sítě v čase a hovoří se o tzv. *spojitém modelu*, kdy stav sítě je (spojitou) funkcí času, která je obvykle v aktivní dynamice zadána diferenciální rovnicí. Většinou se však předpokládá diskrétní čas, tj. na počátku se síť nachází v čase 0 a stav sítě se mění jen v čase $1, 2, 3, \ldots$. V každém takovém časovém kroku je podle daného pravidla aktivní dynamiky vybrán jeden neuron (tzv. *sekvenční* výpočet) nebo více neuronů (tzv. *paralelní* výpočet), které *aktualizují* (mění) svůj stav na základě svých vstupů, tj. stavů sousedních neuronů, jejichž výstupy jsou vstupy aktualizovaných neuronů. Podle toho, zda neurony mění svůj stav nezávisle na sobě nebo je jejich aktualizace řízena centrálně, rozlišujeme *asynchronní* a *synchronní* modely neuronových sítí. Stav výstupních neuronů, který se obecně mění v čase, je tzv. *výstupem* neuronové sítě (tj. výsledkem výpočtu). Obvykle se však uvažuje taková aktivní dynamika, že výstup sítě je po nějakém čase konstantní a neuronová síť tak v aktivním. Nejvíce cyklů je v tzv. *úplné topologii* cyklické neuronové sítě, kde výstup libovolného neuronu je vstupem každého neuronu. Příklad obecné cyklické neuronové sítě je na obrázku 1.8, kde jsou vyznačeny všechny cykly. V acyklických sítích naopak cyklus neexistuje a všechny cesty vedou jedním směrem. Příklad acyklické neuronové sítě je na obrázku 1.9, kde je vyznačena nejdelší cesta.



Obr. 1.9: Příklad acyklické architektury.

U acyklické neuronové sítě lze neurony vždy (disjunktně) rozdělit do tzv. *vrstev*, které jsou uspořádány (např. nad sebou) tak, že spoje mezi neurony vedou jen z nižších vrstev do vyšších a obecně mohou přeskočit jednu nebo více vrstev. Speciálním případem takové architektury je tzv. *vícevrstvá neuronová síť*. V této síti je nultá (dolní), tzv. *vstupní* vrstva tvořena vstupními neurony a poslední (horní), tzv. *výstupní* vrstva se skládá z výstupních neuronů. Ostatní, tzv. *skryté* (*mezilehlé*) vrstvy jsou složeny ze skrytých neuronů. Jak už bylo naznačeno, vrstvy číslujeme od nuly, která odpovídá vstupní vrstvě. Tu potom nepočítáme do počtu vrstev sítě (např. dvouvrstvá neuronová síť se skládá ze vstupní, jedné skryté a výstupní vrstvy). V topologii vícevrstvé sítě jsou neurony jedné vrstvy spojeny se všemi neurony bezprostředně následující vrstvy (příp. chybějící spoje lze implicitně chápat jako spoje s nulovými váhami). Proto architekturu takové sítě lze zadat jen počty neuronů v jednotlivých vrstvách, typicky oddělenými pomlčkou, v pořadí od vstupní k výstupní vrstvě. Také cesta v takové síti vede směrem od vstupní vrstvy k výstupní, přičemž obsahuje po jednom neuronu z každé vrstvy. Příklad architektury třívrstvé neuronové sítě 3–4–3–2 s jednou vyznačenou cestou je na obrázku 1.10, kde kromě vstupní a výstupní vrstvy jsou dvě skryté vrstvy.



Obr. 1.11: Grafy sigmoidních aktivačních funkcí.

konfigurace této vícevrstvé sítě. Na začátku jsou (obecně reálné) stavy neuronů vstupní vrstvy nastaveny na vstup sítě a ostatní (tj. skryté a výstupní) neurony jsou pasivní. Výpočet vícevrstvé sítě dále probíhá v diskrétním čase. V časovém kroku 1 jsou aktualizovány stavy neuronů z první (skryté) vrstvy podle rovnice (1.3). To znamená, že neurony z této vrstvy převezmou své vstupy od vstupních neuronů, spočítají svůj vnitřní potenciál jako zváženou sumu těchto vstupů a svůj stav (výstup) určí ze znaménka této sumy pomocí přenosové funkce. V časovém kroku 2 jsou pak aktualizovány stavy neuronů z druhé (skryté) vrstvy opět podle rovnice (1.3). V tomto případě výstupy

ním režimu realizuje nějakou funkci na vstupním prostoru, tj. ke každému vstupu sítě vypočítá právě jeden výstup. Tato tzv. *funkce neuronové sítě* je tedy dána aktivní dynamikou, jejíž rovnice parametricky závisí na topologii a konfiguraci, které se v aktivním režimu nemění. Je zřejmé, že v aktivním režimu se neuronová síť využívá k vlastním výpočtům.

Aktivní dynamika neuronové sítě také určuje funkci jednoho neuronu, jejíž předpis (matematický vzorec) je většinou pro všechny (nevstupní) neurony v síti stejný (tzv. *homogenní* neuronová síť). Zatím jsme uvažovali jen tvar funkce neuronu daný rovnicí (1.3), která byla inspirovaná funkcí biologického neuronu. U modelů neuronových sítí se však běžně setkáváme s jinými funkcemi, které nemusí mít neurofyziologický vzor, ale vznikly již jen matematickou invencí nebo byly dokonce motivovány jinými, např. fyzikálními teoriemi. Například místo zvážené sumy (1.1) se u tzv. *neuronových sítí vyššího řádu* používá obecně polynom více proměnných (vstupů). Nebo někdy vnitřní potenciál neuronu odpovídá formálně normě určující vzdálenost vstupu od váhového vektoru apod. Také diskrétní přenosová funkce (1.2) bývá aproximována spojitou (příp. diferencovatelnou) aktivační funkcí, popř. nahrazena úplně odlišnou funkcí. V této práci se setkáme např. s následujícími *sigmoidními aktivačními funkcemi*: *ostrá nelinearita* (hard limiter) (1.7), *saturovaná lineární funkce* (saturated-linear function, resp. piecewise-linear function) (1.8), *standardní (logistická) sigmoida* (standard sigmoid, resp. logistic function) (1.9), *hyperbolický tangens* (1.10) apod.

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 1 \\ 0 & \xi < 0 \end{cases} \quad \textit{ostrá nelinearita} \tag{1.7}$$

$$\sigma(\xi) = \begin{cases} 1 & \xi > 1 \\ \xi & 0 \leq \xi \leq 1 \\ 0 & \xi < 0 \end{cases} \quad \textit{saturovaná lineární funkce} \tag{1.8}$$

$$\sigma(\xi) = \frac{1}{1+e^{-\xi}} \quad \textit{standardní (logistická) sigmoida} \tag{1.9}$$

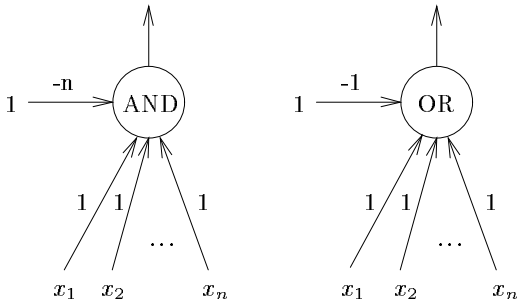
$$\sigma(\xi) = \frac{1-e^{-\xi}}{1+e^{-\xi}} \quad \textit{hyperbolický tangens} \tag{1.10}$$

Grafy těchto sigmoidních funkcí jsou znázorněny na obrázku 1.11. Podle toho, zda je funkce neuronu diskrétní nebo spojitá rozlišujeme *diskrétní* a *analogové* modely neuronových sítí.

Uvedené obecné principy objasníme na příkladě modelu neuronové sítě, u něhož konkrétně popíšeme aktivní dynamiku a nastíníme její geometrickou interpretaci podobně jako u neuronu v odstavci 1.3.1. Uvažujme pevnou architekturu vícevrstvé neuronové sítě (viz obrázek 1.10). Dále předpokládejme, že každý spoj v této síti je ohodnocen synaptickou váhou, tj. je dána

sítě formálně spojeny s tímto neuronem, ale odpovídající váhy jsou nulové, a proto nemají na uvedený neuron žádný vliv. V tomto případě neurony v druhé vrstvě realizují logickou konjunkci (AND), tj. jsou aktivní, právě když všechny relevantní vstupy jsou 1. Na obrázku 1.12 je znázorněno rozdělení vstupního prostoru na čtyři poloprostory P_1, P_2, P_3, P_4 pomocí čtyř neuronů z první vrstvy (srovnejte s příkladem architektury vícevrstvé sítě 3–4–3–2 na obrázku 1.10). Jsou zde vyznačeny tři konvexní oblasti, které jsou průnikem poloprostorů $K_1 = P_1 \cap P_2 \cap P_3$, $K_2 = P_2 \cap P_3 \cap P_4$ a $K_3 = P_1 \cap P_4$ a odpovídají třem neuronům z druhé vrstvy, z nichž každý je aktivní, právě když vstup sítě je z odpovídající oblasti.

Rozdělení vstupního prostoru na konvexní oblasti lze např. využít k rozpoznávání obrazů více písmen (srovnejte s dichotomií obrazů dvou písmen v motivačním příkladu na obrázku 1.5), kde každému písmenu odpovídá jedna konvexní oblast. Někdy nelze část vstupního prostoru odpovídající nějakému obrazu uzavřít do konvexní oblasti. Avšak nekonvexní oblast lze získat sjednocením konvexních oblastí, které je realizováno neuronem třetí (výstupní) vrstvy. To znamená, že výstupní neuron je aktivní, právě když vstup sítě reprezentuje bod ve vstupním prostoru, který leží aspoň v jedné z vybraných konvexních oblastí, které jsou klasifikovány neurony z druhé vrstvy. V tomto případě neurony výstupní vrstvy počítají logickou disjunkci (OR), tj. jsou aktivní, právě když aspoň jeden relevantní vstup je 1. V příkladu na obrázku 1.12 může být výstupní neuron např. aktivní, právě když vstup sítě je z oblasti K_1 nebo K_2 (tj. $K_1 \cup K_2$). Je zřejmé, že obecně lze nekonvexní oblasti klasifikovat pomocí třívrstvé neuronové sítě.

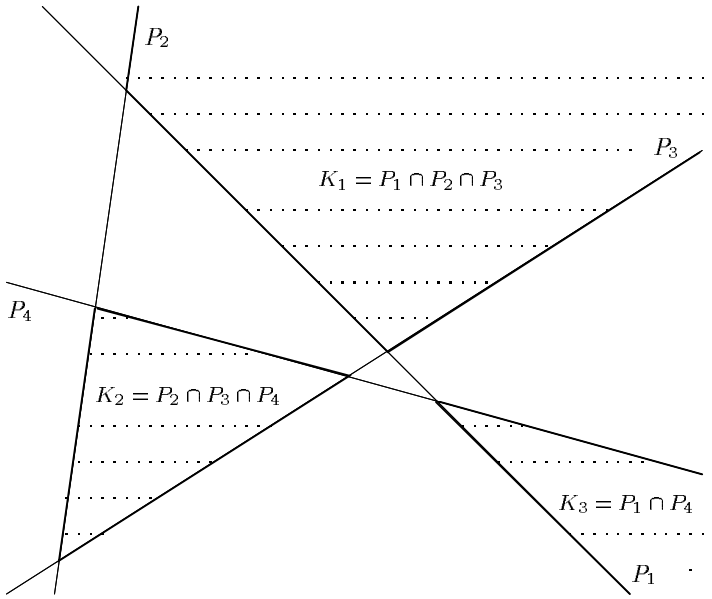


Obr. 1.13: Neuronová realizace funkcí AND a OR.

V předchozích úvahách jsme využili faktu, že pomocí neuronu s aktivní dynamikou (1.3) lze počítat logickou konjunkci (AND) a disjunkci (OR). Na obrázku 1.13 je znázorněna neuronová realizace těchto funkcí. Jednotkové

neuronů z první vrstvy jsou vstupy neuronů z druhé vrstvy. Obdobně v časovém kroku 3 jsou aktualizovány stavy neuronů z třetí vrstvy atd. Výpočet tak probíhá směrem od vstupní vrstvy k výstupní s tím, že v každém časovém kroku jsou paralelně aktualizovány všechny neurony jedné vrstvy, které získají své vstupy ze stavů neuronů předcházející vrstvy. Nakonec jsou aktualizovány stavy neuronů ve výstupní vrstvě, které tvoří výstup sítě, a výpočet vícevrstvé neuronové sítě končí.

Pokusíme se nyní zobecnit naši geometrickou představu funkce neuronu (viz obrázek 1.4) pro funkci třívrstvé neuronové sítě. Je zřejmé, že neurony v první (skryté) vrstvě rozdělí vstupní prostor sítě pomocí příslušných nadrovin na různé poloprostory stejně jako v odstavci 1.3.1, a tedy počet těchto poloprostorů je roven počtu neuronů v první vrstvě. Neurony z druhé vrstvy pak mohou např. klasifikovat průnik některých z těchto poloprostorů, tj. lze pomocí nich reprezentovat konvexní (vypouklé) oblasti ve vstupním prostoru. To znamená, že takový neuron z druhé vrstvy je aktivní, právě když vstup sítě reprezentuje bod ve vstupním prostoru, který leží současně ve všech poloprostorech, které jsou klasifikovány vybranými neurony z první vrstvy. Ostatní neurony z první vrstvy jsou sice v topologii vícevrstvé



Obr. 1.12: Příklad separace konvexních oblastí.

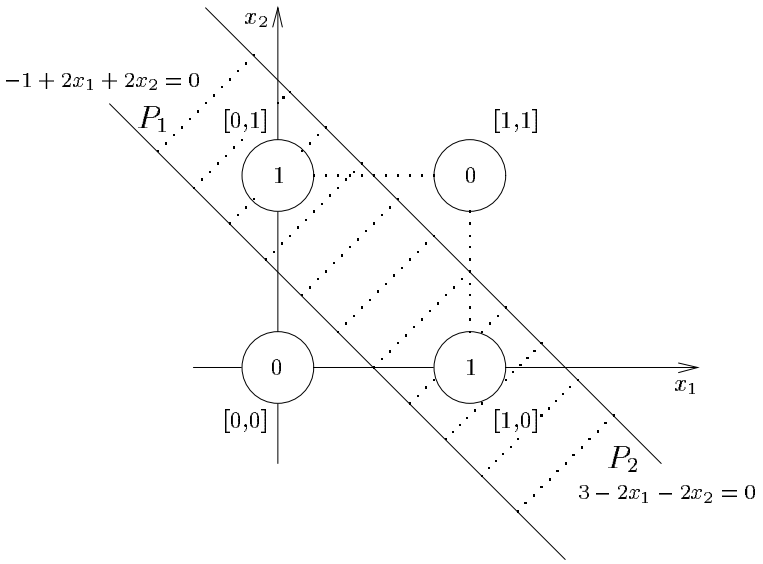
váhy (kromě biasu) zajistí, že zvážená suma skutečných binárních vstupů (z množiny $\{0, 1\}$) bude odpovídat počtu 1 na vstupu, a práh (bias s opačným znaménkem) n v případě funkce AND (resp. 1 v případě funkce OR) zaručuje, že neuron bude aktivní, právě když tento počet bude aspoň n (resp. 1). Samozřejmě, že neurony ve vyšších vrstvách vícevrstvé sítě mohou obecně počítat jiné funkce než logickou konjunkci nebo disjunkci, proto třída možných funkcí vícevrstvé neuronové sítě je bohatší, než jsme v našem příkladě uvažovali.

Výklad o geometrické interpretaci vícevrstvé neuronové sítě budeme ještě ilustrovat na již zmíněném důležitém příkladu logické funkce, vylučovací disjunkce (XOR). Z odstavce 1.3.1 víme, že ji nelze počítat pomocí jednoho neuronu s aktivní dynamikou (1.3). Na obrázku 1.14 (srovnejte s komentářem k obrázku 1.7) vidíme, že vstupy, pro něž výstup funkce XOR je 1, lze pomocí průniku dvou poloprostorů (polorovin) P_1, P_2 ohraničených nadrovinami (přímkami) uzavřít do konvexní oblasti. Funkci XOR lze proto realizovat pomocí dvouvrstvé neuronové sítě 2–2–1 (s jednou skrytou vrstvou), která je znázorněna na obrázku 1.15.

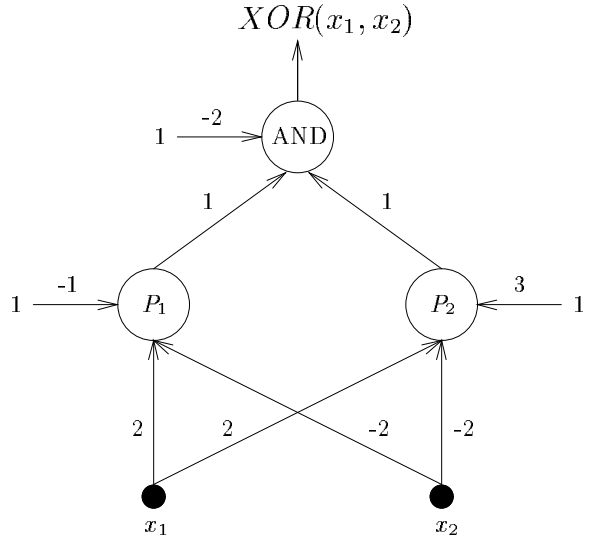
Adaptivní dynamika

Adaptivní dynamika specifikuje *počáteční konfiguraci* sítě a jakým způsobem se mění váhy v síti v čase. Všechny možné konfigurace sítě tvoří tzv. *váhový prostor* neuronové sítě. V adaptivním režimu se tedy na začátku nastaví váhy všech spojů v síti na počáteční konfiguraci (např. náhodně). Po inicializaci konfigurace sítě probíhá vlastní adaptace. Podobně jako v aktivní dynamice se obecně uvažuje spojitý model se spojitým vývojem konfigurace neuronové sítě v čase, kdy váhy sítě jsou (spojitou) funkcí času, která je obvykle v adaptivní dynamice zadána diferenciální rovnicí. Většinou se však předpokládá diskrétní čas adaptace (srovnejte s aktivní dynamikou).

Víme, že funkce sítě v aktivním režimu závisí na konfiguraci. Cílem adaptace je nalézt takovou konfiguraci sítě ve váhovém prostoru, která by v aktivním režimu realizovala předepsanou funkci. Jestliže aktivní režim sítě se využívá k vlastnímu výpočtu funkce sítě pro daný vstup, pak adaptivní režim slouží k *učení* („programování“) této funkce. Existují stovky úspěšných *učících algoritmů* pro různé modely neuronových sítí. Například nejznámější a nejpoužívanější učící algoritmus je backpropagation pro vícevrstvou neuronovou síť, jehož význam pro rozvoj neurovýpočtů jsme objasnili v podkapitole 1.1 a vlastní popis lze najít v podkapitole 2.2. Učení neuronové sítě představuje většinou složitý nelineární optimalizační problém, jehož řešení je i pro menší úlohy časově velmi náročné (desítky hodin i dnů výpočtu na PC).



Obr. 1.14: Geometrické znázornění výpočtu XOR pomocí dvouvrstvé sítě.



Obr. 1.15: Dvouvrstvá síť pro výpočet funkce XOR.

neprogramujeme tak, že bychom popsali přesný postup výpočtu její funkční hodnoty, ale síť sama abstrahuje a zobecňuje charakter funkce v adaptivním režimu v procesu učení ze vzorových příkladů. V tomto smyslu neuronová síť připomíná inteligenci člověka, který získává mnohé své znalosti a dovednosti ze zkušenosti, kterou ani není ve většině případů schopen formulovat analyticky pomocí přesných pravidel či algoritmu. V následujícím výkladu uvedeme opět několik motivačních (trochu nadnesených) příkladů, které nám pomohou tento fenomén pochopit.

Představme si zedníka, který by chtěl svého učně naučit omítat zeď. Pravděpodobně ti, kdo se někdy sami pokoušeli omítnout svůj dům, ví, že první pokusy nebývají moc zdařilé (polovina malty většinou končí na zemi). Jak groteskní by bylo teoretické školení zedníka, který by na tabuli vylekanému učni napsal diferenciální rovnice popisující trajektorii (dráhu) a rychlost pohybu ruky, popř. zápěstí, při nahazování malty na omítanou zeď. I kdyby zednický učeň měl základy v diferenciálním počtu, omítat by se tímto způsobem asi nenaučil. Tuto dovednost může totiž učeň získat jen pozorováním zedníka při omítání a vlastními pokusy korigovanými učitelem.

Nebo již uvedený příklad školáka, který se učí číst (viz podkapitolu 1.3). I zde by analytický popis tvarů ideálních písmen školákovi příliš nepomohl. Pro něj je důležité snažit se pod dozorem učitele přečíst co nejvíce příkladů jednoduchých vět ze slabikáře, ale i vět napsaných rukou učitele na tabuli odlišným písmem. Navíc vlastní rozpoznání písmen není zárukou správného čtení slov, jak ukazuje příklad obrazů dvou anglických slov *THE* (určitý člen), *TEA* (čaj) na obrázku 1.16, kde deformovaný tvar písmene „H“ je totožný s porušeným tvarem písmene „A“. Kontext jednotlivých písmen nebo dokonce význam slov je totiž podstatný pro správné čtení (srovnejte se systémem NETtalk zmíněném v podkapitole 1.1 a odstavci 1.4.2).

Obr. 1.16: Rozlišení písmen „H“ a „A“ na základě kontextu.

Dalším demonstračním příkladem, který je popisován v literatuře [273], je balancování s tyčí koštěte. Byla sestrojena neuronová síť, která dokázala napodobit dovednost cirkusového klauna, který na svém nose udrží koště ve vertikální poloze. Při vlastním experimentu byl použit speciální vozík, na kterém bylo koště volně upevněno (pro jednoduchost v jedné rovině) tak, že

Požadovaná funkce sítě je obvykle zadána tzv. *tréninkovou množinou* (*posloupností*) dvojic vstup/výstup sítě (tzv. *tréninkový vzor*). V motivačním příkladě školáka, který se učí číst (viz odstavec 1.3.1), mohou vstupy tréninkových vzorů být vzorové obrazy písmen (tj. matice intenzit $n = k \times k$ obrazových bodů), kterým jako výstupy odpovídají jednotlivá písmena (např. u výstupního neuronu, který reprezentuje písmeno „A“, je požadovaná hodnota stavu rovna 1, právě když je na vstupu příklad obrazu písmene „A“, a ostatní výstupní neurony jsou v takovém případě pasivní). Tento způsob popisu požadovaného chování sítě modeluje učitele, který pro vzorové vstupy sítě informuje adaptivní mechanismus o správném výstupu sítě. Proto se tomuto typu adaptace říká *učení s učitelem* (supervised learning). Někdy učitel hodnotí kvalitu momentální skutečné odpovědi (výstupu) sítě pro daný vzorový vstup pomocí známky, která je zadána místo požadované hodnoty výstupu sítě (tzv. *klasifikované učení*). Příklady modelů neuronových sítí, které využívají učení s učitelem, budou popsány v kapitole 2 (např. typickým zástupcem z této třídy je již zmiňovaný algoritmus backpropagation).

Jiným typem adaptace je tzv. *samoorganizace*. V tomto případě tréninková množina obsahuje jen vstupy sítě. To modeluje situaci, kdy není k dispozici učitel, proto se tomuto způsobu adaptace také říká *učení bez učitele*. Neuronová síť v adaptivním režimu sama organizuje tréninkové vzory (např. do shluků) a odhaluje jejich souborné vlastnosti. Příklady modelů neuronových sítí, které využívají učení bez učitele, budou popsány v kapitole 4.

1.4 Postavení neuronových sítí v informatice

1.4.1 Neuronové sítě a von neumannovská architektura počítače

Z předchozího výkladu nemusí být zcela zřejmé, v čem spočívá přínos neuronových sítí. Nejprve si uvědomíme, že neuronová síť je schopna v aktivním režimu realizovat požadovanou funkci. V jistém smyslu neuronové sítě představují univerzální výpočetní prostředek (přesněji viz druhou část této knihy), a tedy mají stejnou výpočetní sílu jako klasické počítače např. von neumannovské architektury (tj. pomocí neuronových sítí lze principiálně spočítat vše, co umí např. osobní počítač, a naopak). Tato jejich vlastnost by vzhledem k existenci stovek různých (i velmi exotických) univerzálních výpočetních modelů nebyla tak výjimečná. Navíc je funkce sítě popsána velkým počtem váhových parametrů a vůbec není zřejmé, jak bychom požadovanou funkci v tomto výpočetním modelu programovali.

Hlavní výhodou a zároveň odlišností neuronových sítí od klasické von neumannovské architektury je jejich schopnost učit se. Požadovanou funkci sítě

se totiž nazpaměť naučit vzorové příklady, ale je potřeba umět zobecnit zákonitosti jejich řešení.

Schopnost učit se a zobecňovat je typickou vlastností lidské inteligence. Velkým problémem pro hodnocení generalizační schopnosti neuronové sítě je, že není jasné, jakým způsobem definovat, co je to správná generalizace. Uvažujme otázku z testu inteligence, kdy se má doplnit další člen v číselné posloupnosti $1, 2, 3, \ldots$. Většina lidí by asi doplnila následující číslo 4. Představme, si ale matematika, který si všimne, že číslo 3 je součtem dvou předcházejících čísel 1 a 2, a dle této komplikovanější souvislosti doplní místo čísla 4 číslo 5, které je opět součtem dvou předcházejících čísel 2 a 3. Kromě toho, že bude některými „normálními" lidmi považován za podivína, není vůbec zřejmé, které ze dvou uvedených doplnění je správnou generalizací zákonitosti této posloupnosti. A takových doplnění, které je možné nějakým způsobem zdůvodnit, existuje jistě nekonečně mnoho.

Díky tomu, že neumíme definovat (formalizovat), a tedy ani měřit generalizační schopnosti neuronových sítí, chybí základní kritérium, které by rozhodlo, jaké modely neuronových sítí jsou v konkrétním případě dobré či lepší než jiné apod. Generalizační schopnosti navržených modelů neuronových sítí se většinou ilustrují na jednotlivých příkladech, které (možná díky vhodnému výběru) vykazují dobré vlastnosti, ale tyto vlastnosti nelze nijak formálně ověřit (dokázat). Tento stav je také dle našeho názoru příčinou krize základního výzkumu neuronových sítí.

Na druhou stranu úspěšné aplikace neuronových sítí při řešení důležitých praktických úloh, kde klasické počítače neuspěly, i to, že simulace (velmi zjednodušených) modelů biologických neuronových sítí vykazují prvky podobné lidské inteligenci, naznačují, že tyto modely vystihují určité rysy, důležité pro napodobení inteligentních činností člověka, které počítače von neumannovské architektury postrádají. Základním rysem biologických nervových systémů je hustě propojená síť velkého počtu výpočetních prvků (neuronů), které samy počítají jen jednoduché funkce, což v případě matematických modelů neuronových sítí pravděpodobně vytváří výpočetní paradigma postačující k napodobení inteligentního chování.

Systematická logika a přesnost klasických počítačů je u neuronových sítí nahrazena asociací a neurčitostí, kdy se k novému problému „vybaví" sdružený (podobný) vzorový příklad (tréninkový vzor), ze kterého je zobecněno jeho řešení. Také místo explicitní reprezentace dat v paměti klasických počítačů jsou informace v neuronových sítích zakódovány implicitně a jednotlivým číselným parametrům sítě (kromě vstupů a výstupů) není přiřazen přesný význam. Zatímco klasické počítače jsou citlivé na chybu a změna jednoho bitu může znamenat celkový výpadek systému, neuronové sítě jsou robustní. Je například známo, že po neurochirurgických operacích, kdy je pacientovi odebrána část tkáně mozkové kůry, pacient přechodně zapomíná

by bez zachycení spadlo. Neuronová síť se učila nejprve na základě odchylky (úhlu) koštěte od vertikální polohy a později pomocí filtrovaného obrazu násady koštěte snímaného kamerou určit posuv vozíku (v jedné přímé dráze) tak, aby koště nespadlo. Tréninkové vzory pro její adaptaci, kde vstup odpovídal filtrovanému obrazu koštěte a výstup posuvu vozíku, byly získány od demonstrátora (při zpomalené počítačové simulaci), který nějaký čas pohyboval vozíkem tak, aby koště nespadlo. Po čase neuronová síť sama úspěšně převzala jeho úlohu řízení (již skutečného) vozíku. I zde by bylo teoreticky možné sestavit diferenciální rovnice pro pohyb vozíku, ale než by je klasický počítač von neumannovské architektury vyřešil, koště by pravděpodobně spadlo. Na druhou stranu v tomto jednoduchém demonstračním příkladě (varianta se vstupním úhlem) existuje úspěšný řídící systém založený na klasické teorii řízení.

Podobným příkladem popisovaným v literatuře je řízení přítoku látek potřebných ve složitém výrobním procesu, kde je prakticky nemožné sestavit analytický model. V praxi byla tato činnost prováděna zkušeným pracovníkem, který na základě informací z různých měřidel reguloval pomocí pák přítok jednotlivých látek. Uvedený dobře placený expert by nejen ve vlastním zájmu nebyl ochoten prozradit svou dlouhodobě nabytou zkušenost, ale ani by ji nebyl schopen vyjádřit prostřednictvím přesných pravidel pro pohyb s regulačními pákami. I zde byla zapojena neuronová síť, která se na základě příkladů stavů měřidel a odpovídajících reakcí experta sama po nějakém čase naučila regulovat přítok látek, a expert tak přišel o svou dobře placenou práci.

Z uvedených příkladů vyplývá, že neuronová síť modeluje schopnost člověka učit se z příkladů dovednosti či znalosti, které není schopen řešit algoritmicky pomocí klasických počítačů von neumannovské architektury, protože chybí analytický popis nebo jejich analýza je příliš složitá. Tomu potom odpovídají oblasti aplikace neuronových sítí (viz podkapitolu 1.4.2), kde klasické počítače selhávají. Zřejmě si také nestačí jen pamatovat všechny vzorové příklady (tréninkovou množinu) nazpaměť (např. v tabulce uložené v paměti klasického počítače). Navíc je potřeba zobecňovat (*generalizovat*) jejich zákonitosti, které by umožnily řešit podobné případy, s nimiž se neuronová síť při učení ještě nesetkala. Např. v případě rozpoznávání písmen si není možné pamatovat všechny možné tvary obrazu písmene „A".

Dalším ilustračním příkladem důležitosti generalizační schopnosti lidské inteligence, který uvedeme, je příprava studenta na zkoušku z matematiky. Je zřejmé, že naučení všech vzorových příkladů ve sbírce nazpaměť bez náležitého pochopení postupů řešení nezaručuje úspěšné složení zkoušky. Student pravděpodobně u zkoušky neuspěje, pokud nedostane identický příklad ze sbírky, ale bude mu zadána úloha jen s podobným postupem řešení. Nestačí

některé funkce (např. schopnost mluvit) nebo u nich jen ztrácí určitou obratnost (např. koktá), ale brzy se znovu tyto schopnosti obnoví či zdokonalí, protože jiné neurony převezmou funkci těch původních. Tento jev lze pozorovat i u modelů neuronových sítí, kdy odebráním několika málo neuronů nemusí síť nutně ztratit svoji funkčnost, ale způsobí to třeba jen menší nepřesnosti výsledných odpovědí. Dále u klasických počítačů von neumannovské architektury je sekvenční běh programu lokalizován např. pomocí čítače instrukcí. V neuronových sítích je naopak výpočet distribuován po celé síti a je přirozeně paralelní.

Při srovnávání modelů neuronových sítí s klasickou von neumannovskou architekturou počítače je možné vypozorovat střet dvou inteligencí: biologické a křemíkové. Východiskem, které může nalézt v dnešním přetechnizovaném světě širší uplatnění, je symbióza obou přístupů. Myšlenka vytvořit počítač ke svému obrazu nabývá v poslední době konkrétnější podoby.

1.4.2 Aplikace neuronových sítí

Porovnání modelů neuronových sítí s počítači von neumannovské architektury naznačuje možné oblasti jejich aplikace tam, kde klasické počítače selhávají. Jedná se především o praktické problémy, u kterých není znám algoritmus nebo jejich analytický popis je pro počítačové zpracování příliš komplikovaný. Typicky se neuronové sítě dají použít tam, kde jsou k dispozici příkladová data, která dostatečně pokrývají problémovou oblast. Výhody neuronových sítí oproti klasickým počítačům samozřejmě neznamenají, že by neuronové sítě mohly úplně nahradit současné počítače, protože v případě mechanických výpočtů (např. násobení), které lze jednoduše algoritmicky popsat, nemohou (stejně jako lidé) v rychlosti a přesnosti klasickým počítačům konkurovat. Neuronové sítě ve formě specializovaných modulů pravděpodobně jen obohatí počítače von neumannovské architektury. V následujícím výkladu uvedeme několik možných oblastí aplikace neuronových sítí.

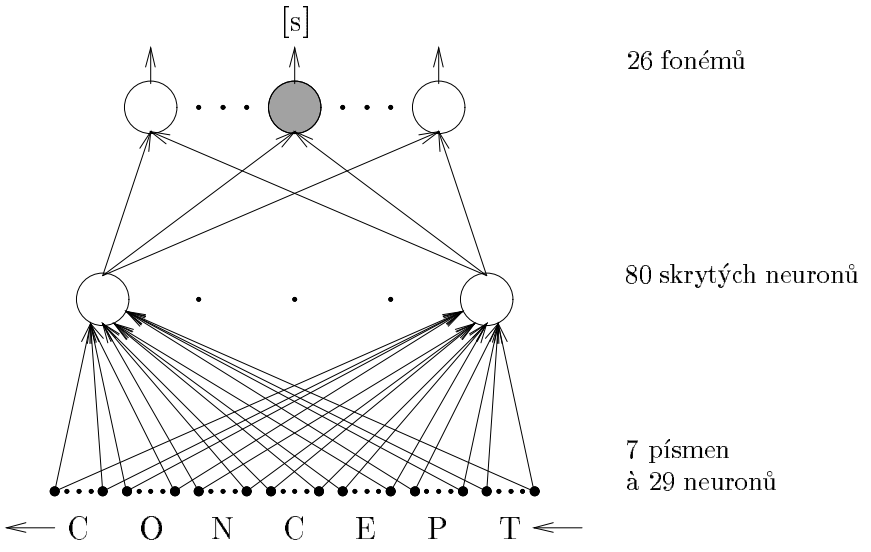
Neuronové sítě lze přirozeným způsobem použít k *rozpoznávání obrazců*, jehož speciální případ — rozpoznání (např. nascanovaných) psaných, resp. tištěných znaků (číslic, písmen apod.) nás jako motivační příklad provázel předchozím výkladem. V tomto případě se takový obraz znaku (např. pomocí klasického počítače) nejprve odseparuje od okolního textu (např. se určí krajní body obrazu) a potom se znormuje, tj. zobrazí do standardizované matice (např. $15 \times 10 = 150$) bodů. Jednotlivé body pak odpovídají vstupům neuronové sítě, které jsou např. aktivní, právě když čára v obrazu zasahuje příslušné body. Každý výstupní neuron v síti představuje možný znak, který je rozpoznán, právě když je tento neuron aktivní. Tréninkovou množinu lze např. vytvořit přepsáním nějakého textu, který je již k dispo-

zici v počítači (odpovídá požadovaným výstupům tréninkových vzorů, tj. identifikovaným znakům), takovým způsobem (např. rukou), pro který budeme neuronovou síť k rozpoznávání potřebovat (představuje odpovídající příklady obrazových vstupů tréninkových vzorů). Neuronovou síť lze pak pomocí této tréninkové množiny učit tak dlouho, dokud není sama schopna rozpoznávat příslušné znaky. Tímto postupem můžeme v relativně krátké době docílit spolehlivosti např. 95% správně rozpoznaných znaků. Podobný postup lze využít např. v robotice pro zpracování vizuálních informací či při vyhodnocování družicových snímků apod.

Další možnou oblastí aplikace neuronových sítí je *řízení* složitých zařízení v dynamicky se měnících podmínkách. V odstavci 1.4.1 jsme uvedli dva motivační příklady z této oblasti: balancování koštěte a regulace přítoku látek ve složitém výrobním procesu. Dalším demonstračním příkladem řídícího systému popisovaného v literatuře [251] je autopilot automobilu, který se v počítačové simulaci pohybuje na dvouproudé dálnici spolu s auty jedoucími stejným směrem. Auto řízené neuronovou sítí určovalo na základě vzdáleností a rychlostí nejbližších aut v obou pruzích svoji vlastní rychlost a změnu pruhu. Dále neuronová síť ovládala volant podle zakřivení dálnice, polohy auta v pruhu a aktuálního úhlu volantu. Je zajímavé, že neuronová síť se kromě úspěšného řízení vozidla (bez kolizí) včetně předjíždění naučila i různé zvyky a styl jízdy (např. riskantní rychlá jízda a časté předjíždění nebo naopak opatrná pomalá jízda) podle řidičů — trenérů, od kterých byly získávány tréninkové vzory.

Jinou důležitou aplikační oblastí neuronových sítí je *predikce* a příp. následné *rozhodování*. Typickými příklady z této oblasti jsou předpověď počasí, vývoj cen akcií na burze, spotřeba elektrické energie apod. Např. při meteorologické předpovědi jsou vstupem neuronové sítě odečty základních parametrů (např. teplota, tlak apod.) v čase a učitelem je skutečný vývoj počasí v následujícím období. Uvádí se, že u předpovědi počasí v rozpětí několika dnů byla síť úspěšnější než meteorologové.

Další možností využití neuronových sítí je *komprese dat* např. pro přenos televizního signálu, telekomunikaci apod. Pro tento účel byla vyvinuta zajímavá technika [49] použití dvouvrstvé neuronové sítě n–$n/4$–n, znázorněná na obrázku 1.17. Počet skrytých neuronů v mezilehlé vrstvě této sítě byl výrazně (čtyřikrát) menší než stejný počet vstupů a výstupů, které reprezentovaly stejný obrazový signál. Dvouvrstvá síť se učila různé obrazové vzory tak, že vstup a výstup tréninkových vzorů představoval totožný obraz a síť pro daný obrazový vstup odpovídala přibližně stejným výstupem. Při vlastním přenosu byl pro daný vstupní obrazový signál $x_1, \ldots, x_n$ u vysílače nejprve vypočten stav skrytých neuronů $t_1, \ldots, t_{\frac{n}{4}}$, jejichž počet byl čtyřikrát menší, a takto kompresovaný obraz byl přenesen informačním kanálem k příjemci, který jej dekódoval výpočtem stavů výstupních neuronů $x'_1, \ldots, x'_n$, a

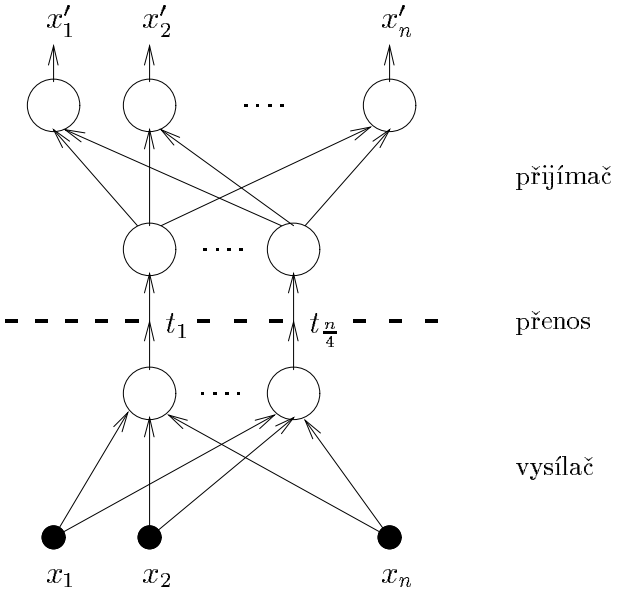


Obr. 1.18: Systém NETtalk.

v anglickém slově *CONCEPT* (pojem) s výslovností ['konsept], kterému odpovídá foném [s]. Stejné písmeno „C“ na začátku tohoto slova však v daném kontextu odpovídalo fonému [k]. Úspěšná implementace systému NETtalk vedla ke snaze vytvořit systém založený na neuronové síti s obrácenou funkcí, která by převáděla mluvený jazyk do psané formy (tzv. fonetický psací stroj).

Jiným příkladem uplatnění neuronových sítí je ***analýza signálů*** jako např. EKG, EEG. Spojitý signál je vzorkován ve stejných časových intervalech a několik posledních diskrétních hodnot úrovně signálu slouží jako vstup např. do dvouvrstvé sítě. Naučená neuronová síť je schopna identifikovat specifický tvar signálu, který je důležitý pro diagnostiku. Např. neuronová síť 40–17–1 byla použita pro klasifikaci EEG signálů se specifickými α-rytmy [219].

Posledním oborem aplikace neuronových sítí, který uvedeme, jsou ***expertní systémy***. Velkým problémem klasických expertních systémů založených na pravidlech je vytvoření báze znalostí, které bývá časově velmi náročnou záležitostí s nejistým výsledkem. Neuronové sítě představují alternativní řešení, kde reprezentace znalostí v bázi vzniká učením z příkladových inferencí. V tomto případě aktivní režim neuronové sítě zastupuje funkci inferenčního stroje. Na druhou stranu implicitní reprezentace znalostí ne-



Obr. 1.17: Komprese při přenosu signálu pomocí dvouvrstvé sítě n–$n/4$–n.

tím získal skoro stejný původní obraz. Při vlastním experimentu se ukázalo, že kvalita přenosu (srovnatelná s jinými způsoby komprese dat) závisí na tom, zda jsou přenášené obrazy podobné tréninkovým vzorům, na které se síť adaptovala.

Další oblastí aplikace neuronových sítí je ***transformace signálů***, jehož příkladem je systém NETtalk [249], zmíněný v podkapitole 1.1, pro převod anglicky psaného textu na mluvený signál. Tento systém je opět založen na dvouvrstvé síti 203–80–26 s 7×29 vstupními neurony pro zakódování kontextu 7 písmen psaného textu (každému z 26 písmen anglické abecedy a čárce, tečce a mezeře odpovídá 1 neuron, který je při jejich výskytu aktivní), 80 skrytými neurony v mezilehlé vrstvě a 26 výstupními neurony reprezentujícími fonémy odpovídajícího mluveného signálu. Funkce sítě je znázorněna na obrázku 1.18, kde se vstupní text postupně přesouvá u vstupních neuronů po jednom písmenu zprava doleva, přitom je aktivní právě výstupní neuron, který reprezentuje foném odpovídající prostřednímu ze sedmi písmen vstupního kontextu. V našem příkladě se čte prostřední písmeno „C“

ropočítače), což se využívá pro velmi rychlé výpočty v reálném čase. Většinou se konstruují tzv. *virtuální* neuropočítače, kde jeden procesor vykonává práci stovek i tisíců neuronů části implementované neuronové sítě.

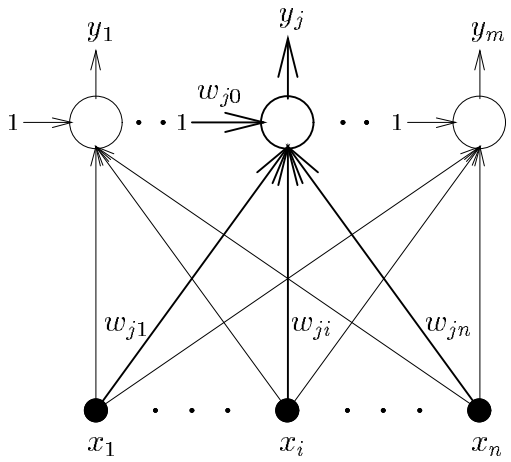
Z hlediska technologie je většina neuropočítačů založena na klasické mikroelektronice (např. VLSI technologie), kde neurony odpovídají hradlům (např. speciálním tranzistorům) a váhy synaptických spojů jsou reprezentovány rezistorovými vazbami. Tento přístup však s sebou přináší technické problémy jako je velká hustota propojení neuronů (roste řádově kvadraticky vzhledem k počtu neuronů) nebo adaptovatelnost vah u všech těchto spojů. Proto adaptivní režim neuronové sítě je někdy předem realizován odděleně pomocí dostupného netwaru na klasickém počítači a výsledná konfigurace sítě je napevno zapojena do příslušného obvodu neuropočítače. Také se stále více uplatňuje optoelektronika a dlouhodobější výhledy počítají s úplně odlišnými technologiemi, jako např. molekulární elektronika, hybridní biočipy apod.

umožňuje pracovat s neúplnou informací a neposkytuje zdůvodnění závěrů, což jsou vlastnosti, bez kterých se prakticky použitelný expertní systém neobejde. Tento problém částečně řeší univerzální neuronový expertní systém *EXPSYS* [265], který obohacuje vícevrstvou neuronovou síť o intervalovou aritmetiku pro práci s nepřesnou informací a o heuristiku analyzující síť, která umožňuje jednoduché vysvětlování závěrů. Systém EXPSYS byl úspěšně aplikován v energetice a medicíně. Například v lékařské aplikaci jsou zakódované příznaky onemocnění a výsledky různých vyšetření vstupem neuronové sítě a diagnózy, popř. doporučená léčba jsou jejím výstupem. Tréninkovou množinu lze získat z kartotéky pacientů.

1.4.3 Implementace neuronových sítí a neuropočítače

Odlišná architektura neuronových sítí vyžaduje speciální hardwarovou realizaci. V této souvislosti hovoříme o tzv. *neuropočítačích*. Avšak vzhledem k rozšířenosti klasických počítačů a kvůli problémům spojeným s hardwarovou realizací neuronových sítí zatím nejjednodušší implementací neuronových sítí, se kterou se nejčastěji (zvláště v České republice) setkáváme, je tzv. *netware*, což je software pro klasické počítače (např. PC), který modeluje práci neuronové sítě. Jedná se většinou o demonstrační programy s efektním uživatelským interfacem, které simulují práci nejznámějších modelů neuronových sítí na jednoduchých příkladech. V některých již dokonalejších programech je možné zadat vlastní aktivní i adaptivní dynamiku, což umožňuje relativně rychle přizpůsobit model neuronové sítě danému praktickému problému nebo ověřit použitelnost navrženého nového modelu. Existují i programovací jazyky (a jejich překladače) pro klasické počítače, které podporují programovou implementaci neuronových algoritmů. Příkladem takového programovacího jazyka je *AXON* [105], který je podobný jazyku *C*. Dokonalejší netware většinou podporuje využití specializovaných koprocesorů (které je možno např. připojit k PC), které efektivně implementují neuronové funkce a urychlují časově náročné učení.

Vlastní neuropočítače většinou nepracují samostatně, ale jsou napojeny na klasické počítače, které mohou realizovat např. uživatelský interface. To je dáno především tím, že neuropočítače nejsou používány jako univerzální počítače, ale převážně fungují jako specializovaná zařízení pro řešení specifických úloh (viz odstavec 1.4.2). Malé neuropočítače jsou spojeny přímo se sběrnicí klasického počítače a větší se mohou uplatnit jako servery na lokální síti. Podle způsobu aktualizace parametrů neuronové sítě rozdělujeme neuropočítače na *spojité* a *diskrétní* a podle typu reprezentace těchto číselných parametrů máme *analogové*, *digitální*, resp. *hybridní* (kombinace analogových a digitálních) neuropočítače. Zřídkakdy jeden neuron v implementované síti odpovídá jednomu procesoru neuropočítače (tzv. *plně implementované* neu-



Obr. 2.1: Architektura sítě perceptronů.

funkce $\sigma : \mathbb{R} \longrightarrow \{0,1\}$, která má tvar ostré nelinearity (1.7). To znamená, že funkce $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \{0,1\}^m$ sítě perceptronů, která závisí na konfiguraci $\mathbf{w}$, je dána následujícím vztahem:

$$y_j = \sigma(\xi_j) \quad j = 1, \ldots, m, \quad \text{kde } \sigma(\xi) = \begin{cases} 1 & \xi \geq 0 \\ 0 & \xi < 0\,. \end{cases} \tag{2.2}$$

V **adaptivním** režimu je požadovaná funkce sítě perceptronů zadána tréninkovou množinou

$$\mathcal{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \mathbb{R}^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \{0,1\}^m \end{array} \quad k = 1, \ldots, p \right\}, \tag{2.3}$$

kde $\mathbf{x}_k$ je reálný vstup k-tého tréninkového vzoru a $\mathbf{d}_k$ je odpovídající požadovaný binární výstup (daný učitelem). Cílem adaptace je, aby síť pro každý vstup $\mathbf{x}_k$ ($k = 1, \ldots, p$) z tréninkové množiny odpovídala v aktivním režimu požadovaným výstupem $\mathbf{d}_k$, tj. aby platilo:

$$\mathbf{y}(\mathbf{w}, \mathbf{x}_k) = \mathbf{d}_k \quad k = 1, \ldots, p\,. \tag{2.4}$$

Samozřejmě podmínku (2.4) nelze vždy splnit, protože ne každou funkci lze počítat jedním perceptronem (viz např. obrázek 1.7 funkce XOR) nebo tréninková množina nemusí být funkcí (tj. k jednomu vstupu jsou požadované

Kapitola 2

Klasické modely neuronových sítí

2.1 Síť perceptronů

Historicky prvním úspěšným modelem neuronové sítě (viz podkapitolu 1.1) byla *síť perceptronů* [238]. **Organizační** dynamika této sítě specifikuje na začátku pevnou architekturu jednovrstvé sítě n-m. To znamená, že síť se skládá z n vstupních neuronů, z nichž každý je vstupem každého z m výstupních neuronů, jak je naznačeno na obrázku 2.1. Označme $x_1, \ldots, x_n$ reálné stavy vstupních neuronů, tj. $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ je vstup sítě, a $y_1, \ldots, y_m$ binární stavy výstupních neuronů, tj. $\mathbf{y} = (y_1, \ldots, y_m) \in \{0,1\}^m$ je výstup sítě. Dále w_{ji} představuje reálnou synaptickou váhu spoje od i-tého vstupního ($i = 1, \ldots, n$) k j-tému výstupnímu ($j = 1, \ldots, m$) neuronu a $w_{j0} = -h_j$ je bias (práh h_j s opačným znaménkem) j-tého výstupního neuronu odpovídající formálnímu jednotkovému vstupu $x_0 = 1$.

Aktivní dynamika sítě perceptronů určuje způsob výpočtu funkce sítě. V tomto případě se reálné stavy neuronů ve vstupní vrstvě nastaví na vstup sítě a výstupní neurony počítají svůj binární stav, který určuje výstup sítě, stejným způsobem jako formální neuron (viz rovnici (1.3)). To znamená, že každý perceptron nejprve vypočte svůj vnitřní potenciál jako příslušnou *afinní kombinaci* vstupů:

$$\xi_j = \sum_{i=0}^{n} w_{ji} x_i \quad j = 1, \ldots, m, \tag{2.1}$$

jejíž koeficienty $\mathbf{w} = (w_{10}, \ldots, w_{1n}, \ldots, w_{m0}, \ldots, w_{mn})$ tvoří konfiguraci sítě. Stav perceptronu se pak určí z jeho vnitřního potenciálu aplikací aktivační

dva různé výstupy). V takovém případě se snažíme, aby se síť naučila co nejvíce vzorů. V praxi je také někdy lepší, aby se síť nenaučila tréninkovou množinu stoprocentně, protože příkladové vzory nemusí být úplně přesné (učitel nemusí být dokonalý).

Na začátku adaptace v čase 0 jsou váhy konfigurace $\mathbf{w}^{(0)}$ nastaveny náhodně blízko nuly, např. $w_{ji}^{(0)} \in \langle -1, 1\rangle$ $(j = 1, \ldots, m,\ i = 0, \ldots, n)$. Síť perceptronů má diskrétní adaptivní dynamiku. V každém časovém kroku učení $t = 1, 2, 3, \ldots$ je síti předložen jeden vzor z tréninkové množiny a síť se ho snaží naučit, tj. adaptuje podle něj své váhy. Pořadí vzorů při učení je dáno tzv. *tréninkovou strategií*, kterou lze např. volit v analogii s lidským učením. Některý student na zkoušku několikrát přečte učebnici, jiný se hned při prvním čtení vše důkladně učí a oba na konci popř. opakují části, které neumí. Adaptace sítě perceptronů obvykle probíhá v tzv. *tréninkových cyklech*, ve kterých se systematicky prochází všechny vzory tréninkové množiny (popř. každý vzor i vícekrát za sebou). Tedy např. v čase $t = (c-1)p + k$ (kde $1 \leq k \leq p$), který odpovídá c-tému tréninkovému cyklu, se síť učí k-tý tréninkový vzor.

Adaptivní dynamika sítě perceptronů, která určuje změnu konfigurace $\mathbf{w}^{(t)}$ v čase $t > 0$, kdy je předložen k-tý tréninkový vzor, je dána následujícím *perceptronovým učícím pravidlem*:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \varepsilon x_{ki} \left(y_j(\mathbf{w}^{(t-1)}, \mathbf{x}_k) - d_{kj} \right) \qquad \begin{matrix} j = 1, \ldots, m \\ i = 0, \ldots, n\,. \end{matrix} \tag{2.5}$$

Rychlost učení (learning rate) $0 < \varepsilon \leq 1$ je mírou vlivu vzorů na adaptaci („motivace“ k učení). Většinou se na začátku volí malá rychlost učení, která později během adaptace roste. To v analogii s přípravou studenta na zkoušku odpovídá prvnímu povrchnímu seznámení s předmětem učení a pozdějšímu důkladnému doučení detailů.

Výraz $y_j(\mathbf{w}^{(t-1)}, \mathbf{x}_k) - d_{kj}$ ve vzorci (2.5) je rozdíl mezi skutečným j-tým výstupem sítě pro vstup k-tého vzoru a požadovanou hodnotou odpovídajícího výstupu tohoto vzoru. Určuje tedy chybu j-tého výstupu sítě pro k-tý tréninkový vzor. Je zřejmé, že pokud je tato chyba nulová, příslušné váhy se neadaptují. V opačném případě může být tato chyba buď 1, nebo -1, protože uvažujeme binární výstupy. V geometrické interpretaci adaptace podle (2.5) znamená, že nadrovina s koeficienty $w_{j0}, \ldots, w_{jn}$ příslušející k j-tému neuronu se ve vstupním prostoru posune ve směru chybně klasifikovaného vzoru $\mathbf{x}_k$, aby jej zahrnula do správného poloprostoru. Objevitel perceptronu Rosenblatt ukázal (viz větu 12.24), že adaptivní dynamika (2.5) zajistí, aby síť po konečném počtu kroků adaptivního režimu nalezla konfiguraci (pokud existuje), pro kterou bude správně klasifikovat všechny tréninkové vzory (tj. chyba sítě bude vzhledem ke tréninkové množině nulová), a tedy bude splněna podmínka (2.4).

Vzhledem k tomu, že perceptronová síť může počítat jen omezenou třídu funkcí, je význam tohoto modelu spíše teoretický. Naznačená věta o konvergenci adaptivního režimu sítě není zárukou efektivity učení, což potvrdily časově náročné praktické experimenty. Také generalizační schopnost tohoto modelu není velká, protože síť perceptronů lze použít jen v případě, kdy klasifikované objekty jsou ve vstupním prostoru oddělitelné nadrovinou (např. speciální úlohy rozpoznávání obrazů apod.). Tento jednoduchý model je však základem složitějších modelů jako je obecná vícevrstvá síť s učícím algoritmem backpropagation.

2.2 Vícevrstvá síť a backpropagation

Nejznámější a nejpoužívanější model neuronové sítě je *vícevrstvá neuronová síť* s učícím algoritmem zpětného šíření chyby (backpropagation) [243], který se používá přibližně v 80% všech aplikací neuronových sítí. Význam uvedeného algoritmu pro rozvoj neurovýpočtů jsme objasnili v podkapitole 1.1. Tento model je zobecněním sítě perceptronů pro architekturu se skrytými vrstvami (tzv. *vícevrstvý perceptron*), a proto se při jeho výkladu budeme na principy modelu sítě perceptronů odvolávat (viz podkapitolu 2.1). Vzhledem k rozšířenosti vícevrstvého perceptronu a jeho určitým nedostatkům existuje mnoho variant tohoto modelu, které se snaží zlepšit jeho vlastnosti. Nejprve popíšeme základní variantu a pak naznačíme některé možné modifikace.

2.2.1 Organizační a aktivní dynamika

Organizační dynamika vícevrstvého perceptronu specifikuje na začátku (většinou) pevnou topologii vícevrstvé neuronové sítě. Standardně se používá dvouvrstvá, resp. třívrstvá síť (viz např. obrázek 1.10), protože význam skrytých neuronů a jejich vazeb, potřebný pro návrh speciální topologie, není znám. V našem výkladu popíšeme model s obecnou acyklickou architekturou. Pro tento účel zavedeme následující značení. Množinu n vstupních neuronů označíme X a množinu m výstupních neuronů Y. Neurony značíme indexy i, j apod. a ξ_j představuje reálný vnitřní potenciál a y_j reálný stav, resp. výstup neuronu j. Podobně jako u sítě perceptronů spoj od neuronu i k nevstupnímu neuronu j je ohodnocen reálnou synaptickou váhou w_{ji} a $w_{j0} = -h_j$ je bias (práh h_j s opačným znaménkem) nevstupního neuronu j odpovídající formálnímu jednotkovému vstupu $y_0 = 1$. Dále $j_{\leftarrow}$ označuje množinu všech neuronů, z nichž vede spoj do neuronu j a které jsou tedy vstupem neuronu j (tj. včetně formálního jednotkového vstupu $0 \in j_{\leftarrow}$), a $j^{\rightarrow}$ je množina neuronů, do nichž vede spoj z neuronu j a kterým je proto neuron j vstupem.

Parciální chyba $E_k(\mathbf{w})$ sítě vzhledem ke k-tému tréninkovému vzoru je úměrná součtu mocnin odchylek skutečných hodnot výstupů sítě pro vstup k-tého tréninkového vzoru od odpovídajících požadovaných hodnot výstupů u tohoto vzoru:

$$E_k(\mathbf{w}) = \frac{1}{2}\sum_{j\in Y}\left(y_j(\mathbf{w},\mathbf{x}_k) - d_{kj}\right)^2 . \tag{2.10}$$

Cílem adaptace je minimalizace chyby sítě (2.9) ve váhovém prostoru. Vzhledem k tomu, že chyba sítě přímo závisí na komplikované nelineární složené funkci vícevrstvé sítě, tento cíl představuje netriviální optimalizační problém. Pro jeho řešení se v základním modelu používá nejjednodušší varianta gradientní metody, která vyžaduje diferencovatelnost chybové funkce.

Na začátku adaptace v čase 0 jsou váhy konfigurace $\mathbf{w}^{(0)}$ nastaveny náhodně blízko nuly, např. $w_{ji}^{(0)} \in \langle -1, 1\rangle$ (nebo důmyslněji řádově $1/\sqrt{|j_{\leftarrow}|}$, kde $|j_{\leftarrow}|$ je počet vstupů $i \in j_{\leftarrow}$ neuronu j). Adaptace probíhá v diskrétních časových krocích, které odpovídají tréninkovým cyklům. Nová konfigurace $\mathbf{w}^{(t)}$ v čase $t > 0$ se vypočte:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)} , \tag{2.11}$$

kde změna vah $\Delta\mathbf{w}^{(t)}$ v čase $t > 0$ je úměrná zápornému gradientu chybové funkce $E(\mathbf{w})$ v bodě $\mathbf{w}^{(t-1)}$:

$$\Delta w_{ji}^{(t)} = -\varepsilon\frac{\partial E}{\partial w_{ji}}\left(\mathbf{w}^{(t-1)}\right) , \tag{2.12}$$

kde $0 < \varepsilon < 1$ je rychlost učení, jejíž význam je podobný jako u sítě perceptronů.

K lepšímu pochopení uvedené gradientní metody nám opět pomůže geometrická představa. Na obrázku 2.2 je schematicky znázorněna chybová funkce $E(\mathbf{w})$ tak, že konfigurace, která představuje (typicky) mnohorozměrný vektor vah $\mathbf{w}$, se promítá na osu x. Chybová funkce určuje chybu sítě vzhledem k pevné tréninkové množině v závislosti na konfiguraci. V adaptivním režimu hledáme takovou konfiguraci, pro kterou je chybová funkce minimální. Začneme s náhodně zvolenou konfigurací $\mathbf{w}^{(0)}$, kdy odpovídající chyba sítě od požadované funkce bude pravděpodobně velká. V analogii s lidským učením to odpovídá počátečnímu nastavení synaptických vah u novorozence, který místo požadovaného chování jako chůze, řeč apod. provádí náhodné pohyby a vydává neurčité zvuky. Při adaptaci sestrojíme v tomto bodě $\mathbf{w}^{(0)}$ ke grafu chybové funkce tečný vektor (gradient) $\frac{\partial E}{\partial \mathbf{w}}(\mathbf{w}^{(0)})$ a posuneme se ve směru tohoto vektoru dolů o ε. Pro dostatečně malé ε tak získáme novou konfiguraci $\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + \Delta\mathbf{w}^{(1)}$, pro kterou je chybová funkce menší než pro původní konfiguraci $\mathbf{w}^{(0)}$, tj. $E(\mathbf{w}^{(0)}) \geq E(\mathbf{w}^{(1)})$. Celý postup konstrukce tečného vektoru opakujeme pro $\mathbf{w}^{(1)}$ a získáme tak $\mathbf{w}^{(2)}$ takové, že

V **aktivním** režimu počítá vícevrstvá síť pro daný vstup funkci $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow (0,1)^m$, která je určena konfigurací $\mathbf{w}$. Výpočet probíhá podle následující diskrétní aktivní dynamiky. V čase 0 jsou odpovídající stavy vstupních neuronů y_i $(i \in X)$ nastaveny na vstup sítě a ostatní neurony nemají určen svůj stav. V čase $t > 0$ jsou vypočteny reálné hodnoty vnitřních potenciálů

$$\xi_j = \sum_{i\in j_{\leftarrow}} w_{ji} y_i \tag{2.6}$$

všech neuronů j, jejichž vstupy (z $j_{\leftarrow}$) již mají určen svůj stav. To znamená, že v čase t jsou aktualizovány neurony v t-té vrstvě. Z vnitřního potenciálu (2.6) je pak stanoven reálný stav $y_j = \sigma(\xi_j)$ neuronu j pomocí diferencovatelné aktivační funkce $\sigma : \mathbb{R} \longrightarrow (0,1)$ standardní sigmoidy (1.9), která spojitě aproximuje ostrou nelinearitu (1.7):

$$\sigma(\xi) = \frac{1}{1+e^{-\lambda\xi}} . \tag{2.7}$$

Diferencovatelnost užité přenosové funkce (2.7) a z ní plynoucí diferencovatelnost funkce sítě je podstatná pro učící algoritmus backpropagation. Reálný parametr *strmosti* (gain) λ určuje nelineární nárůst (pro $\lambda < 0$ pokles) standardní sigmoidy v okolí nuly (pro $\lambda \to \infty$ dostaneme ostrou nelinearitu), tj. míru „rozhodnosti“ neuronu. Graf standardní sigmoidy pro $\lambda > 0$ je znázorněn na obrázku 1.11 (pro $\lambda < 0$ je souměrný podle osy y). V základním modelu se obvykle uvažuje $\lambda = 1$ (viz rovnici (1.9)), avšak obecně může být strmost λ_j (a tedy i aktivační funkce σ_j) pro každý (nevstupní) neuron j různá. Stav neuronu se pak počítá:

$$y_j = \sigma_j(\xi_j), \quad \text{kde } \sigma_j(\xi) = \frac{1}{1+e^{-\lambda_j\xi}} . \tag{2.8}$$

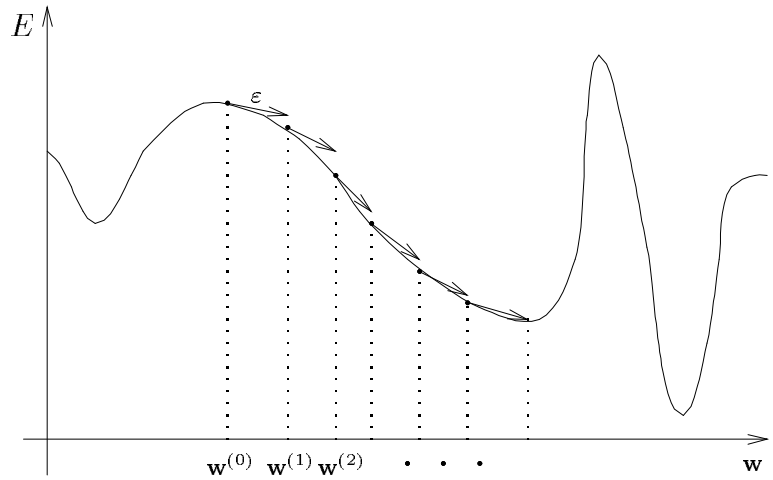
Tímto způsobem jsou v aktivním režimu postupně vypočteny výstupy všech neuronů, protože vícevrstvá síť je acyklická (souvislá) síť. Aktivní režim je ukončen, když je stanoven stav všech neuronů v síti, speciálně výstupních neuronů, které určují výstup sítě, tj. hodnotu funkce sítě pro daný vstup.

2.2.2 Adaptivní dynamika

Adaptivní režim vícevrstvé sítě probíhá podobně jako u sítě perceptronů. Požadovaná funkce je opět zadána tréninkovou množinou (2.3). *Chyba sítě* $E(\mathbf{w})$ vzhledem k této tréninkové množině je definována jako součet parciálních chyb sítě $E_k(\mathbf{w})$ vzhledem k jednotlivým tréninkovým vzorům a závisí na konfiguraci sítě $\mathbf{w}$:

$$E(\mathbf{w}) = \sum_{k=1}^{p} E_k(\mathbf{w}) . \tag{2.9}$$

$E(\mathbf{w}^{(1)}) \geq E(\mathbf{w}^{(2)})$ atd., až se v limitě dostaneme do lokálního minima chybové funkce. Ve vícerozměrném váhovém prostoru tento postup přesahuje naši představivost. I když při vhodné volbě rychlosti učení ε gradientní metoda vždy konverguje k nějakému lokálnímu minimu z libovolné počáteční konfigurace, není vůbec zaručeno, že se tak stane v reálném čase. Obvykle je tento proces časově velmi náročný (několik dnů výpočtu PC) i pro malé topologie vícevrstvé sítě (desítky neuronů).



Obr. 2.2: Gradientní metoda.

Hlavním problémem gradientní metody je, že pokud již nalezne lokální minimum, pak toto minimum nemusí být globální (viz obrázek 2.2). Uvedený postup adaptace se v takovém lokálním minimu zastaví (nulový gradient) a chyba sítě se již dále nesnižuje. To lze v naší analogii s učením člověka interpretovat tak, že počáteční nastavení konfigurace v okolí nějakého minima chybové funkce určuje možnosti jedince učit se, resp. jeho inteligenci. Inteligentnější lidé začínají svoji adaptaci v blízkosti hlubších minim. I zde je však chybová funkce definovaná relativně vzhledem požadovanému „inteligentnímu“ chování (tréninková množina), které však nemusí být univerzálně platné. Hodnotu člověka nelze změřit žádnou chybovou funkcí. Elektrické šoky aplikované v psychiatrických léčebnách připomínají některé metody adaptace neuronových sítí, které v případě, že se učení zastavilo v mělkém lokálním minimu chybové funkce, náhodně vnáší šum do konfigurace sítě,

aby se síť dostala z oblasti atrakce tohoto lokálního minima a mohla příp. konvergovat k hlubšímu minimu.

2.2.3 Strategie zpětného šíření

K realizaci uvedené adaptivní dynamiky (2.11) potřebujeme vypočítat gradient chybové funkce ve vzorci (2.12), což vzhledem ke komplikovanosti chybové funkce není triviální. Nejprve užitím pravidla o derivaci součtu pro (2.9) převedeme tento gradient na součet gradientů parciálních chybových funkcí:

$$\frac{\partial E}{\partial w_{ji}} = \sum_{k=1}^{p} \frac{\partial E_k}{\partial w_{ji}} \,. \tag{2.13}$$

Protože funkce sítě je složena z funkcí jednotlivých neuronů, pro výpočet gradientu parciální chybové funkce je vhodné použít pravidlo o derivaci složené funkce:

$$\frac{\partial E_k}{\partial w_{ji}} = \frac{\partial E_k}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial w_{ji}} \,. \tag{2.14}$$

Parciální derivaci $\frac{\partial \xi_j}{\partial w_{ji}}$ v rovnici (2.14) získáme derivováním vnitřního potenciálu (2.6):

$$\frac{\partial \xi_j}{\partial w_{ji}} = y_i \tag{2.15}$$

a parciální derivaci $\frac{\partial y_j}{\partial \xi_j}$ získáme derivováním aktivační funkce (2.8), jejíž derivaci lze vyjádřit pomocí funkční hodnoty následujícím způsobem:

$$\begin{aligned} \frac{\partial y_j}{\partial \xi_j} &= \frac{\lambda_j e^{-\lambda_j \xi_j}}{\left(1 + e^{-\lambda_j \xi_j}\right)^2} = \\ &= \frac{\lambda_j}{1 + e^{-\lambda_j \xi_j}} \left(1 - \frac{1}{1 + e^{-\lambda_j \xi_j}}\right) = \lambda_j y_j (1 - y_j) \,. \end{aligned} \tag{2.16}$$

Dosazením (2.15) a (2.16) do (2.14) dostáváme:

$$\frac{\partial E_k}{\partial w_{ji}} = \frac{\partial E_k}{\partial y_j} \lambda_j y_j (1 - y_j) y_i \,. \tag{2.17}$$

Pro výpočet parciální derivace $\frac{\partial E_k}{\partial y_j}$ ve vzorci (2.17) se používá strategie zpětného šíření, jak už název učícího algoritmu backpropagation napovídá. Pokud $j \in Y$ je výstupní neuron, pak uvedenou derivaci lze vypočítat přímým derivováním parciální chybové funkce (2.10):

$$\frac{\partial E_k}{\partial y_j} = y_j - d_{kj} \quad j \in Y \,, \tag{2.18}$$

(c) Vypočítej gradient parciální chybové funkce $\frac{\partial E_k}{\partial w_{ji}}(\mathbf{w}^{(t-1)})$ v bodě $\mathbf{w}^{(t-1)}$ podle vzorce (2.17) s využitím parciálních derivací $\frac{\partial E_k}{\partial y_j}(\mathbf{w}^{(t-1)})$ vypočtených v kroku 5b.

(d) Přičti $E'_{ji} := E'_{ji} + \frac{\partial E_k}{\partial w_{ji}}(\mathbf{w}^{(t-1)})$.

6. { Platí $E'_{ji} = \frac{\partial E}{\partial w_{ji}}(\mathbf{w}^{(t-1)})$. }
   Dle (2.12) polož $\Delta w_{ji}^{(t)} := -\varepsilon E'_{ji}$.

7. Dle (2.11) polož $w_{ji}^{(t)} := w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)}$.

8. Vypočítej chybu sítě $E(\mathbf{w}^{(t)})$ pro konfiguraci $\mathbf{w}^{(t)}$ podle vzorců (2.9) a (2.10).

9. Je-li chyba $E(\mathbf{w}^{(t)})$ dostatečně malá, skonči, jinak pokračuj krokem 3.

Ačkoliv vlastní popis učícího algoritmu backpropagation je formulován pro klasický von neumannovský model počítače, přesto je zřejmé, že jej lze implementovat distribuovaně. Pro každý tréninkový vzor probíhá nejprve aktivní režim pro jeho vstup tak, že informace se v neuronové síti šíří od vstupu směrem k jejímu výstupu. Potom na základě externí informace učitele o požadovaném výstupu, tj. o chybě u jednotlivých výstupů, se počítají parciální derivace $\frac{\partial E_k}{\partial y_j}$ (resp. $\frac{\partial E_k}{\partial w_{ji}}$) tak, že signál se šíří zpět směrem od výstupu ke vstupu. Tento zpětný chod připomíná obrácený „aktivní režim“, kdy „vstup“ odpovídá požadovanému výstupu, parciální derivace $\frac{\partial E_k}{\partial y_j}$ reprezentuje „stav“ neuronu j a „aktivní dynamika“ je dána vzorcem (2.18) pro „vstupní“ a (2.19) pro skrytý neuron. Výpočet sítě při zpětném chodu probíhá sekvenčně po vrstvách, přitom v rámci jedné vrstvy může probíhat paralelně.

K uvedenému algoritmu ještě připojíme několik implementačních poznámek pro ty, kteří by chtěli učící algoritmus backpropagation programovat na klasickém počítači. Pro obecnou acyklickou architekturu sítě (pro vícevrstvou topologii dáno implicitně) je nutno pro každý neuron sítě j pamatovat množinu neuronů $j_{\leftarrow}$, od nichž vede spoj k neuronu j, tj. množinu jeho vstupů (včetně příslušných synaptických vah). To je zvláště potřeba v aktivním režimu (výpočet vnitřního potenciálu podle vzorce (2.6)), který se uplatní i v učícím algoritmu backpropagation (krok 5a). Na druhou stranu pro zpětné šíření v tomto algoritmu není potřeba pro každý neuron sítě j explicitně pamatovat množinu neuronů $j^{\rightarrow}$, ke kterým vede spoj od neuronu j, tj. množinu jeho výstupů. V kroku 5b tohoto algoritmu se sice počítá $\frac{\partial E_k}{\partial y_j}$ podle vzorce (2.19), ve kterém se sčítá přes neurony $r \in j^{\rightarrow}$, ale to lze obejít tak, že předem pro každý neuron j budeme $\frac{\partial E_k}{\partial y_j}$ počítat postupně

což odpovídá chybě výstupního neuronu j pro k-tý tréninkový vzor. Pro skrytý neuron $j \notin X \cup Y$ se při výpočtu $\frac{\partial E_k}{\partial y_j}$ znovu uplatní pravidlo o derivování složené funkce, kde opět vypočteme derivace, které lze získat přímým derivováním:

$$\begin{aligned}\frac{\partial E_k}{\partial y_j} &= \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r}\frac{\partial y_r}{\partial \xi_r}\frac{\partial \xi_r}{\partial y_j} = \\ &= \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r}\lambda_r y_r(1-y_r)w_{rj} \quad j \notin X \cup Y\,. \end{aligned} \tag{2.19}$$

Ve vzorci (2.19) jsme výpočet parciální derivace $\frac{\partial E_k}{\partial y_j}$ pro skrytý neuron j převedli na výpočet parciálních derivací $\frac{\partial E_k}{\partial y_r}$ u neuronů $r \in j^{\rightarrow}$, do nichž vede spoj od neuronu j, a proto mají neuron j jako svůj vstup. Při výpočtu $\frac{\partial E_k}{\partial y_j}$ tedy můžeme postupovat od výstupních neuronů, pro které lze tuto parciální derivaci počítat podle (2.18), postupně směrem ke vstupním tak, že u skrytých neuronů vypočteme $\frac{\partial E_k}{\partial y_j}$ podle (2.19), pokud jsou již vypočteny parciální derivace $\frac{\partial E_k}{\partial y_r}$ pro všechny $r \in j^{\rightarrow}$. Korektnost tohoto postupu opět vyplývá z acykličnosti architektury sítě.

2.2.4 Implementace backpropagation

Nyní ještě shrneme adaptivní dynamiku do přehledného algoritmu:

1. Polož diskrétní čas adaptace $t := 0$.

2. Zvol náhodně $w_{ji}^{(0)} \in \langle -1, 1\rangle$.

3. Zvětši $t := t + 1$.

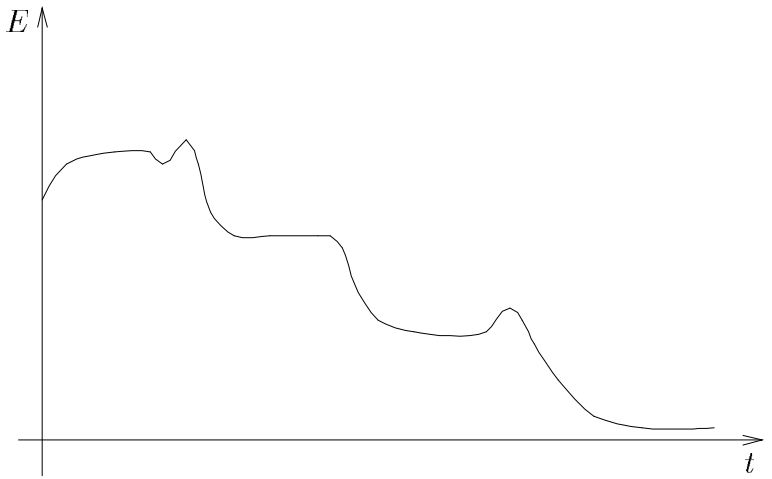
4. Polož $E'_{ji} := 0$ pro každý spoj od i do j.

5. Pro každý tréninkový vzor $k = 1, \ldots, p$ dělej:

   (a) Vypočítej výstup funkce sítě $\mathbf{y}(\mathbf{w}^{(t-1)}, \mathbf{x}_k)$, resp. stavy a vnitřní potenciály všech neuronů (aktivní režim) pro vstup $\mathbf{x}_k$ od k-tého tréninkového vzoru podle vzorců (2.6) a (2.8).

   (b) Zpětným šířením vypočítej pro každý nevstupní neuron $j \notin X$ parciální derivaci $\frac{\partial E_k}{\partial y_j}(\mathbf{w}^{(t-1)})$ parciální chybové funkce k-tého tréninkového vzoru podle stavu neuronu j v bodě $\mathbf{w}^{(t-1)}$ pomocí vzorců (2.18) a (2.19) (využij k tomu hodnoty stavů a vnitřních potenciálů neuronů vypočtené v kroku 5a).

Z uvedeného vyplývá, že v adaptivním režimu, založeném na učícím algoritmu backpropagation, nám jde o to, abychom vhodnou volbou rychlosti učení ε, co nejvíce minimalizovali chybovou funkci sítě $E(\mathbf{w})$ (např. nalezli její globální minimum). Při malém ε metoda konverguje příliš pomalu (chyba nepatrně klesá) a na druhou stranu pro větší ε diverguje (chyba roste). Ladění parametru ε připomíná ladění správné frekvence televizního signálu. Proto je potřeba při řízení učícího procesu mít určitou zkušenost s volbou ε. Následující doporučení lze formulovat jen velmi nepřesně. Pro větší topologie je lepší začínat s menším ε (např. řádově tisíciny, popř. desetitisíciny). Při úspěšné konvergenci lze ε nepatrně zvýšit, při zjevné divergenci či oscilaci hodnotu ε snížíme (např. exponenciálně). I při velkém vzrůstu chyby po předešlé úspěšné konvergenci bez dlouhodobější změny ε, pokračujeme v adaptaci se stejným ε, protože metoda někdy přechází ve váhovém prostoru do oblasti lepší konvergence, což se nám potvrdí nebo vyvrátí v dalších tréninkových cyklech. Někdy je při neúspěchu lépe učící proces opakovat od začátku pro novou počáteční konfiguraci. Graf typického vývoje chyby v průběhu adaptace je načrtnut na obrázku 2.3. Na začátku učení se může chyba mírně zvýšit, později se střídá fáze jejího rychlého (přibližně exponenciálního) poklesu s dlouhodobou stagnací, kdy se chyba shora jen velmi pomalu blíží (pokles chyby se přibližně exponenciálně zmenšuje) k nějaké nenulové hodnotě. Obvykle se po dlouhodobém učení za určitých podmínek podaří nalézt globální minimum (chyba je nulová).



Obr. 2.3: Typický vývoj chyby v čase při učení pomocí backpropagation.

následujícím způsobem. Pokud při zpětném šíření, kdy postupujeme směrem od výstupu sítě k jejímu vstupu, procházíme neuronem $r \in j^{\rightarrow}$ (předpokládejme, že $\frac{\partial E_k}{\partial y_r}$ je vypočtena), pak k postupně počítaným parciálním derivacím $\frac{\partial E_k}{\partial y_i}$ u všech jeho vstupních neuronů $i \in r_{\leftarrow}$ (tedy speciálně také u neuronu j) přičteme $\frac{\partial E_k}{\partial y_r}\lambda_r y_r(1-y_r)w_{ri}$ (viz vzorec (2.19)). Ve chvíli, kdy budeme procházet neuronem j, jsou všechny neurony $r \in j^{\rightarrow}$ zpracovány, a proto hodnota $\frac{\partial E_k}{\partial y_j}$ již bude vypočtena. Také výpočet $\frac{\partial E_k}{\partial w_{ji}}$ v kroku 5c lze realizovat při tomto postupu a kroky 5b a 5c tak probíhají současně.

V uvedeném algoritmu dochází k adaptaci konfigurace vždy po jednom tréninkovém cyklu, tj. s přihlédnutím k celé tréninkové množině. Parciální derivace parciálních chybových funkcí se postupně sčítají (akumulují) v kroku 5d, proto při učení nezáleží na pořadí tréninkových vzorů. Této variantě se říká *akumulované učení*. Na druhou stranu je v tomto případě potřeba pro každý spoj v síti kromě vlastních vah pamatovat i příslušné hodnoty E'_{ji} akumulovaných parciálních derivací. Jinou méně paměťově náročnou možností je adaptovat konfiguraci pro každý tréninkový vzor. V tomto případě nemusíme mít předem k dispozici uzavřenou tréninkovou množinu, ale tréninkové vzory lze generovat (i bez opakování) postupně v pořadí podle zvolené tréninkové strategie. Po výpočtu gradientu parciální chybové funkce $\frac{\partial E_k}{\partial w_{ji}}(\mathbf{w}^{(t-1)})$ pro k-tý tréninkový vzor v kroku 5c se váhy místo kroku 5d hned adaptují s přihlédnutím k tomuto vzoru:

$$w_{ji}^{(t)} := w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)}, \quad \text{kde } \Delta w_{ji}^{(t)} := -\varepsilon \frac{\partial E_k}{\partial w_{ji}}(\mathbf{w}^{(t-1)}). \tag{2.20}$$

To nahrazuje příslušnou adaptaci vah v kroku 6 a 7. Protože gradient parciální chybové funkce se pro další tréninkový vzor počítá pomocí již aktualizovaných vah podle předchozího vzoru, nedochází k minimalizaci celkové chyby sítě vzhledem ke tréninkové množině přesně podle gradientní metody (2.11), (2.12) a podle našich zkušeností tento postup zpomaluje konvergenci.

Výpočet chyby sítě v kroku 8, který dává uživateli informaci o průběhu učení (např. zda algoritmus konverguje nebo se nachází v lokálním minimu apod.) a který je podkladem pro koncovou podmínku učení v kroku 9, je časově náročnou záležitostí (zahrnuje aktivní režim pro každý tréninkový vzor), proto není účelné jej opakovat v každém tréninkovém cyklu, ale stačí jej provést po několika těchto cyklech. Rychlost učení ε lze přizpůsobit vývoji chyby v čase, proto je vhodné, aby aktuální informace o chybě byla během adaptace zobrazena např. na obrazovku. Také je nutné, aby program umožňoval v adaptivním režimu interaktivně aktualizovat rychlost učení. Protože adaptivní režim je typicky dlouhodobou záležitostí, doporučuje se kvůli bezpečnosti jednou za čas uložit (archivovat) konfiguraci (pokud chyba mezitím poklesla) např. na disk.

lépe opisuje tvar chybové funkce $E(\mathbf{w})$, protože bere do úvahy předchozí gradient.

Konvergence gradientní metody závisí na vyladění rychlosti učení ε, popř. parametru momentu α. Proto se často navrhují heuristiky pro automatickou adaptaci těchto parametrů během učení [45, 67, 140, 278]. Například lze pro každou váhu w_{ji} v síti uvažovat zvlášť rychlost učení ε_{ji}. Hodnotu ε_{ji} můžeme např. volit řádově $1/|j_{\leftarrow}|$, kde $|j_{\leftarrow}|$ je počet vstupů neuronu j [226]. Nebo lineárně zvyšujeme rychlost učení $\varepsilon_{ji}^{(t)} = K\varepsilon_{ji}^{(t-1)}$ pro $K > 1$ (např. $K = 1.01$) v čase $t > 1$, pokud by se tím nezměnilo znaménko změny příslušné váhy, tj. $\Delta w_{ji}^{(t)} \Delta w_{ji}^{(t-1)} > 0$, jinak jej exponenciálně snížíme $\varepsilon_{ji}^{(t)} = \varepsilon_{ji}^{(t-1)}/2$.

Při učení neuronových sítí se v aplikacích často opomíjejí známé a propracované účinnější metody nelineární optimalizace [179]. Například *Newtonova metoda* v dostatečně blízkém okolí minima chybové funkce konverguje velmi rychle. Nicméně tato metoda vyžaduje výpočet druhých derivací, je výpočetně velmi náročná (v každém kroku iterace se počítá inverze matice) a numericky nestabilní. Vhodnějším kandidátem pro minimalizaci chybové funkce neuronové sítě je *metoda sdružených gradientů* [162, 187], která využívá jen první derivace.

Na závěr poznamenejme, že uvedený učící algoritmus backpropagation pro vícevrstvou neuronovou síť lze zobecnit pro cyklickou architekturu, pokud rekurentní síť v aktivním režimu konverguje ke stabilnímu stavu. Tato varianta algoritmu se nazývá *rekurentní backpropagation* [7, 6, 221, 222, 223, 237].

2.2.6 Volba topologie a generalizace

Velkým problémem modelu vícevrstvé neuronové sítě s učícím algoritmem backpropagation je (kromě minimalizace chybové funkce) volba vhodné topologie pro řešení konkrétního praktického problému. Zřídkakdy jsou podrobněji známy vztahy mezi vstupy a výstupy, aby se toho dalo využít při návrhu speciální architektury. Většinou se používá vícevrstvá topologie s jednou nebo dvěma skrytými vrstvami a očekává se, že učící algoritmus backpropagation zobecní příslušné vztahy z tréninkové množiny a zohlední je ve váhách jednotlivých spojů mezi neurony. I v tomto případě je však potřeba volit počty neuronů ve skrytých vrstvách. Ukazuje se, že tento problém organizační dynamiky úzce souvisí s adaptací a generalizací neuronové sítě.

Architektura vícevrstvé sítě, tj. počty skrytých neuronů, by měla odpovídat složitosti řešeného problému, tj. počtu tréninkových vzorů, jejich vstupů a výstupů a struktuře vztahů, které popisují. Je zřejmé, že malá síť nemůže řešit komplikovaný problém. Při učení pomocí algoritmu backpropagation se příliš malá síť obvykle zastaví v nějakém mělkém lokálním minimu a je potřeba topologii doplnit o další skryté neurony, aby adaptace měla

2.2.5 Varianty backpropagation

Problémy s minimalizací chybové funkce $E(\mathbf{w})$ v reálném čase se snaží řešit různé varianty základního modelu backpropagation. Např. problém s volbou hodnot strmostí λ_j aktivačních funkcí (2.8) u jednotlivých neuronů j lze řešit tak, že chyba sítě $E(\mathbf{w}, \boldsymbol{\lambda})$ bude kromě synaptických vah $\mathbf{w}$ také funkcí strmostí $\boldsymbol{\lambda}$. Konfigurace sítě je pak dána vektorem všech vah $\mathbf{w}$ a vektorem všech strmostí $\boldsymbol{\lambda}$. Při učení adaptujeme tuto konfiguraci (tj. včetně strmostí) tak, že chybu sítě minimalizujeme gradientní metodou v prostoru vah a strmostí. Tím zvyšujeme stupeň volnosti adaptace, kdy tvar aktivačních funkcí (tj. míra rozhodnosti jednotlivých neuronů) se může přizpůsobit tréninkové množině a snáze se nalezne globální minimum chybové funkce sítě. Na druhou stranu při zvýšení počtu adaptovaných parametrů roste počet numerických operací a učení se zpomaluje. Adaptace strmostí se technicky realizuje podobně jako u vah (2.11), (2.12). Na začátku adaptace v čase 0 jsou strmosti $\boldsymbol{\lambda}^{(0)}$ nastaveny náhodně, např. blízko jedné. V dalším tréninkovém cyklu v čase $t > 0$ se hodnoty strmostí $\boldsymbol{\lambda}^{(t)}$ v nové konfiguraci vypočtou:

$$\lambda_j^{(t)} = \lambda_j^{(t-1)} + \Delta\lambda_j^{(t)}, \quad \text{kde } \Delta\lambda_j^{(t)} = -\varepsilon' \frac{\partial E}{\partial \lambda_j}\left(\mathbf{w}^{(t-1)}, \boldsymbol{\lambda}^{(t-1)}\right) \tag{2.21}$$

a $0 < \varepsilon' < 1$ je rychlost učení strmostí, jejíž význam je stejný jako u synaptických vah. Výpočet parciální derivace $\frac{\partial E}{\partial \lambda_j}$ je opět analogický (viz vzorce (2.13), (2.14) a (2.16)):

$$\frac{\partial E}{\partial \lambda_j} = \sum_{k=1}^{p} \frac{\partial E_k}{\partial \lambda_j}, \quad \text{kde } \frac{\partial E_k}{\partial \lambda_j} = \frac{\partial E_k}{\partial y_j}\frac{\partial y_j}{\partial \lambda_j} = \frac{\partial E_k}{\partial y_j}\xi_j y_j(1 - y_j). \tag{2.22}$$

Parciální derivaci $\frac{\partial E_k}{\partial y_j}$ již umíme počítat pomocí strategie zpětného šíření (2.18), (2.19). V průběhu adaptace se obecně může hodnota parametru strmosti aktivační funkce u některého neuronu stát zápornou.

Popsaná varianta gradientní metody (2.11), (2.12) se vzhledem ke své jednoduchosti často používá, i když není příliš efektivní. Při malé rychlosti učení ε je konvergence této metody pomalá, avšak při větším ε metoda diverguje. Její jednoduchá celkem frekventovaná modifikace, která se snaží tento nedostatek odstranit, zohledňuje při výpočtu změny váhy ve směru gradientu chybové funkce navíc předchozí změnu vah, tzv. *moment* [226]:

$$\Delta w_{ji}^{(t)} = -\varepsilon \frac{\partial E}{\partial w_{ji}}\left(\mathbf{w}^{(t-1)}\right) + \alpha \Delta w_{ji}^{(t-1)}, \tag{2.23}$$

kde $0 < \alpha < 1$ je parametr momentu, který určuje míru vlivu předchozí změny (obvykle se volí např. $\alpha = 0.9$). Pomocí momentu gradientní metoda

několik neuronů a adaptivní režim se celý opakuje pro novou architekturu. Pro test kvality generalizace neuronové sítě se počítá chyba sítě vzhledem k tzv. *testovací množině*, což je část tréninkové množiny, která se záměrně nevyužila při adaptaci.

Důmyslnější metody podle potřeby automaticky během adaptace modifikují architekturu a vytváří tak kombinovaný *organizačně–adaptivní* režim sítě. Jsou možné následující dva odlišné přístupy. V tzv. *konstruktivních algoritmech* [62, 68, 188, 192] se začíná s malou topologií a v případě, že hodnotu chybové funkce již nelze dále snižovat, se přidávají nové neurony. Příklad takového algoritmu pro kaskádové architektury je v podkapitole 2.4. Naopak při tzv. *prořezávání* (pruning) [93, 110, 132, 162, 248] se vychází z dostatečné bohaté topologie a odstraňují se spoje (příp. odpovídající skryté neurony), které mají během učení v absolutní hodnotě malou váhu. To je možné díky velké robustnosti funkce sítě. Zanedbání malých vah lze např. zohlednit již v definici minimalizované chybové funkce sítě přidáním členu, který penalizuje všechny váhy v síti úměrně jejich velikosti:

$$E'(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2}\gamma \sum_{j,i} \frac{w_{ji}^2}{1 + w_{ji}^2}, \tag{2.24}$$

kde parametr $\gamma > 0$ mírou vlivu tohoto členu na chybu sítě a $E(\mathbf{w})$ je původní chybová funkce (2.9).

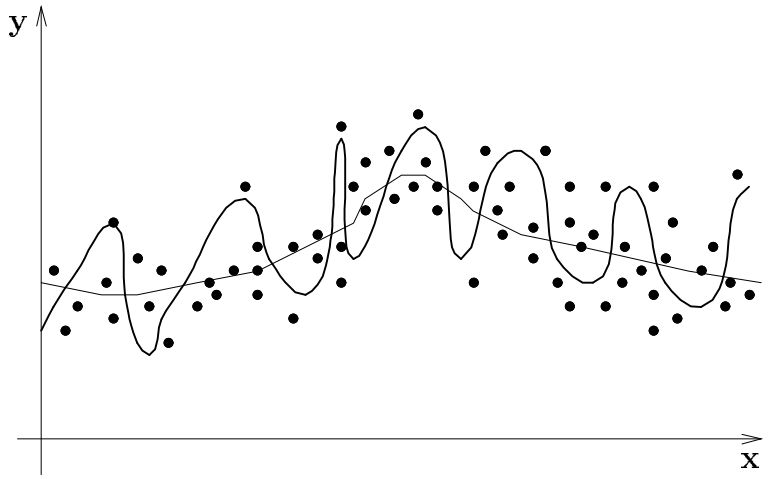
2.3 MADALINE

Další historicky důležitým modelem neuronové sítě je *MADALINE* (**M**ultiple ADALINE), který byl navržen Widrowem a Hoffem [288, 287] (viz podkapitolu 1.1). Základním prvkem v tomto modelu je neuron *ADALINE* (**ADA**ptive **LIN**ear **E**lement), který je velmi podobný perceptronu. Proto model MADALINE je formálně skoro identický se sítí perceptronů, kterou jsme se zabývali v podkapitole 2.1, i když původně vychází z jiných principů.

Organizační dynamika MADALINE (tj. architektura) a značení parametrů sítě je stejné jako u sítě perceptronů (viz obrázek 2.1), avšak místo perceptronu uvažujeme ADALINE. **Aktivní** dynamika se u tohoto modelu liší tím, že výstupy sítě mohou být obecně reálné a jednotlivé ADALINE realizují lineární funkci, tj. chybí nelineární aktivační funkce. Funkce MADALINE $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, která závisí na konfiguraci $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$, kde $\mathbf{w}_1 = (w_{10}, \ldots, w_{1n}), \ldots, \mathbf{w}_m = (w_{m0}, \ldots, w_{mn})$, je tedy dána afinní kombinací vstupů:

$$y_j = \sum_{i=0}^{n} w_{ji} x_i \quad j = 1, \ldots, m\,. \tag{2.25}$$

větší stupeň volnosti. Na druhou stranu bohatá architektura sice při učení mnohdy umožní nalézt globální minimum chybové funkce, i když s větším počtem vah roste výpočetní náročnost adaptace. Avšak nalezená konfigurace sítě obvykle příliš zohledňuje tréninkové vzory včetně jejich nepřesností a chyb a pro neučené vzory dává chybné výsledky, tj. špatně generalizuje. Tomuto přesnému zapamatování tréninkové množiny bez zobecnění zákonitostí v ní obsažených se říká *přeučení* (overfitting). Na obrázku 2.4 jsou graficky znázorněny dvě funkce sítě spolu s tréninkovými vzory (body), ze kterých byly naučeny. Silná čára představuje přeučenou síť, jejíž funkce se přizpůsobila nepřesným tréninkovým vzorům, zatímco tenká čára představuje funkci sítě, která „správně“ generalizovala zákonitosti v tréninkové množině. Zdá se tedy, že existuje optimální topologie, která je na jednu stranu dostatečně bohatá, aby byla schopna řešit daný problém, a na druhou stranu ne moc velká, aby správně zobecnila potřebné vztahy mezi vstupy a výstupy.



Obr. 2.4: Graf funkce přeučené sítě (tučně) a sítě se „správnou“ generalizací.

Existují teoretické výsledky ohledně horního odhadu počtu skrytých neuronů postačujících pro realizaci libovolné funkce z určité třídy (viz třetí část této knihy), avšak pro praktické potřeby jsou příliš nadhodnocené, a tedy nepoužitelné. V praxi se obvykle topologie volí heuristicky, např. v první skryté vrstvě o něco více neuronů než je vstupů a v druhé skryté vrstvě aritmetický průměr mezi počtem výstupů a neuronů v první skryté vrstvě. Po adaptaci se v případě velké chyby sítě příp. přidá, resp. při chudé generalizaci odebere

směru daném znaménkem y_j. Uvedená situace je načrtnuta na obrázku 2.5, kde nadrovina určená stejným výstupem je znázorněna přerušovanou čarou.

V **adaptivním** režimu je požadovaná funkce MADALINE zadána tréninkovou posloupností, kde reálné vstupy tréninkových vzorů $\mathbf{x}_k$ jsou generovány náhodně s daným rozdělením pravděpodobností a u každého je dán požadovaný reálný výstup $\mathbf{d}_k$:

$$(\mathbf{x}_k, \mathbf{d}_k)_{k=1,2,\ldots} \quad \text{kde} \quad \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \mathbb{R}^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \mathbb{R}^m \,. \end{array} \tag{2.29}$$

Chyba j-tého ADALINE vzhledem k tréninkové posloupnosti (2.29) v závislosti na části konfigurace $\mathbf{w}_j$ je definována:

$$E_j(\mathbf{w}_j) = \lim_{p\to\infty} \frac{\frac{1}{2}\sum_{k=1}^{p} \left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right)^2}{p} \quad j = 1, \ldots, m \,, \tag{2.30}$$

což je podle zákona velkých čísel [17] střední hodnota (značíme tučné $\mathbf{E}$) poloviny mocniny rozdílu skutečného stavu j-tého ADALINE a odpovídajícího požadovaného výstupu vzhledem k tréninkové posloupnosti:

$$E_j(\mathbf{w}_j) = \mathbf{E}\left[\frac{1}{2}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right)^2\right] \quad j = 1, \ldots, m \,. \tag{2.31}$$

Cílem adaptace je minimalizace chyby $E_j(\mathbf{w}_j)$ $(j = 1, \ldots, m)$, která ve váhovém prostoru díky tvaru funkce (2.25) určuje paraboloid. Za tímto účelem nejprve vypočteme gradient chybové funkce $E_j(\mathbf{w}_j)$ ze vztahu (2.30) záměnou limity a derivace, dále využijeme pravidlo o derivaci složené funkce a derivujeme (2.25):

$$\frac{\partial E_j}{\partial w_{ji}} = \lim_{p\to\infty} \frac{1}{p} \sum_{k=1}^{p} x_{ki}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right) \quad 0 = 1, \ldots, n \,, \tag{2.32}$$

který lze opět vyjádřit jako střední hodnotu:

$$\frac{\partial E_j}{\partial w_{ji}} = \mathbf{E}\left[x_{ki}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right)\right] \quad i = 0, \ldots, n \,. \tag{2.33}$$

Po dosazení (2.25) do (2.33) a s využitím pravidla o střední hodnotě lineární funkce dostáváme:

$$\frac{\partial E_j}{\partial w_{ji}} = -\mathbf{E}[d_{kj} x_{ki}] + \sum_{r=0}^{n} w_{jr} \mathbf{E}[x_{kr} x_{ki}] \quad i = 0, \ldots, n \,. \tag{2.34}$$

Také geometrický význam funkce j-tého ADALINE se nepatrně liší od perceptronu. Uvažujme vstup $\mathbf{x} = (x_1, \ldots, x_n)$, tj. bod $[x_1, \ldots, x_n]$ v n-rozměrném vstupním prostoru. Nadrovina s koeficienty $\mathbf{w}_j$ pro j-tý ADALINE daná rovnicí

$$w_{j0} + \sum_{i=1}^{n} w_{ji} x_i = 0 \tag{2.26}$$

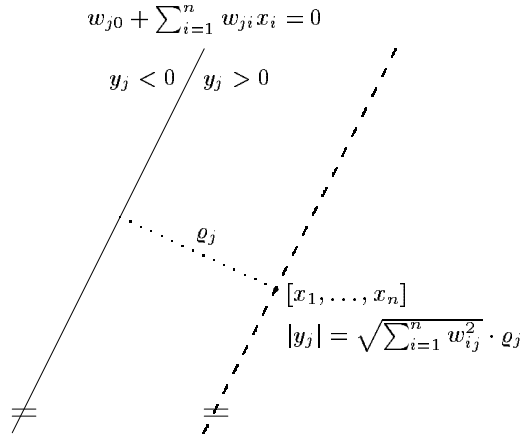
rozděluje tento prostor na dva poloprostory, ve kterých má hodnota výstupu y_j (2.25) odlišné znaménko. Pro body ležící na této nadrovině je hodnota výstupu nulová. Vzdálenost ϱ_j bodu $[x_1, \ldots, x_n]$ od nadroviny (2.26) je dána rovnicí:

$$\varrho_j = \frac{|w_{j0} + \sum_{i=1}^{n} w_{ji} x_i|}{\sqrt{\sum_{i=1}^{n} w_{ji}^2}} = \frac{|y_j|}{\sqrt{\sum_{i=1}^{n} w_{ji}^2}} \,. \tag{2.27}$$

Tedy absolutní hodnota $|y_j|$ výstupu j-tého ADALINE závisí lineárně na vzdálenosti bodu od nadroviny ve vstupním prostoru:

$$|y_j| = \sqrt{\sum_{i=1}^{n} w_{ji}^2} \cdot \varrho_j \,. \tag{2.28}$$

Body ze vstupního prostoru, které mají stejný výstup, leží na jedné nadrovině rovnoběžné s nadrovinou (2.26), která je od ní ve vzdálenosti ϱ_j ve



Obr. 2.5: Geometrická interpretace funkce ADALINE.

Jeden z možných postupů při minimalizaci chybové funkce $E_j(\mathbf{w}_j)$ je položit parciální derivace $\frac{\partial E_j}{\partial w_{ji}} = 0$ rovny nule. Pokud ve vzorcích (2.34) statisticky odhadneme střední hodnoty $\mathbf{E}[d_{kj}x_{ki}]$, $\mathbf{E}[x_{kr}x_{ki}]$, dostaneme soustavu lineárních rovnic:

$$\sum_{r=0}^{n} \mathbf{E}[x_{kr}x_{ki}]w_{jr} = -\mathbf{E}[d_{kj}x_{ki}] \quad i = 0,\ldots,n\,, \tag{2.35}$$

jejímž řešením je konfigurace $\mathbf{w}_j^\star$ pro j-tý ADALINE, která minimalizuje chybovou funkci $E_j(\mathbf{w}_j)$.

Obvykle se však k minimalizaci chyby $E_j(\mathbf{w}_j)$ používá podobně jako u algoritmu backpropagation (viz podkapitolu 2.2) gradientní metoda, u které v tomto případě díky tvaru chybové funkce (paraboloid) nejsou problémy s lokálními minimy. Na začátku adaptace v čase 0 jsou váhy konfigurace $\mathbf{w}^{(0)}$ nastaveny náhodně blízko nuly, např. $w_{ji}^{(0)} \in \langle -1, 1\rangle$ ($j = 1,\ldots,m$, $i = 0,\ldots,n$). V diskrétním čase adaptace $t > 0$ se konfigurace $\mathbf{w}^{(t)}$ mění podle gradientní metody:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)}\,, \quad \text{kde } \Delta w_{ji}^{(t)} = -\varepsilon \frac{\partial E_j}{\partial w_{ji}}(\mathbf{w}^{(t-1)}) \tag{2.36}$$

a $0 < \varepsilon < 1$ je rychlost učení.

V analogii s algoritmem backpropagation bychom měli akumulovat parciální derivace $\frac{\partial E_j}{\partial w_{ji}}$ v (2.36) (tj. podle (2.32) počítat limitu jejich průměrů) pro celou tréninkovou posloupnost. Avšak Widrow a Hoff navrhli pro MADALINE tzv. *Widrowovo* (*Widrow–Hoffovo*) *pravidlo*, někdy také nazývané *LMS* (**L**east–**M**ean–**S**quare) *pravidlo* učení, ve kterém dochází k adaptaci vah po každém tréninkovém vzoru (srovnejte s neakumulovanou implementací algoritmu backpropagation v odstavci 2.2.4). Model MADALINE má tedy diskrétní adaptivní dynamiku, tj. v každém časovém kroku učení $t = 1, 2, \ldots$ je síti předložen k-tý vzor ($k = t$) z tréninkové posloupnosti (2.29), podle kterého jsou adaptovány váhy. Podle Widrowova pravidla změna konfigurace $\mathbf{w}^{(t)}$ v čase $t > 0$ je dána následující rovnicí (srovnejte s (2.36) a (2.32)):

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \varepsilon x_{ki}\left(y_j(\mathbf{w}_j^{(t-1)}, \mathbf{x}_k) - d_{kj}\right) \qquad \begin{matrix} j = 1,\ldots,m \\ i = 0,\ldots,n\,. \end{matrix} \tag{2.37}$$

Je zajímavé, že adaptivní dynamika (2.37) u MADALINE je formálně totožná s adaptací perceptronu podle pravidla (2.5). Widrow a Hoff dokázali, že adaptivní proces podle (2.37) konverguje z libovolné počáteční konfigurace $\mathbf{w}^{(0)}$ ke konfiguraci $\mathbf{w}^\star$, která minimalizuje chybové funkce $E_j(\mathbf{w}_j)$ pro $j = 1,\ldots,m$.

Existují i různé varianty adaptivní dynamiky MADALINE. Například adaptaci vah můžeme provádět podle gradientní metody (2.36) tak, že limitní hodnotu parciální derivace $\frac{\partial E_j}{\partial w_{ji}}$ danou vzorcem (2.32) aproximujeme průměrem přes p tréninkových vzorů:

$$\frac{\partial E_j}{\partial w_{ji}} \doteq \frac{1}{p}\sum_{k=1}^{p} x_{ki}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right)\,. \tag{2.38}$$

Nebo ve Widrowově pravidle můžeme navíc uvažovat moment (viz vzorec (2.23)), tj. pro $t > 1$ dostaneme

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \varepsilon x_{ki}\left(y_j(\mathbf{w}_j^{(t-1)}, \mathbf{x}_k) - d_{kj}\right) + \alpha\left(w_{ji}^{(t-1)} - w_{ji}^{(t-2)}\right)\,, \tag{2.39}$$

kde $0 < \alpha < 1$ je parametr momentu.

2.4 Sítě s kaskádovou architekturou

V této podkapitole uvedeme popis sítí s kaskádovou architekturou, neboli *kaskádových sítí*, a jejich učícího algoritmu *kaskádové korelace (cascade-correlation)* navrženého Fahlmanem a Labierem na přelomu osmdesátých a devadesátých let [62]. Ačkoli, striktně bráno, nepatří tento model mezi „klasické“, uvádíme ho v této kapitole proto, že je jistým zobecněním perceptronových sítí.

Organizační dynamika (architektura) kaskádové sítě je znázorněna na obrázku 2.6. Jde o dopřednou síť s jednou vstupní vrstvou n vstupních jednotek, skrytou vrstvou obsahující h perceptronů, a lineární výstupní vrstvou o m jednotkách. Perceptronové jednotky jsou navíc pospojovány laterálními spoji, které jdou jedním směrem, řekněme zleva doprava ve smyslu očíslování skrytých jednotek. Každý perceptron dostává jako vstupní signály i výstupy ze všech předchozích jednotek ve skryté vrstvě. Takže i-tá jednotka ve skryté vrstvě má jako obvykle n vstupů spojených se vstupními jednotkami a dále $i-1$ vstupů napojených na výstupy jednotek j, kde $j = 1,\ldots,i-1$. Výstupní jednotky jsou spojeny se všemi jednotkami ze skryté i vstupní vrstvy (na obrázku 2.6 nejsou pro přehlednost znázorněny spoje mezi vstupními a výstupními jednotkami).

Funkce $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ realizovaná kaskádovou sítí v **aktivním** režimu se díky rekurentním vztahům mezi skrytými jednotkami zapisuje jedním vzorcem poměrně nesnadno, ale samotný způsob, jakým síť počítá odezvu na předložený vstup $\mathbf{x} \in \mathbb{R}^n$, je jednoduchý. Jednotky ve skryté vrstvě po-

stavec 2.2.5). Když už nedochází k výraznému zlepšení chyby sítě, přidáme do sítě další skrytou jednotku a opakujeme gradientní učení.

Přidání skryté jednotky spočívá v adaptivním nastavení jejích vstupních vah tak, aby jednotka mohla co nejvíce zlepšit chybu sítě. Nejprve si vytvoříme množinu kandidátů na novou jednotku, z nichž každý je svými vstupy připojen do sítě na všechny vstupní a příslušné skryté jednotky. Výstupy kandidátů zatím do sítě nezapojujeme. V této části učícího algoritmu adaptujeme vstupní váhy kandidátů tak, abychom maximalizovali korelaci mezi jejich výstupem a chybou sítě. To je opět realizováno gradientní metodou, a to tak dlouho, než se korelace přestane významně zvětšovat. Pak vybereme nejúspěšnějšího kandidáta, fixujeme jeho vstupní váhy a připojíme do sítě. Následně pokračujeme gradientním minimalizováním chyby, při kterém adaptujeme všechny váhy výstupních jednotek (tedy i od dosavadních skrytých neuronů, i od právě přidaného). Celý proces končí, dosáhneme-li požadovaného prahu chyby.

Motivací právě popsaného postupu je nastavit váhy každého přidávaného neuronu tak, aby neuron mohl co nejvíce snížit celkovou chybu sítě. Proto tedy maximalizujeme jeho korelaci s touto chybou.

Ukažme přesněji, jak obě učící fáze probíhají. Celkovou chybu sítě počítáme jako obvykle:

$$E = \frac{1}{2}\sum_{t=1}^{k}\sum_{j=1}^{m}(y_j^{(t)} - d_j^{(t)})^2\,, \tag{2.42}$$

kde $d_j^{(t)}$ (resp. $y_j^{(t)}$) je j-tá složka požadovaného (resp. aktuálního) výstupu sítě po předložení t-tého tréninkového vzoru. Gradient E spočteme takto:

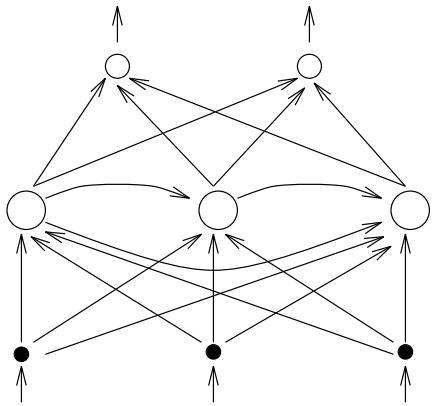
$$e_j^{(t)} = (y_j^{(t)} - d_j^{(t)})\sigma'(\xi^{(t)}) \tag{2.43}$$

$$\frac{\partial E}{\partial w_{ji}} = \sum_{t=1}^{k} e_j^{(t)} I_j^{(t)}\,, \tag{2.44}$$

kde $\sigma'(\xi^{(t)})$ je hodnota derivace aktivační funkce výstupní jednotky po předložení vzoru $\mathbf{x}^{(t)}$, $I_j^{(t)}$ je hodnota vstupní (nebo skryté) jednotky j a w_{ji} je váha spojující vstup i s výstupem j.

Při učení kandidátů maximalizujeme korelaci mezi výstupem kandidáta y a chybou sítě. Korelaci počítáme přes všechny tréninkové vzory t následujícím způsobem:

$$C = \sum_{j=1}^{m}\left|\sum_{t=1}^{k}(y^{(t)} - \bar{y})(e_j^{(t)} - \bar{e}_j)\right|\,, \tag{2.45}$$



Obr. 2.6: Kaskádová síť (laterální spoje jsou označeny oblými šipkami)

stupně dle svého pořadí i spočtou svou výstupní hodnotu z_i podle obvyklé přechodové funkce známé z perceptronů:

$$z_i = \sigma\left(\sum_{j=0}^{n} w_{ij}x_j + \sum_{j=1}^{i-1} w_{i(n+j)}z_j\right) \quad \text{pro } i = 1,\ldots,h\,, \tag{2.40}$$

kde w_{ij} jsou váhy, σ je aktivační funkce.

Pro každý perceptron i jsou tedy váhy příslušející výstupům od předchozích jednotek zařazeny formálně na konec váhového vektoru, odpovídají vždy pozicím $n+1,\ldots,n+i-1$. V další fázi výpočtu potom jednotky ve výstupní vrstvě spočtou lineární kombinace svých vstupů a aplikují přechodovou funkci:

$$y_i = \sigma\left(\sum_{j=0}^{n+h} v_{ij}z_j\right) \quad \text{pro } i = 1,\ldots,m\,. \tag{2.41}$$

Zabývejme se nyní **adaptivní** dynamikou sítě. Algoritmus kaskádové korelace je příkladem konstruktivního či inkrementálního učícího algoritmu. Učení začíná s minimální možnou konfigurací sítě, což v našem případě znamená bez skrytých jednotek. Učení probíhá na dvou časových škálách: v každém velkém cyklu učíme aktuální konfiguraci sítě pomocí lineárního gradientního sestupu. Autoři v [62, 115] používají algoritmus *quickprop*, který je podrobně popsán v [61]. Jde o jednu ze sofistikovanějších variant algoritmu zpětného šíření, která vychází z Newtonovy gradientní metody (srovnej od-

kde $\bar{y}$ (resp. $\bar{e}_j$) jsou průměry hodnot $y^{(t)}$ (resp. $e_j^{(t)}$) počítané přes všechny tréninkové vzory t. Analogicky jako u (2.43) spočítáme gradient C:

$$\delta^{(t)} = \sum_{j=1}^{m} s_j (e_j^{(t)} - \bar{e}^{(t)}) \sigma'(\xi^{(t)}) \tag{2.46}$$

$$\frac{\partial C}{\partial w_j} = \sum_{t=1}^{k} \delta^{(t)} I_j^{(t)} , \tag{2.47}$$

kde s_j je znaménko korelace mezi hodnotou výstupu kandidáta a chybou sítě na výstupu j. Předchozí vzorce platí pro výpočet gradientu u jednoho kandidáta. V průběhu korelační fáze učení inicializujeme množinu kandidátů různými náhodnými vahami a každého kandidáta pak nezávisle adaptujeme.

Pro konkrétní úpravu vah na základě vypočtených gradientů používají autoři metodu quickprop, kterou nyní stručně popíšeme. Uvažujme úpravu váhy w v jedné ze zmíněných fází učení. Podle toho, zda minimalizujeme chybu sítě, nebo maximalizujeme korelaci výstupu jednotky, přiřaďme do hodnoty $S^{(t)}$ buď $\frac{\partial E}{\partial w_{ji}}$ nebo $\frac{\partial C}{\partial w_j}$. Úprava váhy $\Delta w^{(t)}$ se pak řídí následujícím pravidlem:

$$\Delta w^{(t)} = \begin{cases} \varepsilon S^{(t)} & \text{pro} \quad \Delta w^{(t-1)} = 0 \\ \frac{S^{(t)}}{S^{(t-1)} - S^{(t)}} \Delta w^{(t-1)} & \text{je-li} \quad \Delta w^{(t-1)} \neq 0 \\ & \quad \text{a} \ \frac{S^{(t)}}{S^{(t-1)} - S^{(t)}} < \mu \\ \mu \Delta w^{(t-1)} & \text{jinak} , \end{cases} \tag{2.48}$$

kde ε je parametr, který řídí nastartování úprav vah pomocí lineárních kroků a μ řídí maximální velikost kroku v porovnání s předchozím krokem. Detailnější popis a diskusi nastavení hodnot parametrů lze najít v [61].

Hlavní rysy algoritmu kaskádové korelace jsou následující:

- Díky konstruktivnímu učení není třeba dopředu určovat rozměry sítě (v tomto případě počet skrytých jednotek). Učící proces sám vybuduje síť, která daný problém řeší s dostatečnou přesností (není ovšem zaručeno, že tato konfigurace je minimální).

- Inkrementální mechanismus učení umožňuje snadné pozdější doučování sítě novým vzorům.

- Kaskádová korelace je typicky rychlejší než zpětné šíření chyby, protože se v každém okamžiku adaptuje pouze jedna vrstva vah. Obě optimalizační části jsou tedy lineární a rychlé. (Při minimalizaci chyby

měníme váhy u výstupní vrstvy, zatímco algoritmus maximalizující korelaci operuje s vahami přidávaných skrytých neuronů.) Další výhodou může být i to, že algoritmus funguje „hladovým“ (greedy) způsobem: každá přidávaná jednotka se snaží ubrat maximální část chyby, jaké je schopna. Ve standardním algoritmu zpětného šíření se všechny jednotky adaptují naráz a pracují také na různých úkolech, což může mít vliv na rychlost algoritmu.

- Konečně, část, ve které kaskádová korelace tráví nejvíce času—maximalizace korelace s chybou sítě u skupiny přidávaných jednotek, je velmi vhodná k paralelizaci. Jelikož každý z kandidátů provádí izolovaně svůj gradientní výstup, mohou být umístěni na různých procesorech. Jediná komunikace mezi nimi proběhne až na závěr této učící fáze, kdy je třeba vybrat nejlepšího z nich.

Kapitola 3

Asociativní neuronové sítě

3.1 Lineární asociativní síť

Lineární asociativní síť, navržená Andersonem [13] a dále jím a Kohonenem rozpracovaná [14, 15, 16, 151, 152], je příkladem modelu neuronové sítě, který se využívá jako *asociativní paměť*. Na rozdíl od paměti klasických počítačů, kdy klíčem k vyhledání položky v paměti je adresa, u asociativní paměti vybavení příslušné informace probíhá na základě její částečné znalosti (asociace). Např. v databázových aplikacích je znalost některých položek záznamu postačující k vyhledání celého záznamu. Podobně u člověka např. černobílá fotografie přítele pomůže vybavit barvu jeho vlasů či očí, popř. jeho jméno. Budeme rozlišovat v zásadě dva typy asociativní paměti: *autoasociativní* a *heteroasociativní*. U autoasociativní paměti půjde o upřesnění či zúplnění vstupní informace, což v našem motivačním příkladě s černobílou fotografií znamená vybavení odpovídajícího barevného obrazu. Naproti tomu u heteroasociativní paměti dochází k vybavení určité sdružené informace na základě vstupní asociace, což v uvedeném příkladě může odpovídat určení jména osoby na fotografii.

Organizační i **aktivní** dynamika lineární asociativní sítě je téměř identická jako u modelu MADALINE, který jsme popsali v podkapitole 2.3. Jediný rozdíl spočívá v tom, že lineární asociativní síť v aktivním režimu místo afinních kombinací počítá jen lineární kombinace vstupů, tj. chybí formální jednotkový vstup a odpovídající biasy jsou nulové. Formálně lze tedy funkci lineární asociativní sítě $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ zapsat (srovnejte s (2.25)):

$$y_j = \sum_{i=1}^{n} w_{ji} x_i \quad j = 1, \ldots, m\,. \tag{3.1}$$

V geometrické interpretaci to znamená, že příslušné nadroviny (viz obrázek 2.5) odpovídající výstupním neuronům sítě prochází počátkem.

Pro potřeby dalšího výkladu vyjádříme ještě aktivní dynamiku (3.1) formálně v maticovém zápisu. Vstup sítě, resp. výstup, chápeme jako sloupcový vektor $\mathbf{x} = (x_1, \ldots, x_n)$ typu $n \times 1$, resp. $\mathbf{y} = (y_1, \ldots, y_m)$ typu $m \times 1$, a konfigurace sítě je dána váhovou maticí $\mathbf{W}$ typu $m \times n$, jejíž řádky $\mathbf{w}_j = (w_{j1}, \ldots, w_{jn})$ odpovídají synaptickým váhám vstupů (výstupních) neuronů j ($j = 1, \ldots, m$):

$$\mathbf{W} = \begin{pmatrix} w_{11} & \ldots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \ldots & w_{mn} \end{pmatrix}. \tag{3.2}$$

Při tomto označení můžeme aktivní dynamiku lineární asociativní sítě (3.1) formálně zapsat jako maticový součin:

$$\mathbf{y} = \mathbf{W}\mathbf{x}\,. \tag{3.3}$$

V **adaptivním** režimu lineární asociativní sítě je opět požadovaná funkce zadána tréninkovou množinou:

$$\mathcal{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \mathbb{R}^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \mathbb{R}^m \end{array} \quad k = 1, \ldots, p \right\}, \tag{3.4}$$

pro níž v případě autoasociativní paměti platí, že požadovaný výstup odpovídá vstupu sítě, tj. $m = n$ a $\mathbf{x}_k = \mathbf{d}_k$ pro $k = 1, \ldots, p$. V následujících odstavcích 3.1.1 a 3.1.2 popíšeme dvě možné adaptivní dynamiky lineární asociativní sítě.

3.1.1 Adaptace podle Hebbova zákona

Adaptivní dynamiku lineární asociativní sítě popíšeme v tomto odstavci obecně pro případ heteroasociativní paměti, jejímž speciálním případem je autoasociativní paměť. Jeden z možných způsobů adaptace této sítě je motivován neurofyziologickým *Hebbovým zákonem*, který tvrdí, že změna synaptické váhy spoje mezi dvěma neurony je úměrná jejich souhlasné aktivitě, tj. součinu jejich stavů. Donald Hebb tímto způsobem vysvětloval vznik podmíněných reflexů [98] (viz podkapitolu 1.1), kdy současná aktivita (popř. pasivita) prvního neuronu odpovídající podmínce (příčině) a druhého neuronu vyvolávající reflex posiluje synaptickou vazbu spoje směrem od prvního k druhému neuronu. Obráceně opačná aktivita těchto neuronů tuto vazbu zeslabuje.

Hebbův zákon lze tedy formálně shrnout do následující adaptivní dynamiky lineární asociativní sítě. Na začátku adaptace v čase 0 jsou všechny

normě jsou srovnatelné. Za tohoto předpokladu má lineární asociativní síť schopnost tzv. *reprodukce*, tj. předložíme-li této síti vstup $\mathbf{x}_r$ $(1 \leq r \leq p)$ z tréninkové množiny, pak na něj odpoví příslušným požadovaným výstupem $\mathbf{d}_r$. To lze ukázat tak, že do aktivní dynamiky (3.3) dosadíme (3.8) za $\mathbf{W}$, dále využijeme asociativitu a distributivitu maticového násobení (skalárního součinu) a předpoklad ortonormality:

$$\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \left(\sum_{k=1}^{p} \mathbf{d}_k \mathbf{x}_k^{\mathsf{T}}\right) \mathbf{x}_r = \sum_{k=1}^{p} \mathbf{d}_k \left(\mathbf{x}_k^{\mathsf{T}} \mathbf{x}_r\right) = \mathbf{d}_r \,. \tag{3.11}$$

Schopnost reprodukce (3.11) lze považovat za nutnou podmínku asociativní paměti. Avšak lineární asociativní síť by také pro vstup $\mathbf{x}_r + \boldsymbol{\delta}$, který je blízko vstupu $\mathbf{x}_r$ r-tého tréninkového vzoru (tj. norma $\|\boldsymbol{\delta}\| = \sqrt{\sum_{i=1}^{n} \delta_i^2} = \delta$ je dostatečně malá) měla odpovídat požadovaným výstupem $\mathbf{d}_r$. Odpovídající chybu lze vyjádřit jako normu rozdílu skutečného výstupu pro vstup $\mathbf{x}_r + \boldsymbol{\delta}$ a požadovaného výstupu d_r:

$$E_r(\boldsymbol{\delta}) = \left\|\mathbf{y}(\mathbf{x}_r + \boldsymbol{\delta}) - \mathbf{d}_r\right\| = \left\|\mathbf{W}\mathbf{x}_r + \mathbf{W}\boldsymbol{\delta} - \mathbf{d}_r\right\| = \left\|\mathbf{W}\boldsymbol{\delta}\right\| \,. \tag{3.12}$$

Pokud budeme navíc předpokládat, že požadované výstupy $\mathbf{d}_r$ tréninkových vzorů jsou normované, tj. $\mathbf{d}_r^{\mathsf{T}}\mathbf{d}_r = 1$ $(r = 1, \ldots, p)$, což např. pro autoasociativní paměť (kde $\mathbf{d}_r = \mathbf{x}_r$) platí, můžeme chybu $E_r(\boldsymbol{\delta})$ z (3.12) pomocí trojúhelníkové ($\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$) a Cauchy-Schwarzovy nerovnosti ($|\mathbf{x}\mathbf{y}| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$) shora odhadnout:

$$E_r(\boldsymbol{\delta}) = \left\|\mathbf{W}\boldsymbol{\delta}\right\| \leq \sum_{k=1}^{p} \left\|\mathbf{d}_k \mathbf{x}_k^{\mathsf{T}} \boldsymbol{\delta}\right\| \leq \sum_{k=1}^{p} \|\mathbf{d}_k\| \cdot \|\mathbf{x}_k\| \cdot \|\boldsymbol{\delta}\| = p\delta \leq n\delta \,. \tag{3.13}$$

Tedy chyba $E_r(\boldsymbol{\delta}) \to 0$ pro $\delta \to 0$ a pro vstupy blízké ke vzorovým vstupům lineární asociativní síť odpovídá přibližně požadovaným výstupem, a lze ji tedy použít jako asociativní paměť.

3.1.2 Pseudohebbovská adaptace

Pro jednoduchost další možnou adaptivní dynamiku lineární asociativní sítě popíšeme nejprve pro případ autoasociativní paměti a na závěr ji zobecníme pro heteroasociativní paměť. Reprodukce sítě v odstavci 3.1.1, která byla adaptována podle Hebbova zákona, vyžadovala dodatečný předpoklad na ortonormalitu vstupů tréninkových vzorů. Proto byla navržena modifikace Hebbova zákona, tzv. *pseudohebbovská adaptace* [281, 158], která tento předpoklad pomocí matematických úprav zeslabuje, ale již neodpovídá neurofyziologické skutečnosti.

váhy konfigurace nulové, tj. $w_{ji}^{(0)} = 0$ $(j = 1, \ldots, m,\ i = 1, \ldots, n)$. V diskrétním čase adaptace $t = 1, \ldots, p$, kdy je síti předložen k-tý tréninkový vzor $(k = t)$, se váhy adaptují podle Hebbova zákona:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + d_{kj} x_{ki} \qquad \begin{matrix} j = 1, \ldots, m \\ i = 1, \ldots, n \,. \end{matrix} \tag{3.5}$$

Vzhledem k tomu, že adaptivní režim v tomto případě skončí po p krocích, kdy jsou již všechny tréninkové vzory naučeny, lze výslednou konfiguraci formálně vyjádřit jako konečný součet:

$$w_{ji} = \sum_{k=1}^{p} d_{kj} x_{ki} \qquad \begin{matrix} j = 1, \ldots, m \\ i = 1, \ldots, n \,. \end{matrix} \tag{3.6}$$

Adaptivní dynamiku (3.5) lineární asociativní sítě je možno také přehledně vyjádřit pomocí maticového zápisu:

$$\mathbf{W}^{(0)} = \mathbf{0}, \quad \mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \mathbf{d}_k \mathbf{x}_k^{\mathsf{T}}, \quad k = 1, \ldots, p \,, \tag{3.7}$$

kde $^{\mathsf{T}}$ značí transpozici matice, $\mathbf{0}$ je nulová matice (tj. její prvky jsou nulové) a váhová matice $\mathbf{W}^{(k)}$ určuje konfiguraci sítě v čase adaptace $t = k$. Výslednou konfiguraci lze formálně zapsat jako součin matic (srovnejte s (3.6)):

$$\mathbf{W} = \mathbf{W}^{(p)} = \sum_{k=1}^{p} \mathbf{d}_k \mathbf{x}_k^{\mathsf{T}} = \mathbf{D}\mathbf{X}^{\mathsf{T}} \,, \tag{3.8}$$

kde sloupce matice $\mathbf{X}$ typu $n \times p$, resp. matice $\mathbf{D}$ typu $m \times p$, jsou vstupy $\mathbf{x}_k$ $(k = 1, \ldots, p)$, resp. požadované výstupy $\mathbf{d}_k$, tréninkových vzorů (3.4), tj.

$$\mathbf{X} = \begin{pmatrix} x_{11} & \ldots & x_{p1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \ldots & x_{pn} \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} d_{11} & \ldots & d_{p1} \\ \vdots & \ddots & \vdots \\ d_{1m} & \ldots & d_{pm} \end{pmatrix} . \tag{3.9}$$

Speciálně aktivní dynamiku v případě autoasociativní paměti, kde $\mathbf{D} = \mathbf{X}$, lze zapsat:

$$\mathbf{W} = \mathbf{X}\mathbf{X}^{\mathsf{T}} \,. \tag{3.10}$$

Dále budeme předpokládat, že množina vstupních vektorů $\{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ u tréninkových vzorů (3.4) je ortonormální (nutně $p \leq n$). To znamená, že tyto vektory jsou na sebe vzájemně kolmé (ortogonální), tj. $\mathbf{x}_r^{\mathsf{T}}\mathbf{x}_s = 0$ pro $r \neq s$ $(1 \leq r, s \leq p)$, a mají jednotkovou velikost (jsou normované), tj. $\mathbf{x}_r^{\mathsf{T}}\mathbf{x}_r = 1$ $(r = 1, \ldots, p)$. Tento předpoklad lze interpretovat tak, že vstupy u jednotlivých vzorů se kvůli ortogonalitě dostatečně liší a díky jednotkové

prostoru V_{k-1}, který je určen bází $\{\mathbf{x}_1,\ldots,\mathbf{x}_{k-1}\}$, resp. ortogonální bází $\{\mathbf{z}_1,\ldots,\mathbf{z}_{k-1}\}$. Připomeňme, že vektor $\mathbf{x}_k$ neleží v prostoru V_{k-1}, protože vektory $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$ jsou lineárně nezávislé. Chceme tedy ověřit, že vektor $\mathbf{z}_k = \mathbf{x}_k - \mathbf{W}^{(k-1)}\mathbf{x}_k$ je kolmý na bazické vektory $\mathbf{z}_r$ $(r = 1,\ldots,k-1)$. To znamená dokázat, že $\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_r = 0$ pro $r = 1,\ldots,k-1$. Po transpozici $\mathbf{z}_k^{\mathsf{T}} = \mathbf{x}_k^{\mathsf{T}} - \mathbf{x}_k^{\mathsf{T}}(\mathbf{W}^{(k-1)})^{\mathsf{T}}$ dosadíme za $(\mathbf{W}^{(k-1)})^{\mathsf{T}} = \sum_{s=1}^{k-1}\frac{\mathbf{z}_s\mathbf{z}_s^{\mathsf{T}}}{\mathbf{z}_s^{\mathsf{T}}\mathbf{z}_s}$ dle (3.15). Dále využijeme toho, že $\{\mathbf{z}_1,\ldots,\mathbf{z}_{k-1}\}$ jsou dle indukčního předpokladu ortogonální a dostaneme:

$$\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_r = \mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r - \sum_{s=1}^{k-1}\frac{\mathbf{x}_k^{\mathsf{T}}\mathbf{z}_s\mathbf{z}_s^{\mathsf{T}}\mathbf{z}_r}{\mathbf{z}_s^{\mathsf{T}}\mathbf{z}_s} = \mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r - \frac{\mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r\mathbf{z}_r^{\mathsf{T}}\mathbf{z}_r}{\mathbf{z}_r^{\mathsf{T}}\mathbf{z}_r} = 0\,. \tag{3.18}$$

Z uvedeného vyplývá, že pokud vektor $\mathbf{x}$ leží v prostoru V_p, pak splývá se svojí ortogonální projekcí $\mathbf{W}\mathbf{x} = \mathbf{x}$. Speciálně pro vstupní bazické vektory $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$ prostoru V_p dostáváme, že výstup sítě $\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \mathbf{x}_r$ pro $r = 1,\ldots,p$. To znamená, že lineární autoasociativní síť vzniklá pseudohebbovskou adaptací (3.14), (3.15) má schopnost reprodukce. Navíc pro vstup $\mathbf{x}_r + \boldsymbol{\delta}$ v blízkosti r-tého tréninkového vzoru $\mathbf{x}_r$ je výstup sítě $\mathbf{y}(\mathbf{x}_r + \boldsymbol{\delta}) = \mathbf{W}(\mathbf{x}_r + \boldsymbol{\delta})$ ortogonální projekcí do prostoru V_p, tj. jeho nejlepší aproximací v prostoru V_p. Chybu $E_r(\boldsymbol{\delta}) = ||\mathbf{y}(\mathbf{x}_r + \boldsymbol{\delta}) - x_r|| = ||\mathbf{W}\mathbf{x}_r + \mathbf{W}\boldsymbol{\delta} - x_r|| = ||\mathbf{W}\boldsymbol{\delta}||$ lze po dosazení (3.16) za $\mathbf{W}$ podobně jako v (3.13) shora odhadnout:

$$E_r(\boldsymbol{\delta}) = ||\mathbf{W}\boldsymbol{\delta}|| \le \sum_{k=1}^{p}\frac{||\mathbf{z}_k\mathbf{z}_k^{\mathsf{T}}\boldsymbol{\delta}||}{||\mathbf{z}_k||^2} \le p\delta \le n\delta\,, \tag{3.19}$$

kde $\delta = ||\delta||$, a tedy $E_r(\boldsymbol{\delta}) \to 0$ pro $\delta \to 0$.

Na závěr ještě zobecníme pseudohebbovskou adaptivní dynamiku (3.16) lineární asociativní sítě pro heteroasociativní paměť:

$$\mathbf{W} = \mathbf{D}\mathbf{X}^{+} = \mathbf{D}(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\,, \tag{3.20}$$

kde $\mathbf{D}$, $\mathbf{X}$ jsou matice (3.9). Uvedený maticový zápis adaptivní dynamiky (3.20) není vhodný pro distribuovaný adaptivní režim heteroasociativní sítě. Analogie rekurzivního zápisu (3.14), (3.15) v případě heteroasociativní paměti je založena na Grevilleově větě [86, 154]:

$$\mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \frac{\left(\mathbf{d}_k - \mathbf{W}^{(k-1)}\mathbf{x}_k\right)\mathbf{z}_k^{\mathsf{T}}}{\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_k}\,, \tag{3.21}$$

kde $\mathbf{z}_k$ je stejný sloupcový vektor typu $n \times 1$ jako v případě autoasociativní paměti, tj. v jeho definici (3.14) se matice $\mathbf{W}^{(k-1)}$ počítá podle (3.15) (nikoliv dle (3.21)). Pomocí pseudoinverze (3.17) lze $\mathbf{z}_k$ vyjádřit

$$\mathbf{z}_k = \mathbf{x}_k - \mathbf{X}^{(k-1)}\left(\mathbf{X}^{(k-1)}\right)^{+}\mathbf{x}_k\,, \tag{3.22}$$

Předpokládejme, že množina vstupních vektorů (tj. i požadovaných výstupů pro autoasociativní paměť) $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$ tréninkových vzorů (3.4) je lineárně nezávislá (nutně $p \le n$), a tedy tvoří bázi vektorového prostoru V_p, který je podprostorem $\mathbb{R}^n$. Kvůli ortogonalitě vzorů potřebné k jejich reprodukci vytvoříme během pseudohebbovské adaptace pomocí Gram-Schmidtova ortogonalizačního procesu ortogonální bázi $\{\mathbf{z}_1,\ldots,\mathbf{z}_p\}$ vektorového prostoru V_p. Adaptivní pseudohebbovská dynamika lineární asociativní sítě pak probíhá následujícím způsobem. Na začátku adaptace v čase 0 je matice vah nulová, tj. $\mathbf{W}^{(0)} = \mathbf{0}$. V diskrétním čase adaptace $t = 1,\ldots,p$, kdy je síti předložen k-tý tréninkový vzor $(k = t)$, je nejprve určen sloupcový vektor $\mathbf{z}_k$ typu $n \times 1$:

$$\mathbf{z}_k = \mathbf{x}_k - \mathbf{W}^{(k-1)}\mathbf{x}_k\,, \tag{3.14}$$

pomocí kterého je pak adaptována váhová matice:

$$\mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \frac{\mathbf{z}_k\mathbf{z}_k^{\mathsf{T}}}{\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_k}\,. \tag{3.15}$$

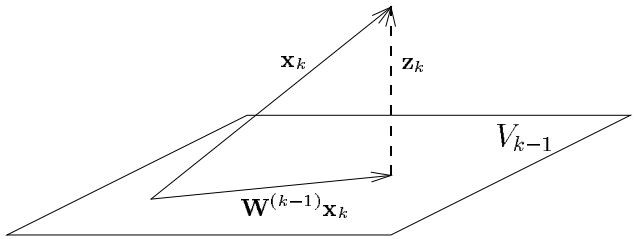
Výslednou váhovou matici lze ze vztahu (3.15) opět formálně vyjádřit jako konečný součet matic, který se dá také převést na součin matic (srovnejte s (3.10)):

$$\mathbf{W} = \mathbf{W}^{(p)} = \sum_{k=1}^{p}\frac{\mathbf{z}_k\mathbf{z}_k^{\mathsf{T}}}{\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_k} = \mathbf{X}\mathbf{X}^{+}\,, \tag{3.16}$$

kde $\mathbf{X}$ je matice (3.9) a

$$\mathbf{X}^{+} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}} \tag{3.17}$$

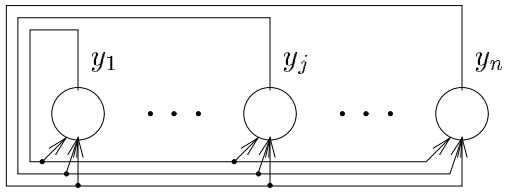
je její *pseudoinverze*.



Obr. 3.1: Geometrická interpretace pseudohebbovské strategie.

Geometrický význam pseudohebbovské adaptace (3.14), (3.15) v k-tém kroku je znázorněn na obrázku 3.1. Matematickou indukcí dle k lze ukázat, že vektor $\mathbf{W}^{(k-1)}\mathbf{x}_k$ je ortogonální projekcí vektoru $\mathbf{x}_k$ do vektorového

není spojen sám ze sebou, tj. odpovídající váhy $w_{jj} = 0$ $(j = 1, \ldots, n)$ jsou nulové.



Obr. 3.2: Topologie Hopfieldovy sítě.

U Hopfieldovy sítě nejprve popíšeme **adaptivní** dynamiku, která se řídí Hebbovým zákonem (viz odstavec 3.1.1). Požadovaná funkce sítě je opět specifikována tréninkovou množinou p vzorů (3.25), z nichž každý je zadán vektorem n bipolárních stavů vstupních, resp. výstupních neuronů, které v případě autoasociativní paměti splývají.

$$\mathcal{T} = \{\mathbf{x}_k \mid \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \{-1, 1\}^n,\ k = 1, \ldots, p\}\ . \tag{3.25}$$

Adaptivní dynamika podle Hebbova zákona probíhá v p diskrétních krocích, ve kterých jsou síti postupně předkládány tréninkové vzory, podle nichž se adaptují synaptické váhy. Výslednou konfiguraci sítě lze zapsat následujícím způsobem (srovnejte s (3.6)):

$$w_{ji} = \sum_{k=1}^{p} x_{kj} x_{ki} \quad 1 \leq j \neq i \leq n\,. \tag{3.26}$$

Nejprve si všimneme, že $w_{ji} = w_{ij}$ $(1 \leq i, j \leq n)$, protože postavení neuronů i, j ve vzorci (3.26) je symetrické. Proto se také někdy Hopfieldově síti říká *symetrická síť*, v níž dva opačně orientované spoje mezi dvěma neurony lze chápat jako jeden neorientovaný spoj. Adaptaci Hopfieldovy sítě podle Hebbova zákona (3.26) můžeme také interpretovat jako hlasování vzorů o vzájemných vazbách neuronů. Váha $w_{ji} = w_{ij}$ totiž představuje rozdíl mezi počtem souhlasných stavů $x_{kj} = x_{ki}$ (tj. $x_{kj}x_{ki} = 1$) neuronů i a j v tréninkových vzorech, které mezi nimi posilují vazbu, a počtem rozdílných stavů $x_{kj} \neq x_{ki}$ (tj. $x_{kj}x_{ki} = -1$), které tuto vazbu zeslabují. Výsledek hlasování se projeví ve znaménku váhy w_{ij}, která je kladná, když v tréninkové množině (3.25) převáží počet souhlasných stavů neuronů i, j nad počtem rozdílných stavů, a je záporná v opačném případě. Absolutní hodnota této váhy určuje, o kolik hlasů ta která strana vyhrála. Je zřejmé, že tréninkové vzory nejsou

kde $\mathbf{X}^{(k-1)}$ je matice typu $n \times (k-1)$, jejíž sloupce tvoří vstupní vektory $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$ prvních $k-1$ tréninkových vzorů (3.4):

$$\mathbf{X}^{(k-1)} = \begin{pmatrix} x_{11} & \ldots & x_{k-1,1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \ldots & x_{k-1,n} \end{pmatrix}\ . \tag{3.23}$$

Výpočet $\mathbf{z}_k$ pomocí (3.14) vyžaduje navíc autoasociativní model sítě, proto se někdy pseudohebbovská adaptivní dynamika (3.20) pro případ heteroasociativní paměti, tj. výpočet $\mathbf{W} = \mathbf{D}\mathbf{X}^+$, aproximuje Widrowovým pravidlem (2.37).

Pseudohebbovská adaptivní dynamika (3.20) zaručuje schopnost heteroasociativní sítě reprodukovat tréninkové vzory (3.4). Při ověřování této vlastnosti budeme pomocí $[\mathbf{X}]_r$ značit r-tý sloupec matice $\mathbf{X}$:

$$\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \mathbf{D}\mathbf{X}^+[\mathbf{X}]_r = \mathbf{D}[(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}(\mathbf{X}^\mathsf{T}\mathbf{X})]_r = \mathbf{d}_r\,. \tag{3.24}$$

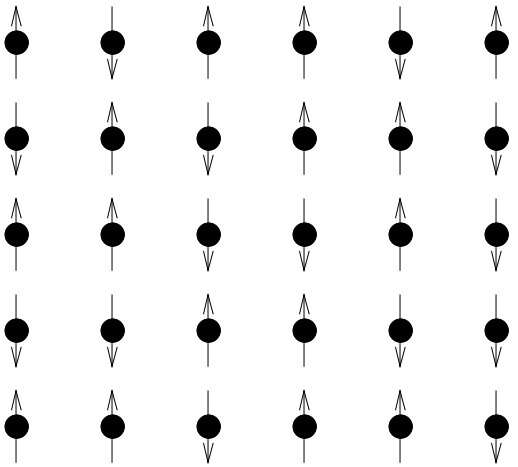
3.2 Hopfieldova síť

Dalším důležitým modelem neuronové sítě, který je předmětem velkého zájmu badatelů, je *Hopfieldova síť*. Tento model byl navržen již McCullochem a Pittsem [189] a později analyzován Amarim [9], W. A. Littlem a G. L. Shawem [176]. Avšak teprve díky Hopfieldovi [122], který při analýze stability této sítě využil průhlednou analogii s fyzikální teorií magnetických materiálů, se tento model sítě stal všeobecně známým (viz podkapitolu 1.1), a proto také nese jeho jméno. Hopfieldova síť se používá jako autoasociativní paměť (viz podkapitolu 3.1). V současnosti existuje mnoho teoretických výsledků a variant uvedeného modelu, které se snaží zlepšit jeho vlastnosti. V této podkapitole se omezíme jen na popis a diskusi základního modelu Hopfieldovy sítě.

3.2.1 Základní model

Organizační dynamika Hopfieldovy sítě specifikuje na začátku pevnou úplnou topologii cyklické sítě s n neurony, kde každý neuron v síti je spojen s každým, tj. má všechny neurony za své vstupy. Dále všechny neurony v síti jsou zároveň vstupní i výstupní. Architektura Hopfieldovy sítě je znázorněna na obrázku 3.2. Označme $\xi_1, \ldots, \xi_n \in \mathbb{Z}$ celočíselné vnitřní potenciály a $y_1, \ldots, y_n \in \{-1, 1\}$ bipolární stavy neuronů. Spoj v síti od neuronu i $(i = 1, \ldots, n)$ směrem k neuronu j $(j = 1, \ldots, n)$ je ohodnocen celočíselnou synaptickou váhou $w_{ji} \in \mathbb{Z}$. V základním modelu neuvažujeme biasy neuronů, tj. všechny prahy neuronů jsou nulové, a podobně žádný neuron

síť. Magnetické materiály lze v těchto modelech chápat jako soubor atomických magnetů, tzv. *spinů*, odpovídajících v Hopfieldově síti neuronům, které jsou uspořádány do pravidelné mřížky reprezentující krystalickou strukturu těchto materiálů, jak je znázorněno na obrázku 3.3. Uvažujeme nejjednodušší případ atomů, kde každý spin může mít dvě možné magnetické orientace, což v Hopfieldově síti modelujeme neuronovými stavy 1 a -1. Fyzikální model je dále specifikován interakcí a dynamikou spinů. Každý spin je ovlivněn okolním magnetickým polem, které je možné rozdělit na externí, které v Hopfieldově síti odpovídá vstupu, a na interní pole vytvořené ostatními spiny. Příspěvek každého atomu do okolního vnitřního pole je úměrný jeho vlastnímu spinu. Součet těchto příspěvků určuje magnetické pole ovlivňující daný spin, což odpovídá aktivní dynamice Hopfieldovy sítě (3.27). Synaptické váhy v rovnici (3.27) modelují vzájemné interaktivní síly spinů, které jsou ve fyzikálním modelu symetrické a mohou být různě velké, kladné nebo záporné v závislosti na makroskopických vlastnostech materiálu. Fyzikální model odpovídá asynchronní Hopfieldově síti.



Obr. 3.3: Model magnetického materiálu.

K lepšímu pochopení aktivní dynamiky Hopfieldovy sítě (3.27), (3.28) byla Hopfieldem v analogii s fyzikálními ději definována tzv. *energetická funkce* $E(\mathbf{y})$ sítě, která každému stavu sítě $\mathbf{y} \in \{-1,1\}^n$ přiřazuje jeho potenciální *energii* podle kvadratické formy:

v Hopfieldově síti uloženy přímo, ale jsou reprezentovány pomocí vztahů mezi stavy neuronů.

Aktivní dynamiku Hopfieldovy sítě popíšeme pro případ sekvenčního synchronního výpočtu. Na začátku aktivního režimu v čase 0 jsou stavy neuronů nastaveny na vstup sítě $\mathbf{x} = (x_1, \ldots, x_n)$, tj. $y_i^{(0)} = x_i$ $(i = 1, \ldots, n)$. V diskrétním čase $t > 0$ výpočtu je aktualizován neuron j (ostatní svůj stav nemění), který je vybrán např. systematicky tak, že $t = \tau n + j$, kde τ je tzv. *makroskopický čas*, tj. počet period, ve kterých jsou aktualizovány všechny neurony. Nejprve je vypočten celočíselný vnitřní potenciál neuronu j:

$$\xi_j^{(t-1)} = \sum_{i=1}^{n} w_{ji} y_i^{(t-1)}, \tag{3.27}$$

jehož znaménko určuje jeho nový bipolární stav:

$$y_j^{(t)} = \begin{cases} 1 & \xi_j^{(t-1)} > 0 \\ y_j^{(t-1)} & \xi_j^{(t-1)} = 0 \\ -1 & \xi_j^{(t-1)} < 0\,. \end{cases} \tag{3.28}$$

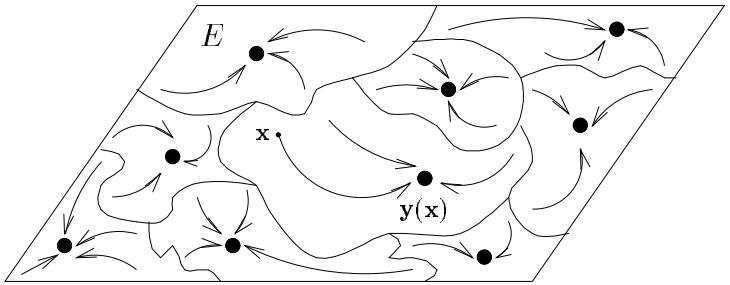
Stanovení výstupu neuronu z vnitřního potenciálu (3.27) podle (3.28) lze chápat jako aplikaci aktivační funkce, která je bipolární verzí ostré nelinearity (1.7). Výpočet (tj. aktivní režim) Hopfieldovy sítě skončí v čase $t^\star$, když se síť nachází v tzv. *stabilním stavu*, tj. stavy neuronů se již nemění: $y_j^{(t^\star+n)} = y_j^{(t^\star)}$ $(j = 1, \ldots, n)$. V tomto případě stavy (výstupních) neuronů určují výstup sítě $\mathbf{y} = (y_1, \ldots, y_n)$, kde $y_j = y_j^{(t^\star)}$ $(j = 1, \ldots, n)$. Dá se ukázat (viz větu 8.14), že za předpokladu symetrie vah sekvenční výpočet Hopfieldovy sítě podle aktivní dynamiky (3.27), (3.28) skončí pro každý vstup. Tedy Hopfieldova síť v aktivním režimu počítá ve vstupním prostoru funkci $\mathbf{y}(\mathbf{w}) : \{-1,1\}^n \longrightarrow \{-1,1\}^n$. Tato funkce kromě konfigurace $\mathbf{w}$ závisí na pořadí, ve kterém jsou aktualizovány neurony.

Také je možné uvažovat paralelní výpočet Hopfieldovy sítě, kdy v jednom časovém kroku aktivního režimu je aktualizován stav podle (3.27), (3.28) u více neuronů současně. V tomto případě však výpočet nemusí obecně skončit a síť po nějakém čase případně začne střídat dva stavy (viz větu 8.16). Podobně v asynchronním modelu Hopfieldovy sítě místo systematické (sekvenční nebo paralelní) aktualizace neuronů dochází k náhodné a nezávislé aktualizaci u jednotlivých neuronů.

3.2.2 Energetická funkce

Jak už bylo v úvodu této podkapitoly řečeno, Hopfieldova síť má svoji přirozenou fyzikální analogii. Některé jednoduché modely magnetických materiálů ve statistické fyzice (např. spinová skla) připomínají Hopfieldovu

by výstup sítě měl odpovídat tomuto vzoru. Z hlediska energie by každý tréninkový vzor z (3.25) měl být lokálním minimem energetické funkce, tj. stabilním stavem sítě. V jeho blízkém okolí, v tzv. *oblasti atrakce*, se nachází všechny vstupy blízké tomuto vzoru. Ty představují počáteční stavy sítě, ze kterých se při minimalizaci energetické funkce v aktivním režimu síť dostane do příslušného minima, tj. stabilního stavu odpovídajícího uvažovanému tréninkovému vzoru. Geometricky se tedy energetická plocha rozpadá na oblasti atrakce lokálních minim a příslušná funkce Hopfieldovy sítě přiřadí v aktivním režimu ke každému vstupu náležejícímu do oblasti atrakce nějakého lokálního minima právě toto minimum. Energetická plocha je graficky znázorněna na obrázku 3.4, kde jsou vyznačena lokální minima jednotlivých oblastí atrakce.



Obr. 3.4: Energetická plocha.

Při učení Hopfieldovy sítě podle Hebbova zákona (3.26) však navíc na energetické ploše vznikají samovolně lokální minima, tzv. *nepravé vzory* (*fantomy*), které neodpovídají žádným tréninkovým vzorům. Výstup sítě pro vstup dostatečně blízký takovému fantomu neodpovídá žádnému vzoru, a tudíž nedává smysl. Existují varianty adaptivní dynamiky Hopfieldovy sítě, při nichž se takto vzniklé fantomy dodatečně odučují. Např. Hopfieldovu síť s konfigurací w_{ji} $(1 \leq j,i \leq n)$ lze odučit fantom $\mathbf{x}' = (x'_1,\ldots,x'_n) \in \{-1,1\}^n$ modifikací Hebbova zákona [124] a získat tak váhy w'_{ji}:

$$w'_{ji} = w_{ji} - x'_j x'_i \quad 1 \leq j \neq i \leq n\,. \tag{3.31}$$

Zbavování sítě fantomů lze v neurofyziologické analogii připodobnit k léčení neuróz. Existují teorie, které tvrdí, že zlepšování paměti odučováním fantomů probíhá u člověka ve snu.

$$E(\mathbf{y}) = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji}y_j y_i\,. \tag{3.29}$$

Z definice energetické funkce $E(\mathbf{y})$ vyplývá, že stavy sítě s nízkou energií (tj. odpovídající sčítance $w_{ji}y_jy_i$ součtu (3.29) jsou dostatečné velké, např. kladné), mají větší stabilitu, protože znaménko váhy w_{ji} spoje mezi neurony j a i odpovídá vzájemnému vztahu mezi jejich stavy y_j a y_i. Víme totiž, že znaménko v případě kladné (resp. záporné) váhy vynucuje souhlasné (resp. rozdílné) stavy těchto neuronů. Obráceně stavy s velkou energií jsou nestabilní ze stejného důvodu. Díky této fyzikální analogii se někdy energetická funkce s opačným znaménkem nazývá *stabilita* (viz (8.10)), resp. *harmonie*.

Pro větší názornost budeme v následujícím výkladu předpokládat, že energetická funkce je spojitá, i když v uvedeném základním modelu Hopfieldovy sítě uvažujeme pouze diskrétní hodnoty. V aktivním režimu počáteční stav $\mathbf{y}^{(0)}$ Hopfieldovy sítě, tj. její vstup, dodá síti energii $E(\mathbf{y}^{(0)})$, která se v průběhu výpočtu sítě ztrácí, tj. energetická funkce $E(\mathbf{y}^{(t)}) \geq E(\mathbf{y}^{(t+1)})$ klesá až do chvíle, kdy se výpočet v čase $t^\star$ zastaví ve stabilním stavu $\mathbf{y}^{(t^\star)}$, který odpovídá lokálnímu minimu energetické funkce $E(\mathbf{y}^{(t^\star)})$. Je zajímavé, že pokles energetické funkce (3.29) podle aktivní dynamiky (3.27), (3.28) připomíná minimalizaci „chyby“ $E(\mathbf{y})$ gradientní metodou. Nový stav $y_j^{(t)}$ vybraného neuronu j v čase $t > 0$ aktivního režimu odpovídá opačnému znaménku gradientu v bodě $y_j^{(t-1)}$:

$$\frac{\partial E}{\partial y_j}(y_j^{(t-1)}) = -\sum_{i=1}^{n} w_{ji}y_i^{(t-1)}\,. \tag{3.30}$$

Hopfieldova síť má ve srovnání s vícevrstvou sítí adaptovanou učícím algoritmem backpropagation (viz podkapitolu 2.2) opačný charakter aktivní a adaptivní dynamiky, proto jsme při jejich výkladu postupovali v obráceném pořadí. Zatímco adaptace Hopfieldovy sítě podle Hebbova zákona (3.26) je jednorázovou záležitostí, jejíž trvání závisí jen na počtu tréninkových vzorů, učící algoritmus backpropagation (viz odstavec 2.2.4) realizuje iterativní proces minimalizující chybu sítě gradientní metodou bez záruky konvergence. Na druhou stranu délka aktivní fáze vícevrstvé sítě je dána pouze počtem vrstev (viz odstavec 2.2.1), zatímco aktivní režim Hopfieldovy sítě podle (3.27), (3.28) představuje iterativní proces minimalizující energii sítě diskrétní variantou gradientní metody v obecném případě (např. při paralelním výpočtu) s nejistou konvergencí.

Cílem adaptace Hopfieldovy sítě podle Hebbova zákona (3.26) je nalezení takové konfigurace, aby funkce sítě v aktivním režimu realizovala autoasociativní paměť. To znamená, že pro vstup blízký nějakému tréninkovému vzoru

3.2.3 Kapacita Hopfieldovy paměti

Nejprve se budeme zabývat schopností reprodukce Hopfieldovy sítě, která je pro autoasociativní paměť nutnou podmínkou (viz odstavec 3.1.1). V našem případě to znamená, že tréninkové vzory jsou lokálními minimy energetické funkce, resp. stabilními stavy sítě. Schopnost reprodukce závisí na poměru p/n počtu tréninkových vzorů p ku počtu neuronů n, kterým je dána tzv. *kapacita* Hopfieldovy autoasociativní paměti. Za předpokladu, že stavy neuronů tréninkových vzorů jsou vybrány náhodně se stejnou pravděpodobností, lze pro dostatečně velký počet neuronů n a tréninkových vzorů p vypočítat pravděpodobnost P, že stav daného neuronu v tréninkovém vzoru bude stabilní [109]:

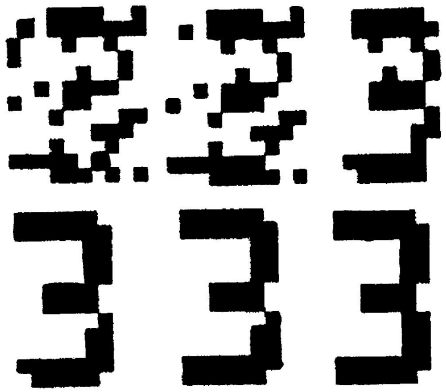
$$P = \frac{1}{2} - \frac{1}{\sqrt{\pi}} \int_0^{\sqrt{n/2p}} e^{-x^2} dx \,. \tag{3.32}$$

Například pro $p = 0.185n$ (tj. pro kapacitu paměti 0.185) lze očekávat, že počet nestabilních stavů neuronů v tréninkových vzorech nepřevýší 1%. Tento výsledek však neříká nic o tom, zda síť při následném výpočtu, kdy se změní příslušný nestabilní stav neuronu, dospěje ke stabilnímu stavu blízkému k uvažovanému tréninkovému vzoru. Podobná analýza [190, 284] pro reprodukci většiny, resp. všech celých vzorů (na rozdíl od reprodukce jen jednotlivých stavů neuronů v tréninkových vzorech), ukazuje, že maximální počet vzorů, které lze uložit do Hopfieldovy autoasociativní paměti, je úměrný $n/\log n$.

Avšak u autoasociativní paměti nejde jen o reprodukci, tj. zapamatování všech tréninkových vzorů, ale též o vybavení těchto vzorů na základě jejich částečné (nepřesné) znalosti. Podrobnější analýza [11, 12, 74, 220] ukazuje, že pro počet tréninkových vzorů $p \leq 0.138n$ (tj. pro kapacitu paměti nejvýše 0.138) tréninkové vzory odpovídají lokálním minimům energetické funkce Hopfieldovy sítě adaptované podle Hebbova zákona (3.26), tj. Hopfieldovu síť lze principiálně použít jako autoasociativní paměť. Naopak pro $p > 0.138n$ lokální minima příslušející k tréninkovým vzorům zanikají. Pro $p < 0.05n$ (tj. pro kapacitu paměti menší než 0.05) tréninkové vzory odpovídají globálním minimům energetické funkce, která jsou hlubší než lokální minima příslušející fantomům. To znamená, že pro Hopfieldovu autoasociativní paměť s dobrými vlastnosti je k zapamatovaní např. 10 tréninkových vzorů potřeba 200 neuronů, což v úplné topologii cyklické sítě odpovídá řádově 40000 spojů ohodnocených (celočíselnými) synaptickými váhami. I když se v praxi ukazuje, že uvedené teoretické odhady jsou poněkud nadhodnocené, přesto základní model Hopfieldovy autoasociativní paměti má díky své malé kapacitě spíše teoretický význam. V literatuře existuje mnoho modifikací tohoto modelu, které se snaží uvedený nedostatek odstranit.

3.2.4 Příklad aplikace Hopfieldovy sítě

Uvedené obecné principy budeme ještě ilustrovat na příkladě využití Hopfieldovy sítě pro redukci šumu rozpoznávaných znaků, v tomto případě číslic. Obraz cifry je rozložen na matici 12×10 černobílých obrazových bodů, které odpovídají 120 neuronům Hopfieldovy sítě tak, že jejich stavy 1, -1 reprezentují po řadě černou a bílou barvu (srovnejte s motivačním příkladem školáka v odstavci 1.3.1). Dále byly vytvořeny vzorové obrazy osmi uvažovaných číslic, které posloužily jako tréninkové vzory při adaptaci Hopfieldovy sítě podle Hebbova zákona (3.26). Naučené síti byl pak předložen vstup odpovídající obrazu cifry, který vznikl 25% šumem vzorového obrazu trojky. Na obrázku 3.5 můžeme sledovat průběh aktivního režimu Hopfieldovy sítě pro tento vstup monitorovaný v makroskopických časových krocích. Hopfieldova síť postupně odstraňuje z obrazu číslice šum, tj. pracuje jako autoasociativní paměť, když vybavuje původní vzorový obraz trojky, který odpovídá stabilnímu stavu sítě.



Obr. 3.5: Příklad aplikace Hopfieldovy sítě.

3.3 Spojitá Hopfieldova síť

V této podkapitole popíšeme spojitou variantu analogové Hopfieldovy sítě [46, 123] a její aplikaci při heuristickém řešení problému obchodního cestujícího. Na rozdíl od analogových modelů neuronových sítí, které jsme doposud uvažovali (např. vícevrstvá neuronová síť s učícím algoritmem backpropagation v podkapitole 2.2), kde reálný stav neuronu je spojitou funkcí

$j = 1, \ldots, n$. Po dosazení $\frac{dy_j}{dt}(t^\star) = 0$ do soustavy (3.33) dostáváme:

$$y_j = \sigma(\xi_j) = \sigma\left(\sum_{i=0}^{n} w_{ji} y_i\right) \quad j = 1, \ldots, n\,. \tag{3.36}$$

Rovnice (3.36) připomínají stabilní stav diskrétní verze Hopfieldovy sítě v aktivním režimu (3.27), (3.28).

Aktivní dynamiku (3.33) spojité Hopfieldovy sítě můžeme také zapsat velmi podobnou soustavou diferenciálních rovnic, jejímž řešením je (místo $\mathbf{y}(t)$) vývoj vnitřních potenciálů $\xi_j(t)$ $(j = 1, \ldots, n)$ v čase:

$$\tau_j \frac{d\xi_j}{dt} = -\xi_j + \sum_{i=0}^{n} w_{ji} y_i = -\xi_j + \sum_{i=0}^{n} w_{ji}\sigma(\xi_i) \quad j = 1, \ldots, n\,. \tag{3.37}$$

Soustava (3.37) má stejné stabilní řešení (3.36) jako soustava (3.33).

Podobně jako u diskrétní verze (3.29) můžeme definovat energii spojité Hopfieldovy sítě, která je závislá na stavu sítě, a proto se také spojitě vyvíjí v čase:

$$E(t) = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji} y_j y_i - \sum_{j=1}^{n} w_{j0} y_j + \sum_{j=1}^{n}\int_0^{y_j} \sigma^{-1}(y)dy\,. \tag{3.38}$$

Pro aktivační funkci (3.34) (podobně pro (3.35)) můžeme integrály v rovnici (3.38) explicitně vyjádřit:

$$\int_0^{y_j} \sigma^{-1}(y)dy = \frac{1}{\lambda}\ln y_j^{y_j}\,(1-y_j)^{1-y_j} \quad j = 1, \ldots, n\,. \tag{3.39}$$

V následujícím ukážeme, že energie $E(t)$ během aktivního režimu (3.33) (resp. (3.37)) neroste, tj. $\frac{dE}{dt} \leq 0$:

$$\begin{aligned}\frac{dE}{dt} \;=\; & -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji}\frac{dy_j}{dt} y_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji} y_j \frac{dy_i}{dt} - \\ & -\sum_{j=1}^{n} w_{j0}\frac{dy_j}{dt} + \sum_{j=1}^{n}\sigma^{-1}(y_j)\frac{dy_j}{dt}\,.\end{aligned} \tag{3.40}$$

S využitím toho, že váhy jsou symetrické, tj. $w_{ji} = w_{ij}$ $(1 \leq i, j \leq n)$, a (3.36), (3.37) dostáváme:

$$\frac{dE}{dt} = -\sum_{j=1}^{n}\frac{dy_j}{dt}\left(\sum_{i=0}^{n} w_{ji} y_i - \xi_j\right) = -\sum_{j=1}^{n}\tau_j \frac{dy_j}{dt}\frac{d\xi_j}{dt}\,. \tag{3.41}$$

vnitřního potenciálu, bude tato síť příkladem modelu, u kterého je vývoj reálného stavu v aktivním režimu navíc spojitou funkcí času. Aktivní (příp. adaptivní) dynamika je v takových případech obvykle zadána diferenciální rovnicí, jejíž řešení nelze explicitně vyjádřit, proto tyto modely, pokud nepracujeme s jejich diskrétní verzí (v našem případě s Hopfieldovou sítí z podkapitoly 3.2), nejsou vhodné pro simulaci na klasických počítačích. Na druhou stranu jsou výhodné pro analogovou hardwarovou implementaci pomocí elektrických obvodů.

3.3.1 Spojitá aktivní dynamika

Předpokládejme model Hopfieldovy sítě s **organizační**, popř. **adaptivní** dynamikou popsanou v odstavci 3.2.1. Navíc předpokládáme, že váhy sítě mohou být reálná čísla, a w_{j0} $(1 \leq j \leq n)$ představuje obecně nenulový reálný bias (práh s opačným znaménkem) j-tého neuronu odpovídající formálnímu jednotkovému vstupu $y_0 = 1$. Připomeňme, že $w_{jj} = 0$ $(j = 1, \ldots, n)$. Pro spojitou Hopfieldovu síť uvažujeme následující **aktivní** dynamiku. Na začátku aktivního režimu v čase 0 jsou stavy neuronů nastaveny na reálný vstup sítě $\mathbf{x} = (x_1, \ldots, x_n)$, tj. $y_i(0) = x_i$ $(i = 1, \ldots, n)$. Vývoj reálného stavu sítě $\mathbf{y}(t)$ v aktivní fázi je spojitou funkcí času $t > 0$ danou soustavou diferenciálních rovnic (v následujících vzorcích kvůli přehlednosti vynecháváme parametr času t):

$$\tau_j \frac{dy_j}{dt} = -y_j + \sigma(\xi_j) = -y_j + \sigma\left(\sum_{i=0}^{n} w_{ji} y_i\right) \quad j = 1, \ldots, n\,, \tag{3.33}$$

kde $\tau_j > 0$ $(j = 1, \ldots, n)$ jsou vhodné časové konstanty, $\xi_j(t)$ je reálný vnitřní potenciál neuronu j, který také závisí na čase t, a σ je spojitá aktivační funkce, např. standardní sigmoida (1.9) s oborem hodnot $(0, 1)$:

$$\sigma(\xi) = \frac{1}{1 + e^{-\lambda\xi}} \tag{3.34}$$

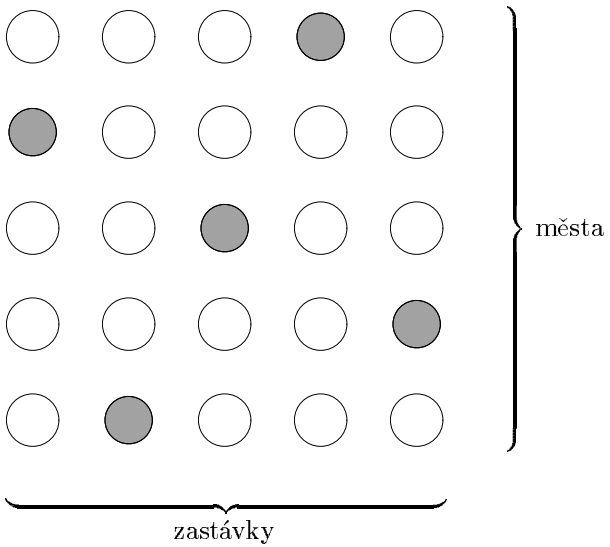
nebo její bipolární tvar — hyperbolický tangens (1.10) s oborem hodnot $(-1, 1)$:

$$\sigma(\xi) = \mathrm{tgh}\left(\frac{1}{2}\lambda\xi\right) = \frac{1 - e^{-\lambda\xi}}{1 + e^{-\lambda\xi}}\,, \tag{3.35}$$

kde $\lambda > 0$ je parametr strmosti (srovnejte s (2.7)). Pro $\lambda \to \infty$ dostaneme diskrétní výstupy neuronů -1 nebo 1 podobně jako u diskrétní Hopfieldovy sítě (3.28).

Výpočet spojité Hopfieldovy sítě končí v čase $t^\star$, kdy se síť nachází ve stabilním stavu $\mathbf{y}(t^\star)$, jehož změna v čase je nulová, tj. $\frac{dy_j}{dt}(t^\star) = 0$ pro

(3.39) jsou při těchto hodnotách nulové. Dále se proto omezíme již jen na zbylé členy v definici energetické funkce (3.38). Stav $y_{ru} \doteq 1$ (řekneme, že příslušný neuron je aktivní) interpretujeme tak, že r-té město se nachází na okružní cestě u-té v pořadí, a $y_{ru} \doteq 0$ (neuron je pasivní), pokud tomu tak není.



Obr. 3.6: Neuronová reprezentace problému obchodního cestujícího.

Je zřejmé, že v uvedené neuronové reprezentaci problému obchodního cestujícího ne všechny stavy sítě odpovídají jeho přípustnému řešení, tj. představují skutečně okružní cesty. Přípustnost řešení je potřeba zohlednit v minimalizované účelové funkci. Nejprve požadujeme, aby obchodní cestující navštívil každé město nejvýše jednou, tj. aby v každém řádku byl aktivní nejvýše jeden neuron, jak je naznačeno na obrázku 3.6. Tomu odpovídá minimalizace následujícího výrazu:

$$\begin{aligned} E_A &= \frac{A}{2} \sum_{r=1}^{N} \sum_{u=1}^{N} \sum_{\substack{v=1 \\ v \neq u}}^{N} y_{ru} y_{rv} = \\ &= -\frac{1}{2} \sum_{\substack{j=1 \\ j=(r,u)}}^{n} \sum_{\substack{i=1 \\ i=(s,v)}}^{n} -A\delta_{rs}(1-\delta_{uv}) y_j y_i \,, \end{aligned} \tag{3.43}$$

Pomocí (3.36) upravíme derivaci $\frac{dE}{dt}$ na tvar, ze kterého je zřejmé, že je nekladná:

$$\frac{dE}{dt} = -\sum_{j=1}^{n} \tau_j \sigma'(\xi_j) \left(\frac{d\xi_j}{dt}\right)^2 \leq 0 \,, \tag{3.42}$$

protože $\tau_j > 0$ $(j = 1, \ldots, n)$ a aktivační funkce $\sigma(\xi)$ je pro $\lambda > 0$ rostoucí, tj. $\sigma'(\xi_j) > 0$. Tedy energie $E(t)$ během aktivního režimu (3.33) klesá, tj. $\frac{dE}{dt} < 0$, až do chvíle, kdy se síť vzhledem k omezenosti energetické funkce (3.38) (reálné stavy neuronů jsou z intervalu $(0,1)$, resp. $(-1,1)$, a příslušné integrály (3.39) jsou omezené) dostane do jejího lokálního minima, kde $\frac{dE}{dt} = 0$, které podle (3.42) odpovídá stabilnímu stavu sítě $\frac{d\xi_j}{dt} = \frac{dy_j}{dt} = 0$. Tím jsme ukázali, že spojitá Hopfieldova síť vždy skončí svůj výpočet ve stabilním stavu.

Spojitou Hopfieldovu síť adaptovanou pomocí Hebbova zákona lze v analogii s její diskrétní verzí použít jako autoasociativní paměť. V následujícím výkladu se budeme zabývat ještě jiným nestandardním využitím tohoto modelu.

3.3.2 Problém obchodního cestujícího

Ukázali jsme, že během aktivního režimu spojité Hopfieldovy sítě se minimalizuje energie E ve stavovém prostoru. Toho se dá využít při heuristickém řešení optimalizačního problému, jehož minimalizovanou účelovou funkci a případná omezení lze vyjádřit ve tvaru „kvadratické formy“ (3.38). Porovnáním příslušné účelové funkce s energetickou funkcí se vyextrahují synaptické váhy spojité Hopfieldovy sítě („adaptivní“ režim). V aktivním režimu pak hledáme optimální přípustné řešení daného problému. Tento postup budeme ilustrovat na známém problému obchodního cestujícího [125].

Je dáno $N > 2$ měst a funkce $d : \{1, \ldots, N\}^2 \longrightarrow \mathbb{R}$, která pro každou dvojici těchto měst $1 \leq r, s \leq N$ určuje vzdálenost $d(r,s)$ z města r do města s, tj. $d(r,r) = 0$ a $d(r,s) = d(s,r)$ $(1 \leq r, s \leq N)$. Problémem obchodního cestujícího je určit pořadí měst nejkratší okružní cesty, při které každé město navštíví právě jednou a skončí ve městě, ve kterém začal. Uvedený problém reprezentujeme pomocí spojité Hopfieldovy sítě s $n = N \times N$ neurony, jejichž stavy y_{ru} indexujeme číslem města r $(1 \leq r \leq N)$ a jeho možným pořadím u $(1 \leq u \leq N)$ při okružní cestě obchodního cestujícího. Maticové uspořádání neuronů v topologii sítě je naznačeno na obrázku 3.6, kde jednotlivé řádky odpovídají městům a sloupce určují pořadí zastávek obchodního cestujícího v těchto městech. Jako aktivační funkci σ uvažujeme standardní sigmoidu (3.34), tj. stavy neuronů $y_{ru} \in (0,1)$ $(1 \leq r, u \leq N)$. Při minimalizaci energetické funkce E v aktivním režimu vynucuje člen s integrály v rovnici (3.38) binární stavy (tj. $y_{ru} \doteq 1$ nebo $y_{ru} \doteq 0$), protože příslušné integrály

$$= -\frac{1}{2} \sum_{\substack{j=1 \\ j=(r,u)}}^{n} \sum_{\substack{i=1 \\ i=(s,v)}}^{n} -Dd(r,s)\,(\delta_{u,v-1} + \delta_{u,v+1})\, y_j y_i \,,$$

kde $D > 0$ je míra vlivu E_D při minimalizaci a cílem úpravy v (3.46) je převést E_D opět na tvar porovnatelný s prvním členem v (3.38). Kvůli jednoduchosti zápisu (3.46) indexy $u-1$ pro $u=1$, resp. $u+1$ pro $u=N$, interpretujeme jako $u=N$, resp. $u=1$.

Tedy výsledná účelová funkce E_{OC} problému obchodního cestujícího, jejíž minimalizací získáme jeho přípustné optimální řešení, tj. pořadí měst nejkratší okružní cesty, je součtem (3.43), (3.44), (3.45) a (3.46):

$$E_{OC} = E_A + E_B + E_C + E_D =$$

$$= -\frac{1}{2} \sum_{\substack{j=1 \\ j=(r,u)}}^{n} \sum_{\substack{i=1 \\ i=(s,v)}}^{n} \Big(-A\delta_{rs}(1-\delta_{uv}) - B\delta_{uv}(1-\delta_{rs}) - \\ -C(1-\delta_{ji}) - Dd(r,s)(\delta_{u,v-1} + \delta_{u,v+1}) \Big) y_j y_i$$

$$- \sum_{j=1}^{n} \left(CN - \frac{C}{2} \right) y_j \,. \qquad (3.47)$$

Porovnáním účelové funkce obchodního cestujícího (3.47) s prvními dvěma členy energetické funkce spojité Hopfieldovy sítě (3.38), vyextrahujeme váhy příslušné sítě, které zajistí minimalizaci E_{OC} během aktivního režimu:

$$\begin{aligned} w_{ji} &= -A\delta_{rs}(1-\delta_{uv}) - B\delta_{uv}(1-\delta_{rs}) - C(1-\delta_{ji}) \\ &\quad -Dd(r,s)(\delta_{u,v-1} + \delta_{u,v+1}) \qquad (3.48) \\ w_{j0} &= CN - \frac{C}{2} \qquad j=(r,u),\ i=(s,v),\ 1 \le r,s,u,v \le N \,. \end{aligned}$$

Z definice vah (3.48) je zřejmé, že $w_{ji} = w_{ij}$ a $w_{jj} = 0$ pro $1 \le i,j \le n$.

Při heuristickém řešení problému obchodního cestujícího nejprve nastavíme váhy spojité Hopfieldovy sítě podle (3.48), přitom volíme vhodné reálné parametry A, B, C, D určující stupeň minimalizace jednotlivých členů v $E_{OC} = E_A + E_B + E_C + E_D$ (např. $A = B = D = 500$, $C = 200$ [125]). Aktivní režim začneme v náhodném počátečním stavu sítě v blízkosti nulového vektoru, např. $y_j^{(0)} \in (0, 0.001)$ $(j = 1, \ldots, n)$. Vlastní výpočet spojité Hopfieldovy sítě probíhá podle (3.33), dokud se síť nenachází ve stabilním stavu odpovídajícím lokálnímu minimu energetické funkce (3.38), resp. účelové funkce problému obchodního cestujícího E_{OC}. Nakonec odečteme výsledek, tj. $y_{ru} \doteq 1$ interpretujeme tak, že obchodní cestující navštíví r-té město na své okružní cestě v u-tém pořadí.

kde $A > 0$ je míra vlivu E_A při minimalizaci a δ_{rs} je Kroneckerovo delta, tj. $\delta_{rs} = 1$, jestliže $r = s$, a $\delta_{rs} = 0$ pro $r \neq s$. Cílem úpravy v (3.43) je převést E_A na tvar porovnatelný s prvním členem energetické funkce (3.38). Podobně požadujeme, aby obchodní cestující byl při každé zastávce na své okružní cestě nejvýše v jednom městě, tj. aby v každém sloupci na obrázku 3.6 byl aktivní nejvýše jeden neuron. Tomu odpovídá minimalizace následujícího výrazu:

$$\begin{aligned} E_B &= \frac{B}{2} \sum_{u=1}^{N} \sum_{r=1}^{N} \sum_{\substack{s=1 \\ s \neq r}}^{N} y_{ru}\, y_{su} = \\ &= -\frac{1}{2} \sum_{\substack{j=1 \\ j=(r,u)}}^{n} \sum_{\substack{i=1 \\ i=(s,v)}}^{n} -B\delta_{uv}(1-\delta_{rs}) y_j y_i \,, \qquad (3.44) \end{aligned}$$

kde $B > 0$ je míra vlivu E_B při minimalizaci a cílem úpravy v (3.44) je převést E_B opět na tvar porovnatelný s prvním členem v (3.38). Nakonec ještě požadujeme, aby obchodní cestující na své okružní cestě navštívil právě N měst, tj. aby bylo právě N neuronů v síti (na obrázku 3.6) aktivních. Tomu odpovídá minimalizace následujícího výrazu:

$$E_C = \frac{C}{2} \left(N - \sum_{r=1}^{N} \sum_{u=1}^{N} y_{ru} \right)^2 = \frac{CN^2}{2} - \qquad (3.45)$$

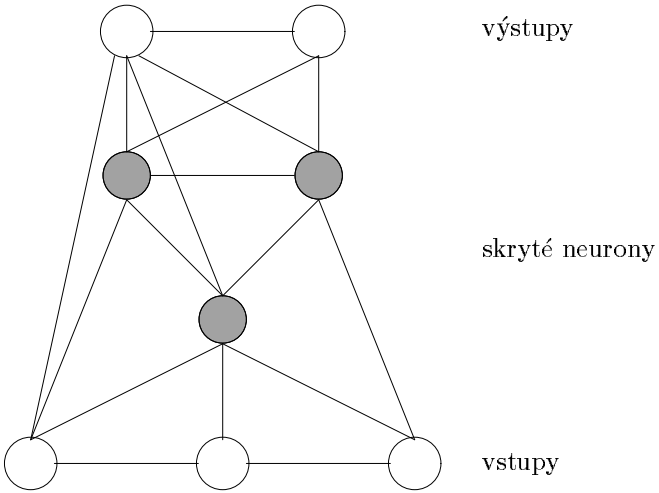
$$-\frac{1}{2} \sum_{\substack{j=1 \\ j=(r,u)}}^{n} \sum_{\substack{i=1 \\ i=(s,v)}}^{n} -C\,(1-\delta_{ji})\, y_j y_i - \sum_{j=1}^{n} \left(CN - \frac{C}{2} \right) y_j \,,$$

kde $C > 0$ je míra vlivu E_C při minimalizaci. Druhý řádek (3.45) je ve tvaru porovnatelném s prvními dvěma členy v (3.38), člen $\frac{CN^2}{2}$ je konstantní a nemá vliv na minimalizaci, proto jej dále v E_C nebudeme uvažovat. Tedy současná minimalizace (3.43), (3.44), (3.45) ve stavovém prostoru vynucuje stavy sítě odpovídající přípustným řešením problému obchodního cestujícího.

A nyní ještě v účelové funkci zohledníme požadavek na to, aby cesta obchodního cestujícího byla co nejkratší, tj. aby příslušné přípustné řešení bylo optimální. Tomu odpovídá minimalizace následujícího výrazu:

$$E_D = \frac{D}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \sum_{u=1}^{N} d(r,s) y_{ru}\, (y_{s,u-1} + y_{s,u+1}) = \qquad (3.46)$$

Použijeme též značení $\bar{j}$ pro množinu všech neuronů, které jsou spojeny s neuronem j.



Obr. 3.7: Příklad topologie Boltzmannova stroje.

Aktivní dynamiku Boltzmannova stroje popíšeme pro případ sekvenčního výpočtu. Na začátku aktivního režimu v čase 0 jsou stavy vstupních neuronů $i \in X$ nastaveny na bipolární vstup sítě $\mathbf{x} = (x_1, \ldots, x_n) \in \{-1, 1\}^n$ a během celého výpočtu jsou fixovány, tj. $y_i^{(t)} = x_i$ $(i \in X)$ pro $t \geq 0$. Stavy ostatních nevstupních neuronů $j \in V \setminus X$ jsou na začátku nastaveny náhodně $y_j^{(0)} \in \{-1, 1\}$. V diskrétním čase $t > 0$ aktivního režimu je pak náhodně vybrán nevstupní neuron $j \in V \setminus X$, který je aktualizován (ostatní svůj stav nemění). Podobně jako u Hopfieldovy sítě v odstavci 3.2.1 uvažujeme makroskopický čas τ, tj. $\tau s' < t \leq (\tau + 1)s'$, kde $s' = s - n$ je počet aktualizovaných (nevstupních) neuronů během jedné časové periody. Nejprve je vypočten reálný vnitřní potenciál neuronu j:

$$\xi_j^{(t-1)} = \sum_{i \in \bar{j}} w_{ji} y_i^{(t-1)} . \tag{3.49}$$

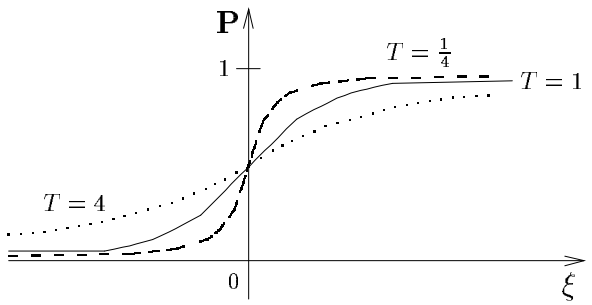
Avšak vzhledem k tomu, že nalezené minimum energetické funkce nemusí být globální, obecně není výsledné řešení problému obchodního cestujícího, získané v aktivním režimu spojité Hopfieldovy sítě, optimální a dokonce nemusí být ani přípustné. Vhodnou volbou parametrů A, B, C, D a příp. více pokusy s různým počátečním stavem sítě lze docílit lepší aproximaci optima. Také vhodná strategie volby parametru strmosti $\lambda > 0$ v aktivační funkci (3.34) může pomoci. Na začátku volíme λ dostatečně malé, aby stavy jednotlivých neuronů v síti měly větší stupeň volnosti, a v průběhu aktivního režimu, kdy se síť ve stavovém prostoru dostane do oblastí hlubších minim energetické funkce, můžeme tento parametr zvyšovat a zrychlovat tak konvergenci výpočtu. Obecně však neexistuje návod na efektivní volbu uvedených parametrů, které by zaručily konvergenci ke globálnímu minimu energetické funkce, tj. nalezení skutečného optima problému obchodního cestujícího, protože se jedná o NP-úplný problém. Uvedený postup je heuristikou, která se navíc nevyrovná známým klasickým algoritmům pro nalezení přibližného řešení problému obchodního cestujícího. Spíše ilustruje možnou nestandardní aplikaci neuronových sítí.

3.4 Boltzmannův stroj

Boltzmannův stroj je model neuronové sítě navržený Hintonem a Sejnowskim [1, 111, 112], který je stochastickou variantou Hopfieldovy sítě se skrytými neurony. Název tohoto modelu pochází ze statistické mechaniky, protože pravděpodobnost stavů při tzv. termální rovnováze sítě je dána Boltzmannovým rozdělením. Boltzmannův stroj lze např. použít jako heteroasociativní paměť (viz podkapitolu 3.1).

3.4.1 Stochastická aktivní dynamika

Organizační dynamika Boltzmannova stroje specifikuje na začátku pevnou architekturu cyklické neuronové sítě se symetrickými spoji, tj. topologii sítě tvoří neorientovaný graf. Množina s neuronů $V = A \cup B$ je disjunktně rozdělena na množinu b skrytých neuronů B a na množinu $a = n + m$ tzv. *viditelných* neuronů $A = X \cup Y$, kde X je množina n vstupních neuronů a Y je množina m výstupních neuronů. Příklad architektury Boltzmannova stroje je znázorněn na obrázku 3.7. Neurony značíme indexy i, j apod., ξ_j představuje reálný vnitřní potenciál a $y_j \in \{-1, 1\}$ bipolární stav, resp. výstup neuronu j. Stav viditelných neuronů budeme označovat $\boldsymbol{\alpha} \in \{-1, 1\}^a$, stav skrytých neuronů $\boldsymbol{\beta} \in \{-1, 1\}^b$ a stav Boltzmannova stroje $\mathbf{y} = (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \{-1, 1\}^s$. Spoj mezi neurony i a j je ohodnocen reálnou symetrickou váhou $w_{ji} = w_{ij}$. Pro jednoduchost opět uvažujeme nulové prahy neuronů a $w_{jj} = 0$ $(j \in V)$.



Obr. 3.8: Graf pravděpodobnostní aktivační funkce pro různé teploty T.

přičemž poměr obou vlivů závisí na teplotě. S klesající teplotou termální fluktuace mizí a v absolutní nule 0K (tj. $-273°$C) se vůbec nevyskytují. Naopak při vysokých teplotách termální fluktuace dominují a magnetická orientace jednoho spinu je nezávislá na magnetickém poli, tj. je s ním v rozporu přibližně stejně často jako mu odpovídá. Tento jev lze formálně matematicky popsat pomocí tzv. *Glauberovy dynamiky* [76], která odpovídá aktivní dynamice Boltzmannova stroje (3.49), (3.50) s tím, že skutečnou teplotu T_K v Kelvinech je ještě potřeba upravit: $T = k_B T_K$, kde $k_B = 1.38 \times 10^{-23} J/K$ je *Boltzmannova konstanta*. Pokles teploty v aktivním režimu lze připodobnit ke skutečnému žíhání materiálu při výrobě tvrdého kovu s pravidelnou krystalickou strukturou, při které dochází k postupnému ochlazování, aby se předešlo anomáliím v krystalické struktuře materiálu.

Aktivní dynamiku Boltzmannova stroje je také možno interpretovat neurofyziologicky. Biologický neuron je kromě vlastního elektrického signálu z dendritů (viz podkapitolu 1.2) ovlivňován jinými okolnostmi. Například impuls generovaný v axonu má po každé různou intenzitu. Navíc dochází ke zpoždění signálu na synapsích, k náhodným fluktuacím apod. Tyto rušivé vlivy můžeme chápat jako šum, který lze v Boltzmannově stroji reprezentovat termálními fluktuacemi, i když v tomto případě parametr T nepředstavuje skutečnou teplotu, ale úroveň šumu.

3.4.3 Rovnovážný stav

Podobně jako u Hopfieldovy sítě v odstavci 3.2.2 se definuje energetická funkce Boltzmannova stroje $E(\mathbf{y})$ v závislosti na jeho stavu $\mathbf{y}$:

$$E(\mathbf{y}) = -\frac{1}{2} \sum_{j \in V} \sum_{i \in \bar{j}} w_{ji} y_j y_i \,. \tag{3.54}$$

Výstup neuronu j v čase t je pak určen stochasticky tak, že neuron j je aktivní s pravděpodobností

$$\mathbf{P}\left\{y_j^{(t)} = 1\right\} = \sigma(\xi_j^{(t-1)})\,, \tag{3.50}$$

resp. pasivní s pravděpodobností

$$\mathbf{P}\left\{y_j^{(t)} = -1\right\} = 1 - \mathbf{P}\left\{y_j^{(t)} = 1\right\} = \sigma(-\xi_j^{(t-1)})\,, \tag{3.51}$$

kde σ je *pravděpodobnostní aktivační funkce*:

$$\sigma(\xi) = \frac{1}{1 + e^{-2\xi/T^{(\tau)}}}\,, \tag{3.52}$$

jejíž tvar odpovídá standardní sigmoidě (1.9).

3.4.2 Simulované žíhání

Parametr $T^{(\tau)} > 0$ v rovnici (3.52), který se může v makroskopickém čase τ vyvíjet, se díky fyzikální analogii nazývá *teplota* a je nepřímo úměrný strmosti pravděpodobnostní aktivační funkce (srovnejte s (3.34)). Je zřejmé, že pro teplotu $T \to \infty$ pravděpodobnosti (3.50), (3.51) obou bipolárních stavů jsou shodně $\frac{1}{2}$ a Boltzmannův stroj se v aktivním režimu chová zcela náhodně. Naopak pro $T \to 0$ je výpočet Boltzmannova stroje podle (3.49), (3.50) deterministický a shodný s aktivním režimem Hopfieldovy sítě (3.27), (3.28). Obvykle na počátku volíme dostatečně velkou teplotu $T^{(0)}$ tak, aby pravděpodobnost (3.50) byla o něco málo vyšší než $\frac{1}{2}$ a výpočet měl velký stupeň volnosti. Kvůli konvergenci pak postupně celý systém „chladíme“, tj. teplotu pomalu snižujeme. Tomuto postupu se díky fyzikální analogii říká *simulované žíhání* (simulated annealing) [149]. Např. teplota $T^{(\tau)} = \eta^c T^{(0)}$ pro $cq \le \tau < (c+1)q$ $(c = 0, 1, 2, \ldots)$ klesá po $q > 0$ makroskopických krocích, kde $0 < \eta < 1$ volíme blízké 1 nepřímo úměrné volbě q (pro menší q např. $\eta = 0.98$ a pro větší $\eta = 0.9$). Nebo se vývoj teploty řídí následující heuristikou [73]:

$$T^{(\tau)} = \frac{T^{(0)}}{\log(1+\tau)} \quad \tau > 0\,. \tag{3.53}$$

Graf pravděpodobnostní aktivační funkce pro různé teploty je znázorněn na obrázku 3.8.

Uvedená aktivní dynamika Boltzmannova stroje dále rozvíjí fyzikální analogii Hopfieldovy sítě z odstavce 3.2.2 tak, že navíc počítá s teplotou. Spiny v magnetických materiálech jsou totiž kromě magnetického pole ovlivňovány tzv. *termálními fluktuacemi*, které mají tendenci často a náhodně měnit magnetickou orientaci spinů a omezují tak vliv magnetického pole,

3.4.4 Boltzmannovo učení

Nyní popíšeme **adaptivní** dynamiku Boltzmannova stroje pro případ heteroasociativní paměti. Požadovaná funkce sítě je v adaptivním režimu zadána pomocí diskrétního rozdělení pravděpodobností stavů viditelných neuronů, které každému možnému stavu $\boldsymbol{\alpha} = (\mathbf{x}, \mathbf{d}) \in \{-1,1\}^{n+m}$ vstupních a výstupních (viditelných) neuronů přiřazuje požadovanou nenulovou pravděpodobnost $p_d(\mathbf{x}, \mathbf{d}) > 0$. V praxi však obvykle toto pravděpodobnostní rozdělení není explicitně známo a také jeho popis by představoval exponenciálně mnoho (2^a, kde a=n+m) nenulových pravděpodobností. Proto se uvažuje tréninková množina

$$\mathcal{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \{-1,1\}^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \{-1,1\}^m \end{array} \quad k = 1, \ldots, p \right\}, \quad (3.58)$$

která obsahuje jen relevantní vstupy $\mathbf{x}_k$ $(k = 1, \ldots, p)$ s odpovídajícími požadovanými výstupy $\mathbf{d}_k$, kde počet vzorů $0 < p \ll 2^a$ je typicky mnohem menší než počet možných stavů viditelných neuronů. Dále se obvykle předpokládá rovnoměrné rozdělení tréninkových vzorů, tj. $p_d(\mathbf{x}_k, \mathbf{d}_k) = \frac{1}{p} > 0$ $(k = 1, \ldots, p)$. Při adaptaci Boltzmannova stroje se pak do tréninkové množiny $\mathcal{T}$ vnáší s malou pravděpodobností šum, který zajistí nenulovou pravděpodobnost výskytu zbylých stavů viditelných neuronů, které nejsou tréninkovými vzory.

Cílem adaptace Boltzmannova stroje je najít takovou konfiguraci sítě $\mathbf{w}$, aby pravděpodobnostní rozdělení $p_A(\boldsymbol{\alpha})$ stavů $\boldsymbol{\alpha} \in \{-1,1\}^a$ viditelných neuronů $A = X \cup Y$ v termální rovnováze odpovídalo požadovanému rozdělení pravděpodobností $p_d(\boldsymbol{\alpha})$ stavů $\boldsymbol{\alpha} = (\mathbf{x}, \mathbf{d})$ tréninkových vzorů (3.58). Pomocí (3.56) lze vyjádřit pravděpodobnost $p_A(\alpha)$ stavu $\boldsymbol{\alpha} \in \{-1,1\}^a$ viditelných neuronů nezávisle na stavu $\boldsymbol{\beta} \in \{-1,1\}^b$ skrytých neuronů:

$$p_A(\boldsymbol{\alpha}) = \sum_{\boldsymbol{\beta} \in \{-1,1\}^b} p_B(\boldsymbol{\alpha}, \boldsymbol{\beta}). \quad (3.59)$$

Vhodnou mírou rozdílu mezi pravděpodobnostními rozděleními p_A a p_d, tj. chybou $\mathcal{E}$, je relativní entropie zvážená pravděpodobností výskytu tréninkových vzorů:

$$\mathcal{E}(\mathbf{w}) = \sum_{\boldsymbol{\alpha} \in \{-1,1\}^a} p_d(\boldsymbol{\alpha}) \log \frac{p_d(\boldsymbol{\alpha})}{p_A(\boldsymbol{\alpha})}. \quad (3.60)$$

Chyba $\mathcal{E}(\mathbf{w})$ je vždy nezáporná a je nulová, právě když $p_A = p_d$. Navíc $\mathcal{E}(\mathbf{w})$ je funkcí konfigurace $\mathbf{w}$ Boltzmannova stroje, protože p_A podle (3.59), (3.56) a (3.54) závisí na synaptických váhách. Můžeme tedy minimalizovat chybovou $\mathcal{E}(\mathbf{w})$ ve váhovém prostoru pomocí gradientní metody.

V průměrném případě se očekává, že energie (3.54) během aktivního režimu klesá, dokud se Boltzmannův stroj nedostane v čase $t^\star$ do tzv. *termální rovnováhy*, kdy stav Boltzmannova stroje sice není vzhledem ke stochastickému výpočtu konstantní, ale lokálně kolísá kolem „stabilního“ stavu:

$$y_j^{(t^\star)} = \begin{cases} 1 & \xi_j^{(t^\star)} > 0 \\ -1 & \xi_j^{(t^\star)} < 0 \end{cases} \qquad j \in V, \quad (3.55)$$

který odpovídá lokálnímu minimu E. Pravděpodobnost, že se Boltzmannův stroj v termální rovnováze při teplotě T nachází ve stavu $\mathbf{y}^\star \in \{-1,1\}^s$, se řídí Boltzmannovým rozdělením ze statistické mechaniky:

$$p_B(\mathbf{y}^\star) = \frac{e^{-E(\mathbf{y}^\star)/T}}{\sum_{\mathbf{y} \in \{-1,1\}^s} e^{-E(\mathbf{y})/T}}. \quad (3.56)$$

Pomocí něj lze vypočítat tzv. *rovnovážný stav* $\overline{\mathbf{y}^\star} \in \{-1,1\}^s$, který je průměrem stavů zvážených příslušnými pravděpodobnostmi:

$$\overline{\mathbf{y}_j^\star} = \sum_{\mathbf{y} \in \{-1,1\}^s} p_B(\mathbf{y}) y_j \quad j \in V. \quad (3.57)$$

Stavy výstupních neuronů $\overline{y_j^\star}$ $(j \in Y)$ rovnovážného stavu pak pro daný vstup určují *průměrný výstup*.

Kromě heteroasociativní paměti lze Boltzmannův stroj použít pro heuristické řešení optimalizačních úloh, pokud optimalizační problém s příslušnou účelovou funkcí (optimalita) a omezením (přípustnost) lze převést na minimalizaci kvadratické formy ve tvaru (3.54) (srovnejte s aplikací spojité Hopfieldovy sítě při řešení problému obchodního cestujícího v odstavci 3.3.2). V tomto případě váhy Boltzmannova stroje obdržíme porovnáním příslušné kvadratické formy s energetickou funkcí sítě E. Optimální přípustné řešení daného problému se pak snažíme v aktivním režimu získat minimalizací energetické funkce stochastickou variantou gradientní metody (3.49), (3.50) (srovnejte s (3.30)). Hlavním problémem nelineární optimalizace, se kterým jsme se již např. setkali u učícího algoritmu backpropagation (viz odstavec 2.2.2), jsou lokální minima. Cílem nelineární optimalizace je nalézt globální minimum, tj. vyhnout se lokálním minimům. Výhoda Boltzmannova stroje oproti (spojité) Hopfieldově síti spočívá v možnosti přejít při vyšší teplotě (díky stochastickému výpočtu) z lokálního minima E do stavu s vyšší energií, ze kterého případně vede cesta k hlubšímu minimu. Vhodnou strategií při simulovaném žíhání lze najít lepší optimum než při deterministickém výpočtu Hopfieldovy sítě. Také v případě asociativní paměti, kdy globální minima energetické funkce reprezentují tréninkové vzory a lokální minima odpovídají nepravým vzorům (viz odstavec 3.2.2), se tak lze v aktivním režimu vyhnout fantomům.

3.4.5 Učící algoritmus

Uvedenou adaptivní dynamiku Boltzmannova stroje shrneme do přehledného algoritmu, který lze realizovat distribuovaně prostřednictvím specializovaného hardwaru [272] nebo jej můžeme simulovat pomocí klasického počítače metodou Monte Carlo:

1. Polož diskrétní čas adaptace $t := 0$.
2. Zvol náhodně s rovnoměrným rozdělením $w_{ji}^{(0)} = w_{ij}^{(0)} \in \langle -1, 1\rangle$.
3. Zvětši adaptivní čas $t := t + 1$.
4. Polož $\varrho_{ji}^{+\,(t-1)} := 0$.
5. Pro každý tréninkový vzor $k = 1, \ldots, p$ vykonej q-krát následující akce:
   (a) Fixuj stavy viditelných neuronů $\boldsymbol{\alpha} = (\mathbf{x}'_k, \mathbf{d}'_k) \in \{-1, 1\}^{n+m}$, kde pravděpodobnost shody s jedním stavem k-tého tréninkového vzoru je např. $\mathbf{P}\{x'_{ki} = x_{ki}\} = 0.9$ $(i = 1, \ldots, n)$ a $\mathbf{P}\{d'_{kj} = d_{kj}\} = 0.9$ $(j = 1, \ldots, m)$. Dále postupuj podle aktivní dynamiky (3.49), (3.50) (pro stavy skrytých neuronů) tak, že pomocí simulovaného žíhání dosáhni termální rovnováhy ve stavu $\mathbf{y}^\star$ s nějakou koncovou teplotou $T^\star$. Stav $\mathbf{y}^\star$ bude v makroskopickém aktualizačním čase $\tau = 0$ počátečním stavem dalšího výpočtu Boltzmannova stroje.
   (b) V aktualizačním makroskopickém čase $\tau = 1, \ldots, r$ vykonej následující akce:
      i. Proveď jeden makroskopický krok výpočtu Boltzmannova stroje při teplotě $T^\star$, tj. aktualizuj v náhodném pořadí stavy všech skrytých neuronů podle (3.49), (3.50).
      ii. Aktualizuj statistiku pro výpočet $\overline{y_j^\star y_i^\star}(A)$ podle aktuálního stavu, tj. $\varrho_{ji}^{+\,(t-1)} := \varrho_{ji}^{+\,(t-1)} + y_j^{(\tau)} y_i^{(\tau)}$.
6. Zprůměruj $\varrho_{ji}^{+\,(t-1)} := \frac{\varrho_{ji}^{+\,(t-1)}}{pqr}$.
   { $\varrho_{ji}^{+\,(t-1)}$ je odhadem aktuálního $\overline{y_j^\star y_i^\star}(A)$. }
7. Polož $\varrho_{ji}^{-\,(t-1)} := 0$.

Na začátku adaptivního režimu Boltzmannova stroje v čase 0 jsou váhy konfigurace $\mathbf{w}^{(0)}$ nastaveny náhodně s rovnoměrným rozdělením blízko nuly, např. $w_{ji}^{(0)} = w_{ij}^{(0)} \in \langle -1, 1\rangle$ $(j \in V, i \in \bar{j})$. Adaptace probíhá v diskrétních časových krocích, které odpovídají tréninkovým cyklům, při nichž se každý tréninkový vzor z $\mathcal{T}$ předkládá vícekrát. Nová konfigurace $\mathbf{w}^{(t)}$ v čase $t > 0$ se vypočte:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)} \quad j \in V, i \in \bar{j}\,, \tag{3.61}$$

kde změna vah $\Delta\mathbf{w}^{(t)}$ v čase $t > 0$ je úměrná zápornému gradientu chybové funkce (3.60) v bodě $\mathbf{w}^{(t-1)}$:

$$\Delta w_{ji}^{(t)} = -\varepsilon \frac{\partial \mathcal{E}}{\partial w_{ji}} \left(\mathbf{w}^{(t-1)}\right) \quad j \in V, i \in \bar{j}\,, \tag{3.62}$$

kde $0 < \varepsilon < 1$ rychlost učení.

K realizaci uvedené adaptivní dynamiky (3.61) potřebujeme vypočítat gradient chybové funkce ve vzorci (3.62). Formálním derivováním chybové funkce (3.60) s využitím (3.59), (3.56), (3.54) a interpretací příslušných pravděpodobností dostaneme (podrobné odvození lze najít např. v [109]):

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} = -\frac{1}{T}\left(\overline{y_j^\star y_i^\star}(A) - \overline{y_j^\star y_i^\star}\right) \quad j \in V, i \in \bar{j}\,. \tag{3.63}$$

Výraz $\overline{y_j^\star y_i^\star}(A)$ v rovnici (3.63) je průměrná hodnota $y_j^\star y_i^\star$ při termální rovnováze, když jsou stavy $\boldsymbol{\alpha} = (\mathbf{x}, \mathbf{d})$ vstupních a výstupních (viditelných) neuronů z $A = X \cup Y$ fixovány:

$$\overline{y_j^\star y_i^\star}(A) = \sum_{\boldsymbol{\alpha} \in \{-1,1\}^a} p_d(\boldsymbol{\alpha}) \sum_{\boldsymbol{\beta} \in \{-1,1\}^b} \frac{p_B(\boldsymbol{\alpha}, \boldsymbol{\beta})}{p_A(\boldsymbol{\alpha})} y_j^\star(\boldsymbol{\alpha}) y_i^\star(\boldsymbol{\alpha})\,, \tag{3.64}$$

kde $y_j^\star(\boldsymbol{\alpha})$ je výstup neuronu $j \in V$ při termální rovnováze Boltzmannova stroje, když jsou stavy viditelných neuronů fixovány vektorem $\boldsymbol{\alpha}$. Podobně člen $\overline{y_j^\star y_i^\star}$ v (3.63) je průměrná hodnota $y_j^\star y_i^\star$ při termální rovnováze Boltzmannova stroje bez fixace viditelných neuronů:

$$\overline{y_j^\star y_i^\star} = \sum_{\mathbf{y} \in \{-1,1\}^s} p_B(\mathbf{y}) y_j^\star y_i^\star\,. \tag{3.65}$$

Po dosazení (3.63) do (3.62) obdržíme *Boltzmannovo pravidlo učení*:

$$\Delta w_{ji} = \frac{\varepsilon}{T}\left(\overline{y_j^\star y_i^\star}(A) - \overline{y_j^\star y_i^\star}\right) \quad j \in V, i \in \bar{j}\,. \tag{3.66}$$

První člen $\overline{y_j^\star y_i^\star}(A)$ (viz (3.64)) ve vzorci (3.66) lze interpretovat jako Hebbovo učení (srovnejte s (3.5)) s fixovanými vstupními a výstupními (viditelnými) neurony a druhý člen $-\overline{y_j^\star y_i^\star}$ (viz (3.65)) odpovídá Hebbovu odučování (srovnejte s (3.31)) bez fixace viditelných neuronů. Adaptivní režim konverguje, když se oba tyto členy rovnají pro $j \in V, i \in \bar{j}$.

8. Vykonej (pq)-krát následující akce:

   (a) Bez fixace viditelných neuronů, tj. pro náhodný počáteční stav Boltzmannova stroje postupuj podle aktivní dynamiky (3.49), (3.50) (pro stavy všech neuronů) tak, že pomocí simulovaného žíhání dosáhni termální rovnováhy ve stavu $\mathbf{y}^\star$ s nějakou koncovou teplotou $T^\star$. Stav $\mathbf{y}^\star$ bude v makroskopickém aktualizačním čase $\tau = 0$ počátečním stavem dalšího výpočtu Boltzmannova stroje.

   (b) V aktualizačním makroskopickém čase $\tau = 1, \ldots, r$ vykonej následující akce:

      i. Proveď jeden makroskopický krok výpočtu Boltzmannova stroje při teplotě $T^\star$, tj. aktualizuj v náhodném pořadí stavy všech neuronů podle (3.49), (3.50).

      ii. Aktualizuj statistiku pro výpočet $\overline{y_j^\star y_i^\star}$ podle aktuálního stavu $\varrho_{ji}^{-(t-1)} := \varrho_{ji}^{-(t-1)} + y_j^{(\tau)} y_i^{(\tau)}$.

9. Zprůměruj $\varrho_{ji}^{-(t-1)} := \frac{\varrho_{ji}^{-(t-1)}}{pqr}$.
   { $\varrho_{ji}^{-(t-1)}$ je odhadem aktuálního $\overline{y_j^\star y_i^\star}$. }

10. Podle (3.66) vypočítej $\Delta w_{ji}^{(t)} := \frac{\varepsilon}{T^\star} \left(\varrho_{ji}^{+(t-1)} - \varrho_{ji}^{-(t-1)} \right)$.

11. Aktualizuj konfiguraci $w_{ji}^{(t)} := w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)}$.

12. Je-li $\Delta w_{ji}^{(t)}$ dostatečně malé, skonči, jinak pokračuj krokem 3.

Z uvedeného algoritmu je zřejmé, že adaptivní režim Boltzmannova stroje je výpočetně náročnou záležitostí. Přesto se v praxi ukazuje, že dosažené výsledky Boltzmannova stroje při následné aplikaci jsou velmi dobré [109].

Kapitola 4

Samoorganizace

V této kapitole se budeme věnovat modelům neuronových sítí, které využívají *soutěžní* strategie učení *(competitive learning)*. Společným principem těchto modelů je, že výstupní neurony sítě spolu soutěží o to, který z nich bude aktivní. Na rozdíl od jiných učících principů (např. Hebbovského učení) je tedy v určitém čase aktivní jen jeden neuron.

První průkopnické práce o soutěžním učení se objevují již v šedesátých letech, např. v Nilssonově knize [208] najdeme popsán princip, který nyní nazýváme Kohonenovým učením. Počátkem sedmdesátých let se soutěžnímu učení věnuje několik autorů, kteří se navzájem ovlivňují, i když jejich práce mají různou motivaci. V roce 1973 navrhl von der Malsburg [186] soutěžní učící algoritmus, který měl modelovat způsob detekování hran v primárním optickém kortexu savců. Jeho algoritmus nebyl lokální — předpokládal, že zvětšení jedné váhy bude mít za následek zmenšení vah ostatních. Tento nedostatek odstranila až Grossbergova práce [88], která se, podobně jako autorův předchozí článek [87] z roku 1972, zabývá adaptivním rozpoznáváním vzorů, které neničí již naučené příklady. Také Willshaw a von der Malsburg [292] přicházejí nezávisle s novou verzí soutěžního učení. Jmenujme ještě pro úplnost práci dalšího nestora umělých neuronových sítí Fukushimy, jenž v roce 1975 navrhl model vícevrstvé samoorganizační sítě známé jako *kognitron* [69].

Asi nejdůležitější neuronovou architekturou vycházející ze soutěžního učení je Kohonenova samoorganizační mapa (Self-Organizing Map), která byla poprvé popsána v roce 1982 [153, 156, 157]. Kohonen se také již od sedmdesátých let zabýval samoorganizačním učením a popsal učící algoritmus nesoucí nyní jeho jméno. I když nebyl jeho prvním objevitelem, má zásluhu na tom, že jako cíl učícího procesu vytkl vytvoření množiny reprezentantů majících stejné pravděpodobnosti výběru, což byl na poli neuronových sítí

nový koncept. (Přesněji, hledáme takové reprezentanty, že vybereme-li náhodný vstupní vektor z rozdělení pravděpodobnosti odpovídající rozdělení tréninkové množiny, bude mít každý reprezentant stejnou pravděpodobnost, že mu je nejblíže.)

Poznamenejme nakonec, že v současné době existuje i řada neurofyziologických prací, které experimentálně potvrzují, že některé oblasti v mozku pracují na soutěžním principu.

Struktura této kapitoly je následující: nejprve se budeme věnovat problému učení bez učitele (unsupervised learning), někdy též nazývané vektorová kvantizace (VQ), a ukážeme si (klasický) Lloydův algoritmus a nejjednodušší variantu soutěžního učení, v literatuře nazývanou *Kohonenovo učení*. V další části probereme Kohonenovy samoorganizační mapy a příbuzné metody učící vektorové kvantizace LVQ (learning vector quantization). Na závěr ukážeme síť typu *counterpropagation* navrženou Hecht-Nielsenem [103] v roce 1986.

4.1 Vektorová kvantizace

Termín a základní algoritmy vektorové kvantizace pocházejí z oblasti zpracování a aproximace signálů [85]. Úkolem VQ je aproximovat hustotu pravděpodobnosti $p(\mathbf{x})$ reálných vstupních vektorů $\mathbf{x} \in \mathbb{R}^n$ pomocí konečného počtu jednotek či reprezentantů (v klasické literatuře označovaných jako *codebook vectors*) $\mathbf{w}_i \in \mathbb{R}^n$; $(i = 1, \ldots, h)$. Máme-li již nějakým způsobem reprezentanty určeny, potom ke každému vektoru $\mathbf{x} \in \mathbb{R}^n$ přiřadíme jako jeho reprezentanta tu jednotku $\mathbf{w}_c$, která je mu nejblíže, tedy:

$$c = \arg \min_{l=1,\ldots,h} \{||\mathbf{x} - \mathbf{w}_l||\} . \tag{4.1}$$

Jedním ze způsobů řešení problému nalezení reprezentantů je minimalizovat chybu VQ definovanou jako:

$$E = \int ||\mathbf{x} - \mathbf{w}_c||^2 p(\mathbf{x})\, d\mathbf{x} ,$$

kde $||\cdot||$ je euklidovská norma a c je definováno předpisem (4.1). Nejčastěji se v praxi setkáváme s případem, že hustotu pravděpodobnosti $p(\mathbf{x})$ neznáme a problém máme zadán konečnou tréninkovou množinu vzorů $T = \{\mathbf{x}^{(t)};\ t = 1, \ldots, k\}$. V tom případě vypočteme chybu VQ jako:

$$E = \frac{1}{k} \sum_{t=1}^{k} ||\mathbf{x}^{(t)} - \mathbf{w}_c||^2 . \tag{4.2}$$

Ač jsou předchozí vzorce zdánlivě jednoduché, uvědomme si, že index c funkčně závisí na vzorech $\mathbf{x}$ i jednotkách $\mathbf{w}$. Také z těchto důvodů není lehké explicitně vyjádřit gradient chyby E vzhledem k parametrům $\mathbf{w}$ a použít pak obvyklého postupu minimalizace. Proto byly navrženy heuristické iterativní algoritmy hledající přibližné řešení. Nyní uvedeme základní *Lloydův algoritmus* navržený původně pro jednorozměrné vstupy a později generalizovaný Lindem, Buzem a Grayem [174].

4.1.1 Lloydův algoritmus

Uvažujme tedy problém zadaný tréninkovou množinou T a parametrem h určujícím počet reprezentantů $\mathbf{w}_i$, kterými kódujeme danou množinu. Na začátku inicializujeme jednotky $\mathbf{w}_i$ náhodně a pak opakujeme následující proces tak dlouho, dokud hodnota chyby VQ neklesne pod nějakou předem stanovenou mez. Projdeme celou tréninkovou množinu a ke každému vstupu $\mathbf{x}^{(t)}$ určíme jemu příslušnou jednotku $\mathbf{w}_c$ podle vztahu (4.1). Dále pro každou jednotku $\mathbf{w}_j$ uvážíme množinu T_j všech vzorů $\mathbf{x}^{(t)}$, pro něž byl $\mathbf{w}_j$ jejich reprezentantem. Tedy

$$T_j = \left\{ \mathbf{x}^{(t)}; j = \arg \min_{l=1,\ldots,h} \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \right\} . \tag{4.3}$$

Nyní spočteme těžiště $\mathbf{t}_j$ množiny T_j:

$$\mathbf{t}_j = \frac{1}{|T_j|} \sum_{\mathbf{x}_j \in T_j} \mathbf{x}_j \tag{4.4}$$

a následně nahradíme $\mathbf{w}_j$ hodnotou $\mathbf{t}_j$. Takto určíme novou polohu všech jednotek $\mathbf{w}_j$ a pokračujeme dalším průchodem tréninkovou množinou.

Uveďme nyní přesnější popis Lloydova algoritmu:

Vstup: Tréninková množina $T = \{\mathbf{x}^{(t)}; t = 1, \ldots, k\}$, počet reprezentantů h, požadovaná hodnota chyby ε.

Výstup: Hodnoty reprezentantů $\mathbf{w}_j; j = 1, \ldots, h$.

Inicializace: Rozmísti $\mathbf{w}_j$ náhodně

Opakuj:

Ke každému vektoru $\mathbf{x}^{(t)} \in T$ přiřaď $\mathbf{w}_c$ dle (4.1).
Spočítej chybu E podle (4.2).
Je-li $E < \varepsilon$, skonči.

Pro každé $j = 1, \ldots, h$ spočítej $\mathbf{t}_j$ dle (4.4).
Přiřaď $\mathbf{w}_j = \mathbf{t}_j$.

Pokračuj v cyklu.

4.1.2 Kohonenovo učení

Právě uvedený algoritmus je nevýhodný v tom, že ke změnám reprezentantů dochází až po průchodu celou tréninkovou množinou. Proto byla vyvinuta jeho on-line varianta, kterou si nyní popíšeme již v terminologii neuronových sítí. Jde o jednoduchou *samoorganizační síť*, jejíž učící algoritmus se nazývá *Kohonenovo učení*.

Organizační dynamika této sítě je jednoduchá, jde o dvouvrstvou síť s úplným propojením jednotek mezi vrstvami. Vstupní vrstvu tvoří n neuronů, které slouží k distribuci vstupních hodnot $\mathbf{x} \in \mathbb{R}^n$. Jednotky ve výstupní vrstvě plní funkci výše popsaných reprezentantů. Váhy $\mathbf{w}_j = (w_{j1}, \ldots, w_{jn}), j = 1, \ldots, h$ náležející jedné výstupní jednotce j určují její polohu ve vstupním prostoru.

Podívejme se nyní na **aktivní dynamiku** sítě. Zatímco vstupy $\mathbf{x} \in \mathbb{R}^n$ sítě mohou být libovolná reálná čísla, výstupy y_j jsou obvykle jen hodnoty 0 nebo 1, přičemž aktivní (s hodnotou výstupu $y_j = 1$) je vždy právě jeden výstupní neuron — ten, který vyhrál kompetici. Výstup každého neuronu v závislosti na jeho vzdálenosti od vstupního vektoru $\mathbf{x}^{(t)}$ se spočítá takto:

$$y_j^{(t)} = \begin{cases} 1 & j = \arg \min_{l=1,\ldots,h} \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \\ 0 & \text{jinak.} \end{cases} \tag{4.5}$$

Principu, který jsme právě popsali, se říká „vítěz bere vše“ (“winner takes all”) a jeden z mechanismů, kterým se realizuje, je tzv. *laterální inhibice*. Můžeme si představit, že všechny výstupní neurony jsou každý s každým navíc spojeny laterálními vazbami, které mezi nimi přenášejí inhibiční signály. Každý výstupní neuron se tak snaží v kompetici zeslabit ostatní neurony silou úměrnou jeho potenciálu, který je tím větším, čím je neuron blíže vstupu. Výsledkem tedy je, že výstupní neuron s největším potenciálem utlumí ostatní výstupní neurony a sám zůstane aktivním. V literatuře jsou mechanismy laterální inhibice a laterální zpětné vazby zmiňovány poměrně podrobně, jak z hlediska jejího výskytu v mozku [153], tak i její realizace v modelech umělých neuronových sítí [97, 10. kapitola], my se jí však dále zabývat nebudeme.

Popišme nyní **adaptivní dynamiku** naší jednoduché samoorganizační sítě řízenou Kohonenovým učením. Princip je jednoduchý: procházíme celou tréninkovou množinu a po předložení jednoho tréninkového vzoru proběhne mezi jednotkami sítě obvyklá kompetice. Její vítěz změní své váhy podle následujícího vzorce:

$$w_{ji}^{(t)} = \begin{cases} w_{ji}^{(t-1)} + \theta(x_i^{(t-1)} - w_{ji}^{(t-1)}) & j = \arg \min_l \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \\ w_{ji} & \text{jinak.} \end{cases} \tag{4.6}$$

Reálný parametr $0 < \theta \leq 1$ určuje míru změny vah. Na počátku učení je obvykle blízký jedné a postupně se zmenšuje.

Geometrický význam popsaného algoritmu je, že vítězný neuron c posune svůj váhový vektor $\mathbf{w}_c$ o určitou poměrnou vzdálenost směrem k aktuálnímu vstupu. Motivací tohoto přístupu je snaha, aby vítězný neuron, který nejlépe reprezentuje předložený vstup (je mu nejblíže), ještě zlepšil svou relativní pozici vůči němu.

Často se v literatuře setkáme s tím, že výše popsaný algoritmus je nazýván *k-means clustering* (hledání k středů), což je úkol nalézt k reprezentantů minimalizujících sumu čtverců (4.2). Cypkin ukázal [53], že Kohonenův algoritmus najde řešení k středů, je-li hodnota θ nepřímo úměrná podílu vektorů $\mathbf{x}$ ležících blíže k vítěznému reprezentantu než k ostatním. Kohonenovo učení představuje tedy inkrementální (on-line) algoritmus pro řešení problému k středů, a tím i inkrementálně řeší VQ.

4.1.3 Modifikace Kohonenova učení

Ačkoliv, jak jsme již zmínili, bylo Kohonenovo učení původně navrženo s úmyslem sestrojit množinu stejně pravděpodobných reprezentantů, přesto tento úkol neplní. Ukazuje se, že algoritmus pracuje dobře jen u problémů, kde hustota pravděpodobnosti je přibližně konstantní v jedné konvexní oblasti vstupního prostoru. Hecht-Nielsen v [105] uvádí jednoduchý příklad ilustrující slabiny tohoto algoritmu. Princip je následující: Uvažujme dvě izolované dostatečně vzdálené oblasti, v nichž se nacházejí vstupní data. Představme si, že na počátku rozmístíme reprezentanty náhodně v jedné z oblastí. Vzor z druhé oblasti „přitáhne“ jednoho z reprezentantů, který pak již pokaždé vyhraje kompetici pro vzory z této množiny. Tak dojde k tomu, že celou druhou oblast bude reprezentovat pouze jedna jednotka.

Popišme nyní tři možnosti, kterými lze tento typ problému odstranit. Jde o přidání *šumu* k tréninkové množině, *radiální růst* a lokální *paměť* jednotek.

Přidání šumu Nejjednodušším, ale také časově nejnáročnějším, přístupem je přidávání náhodných vektorů z rovnoměrného rozdělení k tréninkové množině. Na začátku učení je náhodných tréninkových vzorů více než dat a v průběhu učení jejich podíl klesá, až zbydou jen původní tréninkové vzory.

Radiální růst U radiálního růstu začínáme s váhovými vektory blízkými nulovému vektoru $\mathbf{0} = (0, \ldots, 0)$. Všechny tréninkové vzory jsou vynásobeny reálným číslem β, které je zpočátku také blízké nule. Důsledkem je, že všechny vzory jsou zpočátku blízko všem reprezentantům. V průběhu učení zvětšujeme β až k jedničce, čímž nutíme váhové vektory vzdalovat se od počátku souřadné soustavy, až nakonec opět učíme původní data. I toto

řešení je poměrně časově náročné a navíc, stejně jako přidání šumu k tréninkovým datům, řeší jen polovinu problému. Obě techniky odstraňují vznik málo reprezentovaných oblastí, ale druhou slabinou Kohonenova algoritmu jsou oblasti, kde je reprezentantů více, než by odpovídalo hustotě pravděpodobnosti.

Lokální paměť Metoda lokální paměti jednotek navržená Deseinem [58] je založena na myšlence, že při stejné pravděpodobnosti by každý z k reprezentantů měl zvítězit v kompetici zhruba $1/k$ krát. Každá jednotka si tak udržuje přehled o tom, kolikrát již soutěž vyhrála, a pokud je její dosavadní četnost výher příliš vysoká, vypadává (na nějakou dobu) z další kompetice. Formálně přidáme každé jednotce j dva lokální parametry: odhad relativní četnosti výher v kompetici f_j a práh b_j vypočtený na základě této hodnoty.

Učící algoritmus pak vypadá takto:

V čase t předložíme síti vzor $\mathbf{x}^{(t)}$ a podle (4.5) vypočítáme odezvu výstupních jednotek. Dále pro každou jednotku j spočítáme novou hodnotu parametru f_j:

$$f_j^{(t)} = f_j^{(t-1)} + \beta(y_j - f_j^{(t-1)})\,. \tag{4.7}$$

Podle hodnoty f_j vypočítáme práh b_j jako:

$$b_j^{(t)} = \gamma(1/N - f_j^{(t)})\,. \tag{4.8}$$

Reálné parametry β a γ mají v praxi zhruba tyto hodnoty: $\beta = 10^{-4}$ a $\gamma = 10$). Po určení prahu každé výstupní jednotky proběhne druhá kompetice, která však probíhá vzhledem k hodnotě:

$$||\mathbf{x}_i - \mathbf{w}_l|| - b_i\,. \tag{4.9}$$

Vítězná jednotka potom změní své váhy podle obvyklého vzorce (4.6).

4.2 Kohonenovy samoorganizační mapy

Poněkud složitějším modelem samoorganizační sítě je *Kohonenova samoorganizační mapa* (*Self-Organizing Map*) navržená autorem v roce 1982. Vychází částečně z principů popsaných v pracích von der Malsburga a Amariho [186, 10].

Organizační dynamika sítě je podobná jednoduché samoorganizační síti popsané v odstavci 4.1.2. Výstupní jednotky jsou však navíc uspořádány do nějaké topologické struktury, nejčastěji to bývá dvojrozměrná mřížka nebo jednorozměrná řada jednotek. Tato topologická struktura určuje, které

se středem v c, parametrem $h_0 \in \mathbb{R}$ určujícím maximální míru posunu jednotek, a šířkou $\sigma \in \mathbb{R}$. Jak h_0 tak i σ se přitom s časem zmenšují. Tento přístup je při implementaci časově náročnější, neboť se v každém kroku musejí projít a změnit všechny váhové vektory $\mathbf{w}_j$ v síti.

V [156] uvádí autor některé praktické rady pro volbu parametrů učení Kohonenovy sítě. Podle těchto zkušeností není například způsob inicializace jednotek podstatný, zaručí-li se, že jejich hodnoty jsou vesměs různé. Počet kroků algoritmu je obtížné přesně stanovit, ale měl by být alespoň pětsetkrát větší, než je počet jednotek v síti. Typické hodnoty jsou 10 až 100 tisíc iterací. Většinou v průběhu učení rozeznáváme dvě fáze: hrubé učení, které je krátké (typicky 1000 kroků) a vyznačuje se velkými hodnotami parametrů θ a s. V této fázi dojde ke globálnímu rozmístění jednotek, které jsou potom ve druhé — dolaďovací — části učení relativně málo měněny. Parametr θ by během první fáze měl typicky klesnout z hodnoty blízké 1 (např. 0,9) až na 0,01 a v průběhu druhé fáze dále klesat k nule. Stejně tak velikost okolí s může být zpočátku velká (srovnatelná s velikostí sítě), na konci první fáze dosáhnout hodnoty 1 a postupně v průběhu druhé fáze klesnout až na 0. Experimenty ukazují, že není podstatné, zda parametry θ a s klesají lineárně, exponenciálně, či jinak.

4.3 LVQ

Prozatím jsme uvažovali využití Kohonenovy mapy pro učení bez učitele. Nyní se budeme zabývat tím, jak lze tuto síť použít pro řešení problému *klasifikace* dat do několika kategorií. Ukážeme si způsob, kterým označíme výstupní neurony sítě kategoriemi a uvedeme tři algoritmy *učící vektorové kvantizace*, které se používají pro doučení sítě, jež chceme použít k těmto účelům.

Uvažujme tedy data tvaru $\{(\mathbf{x}^{(t)}, d^{(t)}); t = 1, \ldots, k\}$, kde $\mathbf{x}^{(t)} \in \mathbb{R}$ a $d^{(t)} \in \{C_1, \ldots, C_q\}$. Každý vstupní vektor $\mathbf{x}^{(t)}$ má přiřazenu jednu z konečného počtu kategorií C_k. Učení Kohonenovy sítě bude mít nyní tři fáze:

1. Učení bez učitele dle (4.11).
2. Označení výstupních neuronů kategoriemi.
3. Doučení sítě jedním z algoritmů LVQ.

Nejprve tedy použijeme standardní učící algoritmus Kohonenovy sítě, kterým rozmístíme neurony sítě do vstupního prostoru tak, aby aproximovaly hustotu pravděpodobnosti vzorů. Prozatím jsme nepoužili požadované výstupy $d^{(t)}$ z tréninkové množiny. Ty využijeme ve druhé části učení, kdy procházíme tréninkovou množinu a u každého tréninkového vzoru zjistíme,

jednotky spolu v síti sousedí, a je důležitá v učícím procesu, jak uvidíme dále.

Pro učící proces je důležité zavést pojem *okolí* $N_s(c)$ velikosti s neuronu c v síti, což je množina všech neuronů, jejichž vzdálenost v síti od neuronu c je menší nebo rovna s:

$$N_s(c) = \{j; d(j,c) \leq s\}\,. \tag{4.10}$$

To, jak měříme vzdálenost $d(j,c)$, je závislé právě na topologické struktuře výstupních neuronů.

V **aktivním** režimu síť pracuje stejným způsobem jako předchozí model, sousednost neuronů se zde tedy neprojevuje. Předložíme-li síti vstupní vektor, soutěží výstupní jednotky o to, kdo je mu nejblíže, a tato jednotka se pak excituje, má výstupní hodnotu rovnu jedné, zatímco výstupy ostatních jednotek jsou rovny nule.

Adaptivní dynamika sítě již bere do úvahy uspořádání neuronů a pracuje tak, že namísto samotné vítězné jednotky se upravují váhy i všech neuronů, které jsou v jejím okolí. To znamená, že spolu s vítězným neuronem se posunují i jeho sousedé v síti, kteří by od něj neměli být příliš vzdáleni ani ve vstupním prostoru. Velikost okolí přitom není konstantní, na začátku je okolí obvykle velké (například polovina velikosti sítě) a na konci učení potom zahrnuje jen samotný vítězný neuron.

Celý učící algoritmus je analogický algoritmu z odstavce 4.1.2, pouze adaptace vah se neřídí rovnicí (4.6), ale má tvar:

$$w_{ji}^{(t)} = \begin{cases} w_{ji}^{(t-1)} + \theta(x_i^{(t)} - w_{ji}^{(t-1)}) & j \in N_s(c) \\ w_{ji}^{(t-1)} & \text{jinak,} \end{cases} \tag{4.11}$$

kde $c = \arg\min_{l=1,\ldots,h}\{||\mathbf{x}^{(t)} - \mathbf{w}_l||\}$ a $\theta \in \mathbb{R}$ a $s \in \mathbb{N}$ jsou popsané parametry učení.

Předchozí vztah lze ještě zobecnit. Definujme funkci $h_c(j)$, která pro neurony z okolí neuronu c dává hodnotu θ a pro ostatní neurony je nulová:

$$h_c(j) = \begin{cases} \theta & j \in N_s(c) \\ 0 & \text{jinak.} \end{cases} \tag{4.12}$$

Předpis (4.11) lze pak jednodušeji zapsat jako:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + h_c(j)(x_i^{(t)} - w_{ji}^{(t-1)})\,. \tag{4.13}$$

Funkce $h_c(j)$ je často definována obecněji, aby přechod mezi nulovými a nenulovými hodnotami byl spojitý, což více odpovídá biologickým interakcím. Typicky používanou funkcí je Gaussova funkce

$$h_c(j) = h_0 \exp\left(\frac{-d(j,c)^2}{\sigma^2}\right) \tag{4.14}$$

který neuron je mu nejblíže. U něho si pak zapamatujeme, do jaké kategorie tento vzor patřil. Tak se po průchodu tréninkovou množinou u každého výstupního neuronu vytvoří tabulka četností jednotlivých kategorií, které tento neuron reprezentuje. Jelikož musíme každému neuronu j přiřadit právě jednu kategorii, vybereme tu, kterou reprezentoval nejčastěji a označíme ji v_j. Výsledkem je tedy rozdělení neuronů do skupin, které odpovídají jednotlivým kategoriím.

V další fázi použijeme jeden ze tří následujících algoritmů pro doladění vah výstupních neuronů, které byly postupně navrženy Kohonenem v osmdesátých letech ([155, 157]).

4.3.1 LVQ1

První varianta algoritmu učící vektorové kvantizace vychází z myšlenky posílit správnou klasifikaci vstřícným posunutím neuronu a naopak, snažit se napravit nesprávnou klasifikaci odsunutím neuronu od daného vstupu. Posunutí se tedy týká vždy jen jednoho výstupního neuronu — toho, který zvítězil v kompetici. Navíc se posunutí děje o malou poměrnou část vzdálenosti neuronu od vstupního vzoru.

Přesněji, algoritmus LVQ1 pracuje takto: Předkládáme postupně síti všechny tréninkové vzory. Ke vzoru $(\mathbf{x}^{(t)}, d^{(t)})$ nejprve známým způsobem určíme nejbližší neuron c:

$$c = \arg \min_{l=,1\ldots,h} \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \,. \tag{4.15}$$

Potom provedeme úpravu vah u tohoto neuronu (ostatní neurony zůstávají beze změny):

$$\mathbf{w}_c^{(t)} = \begin{cases} \mathbf{w}_c^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_c^{(t-1)}) & d^{(t)} = v_c \\ \mathbf{w}_c^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_c^{(t-1)}) & d^{(t)} \neq v_c \,. \end{cases} \tag{4.16}$$

Parametr $\alpha \in \mathbb{R}$ by měl mít dle praktických zkušeností počáteční hodnotu poměrně malou, asi 0,01 až 0,02, a během několika desítek až set tisíc iterací klesnout k nule.

Ukazuje se, že hranice, která se vytvoří mezi třídami pomocí algoritmu LVQ1, je poměrně věrnou aproximací tzv. *bayesovské rozhodovací hranice* známé ze statistiky. Bayesovské rozhodovací pravidlo určuje, do které třídy bod připadne podle jeho pozice vzhledem k místu kde se setkávají distribuce vzorů dvou tříd. Je intuitivně vidět, že LVQ1 posunuje reprezentanty směrem od rozhodovací hranice. Na rozdíl od bayesovské hranice se ale rozhodovací hranice vytvořená algoritmem LVQ1 nachází uprostřed spojnice mezi dvěma neurony pocházejícími z různých tříd.

4.3.2 LVQ2

Algoritmus LVQ2 je motivován snahou upravit předchozí algoritmus tak, aby explicitně posunoval svou rozhodovací hranici směrem k bayesovské hranici. Algoritmus v jednom kroku posune vždy dva neurony. Podmínky pro jejich určení jsou poměrně přísné: předložíme síti vzor $\mathbf{x}^{(t)}$ a uvažujeme případ, kdy pro dva neurony nejbližší tomuto vzoru (označme je $\mathbf{w}_i$ a $\mathbf{w}_j$), platí, že jeden z nich klasifikuje správně a druhý špatně. Nerozlišujeme přitom, který z neuronů $\mathbf{w}_i$ a $\mathbf{w}_j$ je vzoru nejblíže.

Další podmínkou pro změnu vah těchto dvou vybraných neuronů je, aby vzor $\mathbf{x}^{(t)}$ neležel příliš blízko ani jednoho z nich. Chceme, aby se $\mathbf{x}^{(t)}$ nacházelo v *okně*, které je definováno v okolí nadroviny umístěné ve středu spojnice $\mathbf{w}_i$ a $\mathbf{w}_j$. Přesněji, říkáme, že $\mathbf{x}^{(t)}$ padne do okna relativní šířky q, pokud:

$$\min\left\{\frac{d_i}{d_j}, \frac{d_j}{d_i}\right\} > s \,, \tag{4.17}$$

kde $s = \frac{1-q}{1+q}, d_i = d(\mathbf{w}_i, \mathbf{x}^{(t)}), d_j = d(\mathbf{w}_j, \mathbf{x}^{(t)})$. Volba správné hodnoty parametru q je kompromisem mezi snahou o co nejužší okno, které umožní přesné umístění hranice, a dostatečnou šířku, která zachytí statisticky významné množství dat. Typicky se používají hodnoty q mezi 0.1 až 0.3 (tedy deset až třicet procent vzdálenosti mezi $\mathbf{w}_i$ a $\mathbf{w}_j$).

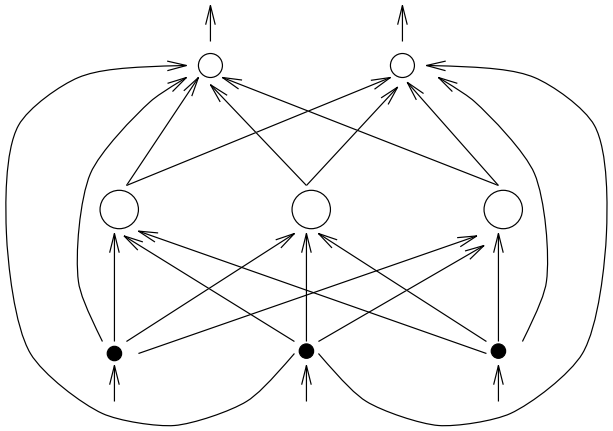
Předpokládejme bez újmy na obecnosti, že například $\mathbf{x}^{(t)}$ a $\mathbf{w}_j$ patří do stejné kategorie. Pak provedeme následující změny vah:

$$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_i^{(t-1)}) \tag{4.18}$$

$$\mathbf{w}_j^{(t)} = \mathbf{w}_j^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t-1)}) \,. \tag{4.19}$$

Praktickými pokusy bylo zjištěno, že algoritmus zpočátku skutečně zlepšuje pozici rozhodovací hranice tím, že ji posunuje směrem k bayesovskému rozhraní. Po větším množství kroků však dochází k tomu, že se jednotky $\mathbf{w}_i$ ze vzorce (4.18) naopak od tohoto rozhraní začínají vzdalovat. Proto se osvědčilo používat algoritmus LVQ2 pouze po kratší počet iterací, typicky kolem 10 000.

Poznámka: Předcházející postup je mírnou modifikací původního algoritmu LVQ2 a bývá autorem přesněji nazýván LVQ2.1. Původně navržený algoritmus LVQ2 pracoval pouze v případě, že jednotka i, označená špatnou kategorií, byla nejbližší vzoru $\mathbf{x}^{(t)}$. Jelikož je posun jednotky úměrný její vzdálenosti od tréninkového vzoru, byla vždy jednotka $\mathbf{w}_j$ změněna více než $\mathbf{w}_i$. To mělo za následek, že se zmenšovala relativní vzdálenost $d(\mathbf{w}_i, \mathbf{w}_j)$,



Obr. 4.1: Síť typu counterpropagation

vstupních neuronů distribuujících do další vrstvy vstupní signály $x_1 \ldots, x_n$. Druhá vrstva je tvořena N samoorganizačními jednotkami (viz odstavec 4.1.2), jejichž váhy označíme $\mathbf{w}_j$ a výstupy z_j. Třetí vrstvu tvoří m Grossbergových jednotek instar s vstupními vahami $\mathbf{u}_r$ $(r = 1, \ldots, m)$. Výstupy jednotek y_r pak tvoří výstup celé sítě.

Síť ve své **aktivní fázi** realizuje zobrazení $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$. Po předložení vstupu $\mathbf{x}$ proběhne kompetice mezi jednotkami druhé vrstvy a jejich výstupy jsou potom určeny následující rovnicí:

$$z_j = \begin{cases} 1 & j = \arg \min_{l=1,\ldots,N} \{||\mathbf{x} - \mathbf{w}_l||\} \\ 0 & \text{jinak.} \end{cases} \tag{4.21}$$

Právě jeden z výstupů z_j samoorganizačních jednotek je tedy roven 1, ostatní mají hodnotu 0. Jednotky třetí vrstvy potom spočítají své výstupy podle následujícího vzorce:

$$y_r = \sum_{j=1}^{N} u_{rj} z_j \,. \tag{4.22}$$

Přechodová funkce výstupního neuronu vybere ze svých vstupů do neuronu ten, který má jediný hodnotu 1, a výstupu pak přiřadí hodnotu odpovídající váhy u_{rj}.

Adaptivní dynamika této sítě sestává ze dvou fází. Nejprve se Kohonenovým učením bez učitele nastaví váhy $\mathbf{w}_j$ samoorganizačních jednotek ve druhé vrstvě. Autor navíc používá mechanismus lokální paměti jedno-

což je vlastnost, které chceme zamezit, protože jednotky i a j reprezentují různé kategorie.

4.3.3 LVQ3

Jak jsme uvedli výše, není ani vylepšená verze LVQ2.1 předchozího algoritmu stabilní — při větším počtu kroků se jednotky $\mathbf{w}_i$ vzdalují ze svých optimálních pozic. Proto byl algoritmus LVQ3 doplněn o další pravidlo, kterým se zajistí, že správně klasifikující neurony se budou navíc pohybovat směrem k předkládanému tréninkovému vzoru (o menší poměrnou část jejich vzdálenosti od vzoru).

Jeden krok LVQ3 vypadá tedy takto:

$$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_i^{(t-1)})$$
$$\mathbf{w}_j^{(t)} = \mathbf{w}_j^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t-1)}) \,,$$

kde i a j je pár výstupních neuronů, jež jsou nejblíže k $\mathbf{x}^{(t)}$, $v_j = d^{(t)}$, $v_i \neq d^{(t)}$ a $\mathbf{x}^{(t)}$ patří do okna dle (4.17). Navíc:

$$\mathbf{w}_r^{(t)} = \mathbf{w}_r^{(t-1)} + \varepsilon\alpha(\mathbf{x}^{(t)} - \mathbf{w}_r^{(t-1)}) \,, \tag{4.20}$$

pro $r \in \{i, j\}$ a $v_i = v_j = d^{(t)}$. Pokusy bylo zjištěno, že hodnota reálného parametru ε závisí na šířce okna (pro užší okna je menší) a měla by se nastavit v rozmezí 0,1 až 0,5. Na rozdíl od parametru $0 < \alpha < 1$, je ε konstantní v čase.

4.4 Síť typu counterpropagation

Nyní uvedeme model umělé neuronové sítě, která se snaží využít samoorganizační síť v kombinaci s dalším přídavným mechanismem k řešení problémů učení s učitelem. V tomto případě jde o síť *counterpropagation*, kterou navrhl Hecht-Nielsen v roce 1986 [103]. Tato síť kombinuje výše popsanou jednoduchou samoorganizační síť (viz odstavec 4.1.2) s jednotkami typu *instar* popsanými Grossbergem např. v [89]. Síť, kterou v dalším popíšeme, je jednou z více variant a vylepšení navržených autorem, jde konkrétně o dopřednou síť typu counterpropagation. Zájemce o podrobnější popis dalších modifikací odkazujeme na [105, 103]. Síť counterpropagation pracuje jako vyhledávací tabulka (lookup table), která k danému vstupu najde nejbližšího reprezentanta a odpoví výstupní hodnotou, která je s tímto reprezentantem spojena.

Zaměřme se nejprve na **organizační dynamiku** tohoto modelu. Síť counterpropagation sestává ze tří vrstev (viz obr. 4.1). Vstupní vrstvu tvoří n

tek a dvojí kompetice popsaný v odstavci 4.1.3, který zajistí, že výsledně rozmístěné jednotky mají stejnou pravděpodobnost výběru.

Po skončení první fáze učení, ve které se používá jen vstupních částí tréninkových vzorů, se váhy $\mathbf{w}_j$ fixují a dojde ke druhé fázi, která nastaví váhy $\mathbf{u}_r$ výstupních jednotek. V této části učícího algoritmu spočítá každá jednotka r svůj aktuální výstup y_r podle rovnice (4.21) a upraví své váhy $\mathbf{u}_r$ podle tzv. *Grossbergova učícího pravidla*:

$$u_{rj}^{(t)} = u_{rj}^{(t-1)} + \eta(-u_{rj}^{(t-1)} + d_r^{(t)})z_j \,, \tag{4.23}$$

kde z_j jsou výstupy jednotek z druhé vrstvy po předložení tréninkového vzoru $\mathbf{x}^{(t)}$, $d_r^{(t)}$ je požadovaná odezva r-té jednotky sítě a $0 < \eta < 1$ je parametr učení.

Je vidět, že ve vzorci (4.23) je u každé jednotky r modifikována pouze ta váha u_{rj}, jež je spojena s vítěznou jednotkou z předcházející vrstvy (pro níž je $z_j = 1$). V průběhu učení je váha upravována tak, aby její hodnota odpovídala průměru těch z hodnot d_r, které náležejí do oblasti excitace jednotky j ve druhé vrstvě. Takže po dostatečné době, kdykoliv ve druhé vrstvě zvítězí v kompetici jednotka j, bude výstup sítě tvořit vektor $\mathbf{v}_j = (u_{1j}, \ldots, u_{mj})$, jehož hodnota je přibližně průměrem požadovaných odpovědí $\mathbf{y}$ odpovídajících těm vstupům $\mathbf{x}^{(t)}$, které způsobily, že jednotka j zvítězila v kompetici.

Naučenou síť si lze představit jako vyhledávací tabulku, kde vstupní vektor $\mathbf{x}$ je porovnán se všemi vektory $\mathbf{w}$, je nalezen jemu nejbližší vektor $\mathbf{w}_c$ a jako výstupní hodnota se použije odpovídající vektor $\mathbf{v}_c$, jenž je průměrem výstupů patřících do oblasti kolem $\mathbf{w}_c$.

Shrňme nyní statistické vlastnosti naučené sítě. Za prvé, díky samoorganizačnímu učení s využitím lokální paměti aproximují vektory $\mathbf{w}_j$ hustotu pravděpodobnosti vzorů. Víme, že jednotky ve druhé vrstvě mají stejnou pravděpodobnost vítězství v kompetici, za předpokladu, že vybíráme vstupy náhodně s rozložením odpovídajícím tréninkové množině. Dále, váhy výstupních jednotek jsou adaptovány tak, aby aproximovaly průměrnou výstupní hodnotu patřící těm vstupům, které aktivovaly odpovídající jednotky ve druhé vrstvě.

Používáme-li tedy síť counterpropagation k aproximování nějakého zobrazení $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, chová se optimálně v tom smyslu, že reprezentanti vstupů jsou zvoleni tak, aby měli stejnou pravděpodobnost výběru, a výstupní hodnoty představují průměr funkčních hodnot v okolí těchto reprezentantů.

Ovšem, to, že síť pracuje právě jako vyhledávací tabulka, ukazuje i její slabiny. Typicky je pro dosažení určité přesnosti potřeba mnohem více jednotek, než např. u vícevrstvého perceptronu. Z experimentů však vyplývá, že učení je typicky výrazně kratší než u algoritmu zpětného šíření. Hecht-

Nielsen v [105] uvádí případy, kdy je použití sítě counterpropagation výhodné:

- Je-li potřeba poznat, kdy nastane konec učící fáze. První fáze učení končí, jsou-li četnosti výběru jednotek druhé vrstvy přibližně stejné. Druhá fáze musí nastavit váhy výstupních jednotek na průměry odpovídajících výstupních hodnot.
- Když má problém strukturu vhodnou pro využití vyhledávací tabulky. Například není-li výstupních hodnot mnoho.
- Je-li důležité, aby chování sítě bylo jednoduše predikovatelné a bezpečné. Je-li např. nutné zabránit, aby se objevily nějaké výrazně odlišné výstupy sítě.
- Chceme-li využít rychlého učení sítě counterpropagation i přes menší přesnost. Například, je-li síť částí nějakého většího systému zpracování dat, můžeme v etapě vytváření prototypu řešení použít counterpropagation, otestovat rámcové chování a v závěru nahradit jinou neuronovou sítí umožňující dosáhnout větší přesnosti.

Kapitola 5

Sítě s lokálními neurony

V této kapitole se zabýváme modely dopředných sítí s jednou skrytou vrstvou obsahující tzv. *lokální jednotky*. Tyto jednotky se chovají v jistém smyslu duálně k perceptronům. Zatímco perceptron globálně rozděluje vstupní prostor na dva podprostory, v nichž má výrazně odlišné hodnoty výstupu, lokální jednotky mají relevantní výstup lokalizován do okolí bodu určeného svými parametry.

Tyto modely umělých neuronových sítí jsou výrazně mladší než perceptrony, objevují se až koncem osmdesátých let a jsou i méně rozšířené. Teoretické i praktické výsledky však ukazují, že svými vlastnostmi představují důstojnou alternativu klasickým modelům. Díky své výrazné odlišnosti se často dobře uplatní v případech, kdy perceptronové sítě mohou mít potíže. Nejvýznamější představitel této kategorie — RBF sítě — o nichž budeme také hovořit, zvládly například učení některých typických testovacích úloh o dva až tři řády rychleji. Bohužel, ukazuje se, že naopak existují úlohy, na něž tyto sítě nejsou vhodné. Jedním z problémů, s nímž si perceptronové sítě typicky poradí lépe, jsou irelevantní vstupy.

Budeme se podrobně zabývat sítěmi typu RBF a sítěmi se semi-lokálními jednotkami. Zvláště RBF sítě jsou zajímavé tím, že jejich učení může probíhat v několika odlišných fázích, v nichž se využijí i jiné neuronové postupy, jako je například samoorganizace.

5.1 Sítě typu RBF

Sítě typu RBF, nebo — jak budeme zkráceně uvádět — *RBF sítě*, představují vícevrstvé neuronové sítě s jednotkami odlišného typu, než jsou například dříve popsané perceptrony. Také motivace jejich zavedení je jiná než u předchozích typů neuronových sítí. Začátkem osmdesátých let se v části nume-

rické matematiky zabývající se interpolací a aproximací dat začaly studovat tzv. *radiální bazické funkce (Radial Basis Functions)* jako jeden z nových způsobů řešení aproximačních problémů.

Hledáme-li funkci, která bude nejlépe (v nějakém smyslu, například dle kritéria nejmenších čtverců) aproximovat daná data, omezujeme se většinou na funkce vyjádřené v nějakém konkrétním tvaru. Často jde o lineární kombinaci tzv. bazických funkcí, což mohou být například polynomy, nebo v tomto případě *radiální funkce*. Radiální funkci si můžeme představit jako funkci určenou nějakým významným bodem — středem — která pro argumenty se stejnou vzdáleností od tohoto středu dává stejné funkční hodnoty. Měříme-li vzdálenost pomocí euklidovské metriky a uvažujeme například dvojrozměrný vstupní prostor, pak množiny se stejnou funkční hodnotou tvoří kružnice, proto tedy hovoříme o radiálních funkcích.

Základní výsledky týkající se aproximace pomocí radiálních bazických funkcí pocházejí od Powella [230, 231] a Micchelliho [194]. První, kdo navrhl využití této techniky pro vytvoření nového modelu umělých neuronových sítí, byli Broomhead a Lowe ve svém článku [42] z roku 1988. Další podstatný podíl na rozvoji RBF sítí měly práce Moodyho a Darkena [198] a Poggia a Girosiho [227].

5.1.1 Motivace

RBF síť si můžeme představit jako třívrstvou neuronovou síť, kde vstupní vrstva neuronů slouží pouze k přenosu vstupních hodnot. Druhá (skrytá) vrstva sestává z tzv. *RBF jednotek*, které realizují jednotlivé radiální funkce. Třetí, výstupní vrstva je lineární.

Při pohledu zvnějšku je RBF jednotka podobná perceptronu — má také n reálných vstupů $\mathbf{x} = (x_1, \ldots, x_n)$, z nichž každý má přiřazen parametr (váhu) c_i. RBF jednotka má také jeden reálný výstup y a může mít další parametr b, kterému budeme říkat *šířka*. Ovšem přechodová funkce RBF jednotky je odlišná: vnitřní potenciál ξ se nepočítá jako skalární součin $\mathbf{w} \cdot \mathbf{x}$, ale je to vzdálenost vstupního vektoru $\mathbf{x}$ od středu $\mathbf{c}$ (případně ještě dělená šířkou b)

$$\xi = \frac{||\mathbf{x} - \mathbf{c}||}{b} .$$

Výstupní hodnotu y získáme aplikováním aktivační funkce φ na potenciál ξ:

$$y = \varphi(\xi) .$$

Často používané aktivační funkce ukazuje tabulka 5.1.

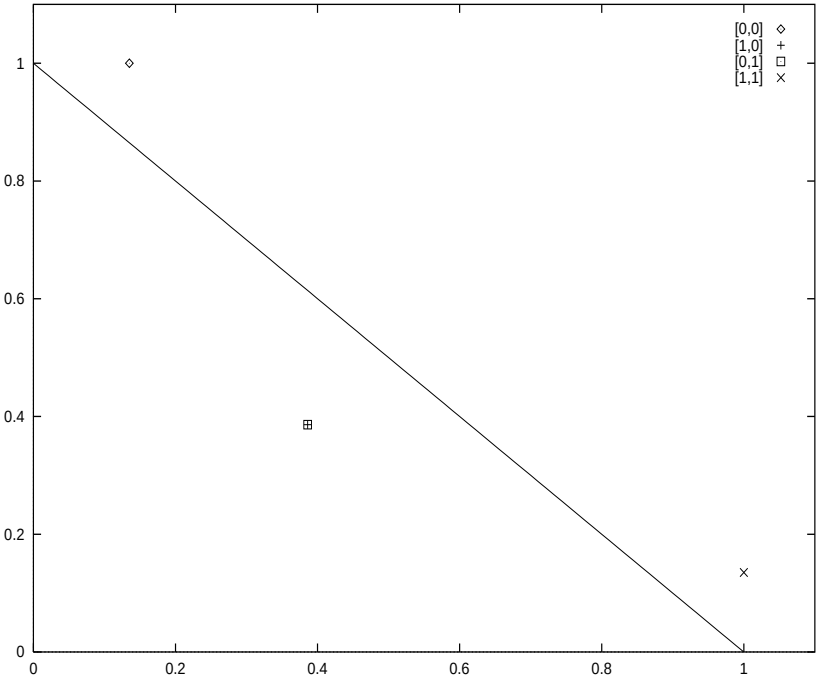
Vstupní vrstva RBF sítě obsahuje n jednotek, odpovídajících n-rozměrným vstupním hodnotám. Každý spoj mezi i-tou vstupní jednotkou a j-tou jednotkou ve skryté vrstvě má přiřazenu váhu c_{ji} $(j = 1, \ldots, h)$. Tato

$\{\{[1,1],[0,0]\},\{[0,1],[1,0]\}\}$. Definujme nyní vhodně vektorové zobrazení $\mathbf{g}: \mathbb{R}^2 \to \mathbb{R}^2$ (povšimneme si, že obor hodnot má stejnou dimenzi jako definiční obor). Zobrazení $\mathbf{g}$ je dvojice vzájemně posunutých Gaussových funkcí:

$$g_1(\mathbf{x}) = e^{-||\mathbf{x}-\mathbf{c}_1||^2} \qquad \mathbf{c}_1 = [1,1]$$

$$g_2(\mathbf{x}) = e^{-||\mathbf{x}-\mathbf{c}_2||^2} \qquad \mathbf{c}_2 = [0,0]\,.$$

Z obrázku 5.1 je zřejmé, že transformovaná dichotomie $\mathbf{g}(\mathbf{x})$ je lineárně separovatelná.



Obr. 5.1: Lineárně separabilní $\mathbf{g}$ obrazy dichotomie XOR.

Předchozí příklad nám ukázal geometrický význam funkce jedné RBF jednotky s gaussovskou aktivační funkcí (případně obdobnými funkcemi). Těmto jednotkám se někdy říká *lokální* (anglicky *local units* nebo *localized receptive fields*), neboť jejich výstupní hodnota je významná jen v určitém okolí středu $\mathbf{c}_i$. Tato vlastnost je duální k perceptronu, který se chová globálně tím, že rozdělí svůj vstupní prostor nadrovinou na dva podprostory, v nichž má odlišné hodnoty výstupu.

$\varphi(z) = z$	*lineární*
$\varphi(z) = z^2 \log x$	
$\varphi(z) = (z^2+\beta)^\alpha, \beta \geq 0, 0 < \alpha < 1$	
$\varphi(z) = e^{(-\frac{z}{\beta})^2}, \beta \geq 0$	*Gaussova*

Tabulka 5.1: Aktivační funkce RBF sítí

váha reprezentuje i-tou souřadnici středu $\mathbf{c}_j$ u j-té RBF jednotky. Výstup j-té jednotky ze skryté vrstvy je spojen s výstupní vrstvou pomocí synapse s vahou w_{js}. Výstupní jednotky počítají váženou sumu svých vstupů.

RBF síť provádí dvě odlišné transformace. První z nich je realizována RBF jednotkami a je to nelineární transformace z $\mathbb{R}^n$ do $\mathbb{R}^h$. Druhá transformace vede z tohoto prostoru skrytých jednotek do výstupního prostoru $\mathbb{R}^m$, je lineární a zprostředkovaná výstupními neurony sítě.

Empirické výsledky ukazují, že je dobré volit počet skrytých jednotek větší než je dimenze vstupu. To odpovídá i teoretickému výsledku publikovanému v polovině šedesátých let Coverem [51], z něhož plyne, že problém klasifikace, který je transformován do vícedimenzionálního prostoru, zde bude spíše lineárním, než v původním prostoru.

Přesněji, mějme dichotomii $\{C_1, C_2\}$ podmnožiny $X \subseteq \mathbb{R}^n$ a vektorovou funkci $\mathbf{g} = (g_1, \ldots, g_h)$ na X. Pro každý vektor $\mathbf{x} \in X$ je $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_h(\mathbf{x}))$. Dichotomii $\{C_1, C_2\}$ říkáme $\mathbf{g}$-*separovatelná*, existuje-li vektor $\mathbf{u} = (u_1, \ldots, u_h)$ takový, že:

$$\mathbf{u} \cdot \mathbf{g}(\mathbf{x}) \geq 0 \qquad \mathbf{x} \in C_1$$

$$\mathbf{u} \cdot \mathbf{g}(\mathbf{x}) < 0 \qquad \mathbf{x} \in C_2$$

(tedy dichotomie, která je indukovaná na obrazech $\mathbf{g}(\mathbf{x})$, je lineárně separovatelná.)

Coverova věta zhruba říká, že pravděpodobnost, že náhodná dichotomie množiny N bodů z $\mathbb{R}^n$ je $\mathbf{g}$-separovatelná, je dána výrazem:

$$p(N,h) = \left(\frac{1}{2}\right)^{N-1} \sum_{m=0}^{h-1} \binom{N-1}{m}.$$

Jinými slovy, čím větší je h, tím blíže má pravděpodobnost $p(N,h)$ k jedničce.

Příklad: Někdy stačí i samotná nelinearita zobrazení $\mathbf{g}$ k tomu, aby se původní lineárně neseparovatelný problém změnil na lineárně separabilní. Uvažme známý problém logické funkce XOR, který je vlastně dichotomií

Shrňme, že co se týče propojení jednotek, je RBF síť obvyklou dopřednou sítí s jednou skrytou vrstvou. Někdy dokonce můžeme u perceptronu i RBF jednotky použít stejnou aktivační funkci (např. Gaussovu), přesto je chování perceptronu a RBF jednotky odlišné. RBF jednotky nám svojí lokalitou více připomenou některé jiné modely umělých neuronových sítí, jako jsou např. samoorganizační sítě.

5.1.2 Interpolace a aproximace

Radiální bazické funkce se studují přibližně od začátku osmdesátých let jako jedna z nových metod v oblasti numerické matematiky zabývající se interpolací a aproximací dat. Uveďme nyní přesnější formulaci těchto problémů a ukažme si, jak je lze pomocí RBF řešit.

Definice 5.1 *Problém přesné* interpolace *reálné funkce více proměnných lze formulovat následujícím způsobem:*

Je-li dána množina h různých vektorů $\{\mathbf{x}_i;\ i=1,\dots,h\}$ v $\mathbb{R}^n$ a h reálných čísel $\{d_i; i=1,\dots,h\}$, nalezněte funkci f splňující

$$f(\mathbf{x}_i) = d_i \quad \forall i = 1,\dots,h\,. \tag{5.1}$$

Různé interpolační metody se liší v tom, jaké stanovují další podmínky pro množinu funkcí, do které má f patřit. U RBF zavádíme množinu *h radiálních bazických funkcí* $\varphi_j : \mathbb{R}^n \to \mathbb{R}$, $j=1,\dots,h$, tvaru $\varphi_j = \varphi(||\mathbf{x}-\mathbf{c}_j||)$, kde $\varphi : \mathbb{R}^+ \to \mathbb{R}, \mathbf{c}_i \in \mathbb{R}^n$ jsou středy těchto funkcí a $||\cdot||$ značí normu na $\mathbb{R}^n$ (obvykle euklidovskou). Funkce $\{\varphi_j; j=1,\dots,h\}$ tvoří lineární funkční prostor. Interpolační funkci f potom hledáme jako prvek tohoto prostoru, neboli lineární kombinaci radiálně bazických funkcí:

$$f(\mathbf{x}) = \sum_{j=1}^{h} w_j \varphi(||\mathbf{x}-\mathbf{c}_j||)\,. \tag{5.2}$$

Dosazením vzorce (5.2) do podmínek (5.1) dostaneme následující soustavu lineárních rovnic s neznámými koeficienty w_j:

$$\begin{pmatrix} d_1 \\ \vdots \\ d_h \end{pmatrix} = \begin{pmatrix} \varphi_{11} & \cdots & \varphi_{1h} \\ \vdots & \ddots & \vdots \\ \varphi_{h1} & \cdots & \varphi_{hh} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_h \end{pmatrix} \tag{5.3}$$

neboli

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}\,,$$

kde

$$\mathbf{\Phi} = (\varphi_{ij})_{i,j} = \varphi(||\,\mathbf{x}_i - \mathbf{c}_j\,||) \quad i,j = 1,\dots,h\,.$$

V případě, že je matice $\mathbf{\Phi}$ regulární, nalezneme snadno jednoznačné řešení soustavy:

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{d}\,. \tag{5.4}$$

Pro obecné funkce φ může být matice $\mathbf{\Phi}$ samozřejmě singulární. Z výsledků Micchelliho a Powella ([194],[231]) ale plyne, že velká třída funkcí — včetně těch, jež jsou uvedeny v tabulce 5.1 — zaručuje, že matice $\mathbf{\Phi}$ je regulární (jsou-li body $\mathbf{x}_i$ navzájem různé).

Interpolační schéma radiálních bazických funkcí lze snadno rozšířit i na vektorové funkce $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, kde $\mathbf{f} = (f_1,\dots,f_m)$, $\mathbf{d} = (d_1,\dots,d_m)$. Podmínka (5.1) je pak zobecněna na množinu podmínek:

$$f_s(x_i) = d_{is} \quad \forall i = 1,\dots,h \quad \forall s = 1,\dots,m\,. \tag{5.5}$$

Takže f_s má tvar:

$$f_s(\mathbf{x}) = \sum_{j=1}^{h} w_{js}\varphi(||\mathbf{x}-\mathbf{c}_j||)\,, \tag{5.6}$$

kde $\mathbf{x} \in \mathbb{R}^n$ a $s = 1,\dots,m$. Koeficienty w_{js} lineární kombinace získáme stejným způsobem jako v (5.4).

Postup, který jsme právě uvedli, má nevýhodu v tom, že počet funkcí φ_i se musí rovnat počtu dat, která aproximujeme. V typických úlohách se ale předpokládá, že dat je mnohem více než volných parametrů úlohy. Takto formulováno jde o problém *aproximace*:

Definice 5.2 *Mějme množinu $\mathbf{X}$, funkci $f(\mathbf{x})$ definovanou na $\mathbf{X}$ a množinu aproximujících funkcí $\{\varphi(\mathbf{v},\mathbf{x}); \mathbf{v} \in \mathbf{P}\}$ které spojitě závisejí na parametrech $\mathbf{v} \in \mathbf{P}$. Dále mějme na množině funkcí na $\mathbf{X}$ zavedenu metriku ρ. Nalezněte hodnotu $\mathbf{v}^\star \in \mathbf{P}$ takovou, že*

$$\rho(\varphi(\mathbf{v}^\star,\mathbf{x}), f(\mathbf{x})) \le \rho(\varphi(\mathbf{v},\mathbf{x}), f(\mathbf{x}))$$

platí pro všechny $\mathbf{v} \in \mathbf{P}$.

Metrika ρ je většinou indukována normou, jako je např. $\mathcal{L}_p$ norma nebo supremová norma (viz kapitola 13). Parametry $\mathbf{v}$ v našem případě reprezentují souřadnice středů $\mathbf{c}_i$, váhy $\mathbf{w}_j$, případně další parametry.

Uvažujme k zadaných dat $\{(\mathbf{x}^{(1)},\mathbf{d}^{(1)}),\dots,(\mathbf{x}^{(k)},\mathbf{d}^{(k)})\}$ a h funkcí φ_j tak, že platí $h < k$. Prvním problémem v tomto případě je určení pozic středů $\mathbf{c}_i$ $(i = 1,\dots,h)$. Ty se většinou umístí buď do pozic některých dat, nebo se využije sofistikovanějších postupů. Podrobněji o tom budeme hovořit v dalším odstavci. Nyní předpokládejme, že jsme již středy nějakým

způsobem umístili a soustřeďme se na koeficienty $\mathbf{w} = \{w_1, \ldots, w_h\}$ lineární kombinace.

V tomto případě, je-li $h < k$, hovoříme o tom, že problém je přespecifikovaný. Matice $\mathbf{\Phi}$ již není čtvercová a neexistuje k ní inverzní matice. Řešení se hledá pomocí lineární optimalizace nějakou z metod řešení nejmenších čtverců. Broomhead a Lowe v [42] používají Moore-Penroseovu pseudoinverzi $\mathbf{\Phi}^+$ (viz 3.17). Připomeňme, že tato matice má vlastnost, že $\mathbf{\Phi}^+\mathbf{\Phi} = \mathbf{E}_h$, kde $\mathbf{E}_h$ jednotková matice $h \times h$. Navíc, řešení $\mathbf{w}$ získané jako $\mathbf{w} = \mathbf{\Phi}^+\mathbf{d}$ má nejmenší normu $||\mathbf{w}||$ mezi všemi vektory $\mathbf{v}$, které minimalizují výraz $||\mathbf{\Phi}^+\mathbf{v} - \mathbf{d}||^2$.

5.1.3 Třífázové učení

Tím, že aproximační schéma popsané v předchozí podkapitole přeložíme do jazyka neuronových sítí, získáme nejen nový úhel pohledu na stejnou věc, ale umožní nám to obohatit původní algoritmus o další „neuronové“ prvky. Jedním z nich bude například využití samoorganizace pro nastavení pozic středů $\mathbf{c}_j$, o němž se zmíníme dále.

Jelikož formálně je síť RBF druhem dopředné sítě, stejně jako například vícevrstvý perceptron, můžeme k jejímu učení použít modifikaci algoritmu zpětného šíření. Analogicky jako u perceptronové sítě lze vyjádřit parciální derivace chybové funkce vzhledem ke všem parametrům sítě a pak je postupně počítat. Jelikož RBF sítě mají jen jednu skrytou vrstvu, je pro ně gradientní algoritmus výrazně jednodušší než pro sítě s více skrytými vrstvami.

Algoritmus zpětného šíření zachází se všemi parametry sítě stejným způsobem. V případě této architektury mají ale různé skupiny parametrů různý význam vzhledem k funkci sítě. To bylo motivací k vytvoření učícího algoritmu, který sestává ze tří fází, z nichž v každé se určují hodnoty jiné skupiny parametrů [198, 114]. V průběhu učení postupujeme v síti vzhůru a postupně se zabýváme souřadnicemi středů jednotek, jejich šířkami a naposledy koeficienty lineární kombinace. Celý algoritmus lze popsat takto (viz tab. 5.2):

Tréninkovou množinu učení RBF sítě tvoří páry vektorů $\{(\mathbf{x}^{(t)}, \mathbf{d}^{(t)}); t = 1, \ldots, k\}$ sestávající ze vstupů $\mathbf{x}^{(t)} \in \mathbb{R}^n$ a požadovaných výstupů $\mathbf{d}^{(t)} \in \mathbb{R}^m$.

První krok algoritmu určuje pozice středů RBF jednotek, jež jsou reprezentovány vahami mezi vstupní a skrytou vrstvou. Jelikož v této fázi jde o aproximaci hustoty výskytu vzorů, používají se různé techniky samoorganizačního učení, které jsme popsali v předchozí kapitole.

Druhá fáze nastavuje hodnoty případných dalších parametrů u RBF jednotek. Obvykle (ne však nutně) mají jednotky nastavitelný parametr, který určuje šířku oblasti kolem středu, v níž má jednotka relevantní výstup.

Učení RBF sítě:

Vstup:	Tréninková množina $\{(\mathbf{x}^{(t)}, \mathbf{d}^{(t)}), t = 1, \ldots, k\}$.
Výstup:	Středy $\mathbf{c}_j$,
	šířky b_j,
	váhy w_{sj}.

1. Spočítej středy $\mathbf{c}_j$ pomocí $\mathbf{x}^{(t)}$.
2. Spočítej šířky b_j.
3. Spočítej váhy w_{sj} pomocí $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$.

Tabulka 5.2: Tři části učení RBF sítě

Třetí krok je obvyklé učení s učitelem, které adaptuje váhy mezi skrytou výstupní vrstvou. Jelikož jde o koeficienty lineární kombinace, je učení lineární a není také nutné šířit chybu do nižších vrstev, protože jejich váhy jsou už fixovány. Toto učení je tedy mnohem jednodušší než například u standardního zpětného šíření. Je však pravda, že jeho úspěch závisí na předchozích krocích.

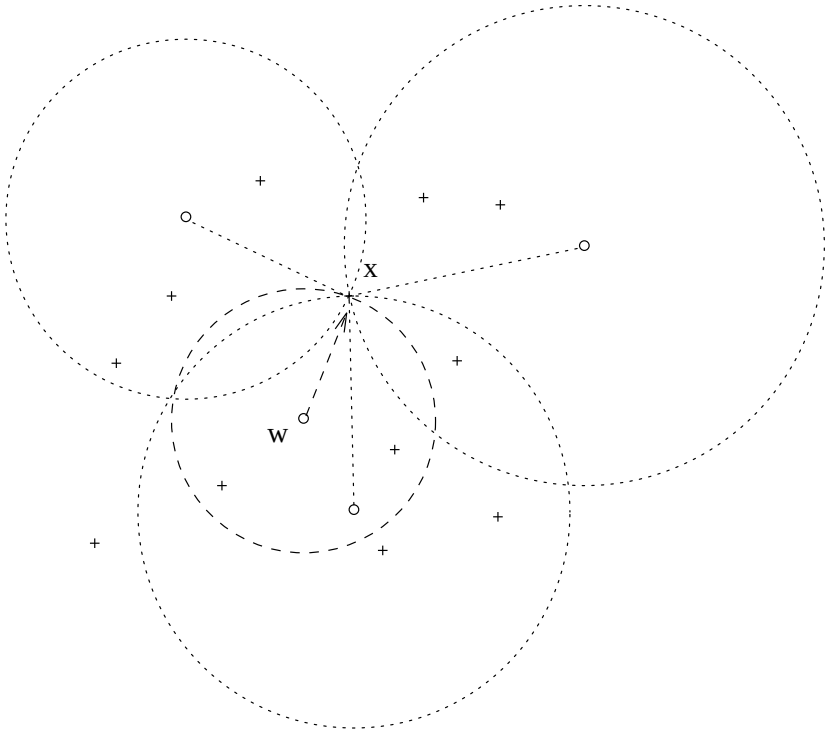
Zaměřme se nyní podrobněji na jednotlivé fáze učení RBF sítí.

Učení bez učitele

Úkolem první fáze učení je tedy umístit h středů RBF jednotek, tj. určit hodnoty vah $\{c_{ji}; i = 1, \ldots, n;\ j = 1, \ldots, h\}$ mezi vstupní a skrytou vrstvou. Existuje řada různých přístupů, jak tento problém řešit. Uveďme si nyní některé, od těch triviálních až po náročnější.

Rovnoměrné rozložení V praktických úlohách často autoři používají co nejjednodušší postup, který zkrátí celkovou dobu učení a dává uspokojivé výsledky. První z takových metod je rovnoměrné pravidelné rozmístění středů jednotek po vstupním prostoru. To je vhodné v případě, že vstupní data jsou také rozmístěna přibližně rovnoměrně. Takové příklady typicky nastávají jsou-li tréninkové vzory výsledkem nějakého pravidelného měření či vzorkování.

Náhodné vzorky Další jednoduchou alternativou je vybrání h náhodných tréninkových vzorů a umístění středů RBF jednotek na jejich vstupní části. Jde také o velmi rychlý postup, který, na rozdíl od předchozího, již v jistém smyslu mapuje rozložení dat.



Obr. 5.2: Jeden krok při nastavování středů

kterou označíme E_2, má tvar:

$$E_2(b_1,\ldots,b_k) = \frac{1}{2}\sum_{r=1}^{h}\left[\sum_{s=1}^{h} e^{-\left(\frac{||c_s-c_r||}{b_r}\right)^2}\left(\frac{||c_s-c_r||}{b_r}\right)^2 - P\right]^2 . \qquad (5.8)$$

Parametr P určuje míru překrývání oblastí patřících jednotlivým jednotkám. V praxi se autoři většinou vyhýbají obecné minimalizaci této chybové funkce a používají jednodušší heuristiky. Často se šířka nastavuje úměrně průměru (euklidovských) vzdáleností q nejbližších sousedů dané jednotky. Hodnota q se přitom často pokládá rovna jedné. Je důležité, že vzorec (5.8) nezávisí na tréninkových vzorech, ale pouze na rozmístění středů $\mathbf{c}_j$. Takže ani při důsledné optimalizaci E_2 nemusíme síti ve druhé fázi učení předkládat tréninkové vzory.

Optimální vzorky Autoři Chen, Cowan a Grant [135] navrhli způsob, který také vybírá některé z tréninkových vzorů a přiřazuje pak jejich souřadnice středům. Jejich výběr není náhodný, ale používá se metoda ortogonálních nejmenších čtverců k tomu, aby se minimalizovala chyba sítě.

Samoorganizace Vyžadujeme-li, aby středy RBF jednotek lépe zachytily rozmístění tréninkových vzorů, musíme použít složitějších algoritmů. Z formulace problému je zřejmé, že jde o učení bez učitele, kde můžeme použít různé samoorganizační algoritmy popsané v předchozí kapitole. Ať již použijeme jakýkoliv z nich, vždy potřebujeme k učení jen vstupní části tréninkových vzorů. Nejčastěji používaným algoritmem je jednoduché samoorganizační učení (k-means clustering, viz. [198, 199] a 4.1.2), které v tomto konkrétním případě vypadá takto (tabulka 5.3, obr 5.2):

(i) rozmísti $\mathbf{c}_j$ náhodně po vstupním prostoru
(ii) v čase t dělej:
  (a) nejdi střed $\mathbf{c}_c$, který je nejblíže vstupu $\mathbf{x}^{(t)}$
  (b) posuň $\mathbf{c}_c$ k $\mathbf{x}^{(t)}$ podle:
$$\mathbf{c}_c := \mathbf{c}_c + \theta(t)||\mathbf{x}^{(t)} - \mathbf{c}_c||,$$
  kde $0 < \theta(t) < 1$ je parametr učení.

Tabulka 5.3: Nastavení středů RBF sítě učením bez učitele

Druhá fáze učení slouží k nastavení dalších parametrů RBF jednotek, existují-li, či jsou-li adaptovány. (Použijeme-li například rovnoměrné rozmístění středů po vstupním prostoru, nemá adaptace šířek valný smysl.) Soustřeďme se nyní na nejčastěji používané Gaussovy radiální bazické funkce, jež mají tvar:

$$\varphi(\mathbf{x}) = e^{-\left(\frac{||\mathbf{x}-\mathbf{c}||}{b}\right)^2} . \qquad (5.7)$$

Parametr b reprezentuje šířku φ a určuje tak velikost radiální oblasti kolem středu $\mathbf{c}$, v němž má daná RBF jednotka významné výstupní hodnoty. Parametry určující šířku kontrolované oblasti mají vliv na generalizační schopnosti sítě. Čím jsou menší, tím horší generalizaci lze očekávat. Na druhou stranu, při příliš širokém okolí ztrácejí jednotky svůj lokální charakter.

Obecně můžeme vyjádřit chybovou funkci závislou na parametrech b určujících šířku, a pak ji vzhledem k těmto parametrům minimalizovat. Chyba,

5.1.4 Regularizace

Obvykle po naučené neuronové síti žádáme nejen, aby věrně aproximovala zadané tréninkové vzory, ale také, aby dávala „rozumné“ odpovědi na vstupy, které nebyly obsaženy v tréninkové množině. Tato vlastnost, jíž se říká generalizace, je předmětem různých kontroverzních definic i diskusí. My se spokojíme s velmi jednoduchým pojetím generalizace — budeme požadovat, aby zobrazení realizované sítí bylo dostatečně hladké. To může být problém, jsouli například tréninkové vzory ovlivněné šumem. Obecné metody pro ovlivnění hladkosti aproximujícího zobrazení přináší teorie regularizace [227], kterou můžeme použít ve třetí fázi učícího procesu.

Principem regularizace je, že k chybě E_3 přidáme další člen E_R, který má za úkol penalizovat funkce s nechtěnými vlastnostmi, jako je třeba velká oscilace. Člen E_R tak reprezentuje naše implicitní předpoklady o typu aproximující funkce. Výsledná chybová funkce má tedy tvar:

$$E_3'(\mathbf{w}) = E_3 + \gamma E_R \,. \tag{5.12}$$

Reálný parametr γ určuje poměr mezi E_3 a E_R. Jeho správné nastavení je důležité, ale bohužel také závislé na problému, který řešíme. Obecně lze říci, že při malé hodnotě γ ztrácí generalizační člen smysl a velká γ způsobí, že zobrazení bude sice např. velmi hladké, ale nebude již věrně reprezentovat data.

Jedním z konkrétních příkladů regularizačního členu je součet druhých parciálních derivací:

$$E_R(\mathbf{w}) = \frac{1}{2}\sum_{t=1}^{k}\sum_{s=1}^{m}\sum_{i=1}^{n}\left(\frac{\partial^2 y_s^{(t)}}{\partial x_i^2}\right)^2 . \tag{5.13}$$

Kromě toho, že E_R penalizuje funkce s velkými druhými derivacemi, má další výhodnou vlastnost: je bilineární vzhledem k vahám w_{qr}, takže výsledná optimalizace zůstává lineární.

Dosazením (5.13) do (5.12) a položením prvních derivací E_3' rovných nule spočteme rovnice pro w_{qr} stejně jako v (5.10). Výsledek je:

$$w_{qr} = \sum_{j=1}^{h}(\mathbf{\Psi}^+)_{jr}\left[\sum_{t=1}^{k}\varphi_j(\mathbf{x}^{(t)})y_s^{(t)}\right] , \tag{5.14}$$

kde $\mathbf{\Psi}$ má tvar:

$$(\mathbf{\Psi})_{jr} = \sum_{t=1}^{k}\left[\varphi_j(\mathbf{x}^{(t)})\varphi_r(\mathbf{x}^{(t)}) + \gamma\sum_{i=1}^{n}\frac{\partial^2\varphi_j(\mathbf{x}^{(t)})}{\partial x_i^2}\frac{\partial^2\varphi_r(\mathbf{x}^{(t)})}{\partial x_i^2}\right] . \tag{5.15}$$

Učení s učitelem

Po naučení vah c_{ji}, případně dalších parametrů RBF jednotek, zbývá nastavit váhy w_{sr} lineární kombinace. To zajistíme minimalizací obvyklé chybové funkce:

$$E_3(\mathbf{w}) = \frac{1}{2}\sum_{t=1}^{k}||\mathbf{d}^{(t)} - \mathbf{y}^{(t)}||^2 = \frac{1}{2}\sum_{t=1}^{k}\sum_{s=1}^{m}(d_s^{(t)} - y_s^{(t)})^2 , \tag{5.9}$$

kde $\mathbf{y}^{(t)}$ (respektive $y_s^{(t)}$) označuje aktuální výstup sítě (resp. její s-té výstupní jednotky) po předložení vstupního vektoru $\mathbf{x}^{(t)}, \mathbf{w} = (w)_{qr}$, $q = 1, \ldots, m$ a $r = 1, \ldots, h$.

Položením parciálních derivací E_3 podle w_{qr} rovných nule získáme následující vztah pro w_{qr}:

$$w_{qr} = \sum_{j=1}^{h}(\mathbf{\Phi}^+)_{jr}\left[\sum_{t=1}^{k}\varphi_j(\mathbf{x}^{(t)})d_s^{(t)}\right] , \tag{5.10}$$

kde

$$(\mathbf{\Phi})_{\mathbf{jr}} = \sum_{\mathbf{t=1}}^{\mathbf{k}}\varphi_{\mathbf{j}}(\mathbf{x^{(t)}})\varphi_{\mathbf{r}}(\mathbf{x^{(t)}}) \tag{5.11}$$

a $\mathbf{\Phi}^+$ je Moore-Penroseova pseudoinverze (srovnej (3.17)).

Silným nástrojem pro řešení problému lineárních nejmenších čtverců je *SVD (Singular Value Decomposition) rozklad*. Tato metoda je obecně charakterizována následující větou:

Věta 5.3 *Buď $\mathbf{A} \in \mathbb{C}^{m,n}$ matice řádu h. Pak existují unitární matice $\mathbf{U} \in \mathbb{C}^{m,m}$ a $\mathbf{V} \in \mathbb{R}^{n,n}$ takové, že*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T} \qquad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_h & 0 \\ 0 & 0 \end{pmatrix} ,$$

kde $\mathbf{\Sigma} \in \mathbb{R}^{m,n}, \mathbf{\Sigma}_h = diag(\sigma_1, \sigma_2, \ldots, \sigma_h)$ a

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_h > 0 \,.$$

Čísla σ_i se nazývají singulární hodnoty $\mathbf{A}$.

Uvažujme náš problém minimalizace E_3 a vyjádřeme si $\mathbf{\Phi}$ jako $\mathbf{\Phi} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$. Potom plyne, že řešení nejmenších čtverců, které má nejmenší normu, získáme jako $\mathbf{x} = \mathbf{\Phi}^+\mathbf{w}$, kde

$$\mathbf{\Phi}^+ = \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}_h^{-1} & 0 \\ 0 & 0 \end{pmatrix}\mathbf{U}^\mathsf{T} \in \mathbb{R}^{n,m}$$

je pseudoinverze $\mathbf{A}$.

počítat jakousi disjunkci podmínek z jednotlivých dimenzí. Konkrétně, přechodovou funkcí $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ této jednotky je vážená suma jednorozměrných Gaussových funkcí:

$$\psi(x_1, \ldots, x_n) = \sum_{j=1}^{n} w_j e^{-\left(\frac{x_j - c_j}{\sigma_j}\right)^2}. \tag{5.17}$$

Celá síť počítá funkci $\mathbf{y} = \mathbf{f}(\mathbf{x})$, $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$y_i = \sum_{j=1}^{n} w_{ij} e^{-\left(\frac{x_{ij} - c_{ij}}{\sigma_{ij}}\right)^2}.$$

Adaptivní dynamika semi-lokálních sítí je určena jednoduchým gradientním algoritmem. Jako obvykle počítáme chybu sítě E a pak vyjádříme parciální derivace chyby podle všech parametrů. Jelikož máme síť pouze s jednou skrytou vrstvou, jsou vzorce pro výpočet derivací přímočaré:

$$-\frac{\partial E}{\partial w_{ij}} = \delta_i e^{-\left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2} \tag{5.18}$$

$$-\frac{\partial E}{\partial c_{ij}} = \delta_i w_{ij} \frac{1}{\sigma_{ij}} \left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right) e^{-\left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2} \tag{5.19}$$

$$-\frac{\partial E}{\partial \sigma_{ij}} = \delta_i w_{ij} \frac{1}{\sigma_{ij}} \left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2 e^{-\left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2}, \tag{5.20}$$

kde $\delta_i = -\frac{\partial E}{\partial y_i}$.

Semi-lokální jednotky vznikly motivovány snahou skloubit výhody perceptronů a RBF sítí. V praxi se ukazuje, že RBF sítě se učí mnohem rychleji, než vícevrstvé perceptrony, ale jejich nevýhodou je, že mají většinou horší generalizační vlastnosti a nedovedou si dobře poradit s irelevantními vstupními dimenzemi. V několika praktických aplikacích skutečně semi-lokální jednotky ukázaly velmi dobré výsledky, ale z hlediska aproximačních vlastností se ukazuje, že jsou slabším výpočetním prostředkem, než perceptronové i RBF sítě (viz podkapitolu 13.5).

Regularizační techniku použil například Bishop v [36] při řešení problému aproximace sinusoidy na základě dat ovlivněných náhodnou chybou. Autor použil stejné množství RBF jednotek jako tréninkových vzorů a jelikož data byla generována ve stejných intervalech, bylo i rozložení středů a šířky Gaussových funkcí uniformní.

Poggio a Girosi [227] využili obecné regularizační schéma k odvození tzv. *HyperBF sítí*, které mají obecné Greenovy funkce jako své aktivační funkce ve skrytých jednotkách. RBF sítě jsou potom speciálním případem HyperBF funkcí.

5.2 Sítě se semi-lokálními jednotkami

Již víme, že RBF jednotky jsou typicky lokální, reprezentují omezenou oblast vstupního prostoru kolem svého středu. Naproti tomu perceptron se sigmoidální aktivační funkcí provádí globální rozdělení vstupního prostoru na dva podprostory pomocí nadroviny. Hartman a Keeler se v [95] pokusili vytvořit architekturu sítě, která by byla jakýmsi spojením těchto dvou diametrálně odlišných koncepcí. Svůj model anglicky nazvali, patrně inspirováni vzhledem aktivační funkce, *Gaussian bars*. My budeme hovořit o semi-lokálních jednotkách.

Existuje několik variant sítí se semi-lokálními jednotkami. Nejjednodušším modelem je síť se vstupní vrstvou a vrstvou semi-lokálních jednotek, jejichž výstupy tvoří výstupy celé sítě. Další variantou je obdoba RBF sítě se skrytou vrstvou obsahující semi-lokální jednotky a linární výstupní vrstvou. Ještě další možností, kterou autoři použili v aplikacích, je síť se dvěma vrstvami semi-lokálních jednotek. V následujícím se omezíme na nejjednodušší variantu, neboť nám postačí k popisu odlišností od předchozího modelu. Aktivní i adaptivní dynamiku obdoby RBF sítě se semi-lokálními jednotkami si na základě těchto dvou podkapitol dokážeme snadno odvodit.

Abychom objasnili, jak pracuje jedna semi-lokální jednotka, podívejme se nejprve na obvyklou vícerozměrnou Gaussovu funkci z jiného pohledu. Gaussovu funkci s n-rozměrným vstupem $\mathbf{x}$ si lze představit jako součin jednorozměrných Gaussových funkcí:

$$\gamma(\mathbf{x}) = \exp\left(-\left(\frac{\sum_{i=1}^{n}(x_i - c_i)^2}{b^2}\right)\right) = \prod_{i=1}^{n} \gamma(x_i). \tag{5.16}$$

Každou z jednorozměrných Gaussových funkcí $\gamma(x_i)$ si lze představit jako lokální podmínku určující hodnotu v dané dimenzi i. Gaussovská RBF jednotka potom počítá logický součin těchto podmínek pro jednotlivé dimenze. Idea Hartmana a Keelera byla nahradit tuto konjunkci požadující, aby podmínky platili zároveň, logickým součtem. Semi-lokální jednotka bude tedy

Část II

Složitost neuronových sítí

Druhá část knihy formalizuje některé architektury neuronových sítí jako výpočetní modely a zabývá se jejich deskriptivní složitostí a výpočetní sílou. Z hlediska teorie složitosti jsou architektury neuronových sítí dalšími zajímavými výpočetními modely srovnatelnými s konečnými automaty, booleovskými obvody, Turingovými stroji apod. Neuronové sítě mají navíc schopnost učit se z příkladů, proto jsou poslední dvě kapitoly v této části věnovány složitosti problému učení a generalizace i efektivitě učících heuristik. Dosažené teoretické výsledky naznačují výpočetní možnosti neuronových sítí a potvrzují praktické zkušenosti s jejich aplikací (např. časová náročnost učení apod.). Je zřejmé, že efektivita modelů neuronových sítí je základním předpokladem jejich praktické použitelnosti v umělé inteligenci. Vzhledem k tomu, že modely neuronových sítí byly inspirovány nervovými systémy živých organismů, které vykonávají příslušné funkce efektivně, tento přístup by mohl vést ke složitostní definici inteligence: způsob *efektivní* reprezentace znalostí. Efektivitu můžeme chápat ve trojím smyslu: efektivní vytváření a adaptace této reprezentace (složitost učení), její paměťovou náročnost (deskriptivní složitost) a efektivní vybavování znalostí (výpočetní síla). Teorie složitosti může nejen verifikovat oprávněnost výpočetních modelů neuronových sítí, ale obohacuje i náš pohled na umělou inteligenci.

Šestá kapitola formalizuje funkci perceptronu, tj. tzv. lineární prahovou funkci, a zabývá se její reprezentací pomocí vah a prahu. V případě reálného definičního oboru zkoumá jednoznačnost této reprezentace. Pro omezenou doménu ukazuje existenci celočíselné reprezentace pro tzv. separabilní lineární prahovou funkci a v konečné doméně odhaduje její velikost. V případě booleovského (binárního) definičního oboru se hovoří o tzv. booleovských prahových funkcích. Je odhadnut jejich počet a optimální velikost jejich reprezentace. Navíc je ukázáno, že rozhodnout, zda daná booleovská funkce v normální formě je prahová, je obecně těžký problém.

V sedmé kapitole se zabýváme složitostí obvodů, které modelují acyklické neuronové sítě. První podkapitola je věnována logickým obvodům, kdy je funkce neuronu omezena jen na konjunkci a disjunkci. Výpočetní síla normálního tvaru logických obvodů (tzv. alternující obvody) je ilustrována při implementaci obecné booleovské funkce a součtu celých čísel. Pozornost je také věnována speciální podtřídě logických obvodů, tzv. klasickým obvodům, u nichž každé hradlo má nejvýše dva vstupy. Ukazuje se, že paralelní výpočet acyklické neuronové sítě je díky hustotě propojení nepatrně rychlejší než u obvodů klasických počítačů. Pro porovnání s Turingovskými výpočty se dále uvažují (uniformní) posloupnosti obvodů, které pro obvody polynomiální velikosti a polylogaritmické hloubky definují třídy složitosti a jejich možné hierarchie. V druhé podkapitole jsou uvedené výsledky zobecněny pro prahové obvody, které modelují acyklické neuronové sítě s perceptrony. Ukazuje, se že prahové obvody polynomiální velikosti s konstantním po-

čtem vrstev jsou silným výpočetním prostředkem, např. počítají symetrické booleovské funkce nebo aproximují analytické funkce apod. Také uvažujeme analogové prahové obvody se standardní sigmoidní aktivační funkcí. Nakonec studujeme možné hierarchie tříd složitosti pro prahové obvody. Je dokázáno, že pro polynomiální počet neuronů jsou třívrstvé neuronové sítě silnější než dvouvrstvé.

Osmá kapitola je věnována cyklickým neuronovým sítím. Nejprve je definován formální model, ukázána ekvivalence jeho variant (asynchronní výpočet, fixované vstupy) a prezentována konstrukce prahového obvodu ekvivalentního s konvergentní cyklickou sítí. Dále se zabýváme otázkou zastavení obecné neuronové sítě, jehož algoritmická predikce je těžký problém. Významnou podtřídou cyklických sítí jsou Hopfieldovy sítě se symetrickými spoji, jejichž výpočet se za velmi obecných předpokladů zastaví pro libovolný počáteční stav. Je dokázán exponenciální dolní odhad pro čas výpočtu symetrické sítě v nejhorším případě. Navíc konvergentní výpočet obecné cyklické sítě lze efektivně simulovat pomocí Hopfieldovy sítě, což dokumentuje jejich stejnou výpočetní sílu. V závěru je pak ukázáno, že pravděpodobně neexistuje efektivní algoritmus pro určení počtu stabilních stavů, velikosti oblastí atrakce a pro minimalizaci energie v Hopfieldově síti.

V deváté kapitole se zabýváme pravděpodobnostními neuronovými sítěmi, které modelují nedeterministické a nespolehlivé chování neuronů. U tzv. pravděpodobnostních prahových obvodů s přidanými nedeterministické vstupy, které jsou s danými pravděpodobnostmi aktivní, lze pravděpodobnost chyby výstupu obvodu libovolně zmenšit, příp. je možno stejnou funkci počítat pomocí deterministického prahového obvodu. V analogii s deterministickými obvody se definují posloupnosti pravděpodobnostních prahových obvodů s rostoucí pravděpodobností chyby a odpovídající hierarchie tříd složitosti pro konstantní hloubku obvodů. Je ukázáno, že pravděpodobnostní posloupnosti prahových obvodů polynomiální velikosti a hloubky 2, s malými váhami mohou počítat více funkcí než posloupnosti prahových obvodů stejných parametrů. Zavádíme také tzv. Boltzmannovy obvody, jejichž definice je bližší modelům používaným v praktických aplikacích, i když mají v podstatě stejnou výpočetní sílu a deskriptivní složitost jako pravděpodobnostní prahové obvody. Nakonec se zabýváme robustními neuronovými sítěmi, které počítají danou funkci spolehlivě s nespolehlivými neurony.

V desáté kapitole studujeme výpočetní sílu (obecně cyklických) neuronových sítí tak, že je využíváme jako akceptory jazyků. Konečné diskrétní neuronové sítě (tzv. neuromaty) rozpoznávají právě regulární jazyky. Ukazuje se, že neuromaty mají v jistém smyslu větší deskriptivní sílu než regulární výrazy. Speciálně se zabýváme neuronovými akceptory binárních řetězců a charakterizujeme třídu tzv. Hopfieldových jazyků, které jsou rozpoznávány symetrickými neuromaty a tvoří vlastní podtřídu regulárních jazyků. Dále

studujeme výpočetní sílu konečných analogových neuronových sítí, která závisí na deskriptivní složitosti váhových parametrů. Pro racionální váhy lze docílit reálnou simulaci Turingova stroje. V případě reálných vah polynomiální výpočty neuronové sítě odpovídají třídě (neuniformní) složitosti $P/poly$ a v exponenciálním čase lze rozpoznat již všechny jazyky. Existuje nekonečná hierarchie tříd jazyků mezi P a $P/poly$, které jsou akceptovány sítěmi s rostoucí kolmogorovskou složitostí vah. Také uvažujeme nekonečné (neuniformní) posloupnosti diskrétních cyklických (popř. symetrických) sítí polynomiální velikosti, které rozpoznávají jazyky z třídy $PSPACE/poly$.

Jedenáctá kapitola je věnována složitosti učení neuronových sítí. Nejprve je definován tzv. tréninkový problém a diskutována jeho relevance pro učení acyklických neuronových sítí. Je ukázáno, že tréninkový problém je obecně těžký problém nezávisle na volbě neuronové funkce. Z důkazu je zřejmé, že ani mnohá silná omezení architektur či tréninkových množin nepomohou odstranit nepříznivou složitost tohoto problému. Dále jsou studovány tzv. mělké architektury, pro které je za velmi omezujících předpokladů navržen polynomiální učící algoritmus. Jeho praktická použitelnost závisí na složitosti tréninkového problému pro hluboké architektury, který je však také těžký. Původní důkaz složitosti tréninkového problému je založen na nepravidelnosti architektury sítě. Proto je uveden i alternativní důkaz pro dvouvrstvou perceptronovou síť s regulární architekturou, speciálně jen se třemi neurony, a pro kaskádovou architekturu dokonce jen se dvěma neurony. Tento výsledek je zobecněn pro standardní sigmoidní aktivační funkci neuronu, kterou využívá učící algoritmus backpropagation, což implikuje jeho neefektivitu. Kapitolu uzavírá poznámka o složitosti učení cyklických neuronových sítí.

Ve dvanácté kapitole se zabýváme generalizačními vlastnostmi neuronových sítí. Za tímto účelem využíváme známý PAC-model, který je základním rámcem obecné výpočetní teorie učení. Učící algoritmus v tomto modelu by měl na základě příkladů neznámého cílového konceptu s neznámým pevným rozdělením pravděpodobností určit v polynomiálním čase hypotézu z dané třídy konceptů, která by se s velkou pravděpodobností shodovala s cílovým konceptem v uvedené distribuci. Nejprve studujeme tzv. vzorkovou složitost, tj. počet tréninkových vzorů nutných ke správné generalizaci, pomocí tzv. Vapnik-Chervonenkisovy dimenze. Určením její hodnoty pro třídy konceptů reprezentované acyklickými neuronovými sítěmi pro ně získáme polynomiální horní a dolní odhad vzorkové složitosti. Dále charakterizujeme PAC-naučitelnost pomocí polynomiální VC-dimenze a existence polynomiálního pravděpodobnostního algoritmu pro tréninkový problém. Jako důsledek ukážeme, že mnohé architektury (analogových) acyklických neuronových sítí nejsou za obecně přijímaných předpokladů z teorie složitosti PAC-naučitelné. V závěru se zabýváme analýzou perceptronového učícího

algoritmu, který obecně není efektivní, i když pomocí lineárního programování je jeden perceptron PAC-naučitelný.

Poznatky v této části knihy jsou vybrány ze zahraničních monografií, resp. z přehledových prací o složitosti neuronových sítí, které jsou doplněny o nejnovější výsledky dostupné jen v původních článcích. Při její kompozici jsme vycházeli zejména z prací [66, 100, 143, 210, 212, 213, 241, 268, 290].

Nechť naopak platí (6.2) a počítáme

$$f(x_1,\ldots,x_n) - f(x_1,\ldots,x_{i-1},x',x_{i+1},\ldots,x_n) =$$

$$= \sum_{j=1}^{n} w_j x_j - \sum_{j=1}^{i-1} w_j x_j - w_i x' - \sum_{j=i+1}^{n} w_j x_j = w_i(x_i - x')\,.$$

□

Definice 6.3 *Funkce* $f : \mathbb{R}^n \longrightarrow \{0,1\}$ *se nazývá* lineární prahová funkce, *jestliže existuje lineární funkce* $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ *taková, že pro všechna* $\mathbf{x} = (x_1,\ldots,x_n) \in \mathbb{R}^n$ *platí* $f(\mathbf{x}) = 1$, *právě když* $g(\mathbf{x}) \geq 0$, *tj. jestliže existují tzv.* váhy $w_1,\ldots,w_n \in \mathbb{R}$ *takové, že pro všechna* $x_1,\ldots,x_n \in \mathbb{R}$ *platí:*

$$f(\mathbf{x}) = 1\,, \quad \textit{právě když} \quad \sum_{i=1}^{n} w_i x_i \geq -g(0,\ldots,0)\,. \tag{6.3}$$

Označme $h = -g(0,\ldots,0)$ práh *a* $(w_1,\ldots,w_n;h)$ reprezentaci f.

Ze vztahu (6.3) je zřejmé, že reprezentace určuje jednoznačně lineární prahovou funkci. Na druhou stranu, pro danou lineární prahovou funkci existuje nekonečně mnoho reprezentací, které se však, jak uvidíme, liší jen o násobek. Pro tento účel budeme ještě předtím charakterizovat reprezentace lineární prahové funkce, jejíž funkční hodnota je nezávislá na nějaké její proměnné.

Definice 6.4 *Řekneme, že lineární prahová funkce* $f : \mathbb{R}^n \longrightarrow \{0,1\}$ *je* degenerovaná v i-té proměnné, *jestliže pro všechna* $x_1,\ldots,x_n \in \mathbb{R}$ *a* $x' \in \mathbb{R}$ *platí* $f(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},x',x_{i+1},\ldots,x_n)$.

Věta 6.5 *Lineární prahová funkce* f *je degenerovaná v* i*-té proměnné, právě když pro všechny reprezentace* f *platí* $w_i = 0$.

Důkaz: Nechť f je degenerovaná v i-té proměnné a $(w_1,\ldots,w_n;h)$ je její reprezentace. Navíc nejprve předpokládejme, že existuje vstup $x_1,\ldots,x_n$, pro který je $f(x_1,\ldots,x_n) = 0$. Pro spor dále předpokládejme, že $w_i \neq 0$. Položme

$$x' = \frac{h - \sum_{j=1}^{i-1} w_j x_j - \sum_{j=i+1}^{n} w_j x_j}{w_i} \tag{6.4}$$

a počítejme $\sum_{j=1}^{i-1} w_j x_j + w_i x' + \sum_{j=i+1}^{n} w_j x_j = h$. To znamená, že $f(x_1,\ldots,x_{i-1},x',x_{i+1},\ldots,x_n) = 1$, což je spor s degenerovaností f, tedy $w_i = 0$. Pro $f \equiv 1$ je důkaz podobný (čitatel x' v (6.4) volíme např. o 1 menší). Obrácená implikace je triviální. □

Kapitola 6

Lineární prahová funkce

6.1 Reálná doména

V této podkapitole budeme formalizovat funkci formálního neuronu (1.3), tj. perceptronu s reálnými vstupy (viz podkapitolu 2.1) a s aktivační funkcí ostrou nelinearitou (1.7). Budeme ji nazývat *lineární prahová funkce*. Nejprve si však připomeneme definici lineární funkce více proměnných a jako jednoduché cvičení ukážeme, že ji lze zapsat pomocí afinní kombinace těchto proměnných.

Definice 6.1 *Reálná funkce* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ *s* n *reálnými proměnnými* $x_1,\ldots,x_n$ *se nazývá* lineární, *jestliže je lineární ve všech svých proměnných, tj. existují reálné koeficienty* $w_1,\ldots,w_n$ *takové, že pro každé* $x_1,\ldots,x_n \in \mathbb{R}$, $x' \in \mathbb{R}$ *a* $1 \leq i \leq n$ *platí:*

$$f(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},x',x_{i+1},\ldots,x_n) + w_i(x_i - x')\,. \tag{6.1}$$

Věta 6.2 *Funkce* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ *je lineární, právě když existují reálné koeficienty* $w_1,\ldots,w_n$ *takové, že pro všechna* $x_1,\ldots,x_n \in \mathbb{R}$ *platí:*

$$f(x_1,\ldots,x_n) = f(0,\ldots,0) + \sum_{i=1}^{n} w_i x_i\,. \tag{6.2}$$

Důkaz: Předpokládejme nejprve, že f je lineární. Ukážeme indukcí dle n, že platí (6.2). Pro $n = 1$ položme $x' = 0$ v (6.1) a dostáváme $f(x_1) = f(0) + w_1 x_1$. Dále předpokládejme, že tvrzení platí pro $n-1$. Definujeme lineární funkci $f_0(x_1,\ldots,x_{n-1}) = f(x_1,\ldots,x_{n-1},0)$. Víme, že f je lineární, tedy využijeme (6.1) pro $x' = 0$ a $i = n$ a dostáváme $f(x_1,\ldots,x_n) = f_0(x_1,\ldots,x_{n-1}) + w_n x_n$, což lze dle indukčního předpokladu pro f_0 přepsat $f(x_1,\ldots,x_n) = f_0(0,\ldots,0) + \sum_{i=1}^{n-1} w_i x_i + w_n x_n = f(0,\ldots,0) + \sum_{i=1}^{n} w_i x_i$.

Lemma 6.7 *Pro každou lineární prahovou funkci $f : \mathbb{R}^n \longrightarrow \{0,1\}$ existuje lineární prahová funkce $g : \mathbb{R}^{n+1} \longrightarrow \{0,1\}$ s nulovým prahem taková, že pro všechna $x_1, \ldots, x_n \in \mathbb{R}$ platí $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n, 1)$.*

Důkaz: Nechť f je lineární prahová funkce s reprezentací $(w_1, \ldots, w_n; h)$. Zřejmě $(w_1, \ldots, w_n, -h, 0)$ je reprezentace g, protože $g(x_1, \ldots, x_n, 1) = 1$, právě když $\sum_{i=1}^{n} w_i x_i - h \geq 0$, tj. $f(x_1, \ldots, x_n) = 1$. □

6.2 Omezená doména

V předchozí podkapitole jsme uvažovali obecně v absolutní hodnotě libovolně velké reálné vstupy neuronů, avšak v praktických aplikacích je obvykle vstupní prostor omezený a také vstupy neuronů jsou v neuronové síti tvořeny omezenými výstupy jiných neuronů. Proto budeme v této podkapitole analyzovat lineární prahovou funkci a její reprezentaci pro případ omezeného definičního oboru. Uvidíme, že na rozdíl od obecného reálného definičního oboru má lineární prahová funkce v omezené doméně bohatší prostor svých reprezentací. Pro tento účel nejprve zavedeme pojem *separability* lineární prahové funkce a definujeme *váhu reprezentace*, což je jednoduchá míra deskriptivní složitosti reprezentace.

Definice 6.8 *Řekneme, že reprezentace $(w_1, \ldots, w_n; h)$ lineární prahové funkce f je* δ-separabilní *v definičním oboru $\Xi_n \subseteq \mathbb{R}^n$ pro kladné reálné $\delta \in \mathbb{R}^+$, jestliže pro všechna $(x_1, \ldots, x_n) \in \Xi_n$ platí:*

$$f(x_1, \ldots, x_n) = 0\,, \quad \textit{právě když} \quad \sum_{i=1}^{n} w_i x_i \leq h - \delta\,. \tag{6.7}$$

Lineární prahová funkce je separabilní *v definičním oboru $\Xi_n \subseteq \mathbb{R}^n$, jestliže má v této doméně δ-separabilní reprezentaci pro nějaké $\delta \in \mathbb{R}^+$. V absolutní hodnotě maximální váhu $|w| = \max_{1 \leq i \leq n} |w_i|$ nazveme* váhou reprezentace $(w_1, \ldots, w_n; h)$.

Separabilní lineární prahová funkce v oboru $\Xi_n \subseteq \mathbb{R}^n$ modeluje funkci perceptronu, u kterého je supremum vnitřních potenciálů s nulovým výstupem pro vstupy z Ξ_n menší než práh. Ukážeme, že pro libovolné kladné δ existuje reprezentace separabilní lineární prahové funkce taková, že toto supremum je o δ menší než práh.

Lemma 6.9 *Nechť f je lineární prahová funkce s doménou $\Xi_n \subseteq \mathbb{R}^n$ a $\delta \in \mathbb{R}^+$ je kladné reálné číslo. Jestliže f má λ-separabilní reprezentaci v Ξ_n s váhou $|w|$ pro nějaké $\lambda > 0$, pak f má δ-separabilní reprezentaci v Ξ_n s váhou $(\delta/\lambda)|w|$.*

Věta 6.6 *Soubory $(w_1, \ldots, w_n; h)$ a $(v_1, \ldots, v_n; r)$ jsou reprezentace jedné lineární prahové funkce f, právě když existuje kladné reálné číslo $\mu \in \mathbb{R}^+$ takové, že $h = \mu r$ a $w_i = \mu v_i$ pro $i = 1, \ldots, n$.*

Důkaz: Nechť $(w_1, \ldots, w_n; h)$ a $(v_1, \ldots, v_n; r)$ jsou reprezentace lineární prahové funkce f. Nejprve pro jednoduchost předpokládejme, že f není degenerovaná a h, r jsou nenulové. Prahy h a r mají stejná znaménka, protože $h > 0$, právě když $f(0, \ldots, 0) = 0$, právě když $r > 0$.

Ukážeme, že pro $w_i > 0$ $(1 \leq i \leq n)$ existuje právě jedno reálné $t_i \in \mathbb{R}$ takové, že

$$f(0, \ldots, 0, x_i, 0, \ldots, 0) = 1\,, \quad \text{právě když} \quad x_i \geq t_i\,. \tag{6.5}$$

Položme $t_i = h/w_i$. Potom $f(0, \ldots, 0, x_i, 0, \ldots, 0) = 1$, právě když $w_i x_i \geq h$, tj. $x_i \geq t_i$. Jednoznačnost t_i dokážeme sporem. Nechť tedy existuje reálné $s_i \neq t_i$ takové, že

$$f(0, \ldots, 0, x_i, 0, \ldots, 0) = 1\,, \quad \text{právě když} \quad x_i \geq s_i\,. \tag{6.6}$$

Bez újmy na obecnosti nechť např. $s_i < t_i$. Podle (6.5) to znamená, že $f(0, \ldots, 0, s_i, 0, \ldots, 0) = 0$. Na druhou stranu z (6.6) vyplývá, že $f(0, \ldots, 0, s_i, 0, \ldots, 0) = 1$, což je spor.

Podobně lze ukázat, že pro $w_i < 0$ $(1 \leq i \leq n)$ existuje jediné $t_i = h/w_i$ takové, že $f(0, \ldots, 0, x_i, 0, \ldots, 0) = 1$, právě když $x_i \leq t_i$. Tedy máme $t_1, \ldots, t_n$ pro f jednoznačné a na její reprezentaci nezávislé, tj. $h/w_i = r/v_i$ pro $i = 1, \ldots, n$. Položme $\mu = h/r > 0$ a dostáváme, že $h = \mu r$ a $w_i = (h/r)v_i = \mu v_i$ pro $i = 1, \ldots, n$.

Pokud $h = 0$ nebo $r = 0$, pak jsou oba prahy nulové, tj. $h/w_i = r/v_i = 0$, a podobně lze ukázat, že např. pro $w_i > 0$ $(1 \leq i \leq n-1)$ existuje právě jedno reálné $t_i \in \mathbb{R}$ takové, že $f(0, \ldots, 0, x_i, 0, \ldots, 0, -1) = 1$, právě když $x_i \geq t_i$. V tomto případě polož $\mu = w_n/v_n > 0$ atd. Pokud f je degenerovaná v i-té proměnné, pak podle věty 6.5 jsou odpovídající váhy nulové, tj. $w_i = v_i = 0$, a pro ostatní váhy použijeme předchozí postup.

Obrácená implikace je jednoduchá, protože $\sum_{i=1}^{n} w_i x_i \geq h$, právě když $\sum_{i=1}^{n} \mu v_i x_i \geq \mu r$, právě když $\sum_{i=1}^{n} v_i x_i \geq r$. □

Pokud budeme dvě reprezentace, které jsou násobkem, považovat za ekvivalentní, pak se množina všech reprezentací rozpadá na třídy této ekvivalence a každá lineární prahová funkce s reálným definičním oborem odpovídá podle věty 6.6 právě jedné takové třídě.

Na závěr této podkapitoly ještě zopakujeme formální úpravu, kterou jsme používali v první části této knihy (srovnejte s (1.1), (1.2) a (1.3)) a pomocí níž se lze omezit na reprezentace s nulovým prahem.

Důkaz: Nechť $(w_1,\ldots,w_n;h)$ je λ-separabilní reprezentace v Ξ_n s váhou $|w|$, tj. $f(x_1,\ldots,x_n)=0$, právě když $\sum_{i=1}^n w_i x_i \le h-\lambda$. Položme $\mu=\delta/\lambda>0$ a aplikujeme větu 6.6. To znamená, že $f(x_1,\ldots,x_n)=0$, právě když $\sum_{i=1}^n \mu w_i x_i \le \mu h-\delta$ a $(\mu w_1,\ldots,\mu w_n;\mu h)$ je δ-separabilní reprezentace v Ξ_n s váhou $(\delta/\lambda)|w|$. □

Důsledek 6.10 *Pro každé $\delta\in\mathbb{R}^+$ má separabilní lineární prahová funkce δ-separabilní reprezentaci.*

Nyní dokážeme větu, která nám umožní omezit se na domény, které jsou podmnožinou n-rozměrné krychle $\langle 0,1\rangle^n$.

Věta 6.11 *Nechť $\Xi_n\subseteq\langle -a,a\rangle^n$ $(a>0)$ je omezená doména a f je lineární prahová funkce v Ξ_n. Pak existuje omezený definiční obor $\Xi'_n\subseteq\langle 0,1\rangle^n$ a vzájemně jednoznačné zobrazení $\Phi:\Xi_n\longrightarrow\Xi'_n$ takové, že $f\circ\Phi^{-1}$ je lineární prahová funkce v Ξ'_n a $(w_1,\ldots,w_n;h)$ je reprezentace f, právě když*

$$\left(w_1,\ldots,w_n;\frac{1}{2}\left(\frac{h}{a}+\sum_{i=1}^n w_i\right)\right) \tag{6.8}$$

je reprezentace $f\circ\Phi^{-1}$.

Důkaz: Zavedeme lineární transformaci $l:\langle -a,a\rangle\longrightarrow\langle 0,1\rangle$ tak, že $l(x)=\frac{1}{2}(\frac{x}{a}+1)$. Pomocí ní definujeme bijekci $\Phi:\Xi_n\longrightarrow\langle 0,1\rangle$ po složkách $\Phi(x_1,\ldots,x_n)=(l(x_1),\ldots,l(x_n))$ a označíme Ξ'_n obor hodnot Φ. Nyní nechť $(w_1,\ldots,w_n;h)$ je reprezentace f. Pak pro každé $(x'_1,\ldots,x'_n)\in\Xi'_n$ je $f\circ\Phi^{-1}(x'_1,\ldots,x'_n)=1$, právě když

$$f(l^{-1}(x'_1),\ldots,l^{-1}(x'_n))=f(a(2x'_1-1),\ldots,a(2x'_n-1))=1\,, \tag{6.9}$$

právě když $\sum_{i=1}^n w_i a(2x'_i-1)\ge h$, tj.

$$\sum_{i=1}^n w_i x'_i \ge \frac{1}{2}\left(\frac{h}{a}+\sum_{i=1}^n w_i\right)\,. \tag{6.10}$$

Tedy $f\circ\Phi^{-1}$ je lineární prahová funkce v Ξ'_n s reprezentací (6.8).

Nechť naopak (6.8) je reprezentace $f\circ\Phi^{-1}$. Pak pro každé $(x_1,\ldots,x_n)\in\Xi_n$ je $f(x_1,\ldots,x_n)=1$, právě když $(f\circ\Phi^{-1})\circ\Phi(x_1,\ldots,x_n)=1$, právě když $(f\circ\Phi^{-1})(l(x_1),\ldots,l(x_n))=1$, tj.

$$\sum_{i=1}^n w_i\frac{1}{2}\left(\frac{x}{a}+1\right)\ge\frac{1}{2}\left(\frac{h}{a}+\sum_{i=1}^n w_i\right) \tag{6.11}$$

a po úpravě $\sum_{i=1}^n w_i x_i\ge h$. Tedy $(w_1,\ldots,w_n;h)$ je reprezentace f v Ξ_n. □

Uvažujme případ, kdy chceme dokázat nějakou vlastnost o váhách lineární prahové funkce v obecné omezené doméně $\Xi_n\subseteq\langle -a,a\rangle^n$. Podle věty 6.11 můžeme pomocí Φ transformovat definiční obor Ξ_n na $\Xi'_n\subseteq\langle 0,1\rangle^n$ a tuto vlastnost dokázat pro váhy $f\circ\Phi^{-1}$, protože pro každou reprezentaci $f\circ\Phi^{-1}$ existuje reprezentace f se stejnými váhami, pro níž příslušný výsledek platí. Pro jednoduchost se proto dále omezíme na definiční obory, které jsou obsaženy v $\langle 0,1\rangle^n$. Podobně zjednodušíme situaci tím, že budeme uvažovat jen kladné váhy. K tomu nás opravňuje následující věta.

Věta 6.12 *Nechť $\Xi_n\subseteq\langle 0,1\rangle^n$ je omezená doména, f je lineární prahová funkce v Ξ_n a $1\le i\le n$. Pak existuje omezený definiční obor $\Xi'_n\subseteq\langle 0,1\rangle^n$ a bijekce $\Phi:\Xi_n\longrightarrow\Xi'_n$ taková, že $f\circ\Phi^{-1}$ je lineární prahová funkce v Ξ'_n a $(w_1,\ldots,w_n;h)$ je reprezentace f, právě když*

$$(w_1,\ldots,w_{i-1},-w_i,w_{i+1},\ldots,w_n;h-w_i) \tag{6.12}$$

je reprezentace $f\circ\Phi^{-1}$.

Důkaz: Definujme bijekci $\Phi(x_1,\ldots,x_n)=(x_1,\ldots,x_{i-1},1-x_i,x_{i+1},\ldots,x_n)$ a označme Ξ'_n obor hodnot Φ. Důkaz dále pokračuje podobně jako důkaz věty 6.11. □

Při implementaci funkce neuronu nelze pracovat s neomezenou přesností reprezentace, proto se v následujícím výkladu budeme zabývat *celočíselnou reprezentací* a její váhou. Ukážeme, že separabilní lineární prahové funkce mají celočíselnou reprezentaci.

Definice 6.13 *Reprezentaci $(w_1,\ldots,w_n;h)$ lineární prahové funkce nazveme* celočíselnou reprezentací, *jestliže $w_i\in\mathbb{Z}$ pro $i=1,\ldots,n$ a $h\in\mathbb{Z}$ jsou celá čísla.*

Věta 6.14 *Každá lineární prahová funkce f s δ-separabilní reprezentací v omezeném definičním oboru $\Xi_n\subseteq\langle 0,1\rangle^n$ s váhou $|w|$ má celočíselnou reprezentaci s váhou nejvýše $((n+1)/\delta)|w|$.*

Důkaz: Nechť f je lineární prahová funkce s δ-separabilní reprezentací v omezené doméně $\Xi_n\subseteq\langle 0,1\rangle^n$ s váhou $|w|$. Podle lemmy 6.9 potom pro f v Ξ_n existuje $(n+1)$-separabilní reprezentace $(w_1,\ldots,w_n;h)$ s váhou $((n+1)/\delta)|w|$. Bez újmy na obecnosti můžeme předpokládat, že $w_i\ge 0$ pro $i=1,\ldots,n$. Pro $w_i<0$ lze totiž použít větu 6.12 a pokračovat v následujícím důkazu. Nakonec pomocí stejné věty obdržíme celočíselné váhy pro původní lineární prahovou funkci a její doménu.

Nyní již uvažujme celočíselné váhy $v_i\in\mathbb{Z}$ a práh $r\in\mathbb{Z}$ takové, že $w_i-1\le v_i<w_i$ pro $i=1,\ldots,n$ a $h-n-1\le r<h-n$. Dokážeme, že $(v_1,\ldots,v_n;r)$ je celočíselná reprezentace f v Ξ_n. Nechť $(x_1,\ldots,x_n)\in\Xi_n$.

Definice 6.18 Váha lineární prahové funkce *v konečném oboru je minimální váha její celočíselné reprezentace.*

V následujícím výkladu budeme odhadovat váhu lineární prahové funkce. Za tímto účelem nejprve zavedeme pojem *jádra* lineární prahové funkce a pro konečnou doménu ukážeme jeho existenci.

Definice 6.19 *Nechť f je lineární prahová funkce v omezené doméně $\Xi_n \subseteq \langle 0,1\rangle^n$. Řekneme, že $K \subseteq \Xi_n$ je* jádro *f v Ξ_n, jestliže obsahuje $|K| = n+1$ prvků a existuje reprezentace $(w_1,\ldots,w_n;h)$ pro f v Ξ_n taková, že pro všechna $(x_1,\ldots,x_n) \in K$ platí $\sum_{i=1}^n w_i x_i \in \{h-1, h\}$.*

Věta 6.20 *Každá nedegenerovaná lineární prahová funkce v konečné doméně má jádro.*

Důkaz: Nechť f je nedegenerovaná lineární prahová funkce v konečné doméně Ξ_n. Podle věty 6.16 je f separabilní a podle důsledku 6.10 má 1-separabilní reprezentaci, která je řešením následující soustavy $|\Xi_n|$ nerovnic s $n+1$ neznámými $w_1,\ldots,w_n,h$:

$$\sum_{i=1}^n x_i w_i \begin{cases} \leq h-1 & \text{pro } f(x_1,\ldots,x_n)=0 \\ \geq h & \text{pro } f(x_1,\ldots,x_n)=1 \end{cases} \quad (x_1,\ldots,x_n)\in\Xi_n\,. \qquad (6.14)$$

Prostor řešení soustavy (6.14), který odpovídá všem 1-separabilním reprezentacím f, je tedy v $(n+1)$-rozměrném euklidovském prostoru E_{n+1} neprázdným průnikem poloprostorů ohraničených nadrovinami

$$\sum_{i=1}^n x_i w_i - h = \begin{cases} -1 & \text{pro } f(x_1,\ldots,x_n)=0 \\ 0 & \text{pro } f(x_1,\ldots,x_n)=1 \end{cases} \quad (x_1,\ldots,x_n)\in\Xi_n \qquad (6.15)$$

s proměnnými $w_1,\ldots,w_n,h$. Navíc f je nedegenerovaná, proto v E_{n+1} existuje řešení soustavy (6.15) odpovídající průniku právě $n+1$ nadrovin. To znamená, že následující soustava $n+1$ rovnic s $n+1$ neznámými $w_1,\ldots,w_n,h$ má řešení:

$$\sum_{i=1}^n s_{ki} w_i - h = t_k \qquad k=1,\ldots,n+1 \qquad (6.16)$$

pro nějaké $\mathbf{s}_k = (s_{k1},\ldots,s_{kn}) \in \Xi_n$ a $t_k = f(s_{k1},\ldots,s_{kn}) - 1 \in \{-1,0\}$ $(k = 1,\ldots,n+1)$. Pro řešení $w_1,\ldots,w_n,h$ soustavy (6.16) platí, že $\sum_{i=1}^n w_i s_{ki} \in \{h-1,h\}$ pro $k=1,\ldots,n+1$, a tedy množina $K = \{\mathbf{s}_k \mid k=1,\ldots,n+1\}$ tvoří jádro f v Ξ_n. □

Nyní budeme pomocí jádra definovat *objem lineární prahové funkce*, o kterém ukážeme, že souvisí s její deskriptivní složitostí.

Pro $f(x_1,\ldots,x_n) = 1$ platí $\sum_{i=1}^n w_i x_i \geq h$. Z toho vyplývá, že $\sum_{i=1}^n (v_i+1)x_i \geq h$, protože $v_i + 1 \geq w_i$. Po úpravě dostáváme $\sum_{i=1}^n v_i x_i \geq h - \sum_{i=1}^n x_i$, což implikuje $\sum_{i=1}^n v_i x_i \geq h - n$, protože $x_i \in \langle 0,1\rangle$. Odtud $\sum_{i=1}^n v_i x_i \geq r$, protože $h - n > r$. Naopak $f(x_1,\ldots,x_n) = 0$ díky $(n+1)$-separabilitě reprezentace $(w_1,\ldots,w_n;h)$ implikuje $\sum_{i=1}^n w_i x_i \leq h - n - 1$, z čehož vyplývá, že $\sum_{i=1}^n v_i x_i < r$, protože $v_i < w_i$ a $h - n - 1 \leq r$. Nakonec váha reprezentace $(v_1,\ldots,v_n;r)$ je menší než $((n+1)/\delta)|w|$, protože $v_i < w_i$ pro $i = 1,\ldots,n$. □

Důsledek 6.15 *Každá lineární prahová funkce separabilní v omezeném definičním oboru má celočíselnou reprezentaci.*

Důkaz: Aplikuj větu 6.11 na tvrzení o celočíselných váhách ve větě 6.14. □

Důsledek 6.15 znamená, že pokud budeme podobně jako v podkapitole 6.1 považovat dvě reprezentace, které určují jednu separabilní lineární prahovou funkci, za ekvivalentní, pak v příslušném rozkladu množiny separabilních reprezentací na třídy ekvivalence, má každá taková třída celočíselného reprezentanta.

6.3 Konečná doména

V této podkapitole se dále omezíme na konečnou doménu, pro kterou má lineární prahová funkce celočíselnou reprezentaci, protože je v ní separabilní. Díky tomu můžeme v konečné doméně definovat *váhu lineární prahové funkce* jako nejmenší váhu její celočíselné reprezentace.

Věta 6.16 *Každá lineární prahová funkce je v konečné doméně separabilní.*

Důkaz: Nechť f je lineární prahová funkce v konečném oboru $\Xi_n \subseteq \mathbb{R}^n$ s reprezentací $(w_1,\ldots,w_n;h)$. Položme

$$\delta = h - \max\left\{ \sum_{i=1}^n w_i x_i \;\middle|\; f(\mathbf{x})=0, \mathbf{x}=(x_1,\ldots,x_n)\in\Xi_n \right\}. \qquad (6.13)$$

Pak pro $\mathbf{x} = (x_1,\ldots,x_n) \in \Xi_n$ platí, že $f(\mathbf{x}) = 0$, právě když $\sum_{i=1}^n w_i x_i \leq \max\{\sum_{i=1}^n w_i x_i \mid f(\mathbf{x}) = 0\} = h - \delta$. Tedy $(w_1,\ldots,w_n;h)$ je δ-separabilní reprezentace f v Ξ_n. □

Důsledek 6.17 *Každá lineární prahová funkce v konečném definičním oboru má celočíselnou reprezentaci.*

Důkaz: Tvrzení je přímým důsledkem věty 6.16 a důsledku 6.15. □

Determinanty Δ_i $(i = 1, \ldots, n)$ jsou řádu $n+1$ a obsahují prvky z $\langle -1, 1\rangle$, proto jejich absolutní hodnoty lze pomocí (6.20) odhadnout $|\Delta_i| \leq (\sqrt{n+1})^{n+1} = (n+1)^{\frac{n+1}{2}}$. Absolutní hodnota determinantu $|\Delta|$ soustavy (6.16) odpovídá objemu V. Tedy $|w_i| \leq (n+1)^{\frac{n+1}{2}}/V$. □

Důsledek 6.23 *Každá lineární prahová funkce f v konečné doméně $\Xi_n \subseteq \langle 0, 1\rangle^n$ s objemem V má celočíselnou reprezentaci s váhou nejvýše $(n+1)^{\frac{n+3}{2}}/V$.*

Důkaz: Podle věty 6.22 existuje 1-separabilní reprezentace f v Ξ_n s váhou nejvýše $(n+1)^{\frac{n+1}{2}}/V$ a aplikací věty 6.14 získáme celočíselnou reprezentaci s váhou nejvýše $((n+1)/1)(n+1)^{\frac{n+1}{2}}/V = (n+1)^{\frac{n+3}{2}}/V$. □

6.4 Booleovská doména

V této podkapitole se budeme zabývat tzv. *booleovskými prahovými funkcemi*, které modelují funkci perceptronu s binárními vstupy.

6.4.1 Booleovská prahová funkce

Nejprve učiníme jednoduchá pozorování o tom, které z klasických booleovských funkcí jsou prahové funkce (srovnejte s obrázky 1.13 a 1.7). Dále nahlédneme triviální vlastnosti negace booleovské prahové funkce a ukážeme existenci tzv. *rozhodující reprezentace*.

Definice 6.24 *Lineární prahová funkce v booleovské doméně $\{0, 1\}^n$ se nazývá* booleovská prahová funkce.

Podobně je možné pro bipolární logiku uvažovat doménu $\{-1, 1\}^n$. Následující věty pro booleovskou prahovou funkci platí i pro tento definiční obor, protože lze využít transformaci domény $\{-1, 1\}^n$ na $\{0, 1\}^n$ podle věty 6.11.

Definice 6.25 *Připomeňme následující značení booleovských funkcí $f : \{0, 1\}^n \longrightarrow \{0, 1\}$.* Konjunkce *$AND(x_1, \ldots, x_n) = x_1 \wedge \ldots \wedge x_n = 1$, právě když $x_i = 1$ pro všechna $i = 1, \ldots, n$.* Disjunkce *$OR(x_1, \ldots, x_n) = x_1 \vee \ldots \vee x_n = 1$, právě když existuje $1 \leq i \leq n$ takové, že $x_i = 1$.* Negace *booleovské proměnné $NOT(x) = \bar{x} = 1$, resp. funkce $\bar{f}(x_1, \ldots, x_n) = 1$, právě když $x = 0$, resp. $f(x_1, \ldots, x_n) = 0$.* Vylučovací disjunkce *(*parita*) $XOR(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n = 1$, právě když počet i takových, že $x_i = 1$, je lichý, tj. $|\{i \mid x_i = 1\}|$ je liché číslo.* Konsenzuální *(*většinová*) funkce $MAJORITY(x_1, \ldots, x_n) = 1$, právě když je aspoň polovina i takových, že $x_i = 1$, tj. $|\{i \mid x_i = 1\}| \geq n/2$.*

Definice 6.21 *Nechť f je lineární prahová funkce v omezené doméně $\Xi_n \subseteq \langle 0, 1\rangle^n$ a $K = \{\mathbf{s}_k \in \Xi_n \mid \mathbf{s}_k = (s_{k1}, \ldots, s_{kn}), 1 \leq k \leq n+1\}$ je jádro f v Ξ_n. Absolutní hodnotu determinantu*

$$\text{abs} \begin{vmatrix} s_{1,1} & \cdots & s_{1,n} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ s_{n+1,1} & \cdots & s_{n+1,n} & 1 \end{vmatrix} \tag{6.17}$$

nazveme objemem jádra K. *Objem lineární prahové funkce f v Ξ_n je definován jako maximální objem jádra přes všechny jádra f v Ξ_n.*

Geometricky lze objem jádra chápat jako objem rovnoběžnostěnu v $(n+1)$-rozměrném euklidovském prostoru E_{n+1}, jehož strany jsou umístění vektorů z jádra (rozšířených o jednotkovou $(n+1)$-ní souřadnici) v počátku. V následujícím výkladu ukážeme vztah mezi objemem lineární prahové funkce a její váhou.

Věta 6.22 *Každá lineární prahová funkce f v konečné doméně $\Xi_n \subseteq \langle 0, 1\rangle^n$ s objemem V má 1-separabilní reprezentaci s váhou nejvýše $(n+1)^{\frac{n+1}{2}}/V$.*

Důkaz: Tvrzení ukážeme pro nedegenerovanou f, zatímco důkaz pro degenerovanou f probíhá obdobně (navíc dle analogie věty 6.5 pro konečnou doménu jsou některé váhy nulové). Tedy podle věty 6.20 má f jádro v Ξ_n s objemem V. Podle důkazu této věty existuje 1-separabilní reprezentace $(w_1, \ldots, w_n; h)$, která je řešením soustavy $n+1$ rovnic (6.16) (pro f degenerovanou v k proměnných uvažujeme obdobnou soustavu $n-k+1$ rovnic a stejný počet nenulových neznámých). Její váhy lze vyjádřit pomocí Cramerova pravidla $w_i = \Delta_i/\Delta$ pro $i = 1, \ldots, n$, kde

$$\Delta = \begin{vmatrix} s_{1,1} & \cdots & s_{1,n} & -1 \\ \vdots & \ddots & \vdots & \vdots \\ s_{n+1,1} & \cdots & s_{n+1,n} & -1 \end{vmatrix} \tag{6.18}$$

$$\Delta_i = \begin{vmatrix} s_{1,1} & \cdots & s_{1,i-1} & t_1 & s_{1,i+1} & \cdots & s_{1,n} & -1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{n+1,1} & \cdots & s_{n+1,i-1} & t_{n+1} & s_{n+1,i+1} & \cdots & s_{n+1,n} & -1 \end{vmatrix}. \tag{6.19}$$

Dále využijeme Hadamardovu nerovnost z lineární algebry, která shora odhaduje absolutní hodnotu determinantu matice pomocí součinu norem jejích řádků, tj.

$$\text{abs} \begin{vmatrix} x_{1,1} & \cdots & x_{1,p} \\ \vdots & \ddots & \vdots \\ x_{p,1} & \cdots & x_{p,p} \end{vmatrix} \leq \prod_{i=1}^{p} \sqrt{\sum_{j=1}^{p} x_{ij}^2}\,. \tag{6.20}$$

Věta 6.26 *Konjunkce $AND(x_1,\ldots,x_n)$, disjunkce $OR(x_1,\ldots,x_n)$, negace $NOT(x)$ a konsenzuální funkce $MAJORITY(x_1,\ldots,x_n)$ jsou booleovské prahové funkce, zatímco vylučovací disjunkce $XOR(x_1,\ldots,x_n)$ pro $n \geq 2$ není booleovská prahová funkce.*

Důkaz: Soubory $(1,\ldots,1;n)$, $(1,\ldots,1;1)$, $(-1;0)$ a $(1,\ldots,1;n/2)$ jsou po řadě reprezentace funkcí AND, OR, NOT a $MAJORITY$. Důkaz neexistence reprezentace pro $XOR(x_1,\ldots,x_n)$ probíhá sporem indukcí dle $n \geq 2$. Nejprve pro $n = 2$ předpokládejme, že $XOR(x_1,x_2)$ je booleovská prahová funkce s reprezentací $(w_1,w_2;h)$. Z $XOR(1,0) = XOR(0,1) = 1$ plyne $w_1 \geq h$ a $w_2 \geq h$, což implikuje $w_1 + w_2 \geq 2h$. Na druhou stranu pro $XOR(1,1) = 0$ obdržíme $w_1 + w_2 < h$. Tedy $h > 2h$, tj. $h < 0$, což je ve sporu s $XOR(0,0) = 0$. Nechť dále $n > 2$ a $XOR(x_1,\ldots,x_{n-1})$ není booleovská prahová funkce. Pro spor předpokládejme, že $(w_1,\ldots,w_n;h)$ je reprezentace $XOR(x_1,\ldots,x_n)$. Potom $XOR(x_1,\ldots,x_{n-1}) = 1$, právě když $XOR(x_1,\ldots,x_{n-1},0) = 1$, právě když $\sum_{i=1}^{n-1} w_i x_i \geq h$. To znamená, že $XOR(x_1,\ldots,x_{n-1})$ je booleovská prahová funkce s reprezentací $(w_1,\ldots,w_{n-1};h)$, což je spor s indukčním předpokladem. □

Následující lemma ukazuje, že booleovská prahová funkce je po negaci proměnných, resp. funkční hodnoty, opět booleovská prahová funkce se stejnou váhou. To znamená, že u perceptronu s binárními vstupy lze negovat vstupy i výstup, aniž bychom ztráceli efektivní neuronovou implementaci.

Lemma 6.27

(i) *Nechť f je booleovská prahová funkce s celočíselnou reprezentací $(w_1,\ldots,w_n;h)$, pak negace $\bar{f}$ je booleovská prahová funkce s celočíselnou reprezentací $(-w_1,\ldots,-w_n;1-h)$.*

(i') *Nechť f je booleovská prahová funkce, pak negace $\bar{f}$ je booleovská prahová funkce.*

(ii) *Nechť f je booleovská prahová funkce s reprezentací $(w_1,\ldots,w_n;h)$ a $1 \leq i \leq n$, pak funkce $f(x_1,\ldots,x_{i-1},\bar{x}_i,x_{i+1},\ldots,x_n)$ je booleovská prahová funkce s reprezentací $(w_1,\ldots,w_{i-1},-w_i,w_{i+1},\ldots,w_n;h-w_i)$.*

(ii') *Nechť f je booleovská prahová funkce s reprezentací $(w_1,\ldots,w_n;h)$, pak funkce $f(\bar{x}_1,\ldots,\bar{x}_n)$ je booleovská prahová funkce s reprezentací $(-w_1,\ldots,-w_n;h-\sum_{i=1}^n w_i)$.*

(iii) *Nechť $\bar{f}$ je booleovská prahová funkce s celočíselnou reprezentací $(\bar{w}_1,\ldots,\bar{w}_n;h)$, pak negace $f(\bar{x}_1,\ldots,\bar{x}_n)$ je booleovská prahová funkce s celočíselnou reprezentací $(\bar{w}_1,\ldots,\bar{w}_n;1-h+\sum_{i=1}^n \bar{w}_i)$.*

Důkaz:

(i) Platí, že $\bar{f}(x_1,\ldots,x_n) = 1$, tj. $f(x_1,\ldots,x_n) = 0$, právě když $\sum_{i=1}^n w_i x_i < h$, tj. díky celočíselné reprezentaci $\sum_{i=1}^n w_i x_i \leq h-1$, což lze upravit $\sum_{i=1}^n -w_i x_i \geq 1-h$.

(i') Booleovská prahová funkce má konečnou doménu, a tedy podle důsledku 6.17 existuje její celočíselná reprezentace. Tvrzení pak plyne z (i).

(ii) Platí, že $f(x_1,\ldots,x_{i-1},\bar{x}_i,x_{i+1},\ldots,x_n) = 1$, právě když $\sum_{j=1}^{i-1} w_j x_j + w_i(1-x_i) + \sum_{j=i+1}^n w_j x_j \geq h$, tj. $\sum_{j=1}^{i-1} w_j x_j - w_i x_i + \sum_{j=i+1}^n w_j x_j \geq h - w_i$.

(ii') Aplikuj (ii) pro $i = 1,\ldots,n$.

(iii) Tvrzení je přímým důsledkem (i) a (ii'). □

Je zajímavé, že podle (iii) z lemmy 6.27 dostaneme s využitím věty 6.26 de Morganovy vzorce, tj. $\overline{AND}(x_1,\ldots,x_n) = OR(\bar{x}_1,\ldots,\bar{x}_n)$ resp. $\overline{OR}(x_1,\ldots,x_n) = AND(\bar{x}_1,\ldots,\bar{x}_n)$, nebo v případě konsenzuální funkce $\overline{MAJORITY}(x_1,\ldots,x_n) = MAJORITY(\bar{x}_1,\ldots,\bar{x}_n)$ pro lichá n.

Definice 6.28 *Reprezentace $(w_1,\ldots,w_n;h)$ booleovské prahové funkce se nazývá* rozhodující, *jestliže pro všechna $x_1,\ldots,x_n \in \{0,1\}$ platí $\sum_{i=1}^n w_i x_i \neq h$.*

Lemma 6.29 *Každá booleovská prahová funkce f s celočíselnou reprezentací $(w_1,\ldots,w_n;h)$ s váhou $|w|$ má rozhodující celočíselnou reprezentaci s váhou $2|w|$.*

Důkaz: Platí, že $f(x_1,\ldots,x_n) = 1$, právě když $\sum_{i=1}^n w_i x_i \geq h$, tj. $\sum_{i=1}^n 2w_i x_i \geq 2h > 2h-1$. Na druhou stranu $f(x_1,\ldots,x_n) = 0$, právě když $\sum_{i=1}^n 2w_i x_i < 2h$, což je díky sudému číslu na levé straně nerovnosti ekvivalentní s $\sum_{i=1}^n 2w_i x_i < 2h-1$. Tedy $(2w_1,\ldots,2w_n;2h-1)$ je celočíselná rozhodující reprezentace f s váhou $2|w|$. □

6.4.2 Váha booleovské prahové funkce

V následujícím výkladu se budeme zabývat velikostí reprezentace booleovské prahové funkce a také určíme počet booleovských prahových funkcí n proměnných. Uvedené otázky si matematici kladli již v 60. letech a výsledky jejich výzkumu z tohoto období jsou shrnuty v práci [200]. Nejprve zpřesníme horní odhad váhy v důsledku 6.23 pro booleovskou prahovou funkci, a tím shora omezíme velikost absolutní hodnoty váhy perceptronu s binárními vstupy.

Věta 6.30 (Muroga, Toda, Takasu [201]) *Každá booleovská prahová funkce f má celočíselnou reprezentaci s váhou nejvýše $(n+1)^{\frac{n+1}{2}}/2^n$.*

Důkaz: Podle důkazu věty 6.22 pro konečnou doménu (opět pro jednoduchost uvažujeme jen nedegenerovaný případ) existuje 1-separabilní reprezentace $(w_1,\ldots,w_n;h)$, jejíž váhy $w_i=\Delta_i/\Delta$ $(i=1,\ldots,n)$ a práh $h=\Delta_{n+1}/\Delta$ lze vyjádřit pomocí determinantů (6.18) a (6.19). Položme $\mu=|\Delta|>0$ a $v_i=\mu w_i$ $(i=1,\ldots,n)$, $r=\mu h$. Podle věty 6.6 je $(v_1,\ldots,v_n;r)$ reprezentace f. Navíc $|v_i|=|\Delta_i|$ a $|r|=|\Delta_{n+1}|$ jsou celá čísla, protože prvky $s_{j,k}\in\{0,1\}$, $t_j\in\{-1,0\}$ $(1\le j\le n+1,\ 1\le k\le n)$ v determinantu (6.19) jsou z množiny $\{-1,0,1\}$. Tedy $(v_1,\ldots,v_n;r)$ je celočíselná reprezentace f. Absolutní hodnotu váhy $|v_i|$ $(i=1,\ldots,n)$ lze vyjádřit:

$$|v_i|=\operatorname{abs}\begin{vmatrix} s_{1,1} & \cdots & s_{1,i-1} & t_1 & s_{1,i+1} & \cdots & s_{1,n} & -1\\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ s_{n+1,1} & \cdots & s_{n+1,i-1} & t_{n+1} & s_{n+1,i+1} & \cdots & s_{n+1,n} & -1 \end{vmatrix}. \quad (6.21)$$

Nejprve vynásobíme i-tý sloupec determinantu (6.21) číslem -1, pak prvních n sloupců vynásobíme dvěma a přičteme k nim poslední sloupec:

$$2^n|v_i|=\operatorname{abs}\begin{vmatrix} 2s_{1,1}-1 & \cdots & -2t_1-1 & \cdots & 2s_{1,n}-1 & -1\\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots\\ 2s_{n+1,1}-1 & \cdots & -2t_{n+1}-1 & \cdots & 2s_{n+1,n}-1 & -1 \end{vmatrix}. \quad (6.22)$$

Nyní prvky determinantu v (6.22) jsou z množiny $\{-1,1\}$. Pomocí Hadamardovy nerovnosti (6.20) dostaneme $2^n|v_i|\le(n+1)^{\frac{n+1}{2}}$, tj. $|v_i|\le(n+1)^{\frac{n+1}{2}}/2^n$. □

Dále odhadneme zdola počet booleovských prahových funkcí n proměnných.

Věta 6.31 (Yajima, Ibaraki [293]) *Existuje aspoň $2^{\frac{n(n-1)}{2}}$ různých booleovských prahových funkcí n proměnných.*

Důkaz: Pro dolní odhad stačí uvažovat reprezentace booleovských prahových funkcí s nulovým prahem. Označme $C(n)$ počet různých booleovských prahových funkcí n proměnných, které mají reprezentaci s nulovým prahem. Díky negaci $NOT(x)$ a konstantní jednotkové funkci je $C(1)=2$.

Dále uvažujme booleovskou prahovou funkci f s $n-1$ proměnnými, která má podle věty 6.16 δ-separabilní reprezentaci $(w_1,\ldots,w_{n-1};0)$ pro nějaké $\delta>0$. Tuto reprezentaci upravíme tak, aby pro každé dva různé vstupy $(x_1,\ldots,x_{n-1})\ne(x'_1,\ldots,x'_{n-1})\in\{0,1\}^{n-1}$ platilo, že

$$\sum_{j=1}^{n-1}w_jx_j\ne\sum_{j=1}^{n-1}w_jx'_j\,. \quad (6.23)$$

Nechť (6.23) neplatí pro nějaké dva vstupy $(a_1,\ldots,a_{n-1})\ne(b_1,\ldots,b_{n-1})\in\{0,1\}^{n-1}$, které se liší v i-té proměnné, tj. $\sum_{j=1}^{n-1}w_ja_j=\sum_{j=1}^{n-1}w_jb_j$ a $a_i\ne b_i$ pro nějaké $1\le i\le n-1$. Zvolme $\varepsilon>0$ tak, že

$$\varepsilon<\min\left(\delta,\min\left\{\left|\sum_{j=1}^{n-1}w_jx_j-\sum_{j=1}^{n-1}w_jx'_j\right|>0\ \middle|\ \mathbf{x},\mathbf{x}'\in\{0,1\}^{n-1}\right\}\right). \quad (6.24)$$

Z δ-separability původní reprezentace vyplývá, že $(w_1,\ldots,w_{i-1},w_i+\varepsilon,w_{i+1},\ldots,w_{n-1};0)$ je také reprezentace f. Navíc pro ní platí (6.23) pro vstupy $(a_1,\ldots,a_{n-1})$, $(b_1,\ldots,b_{n-1})$, protože příslušné zvážené sumy se v nerovnici (6.23) liší o ε. Díky volbě ε v (6.24) se platnost (6.23) pro další dvojice vstupů nezmění. Celý postup se opakuje, dokud (6.23) neplatí pro všechny dvojice vstupů.

Nechť tedy reprezentace $(w_1,\ldots,w_{n-1};0)$ funkce f splňuje (6.23). Potom pro každé $k=0,\ldots,2^{n-1}$ můžeme volit váhu $w_n^{(k)}$ tak, aby počet vstupů $(x_1,\ldots,x_{n-1},1)\in\{0,1\}^n$, pro které booleovská prahová funkce f_k s n proměnnými, zadaná reprezentací $(w_1,\ldots,w_{n-1},w_n^{(k)};0)$, nabývá hodnotu 0, byl právě k, tj.

$$\left|\left\{(x_1,\ldots,x_{n-1},1)\in\{0,1\}^n\ \middle|\ \sum_{i=1}^{n-1}w_ix_i+w_n^{(k)}<0\right\}\right|=k \quad (6.25)$$

pro $0\le k\le 2^{n-1}$. Tím získáme z jedné booleovské prahové funkce f s $n-1$ proměnnými $2^{n-1}+1$ různých booleovských prahových funkcí f_k $(k=0,\ldots,2^{n-1})$ s n proměnnými. Počet různých voleb f je $C(n-1)$ a každá z nich určuje jiná f_k, protože $f_k(x_1,\ldots,x_{n-1},0)=f(x_1,\ldots,x_{n-1})$ se liší pro nějaké $(x_1,\ldots,x_{n-1},0)\in\{0,1\}^n$. Odtud dostáváme rekurentní vztah pro dolní odhad $C(n)\ge(2^{n-1}+1)C(n-1)$, který lze indukcí dle n rozepsat:

$$C(n)\ge\prod_{i=0}^{n-1}(2^i+1)>2^{\frac{n(n-1)}{2}}\,. \quad (6.26)$$

□

Dolní odhad ve větě 6.31 lze např. vylepšit na $2^{\frac{n(n-1)}{2}+16}$ tak, že v jejím důkazu využijeme pozorování, že $C(8)>2^{44}$ [202]. Následující věta však ukazuje, že kvadratický exponent v uvedeném dolním odhadu nelze již řádově zvýšit.

Věta 6.32 *Počet různých nedegenerovaných booleovských prahových funkcí n proměnných je méně než $2^{(n+1)^2}$.*

Důkaz: Chceme dokázat, že neexistuje celočíselná reprezentace f s váhou menší než 2^k. Nechť tedy $(w_1, \ldots, w_n; h)$ je libovolná celočíselná reprezentace f. Podle věty 6.12 můžeme předpokládat kladné váhy $w_i > 0$ $(i = 1, \ldots, n)$. Ukážeme, že $w_{k+1} \geq 2^k$. Za tímto účelem definujme následující vstupy $\mathbf{a}_j, \mathbf{b}_j \in \{0,1\}^n$ $(j = 1, \ldots, k)$ pro f:

$$\mathbf{a}_j = (\overbrace{\underbrace{0, \ldots, 0}_{j}, 1, 0, \ldots, 0}^{k+1}, \overbrace{1, \ldots, 1, \underbrace{0, \ldots, 0}_{j}}^{k}) \tag{6.30}$$

$$\mathbf{b}_j = (\overbrace{\underbrace{1, \ldots, 1}_{j}, 0, 0, \ldots, 0}^{k+1}, \overbrace{1, \ldots, 1, \underbrace{0, \ldots, 0}_{j}}^{k}) . \tag{6.31}$$

S využitím reprezentace (6.29) funkce f dostaneme, že $f(\mathbf{a}_j) = 1$ $(j = 1, \ldots, k)$, protože odpovídající zvážená suma $2^j + (2^k - 1 - \sum_{i=0}^{j-1} 2^i) = 2^k$, a $f(\mathbf{b}_j) = 0$ $(j = 1, \ldots, k)$, protože $\sum_{i=0}^{j-1} 2^i + (2^k - 1 - \sum_{i=0}^{j-1} 2^i) = 2^k - 1$. Tedy pro reprezentaci $(w_1, \ldots, w_n; h)$ platí, že

$$w_{j+1} + \sum_{i=k+2}^{n-j} w_i \geq h \quad j = 1, \ldots, k\,, \tag{6.32}$$

protože $f(\mathbf{a}_j) = 1$, a

$$\sum_{i=1}^{j} w_i + \sum_{i=k+2}^{n-j} w_i < h \quad j = 1, \ldots, k\,, \tag{6.33}$$

protože $f(\mathbf{b}_j) = 0$. K nerovnosti (6.32) přičteme nerovnost (6.33) vynásobenou -1 a obdržíme:

$$w_{j+1} > \sum_{i=1}^{j} w_i \quad j = 1, \ldots, k\,. \tag{6.34}$$

Víme, že váhy w_j jsou celá kladná čísla, tedy např. $w_1 \geq 1$. Indukcí pomocí (6.34) lze nahlédnout, že $w_{j+1} \geq 2^j$ pro $j = 1, \ldots, k$, speciálně $w_{k+1} \geq 2^k$. □

Bez důkazu uvedeme ještě jeden příklad booleovské prahové funkce s váhou větší než 1.61^n, což je více než ve větě 6.35.

Věta 6.36 *Nechť $F_1 = F_2 = 1$ a $F_n = F_{n-2} + F_{n-1}$ pro $n \geq 3$ jsou Fibonacciho čísla. Pak booleovská prahová funkce s reprezentací $(F_1, \ldots, F_n; F_{n+1})$ má váhu aspoň $(\frac{1+\sqrt{5}}{2})^n/\sqrt{5}$.*

Důkaz: Podle důkazu věty 6.20 odpovídá každá nedegenerovaná booleovská prahová funkce v $(n+1)$-rozměrném euklidovském prostoru reprezentací E_{n+1} aspoň jednomu průniku $n+1$ nadrovin. Každé dvě takové (rovnoběžné) nadroviny (6.15) odpovídají jednomu možnému vstupu, což v případě binárního vstupu představuje 2^{n+1} nadrovin. Počet uvedených průniků nadrovin lze tedy shora odhadnout pomocí

$$\binom{2^{n+1}}{n+1} < \frac{2^{(n+1)^2}}{(n+1)!} < 2^{(n+1)^2}\,. \tag{6.27}$$

Tedy existuje méně než $2^{(n+1)^2}$ různých nedegenerovaných booleovských prahových funkcí n proměnných. □

Odhad ve větě 6.32 lze opět vylepšit na $2^{n^2 - O(n \log n)}$, pokud v nerovnosti (6.27) odhadneme faktoriál pomocí Stirlingova vzorce. Nakonec z vět 6.31, 6.32 dostáváme odhad počtu perceptronů s n binárními vstupy, které počítají různé funkce.

Důsledek 6.33 *Existuje právě $2^{\Theta(n^2)}$ booleovských prahových funkcí n proměnných.*

Důsledek 6.34 *Existuje booleovská prahová funkce n proměnných s váhou $|w|$ takovou, že*

$$2^{\frac{n-1}{2}} \leq |w| \leq \frac{(n+1)^{\frac{n+1}{2}}}{2^n}\,, \tag{6.28}$$

tj. na reprezentaci jedné váhy booleovské prahové funkce je v nejhorším případě potřeba aspoň $\Omega(n)$ bitů a stačí $O(n \log n)$ bitů.

Důkaz: Kdyby všechny booleovské prahové funkce n proměnných měly váhu $|w| < 2^{\frac{n-1}{2}}$, tj. na reprezentaci jedné váhy by u nich stačilo $\frac{n-1}{2}$ bitů, pak by jich bylo méně než $2^{\frac{n(n-1)}{2}}$, což je ve sporu s větou 6.31. Horní odhad váhy udává věta 6.30. □

Argument v důkazu důsledku 6.34 neposkytuje žádný příklad booleovské prahové funkce s velkou váhou. Následující věta popisuje takový příklad booleovské prahové funkce s váhou vyšší než 1.41^n.

Věta 6.35 *Nechť n je liché číslo a označme $k = \frac{n-1}{2}$. Pak booleovská prahová funkce f s n proměnnými, zadaná reprezentací*

$$(1, 2, 4, \ldots, 2^{k-1}, 2^k, 2^{k-1}, \ldots, 4, 2, 1; 2^k)\,, \tag{6.29}$$

má váhu 2^k.

α	$\Phi(\alpha)$							
	$\mathbf{b}_1$ (-1,-1,-1)	$\mathbf{b}_2$ (-1,-1,1)	$\mathbf{b}_3$ (-1,1,-1)	$\mathbf{b}_4$ (-1,1,1)	$\mathbf{b}_5$ (1,-1,-1)	$\mathbf{b}_6$ (1,-1,1)	$\mathbf{b}_7$ (1,1,-1)	$\mathbf{b}_8$ (1,1,1)
$\alpha_1 = \emptyset$	1	1	1	1	1	1	1	1
$\alpha_2 = \{3\}$	-1	1	-1	1	-1	1	-1	1
$\alpha_3 = \{2\}$	-1	-1	1	1	-1	-1	1	1
$\alpha_4 = \{1\}$	-1	-1	-1	-1	1	1	1	1
$\alpha_5 = \{1,2\}$	1	1	-1	-1	-1	-1	1	1
$\alpha_6 = \{1,3\}$	1	-1	1	-1	-1	1	-1	1
$\alpha_7 = \{2,3\}$	1	-1	-1	1	1	-1	-1	1
$\alpha_8 = \{1,2,3\}$	-1	1	1	-1	1	-1	-1	1

Table 6.1: Příklad uspořádání $\mathcal{P}(M)$ s minimální změnou určující Φ.

Velikost váhy u příkladů booleovské prahové funkce ve větách 6.35, 6.36 zhruba odpovídá dolnímu odhadu v důsledku 6.34, který však zdaleka nedosahuje úrovně horního odhadu ve větě 6.30. Otázka zlepšení dolního, resp. horního odhadu v důsledku 6.34 představovala dlouho otevřený problém, který se podařilo vyřešit teprve nedávno. Na konci tohoto odstavce definujeme příklad booleovské prahové funkce s dosud největší známou váhou a zformulujeme příslušné tvrzení bez důkazu, který je netriviální.

Definice 6.37 *Nechť $n = 2^m$, $M = \{1, \ldots, m\}$ a $\alpha_1, \ldots, \alpha_n \subseteq M$ jsou všechny různé podmnožiny M. Dá se ukázat, že pro každé m existuje uspořádání $\alpha_1, \ldots, \alpha_n$ s* minimální změnou*, pro které platí:*

1. $|\alpha_i| \leq |\alpha_{i+1}| \quad i = 1, \ldots, n-1$
2. $|\alpha_i \Delta \alpha_{i+1}| \leq 2 \quad i = 1, \ldots, n-1\,,$

kde $\alpha_i \Delta \alpha_{i+1} = (\alpha_i \setminus \alpha_{i+1}) \cup (\alpha_{i+1} \setminus \alpha_i)$ je symetrická diference množin α_i, α_{i+1}. Dále uvažujeme bipolární logiku (viz poznámku k definici 6.24) a označme $\mathbf{b}_i = (b_{i1}, \ldots, b_{im}) \in \{-1, 1\}^m$ bipolární reprezentaci celého kladného čísla $i-1$ ($1 \leq i \leq n$), tj. binární zápis čísla $i-1$, ve kterém číslici 0 nahradíme -1. Nyní definujme zobrazení $\Phi : \mathcal{P}(M) \longrightarrow \{-1, 1\}^n$ potenční množiny $\mathcal{P}(M) = \{\alpha \subseteq M\}$ do množiny n-rozměrných bipolárních vektorů $\{-1, 1\}^n$:

$$\Phi(\alpha) = \left(\prod_{j \in \alpha} b_{1j}, \ldots, \prod_{j \in \alpha} b_{ij}, \ldots, \prod_{j \in \alpha} b_{nj} \right) \quad \alpha \subseteq M\,. \tag{6.35}$$

Konečně definujme bipolární funkci $F : \{-1, 1\}^n \longrightarrow \{-1, 1\}$:

$$F(\mathbf{x}) = \begin{cases} -1 & \mathbf{x} \cdot \Phi(\alpha_k) < 0 \\ 1 & \mathbf{x} \cdot \Phi(\alpha_k) > 0 \end{cases} \qquad \mathbf{x} \in \{-1, 1\}^n\,, \tag{6.36}$$

kde $k = \max\{i \,|\, \mathbf{x} \cdot \Phi(\alpha_i) \neq 0,\, 1 \leq i \leq n\}$, jehož existenci lze ukázat.

V tabulce 6.1 je příklad uspořádání $\mathcal{P}(M)$ s minimální změnou určující zobrazení Φ pro případ $n = 8$, $m = 3$ a $M = \{1, 2, 3\}$. Tedy například $F(1,1,1,1,1,1,1,1) = 1$ ($k = 1$) nebo $F(-1,1,1,-1,-1,1,1,-1) = -1$ ($k = 7$) apod.

Věta 6.38 (Håstad [96]) *Funkce F z definice 6.37 je booleovská prahová funkce n proměnných s váhou $|w|$, pro kterou platí:*

$$|w| \geq \frac{n^{\frac{n-1-\log n}{2}}}{2^{n-1}} e^{\gamma(\log^2 n + 3\log n + 8 - 8n^{\beta})} (\log^2 n - \log n + 1)\,, \tag{6.37}$$

kde $\beta = \log \frac{3}{2} \doteq 0.585$ a $\gamma = \ln \frac{4}{3} \doteq 0.288$.

prahová, právě když neexistuje řešení následující soustavy 2^n lineárních nerovnic s $n+1$ neznámými $w_1, \ldots, w_n, h$, které tvoří reprezentaci f:

$$\sum_{i=1}^{n} x_i w_i \begin{cases} \geq h & \text{pro } f(x_1,\ldots,x_n)=1 \\ < h & \text{pro } f(x_1,\ldots,x_n)=0 \end{cases} \quad (x_1,\ldots,x_n) \in \{0,1\}^n . \quad (6.39)$$

Soustava (6.39) tvoří soubor $k = 2^n$ konvexních množin v euklidovském prostoru E_d dimenze $d = n+1$. Proto podle věty 6.40 nemá řešení, právě když existuje podsystém $n+2$ řádků této soustavy, který nemá řešení. Nedeterministický algoritmus uhádne těchto $n+2$ řádků a pomocí lineárního programování ověří v polynomiálním čase [145], zda příslušný podsystém nemá řešení.

Důkaz, že $LINSEP$ je $co-NP$-těžký problém, probíhá standardním postupem [72], tj. polynomiální redukcí známého $co-NP$-úplného problému, v našem případě *problému tautologie* $DNF-TAUT$, na problém $LINSEP$.

Problém tautologie $DNF-TAUT$ (**TAUT**ology Problem):
instance: Booleovská funkce $g : \{0,1\}^m \longrightarrow \{0,1\}$ s m proměnnými v disjunktivní normální formě.
otázka: Je g tautologie, tj. pro všechna ohodnocení booleovských proměnných je hodnota $g \equiv 1$?

Je všeobecně známo, že $DNF-TAUT$ je $co-NP$-úplný problém [47]. Nechť tedy $g : \{0,1\}^m \longrightarrow \{0,1\}$ je instance $DNF-TAUT$. V polynomiálním čase zkonstruujeme odpovídající instanci $f_g : \{0,1\}^n \longrightarrow \{0,1\}$ problému $LINSEP$ s $n = m+2$ proměnnými takovou, že g je tautologie, právě když f_g je booleovská prahová funkce:

$$f_g(\mathbf{x}, x_{m+1}, x_{m+2}) = (g(\mathbf{x}) \wedge x_{m+2}) \vee (x_{m+1} \wedge x_{m+2}) \vee (\bar{x}_{m+1} \wedge \bar{x}_{m+2}) , \quad (6.40)$$

kde $\mathbf{x} \in \{0,1\}^m$ a $x_{m+1}, x_{m+2} \in \{0,1\}$. Zřejmě f_g po rozepsání konjunkce $g(\mathbf{x}) \wedge x_{m+2}$ v (6.40) je v disjunktivní normální formě, protože g je v disjunktivní normální formě.

Nechť tedy nejprve g je tautologie. Potom formuli (6.40) můžeme díky $g \equiv 1$ upravit:

$$\begin{aligned} f_g(\mathbf{x}, x_{m+1}, x_{m+2}) &= x_{m+2} \vee (x_{m+1} \wedge x_{m+2}) \vee (\bar{x}_{m+1} \wedge \bar{x}_{m+2}) = \\ &= \bar{x}_{m+1} \vee x_{m+2} , \end{aligned} \quad (6.41)$$

a tedy např. $(0,\ldots,0,-1,1;0)$ je reprezentace f_g, která dosvědčuje, že f_g je lineární prahová funkce. Naopak nechť g není tautologie, tj. existuje $\mathbf{a} \in \{0,1\}^m$ takové, že $g(\mathbf{a}) = 0$. Potom ale podle (6.40) platí, že

$$\begin{aligned} f_g(\mathbf{a}, x_{m+1}, x_{m+2}) &= (x_{m+1} \wedge x_{m+2}) \vee (\bar{x}_{m+1} \wedge \bar{x}_{m+2}) = \\ &= \overline{XOR}(x_{m+1}, x_{m+2}) , \end{aligned} \quad (6.42)$$

Dolní odhad váhy F ve větě 6.38 je jen o faktor $2^{O(n^\beta)}$ menší než horní odhad váhy booleovské prahové funkce ve větě 6.30.

Důsledek 6.39 *Pro reprezentaci jedné váhy booleovské prahové funkce n proměnných je v nejhorším případě potřeba a stačí $\Theta(n \log n)$ bitů.*

Důsledek 6.39 znamená, že při implementaci libovolného perceptronu s n binárními vstupy stačí na reprezentaci jeho n vah $\Theta(n^2 \log n)$ bitů, přitom obecně nelze tento odhad zlepšit.

6.4.3 Problém lineární separability

Věta 6.26 ukazuje několik příkladů klasických booleovských funkcí, které jsou prahové. V tomto odstavci ukážeme, že rozhodnout algoritmicky, zda daná booleovská funkce v normální formě je prahová, je obecně těžký problém. Tento výsledek má své důsledky v teorii učení neuronových sítí (viz podkapitolu 11.1).

Problém lineární separability $LINSEP$ (**LIN**ear **SEP**arability Problem):
instance: Booleovská funkce $f : \{0,1\}^n \longrightarrow \{0,1\}$ s n proměnnými v disjunktivní normální formě.
otázka: Je f booleovská prahová funkce?

Důkaz příslušného tvrzení o složitosti problému $LINSEP$ využívá klasickou větu z geometrie euklidovských prostorů o průniku konvexních množin.

Věta 6.40 (Helly [107]) *Nechť $\{Q_1, \ldots, Q_k\}$ je konečný soubor k různých konvexních množin v euklidovském prostoru E_d dimenze d a nechť $k \geq d+1$. Potom*

$$\bigcap_{i=1}^{k} Q_i = \emptyset , \quad \textit{právě když}$$

$$(\exists\, 1 \leq i_1 < i_2 < \cdots < i_{d+1} \leq k) \bigcap_{j=1}^{d+1} Q_{i_j} = \emptyset . \quad (6.38)$$

Věta 6.41 (Hegedüs, Megiddo [102]) *$LINSEP$ je $co-NP$-úplný problém.*

Důkaz: Nejprve ukážeme, že $LINSEP \in co-NP$, tj. že existuje nedeterministický algoritmus, který v polynomiálním čase rozhodne, zda f *není* booleovská prahová funkce. Podobně jako v důkazu věty 6.20 funkce f není

a tedy podle (i') z lemmy 6.27 a věty 6.26 f_g není booleovská prahová funkce. □

Problém $LINSEP$ je $co-NP$-úplný i pro booleovské funkce v konjunktivní normální formě, které lze efektivně převést na negaci disjunktivní normální formy a využít faktu, že třída booleovských prahových funkcí je podle (i') z lemmy 6.27 uzavřená na negaci.

Kapitola 7

Složitost obvodů

Acyklická neuronová síť (viz odstavec 1.3.2) odpovídá v teorii booleovské složitosti obecnému *obvodu*, který se skládá z tzv. *hradel* reprezentujících neurony. Obecně uvažujeme dostatečně širokou *třídu hradlových funkcí*, abychom pokryli co nejvíce modelů acyklických neuronových sítí. Nejprve uvedeme formální definici obvodu a jeho výpočtu.

Definice 7.1 Obvod *je pětice* $C = (V, X, Y, E, \ell)$, *kde* V *je konečná množina* hradel, X ($X \cap V = \emptyset$) *je konečná množina* $n = |X|$ vstupů *a* $Y \subseteq V$ *množina* $m = |Y|$ výstupů, $(V \cup X, E)$ *je acyklický orientovaný graf s množinou vrcholů* $V \cup X$ *a množinou hran* $E \subseteq (V \cup X) \times V$, *který spolu s* X *a* Y *tvoří* architekturu (topologii) $A = (V, X, Y, E)$ *obvodu, a zobrazení* $\ell : V \longrightarrow \mathcal{F}$ *je* konfigurace, *která každému hradlu přiřadí jeho funkci z dané třídy* $\mathcal{F}$ hradlových funkcí *(navíc požadujeme, aby funkce výstupních hradel měly binární obor hodnot). Počet hradel obvodu* $s = |V|$ *(nepočítáme vstupy) se nazývá* velikost *obvodu. Řekneme, že hradlo (resp. vstup) obvodu se nachází v i-té* vrstvě *(*$i \geq 0$*), pokud délka (počet hran) nejdelší orientované cesty od vstupu k tomuto hradlu je i hran. Počet nevstupních vrstev, tj. délka nejdelší orientované cesty od vstupu k výstupu, určuje* hloubku *obvodu* d. *Pokud neuvedeme jinak, předpokládáme, že poslední vrstva se skládá z výstupních hradel* Y.

V definici 7.1 můžeme tedy obecný obvod C interpretovat jako acyklickou neuronovou síť s množinou (nevstupních) neuronů V, které počítají libovolné funkce z třídy hradlových funkcí $\mathcal{F}$ předepsané pomocí ℓ. Množina X obsahuje vstupní neurony, množina Y představuje výstupní neurony a E je množina spojů neuronů. Víme, že acyklickou neuronovou síť lze vždy strukturovat do vrstev (např. vícevrstvá neuronová síť), a podobně uvažujeme rozdělení obvodu na $d+1$ vrstev (tj. včetně vstupní vrstvy) a výpočet po-

mocí obvodu pak také probíhá po vrstvách. Při výpočtu lze hodnoty výstupů hradel interpretovat jako stavy neuronů v aktivním režimu neuronové sítě.

Definice 7.2 *Při výpočtu obvodu* $C = (V, X, Y, E, \ell)$ *hloubky* d *jsou nejprve zadány obecně reálné hodnoty vstupů z* X *v nulté vrstvě. Pokud neuvedeme jinak, budeme vždy uvažovat binární vstupy* $x_1, \ldots, x_n \in \{0,1\}$. *Poté jsou vypočteny výstupy hradel v první vrstvě aplikací příslušných funkcí pro hodnoty těchto vstupů, obdobně ve druhé vrstvě atd. Obecně v i-tém kroku (*$i = 1, \ldots, d$*) jsou vypočteny výstupy hradel v i-té vrstvě na základě výstupů hradel v předchozích vrstvách (popř. vstupů obvodu). Na konci jsou určeny výstupy všech hradel v* Y, *které určují výsledek výpočtu. Definujeme tak* funkci obvodu $C : \{0,1\}^n \longrightarrow \{0,1\}^m$ *(obecně* $C : \Xi_n \longrightarrow \{0,1\}^m$, *kde* $\Xi_n \subseteq \mathbb{R}^n$*), která ke každému vstupu* $\mathbf{x} \in \{0,1\}^n$ *(*$n = |X|$*) přiřadí výstup* $C(\mathbf{x}) \in \{0,1\}^m$ *(*$m = |Y|$*). Řekneme, že dva obvody jsou* ekvivalentní, *pokud počítají stejnou funkci.*

Jak jsme již uvedli, pro jednotlivé třídy hradlových funkcí dostáváme modely neuronových sítí s různou aktivní dynamikou. V této kapitole se nejprve zaměříme na nejjednodušší případ logických obvodů, které se nejčastěji studují v teorii booleovské složitosti a které mohou také posloužit jako metodický základ pro následný výklad prahových obvodů, které jsou bližší modelu vícevrstvého perceptronu.

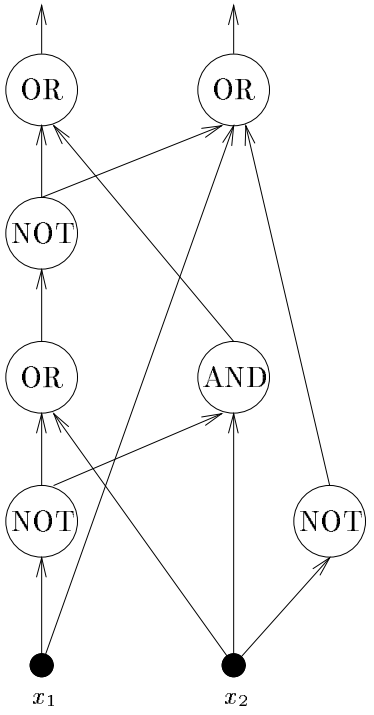
7.1 Logické obvody

Pokud jsou funkce hradel v obvodu omezeny jen na základní logické funkce (AND, OR, NOT z definice 6.25), hovoříme o tzv. *logickém obvodu*. Díky větě 6.26 představuje logický obvod jednoduchou diskrétní acyklickou neuronovou síť s binárními stavy neuronů. V této podkapitole budeme nejprve zkoumat výpočetní sílu a deskriptivní složitost (tj. velikost a hloubku) logického obvodu pro výpočet obecné booleovské funkce, popř. součtu celých čísel. Kvůli hardwarové realizaci se také tradičně uvažuje tzv. *klasický obvod*, který navíc u každého hradla předpokládá omezený počet vstupů (nejvýše 2). Avšak toto omezení již není v souladu s modely neuronových sítí, jejichž základní charakteristikou, korespondující s neurofyziologickou skutečností, je velká hustota propojení neuronů. Budeme se zabývat porovnáním deskriptivní složitosti těchto modelů. Nyní zformalizujeme uvedené pojmy.

Definice 7.3 *Řekneme, že obvod* $C = (V, X, Y, E, \ell)$ *(viz definici 7.1) je* logický obvod, *jestliže* $\mathcal{F} = \{AND, OR, NOT, 0, 1\}$ *je třída jeho hradlových funkcí, kde 0, 1 jsou označení pro příslušné konstantní funkce. Logický obvod se nazývá* klasický obvod, *jestliže počet vstupů každého jeho hradla je nejvýše 2.* Alternující obvod *je logický obvod, u něhož zobrazení* ℓ *přiřazuje všem*

hradlům v jedné vrstvě buď funkci AND, nebo funkci OR, a to střídavě po vrstvách, přitom hradla s funkcí NOT jsou formálně v X, tj. každý vstup x_i $(i = 1, \ldots, n)$ je navíc sdružen s přidaným vstupem $\bar{x}_i$ odpovídajícím jeho negaci.

Příklad logického obvodu je na obrázku 7.1.



Obr. 7.1: Příklad logického obvodu.

7.1.1 Alternující obvody

Ukážeme, že obecný logický obvod lze převést na normální tvar, tj. na alternující obvod, za cenu nejvýš zdvojnásobení velikosti.

Věta 7.4 *Pro každý logický obvod C velikosti s a hloubky d existuje ekvivalentní alternující obvod C' velikosti nejvýše $2s$ a hloubky nejvýše $d+1$.*

Důkaz: Při normalizaci logického obvodu $C = (V, X, Y, E, \ell)$ přesuneme všechna hradla, která počítají negaci, do vstupní vrstvy následujícím způsobem. Nejprve ke každému vstupu x_i $(i = 1, \ldots, n)$ přidáme podle definice alternujícího obvodu jeho negaci $\bar{x}_i$, tj. $X' = X \cup \{\bar{x}_1, \ldots, \bar{x}_n\}$. Dále zrušíme všechna hradla g s funkcí $\ell(g) = NOT$, tj. $\bar{V} = V \setminus \{g \in V \mid \ell(g) = NOT\}$. Poté ke každému hradlu $g \in \bar{V}$ přidáme do stejné vrstvy hradlo $\bar{g}$, které bude počítat jeho negaci, aniž by přímo počítalo funkci NOT, tj. $\bar{V}' = \bar{V} \cup \{\bar{g} \mid g \in \bar{V}\}$. Při výpočtu výstupu hradla $\bar{g}$ jsou totiž díky de Morganovým vzorcům (viz poznámku k lemmě 6.27) případně potřeba jen negace vstupů obvodu a dále výstupy hradel z předchozích vrstev, které však již také obsahují hradla počítající negace podobným způsobem, tj. bez použití funkce NOT. Podle toho novým hradlům přiřadíme následující funkce:

$$\ell'(\bar{g}) = \begin{cases} AND & \ell(g) = OR \\ OR & \ell(g) = AND \\ 1 & \ell(g) = 0 \\ 0 & \ell(g) = 1\,, \end{cases} \tag{7.1}$$

zatímco funkce původních hradel se nemění, tj. $\ell'(g) = \ell(g)$ pro $g \in \bar{V}$. Také musíme přizpůsobit architekturu obvodu. Ke každé hraně $(g_1, g_2) \in E$ pro $g_1, g_2 \in \bar{V} \cup X$ přidáme hranu $(\bar{g}_1, \bar{g}_2)$. Navíc pokud současně $(g_1, g), (g, g_2) \in E$ pro $\ell(g) = NOT$ a $g_1, g_2 \in \bar{V} \cup X$, pak v upravené topologii budou příslušná hradla spojena hranami $(g_1, \bar{g}_2)$ a $(\bar{g}_1, g_2)$. Tak získáme množinu hran

$$\begin{aligned} \bar{E} \;&=\; (E \cap ((\bar{V} \cup X) \times \bar{V})) \cup \\ &\cup\; \{(\bar{g}_1, \bar{g}_2) \mid (g_1, g_2) \in E;\, g_1, g_2 \in \bar{V} \cup X\} \cup \\ &\bigcup \left\{ \begin{array}{ll|ll} (g_1, \bar{g}_2) & & (g_1, g), (g, g_2) \in E & g \in V \\ (\bar{g}_1, g_2) & & \ell(g) = NOT & g_1, g_2 \in \bar{V} \cup X \end{array} \right\}. \end{aligned} \tag{7.2}$$

Tím získáme logický obvod $\bar{C} = (\bar{V}', X', Y', \bar{E}, \ell')$ velikosti nejvýše 2s, který má negace jen na vstupu a je ekvivalentní s C.

Avšak v alternujícím obvodu se navíc vrstvy hradel počítající AND střídají s vrstvami hradel OR. Tomu přizpůsobíme $\bar{C}$ tak, že vypustíme každé hradlo $g_1 \in \bar{V}'$, které je spojeno s hradlem $g_2 \in \bar{V}'$ se stejnou funkcí $\ell(g_2) = \ell(g_1)$. Hrany $(h, g_1) \in \bar{E}$, které vedou k hradlu g_1, pak můžeme díky asociativitě operací AND a OR přesměrovat k hradlu g_2, tj.

$$V' \;=\; \bar{V}' \setminus \{g_1 \mid (g_1, g_2) \in \bar{E},\, \ell(g_1) = \ell(g_2)\} \tag{7.3}$$

$$\begin{aligned} E' \;&=\; (\bar{E} \cap ((V' \cup X') \times V')) \cup \\ &\bigcup \left\{ (h, v_k) \in \bar{E} \;\middle|\; \begin{array}{l} \exists\, (v_i, v_{i+1}) \in \bar{E}\;\; \ell(v_i) = \ell(v_k) \\ (h, v_i) \in \bar{E},\; i = 1, \ldots, k-1 \\ h \in V' \cup X',\; v_k \in V' \end{array} \right\}. \end{aligned} \tag{7.4}$$

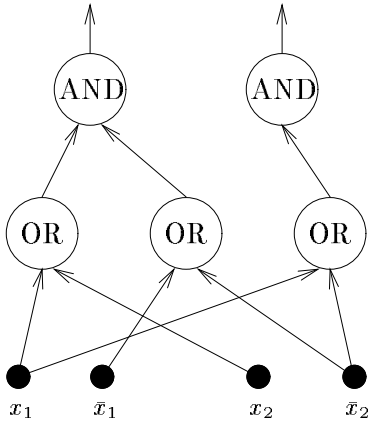
velikosti s_1, hloubky d_1 a $C_2 : \{0,1\}^{m'} \longrightarrow \{0,1\}^m$ je funkce alternujícího obvodu C_2 velikosti s_2, hloubky d_2 a hradla z poslední vrstvy C_1 a hradla z první vrstvy C_2 počítají stejné funkce. Potom složené zobrazení $C_2 \circ C_1 : \{0,1\}^n \longrightarrow \{0,1\}^m$ lze počítat alternujícím obvodem velikosti $s_1 + s_2$ a hloubky $d_1 + d_2 - 1$.

(ii) inverze: *Každé zobrazení $C : \{0,1\}^n \longrightarrow \{0,1\}^m$, které lze počítat alternujícím obvodem C hloubky 2 s p hradly AND v první vrstvě a m hradly OR ve druhé vrstvě, lze počítat alternujícím obvodem C' hloubky 2 a velikosti $n^p m + m$, který má hradla OR v první vrstvě a hradla AND ve druhé vrstvě. Tvrzení platí i při záměně AND a OR.*

Důkaz:

(i) Alternující obvod, který počítá složené zobrazení $C_2 \circ C_1$, vznikne spojením obvodů C_1 a C_2 tak, že díky stejným hradlům první vrstva C_2 splyne s poslední vrstvou C_1 podobně jako v (7.3) a (7.4), čímž se součet hloubek zmenší o 1.

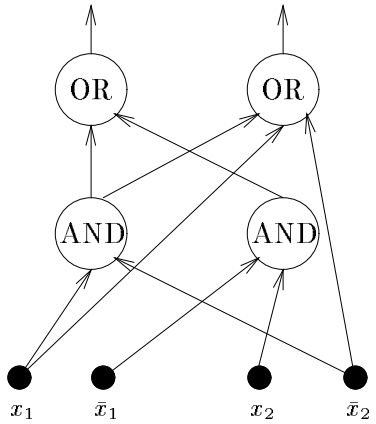
(ii) Uvažujme příslušnou disjunktivní normální formu (s p monomy) booleovské funkce n proměnných, kterou počítá jedno z m výstupních hradel alternujícího obvodu C. Pomocí distributivního zákona ji převedeme na konjunktivní normální formu s n^p klauzulemi, kterou realizujeme alternujícím obvodem hloubky 2 a velikosti $n^p + 1$. Alternující obvod C' se skládá z m takových obvodů, tedy C' má požadovanou velikost $n^p m + m$. Příklad konstrukce obvodu C' je na obrázku 7.3 □



Obr. 7.3: Inverze alternujícího obvodu z obrázku 7.2.

Nyní všechny cesty od vstupu k výstupu střídají hradla AND a OR, avšak nemusí začínat stejných hradlem. Abychom tento nesoulad v alternujícím obvodu odstranili, první vrstvu v tomto případě chápeme formálně jako dvě vrstvy. Tím získáme výsledný alternující obvod $C' = (V', X', Y', E', \ell')$ velikosti nejvýše $2s$ a hloubky nejvýše $d + 1$, který je ekvivalentní s C. □

Definice 7.1, 7.2 (logického) obvodu jsou formálně poměrně komplikované, a proto formální důkaz věty 7.4 je zbytečně dlouhý, ačkoliv jeho myšlenka je celkem jednoduchá. Proto se dále tam, kde to nebude na úkor srozumitelnosti, omezíme pouze na neformální důkaz. Příklad alternujícího obvodu, který je podle věty 7.4 ekvivalentní s obecným logickým obvodem na obrázku 7.1, je znázorněn na obrázku 7.2. Díky této větě se v následujícím výkladu budeme většinou zabývat logickými obvody, které jsou alternující.



Obr. 7.2: Alternující obvod ekvivalentní s logickým obvodem na obrázku 7.1.

Zformulujeme lemmu, která představí dvě triviální techniky (kompresi a inverzi), které se uplatní při konstrukcích alternujících obvodů. Předtím si ještě uvědomíme, že alternující obvod s jedním výstupním hradlem ($|Y| = 1$) hloubky 2 lze chápat jako normální formu příslušné booleovské formule, kdy vstupy včetně jejich negací odpovídají literálům, hradla první vrstvy počítají OR (klauzule) v případě konjunktivní normální formy, resp. AND (monomy) při disjunktivní normální formě.

Lemma 7.5

(i) komprese: *Nechť $n, m', m \in \mathbb{N}$ jsou přirozená čísla. Dále předpokládejme, že $C_1 : \{0,1\}^n \longrightarrow \{0,1\}^{m'}$ je funkce alternujícího obvodu C_1*

Tedy $x_i \oplus y_i$ a $\overline{x_i \oplus y_i}$ $(1 \leq i \leq n)$ lze vyčíslit v prvních dvou vrstvách alternujícího obvodu. Dále $(x_i \oplus y_i) \oplus c_{i+1}$ $(1 \leq i < n)$ můžeme opět podle (7.7) počítat ve třetí a čtvrté vrstvě tak, že třetí vrstvu s hradly OR vytvoříme z třetí vrstvy obvodu pro c_i, resp. $\bar{c}_i$, pomocí komprese podle (i) z lemmy 7.5. Tak obdržíme alternující obvod velikosti $O(n^2)$ a hloubky 4 pro výpočet požadovaného součtu, který má ve čtvrté výstupní vrstvě $n+1$ hradel AND (pro výpočet z_i, $i = 1, \ldots, n+1$), každé se dvěma vstupy od hradel OR (pro výpočet $(x_i \oplus y_i) \vee c_{i+1}$ a $(\overline{x_i \oplus y_i}) \vee \bar{c}_{i+1}$, $1 \leq i < n$) ze třetí vrstvy, které mají nejvýše n vstupů ($x_i \oplus y_i$ a f_{ij}, resp. $\overline{x_i \oplus y_i}$ a e_{ij} pro $j = i, \ldots, n$). Pomocí inverze podle (ii) z lemmy 7.5 můžeme třetí a čtvrtou vrstvu vyměnit tak, že nový obvod má $n+1$ výstupních hradel OR ve čtvrté vrstvě pro výpočet z_i $(i = 1, \ldots, n+1)$ a $O(n^3)$ hradel AND hradel ve třetí vrstvě pro výpočet $f_{ij} \wedge e_{ik}$ $(1 \leq i < n, i \leq j, k \leq n)$ a $(x_i \oplus y_i) \wedge e_{ij}$, resp. $(\overline{x_i \oplus y_i}) \wedge f_{ij}$ a $(x_i \oplus y_i) \wedge (\overline{x_i \oplus y_i})$. Výsledný alternující obvod velikosti $O(n^3)$ a hloubky 3 pro výpočet požadovaného součtu dostaneme pomocí komprese druhé a třetí vrstvy s hradly AND podle (i) v lemmě 7.5. □

7.1.2 Implementace booleovské funkce

Alternující obvod s jedním výstupním hradlem ($|Y| = 1$) budeme využívat k výpočtu obecné booleovské funkce $f : \{0,1\}^n \longrightarrow \{0,1\}$ s n proměnnými, které odpovídají vstupům obvodu. Ukážeme jednoduché pozorování, že pomocí alternujícího obvodu exponenciální velikosti a hloubky 2, tj. pomocí dvouvrstvé neuronové sítě s exponenciálním počtem neuronů (negace na vstupu lze dle lemmy 6.27 realizovat neurony v první vrstvě), můžeme již počítat každou booleovskou funkci.

Věta 7.7 *Každou booleovskou funkci f s n proměnnými lze počítat alternujícím obvodem velikosti $2^{n-1}+1$ a hloubky 2.*

Důkaz: Alternující obvod, který počítá funkci f bude přímočarou realizací její efektivnější normální formy. Tj., pokud počet vstupů $\mathbf{x} \in \{0,1\}^n$, pro které je $f(\mathbf{x}) = 1$, je menší než počet těch, pro které je $f(\mathbf{x}) = 0$, pak jako alternující obvod implementujeme disjunktivní normální formu. V tomto případě hradla v první vrstvě odpovídají méně než 2^{n-1} monomům (AND) a výstupní hradlo počítá OR. V opačném případě implementujeme konjunktivní normální formu a opět počet hradel v první vrstvě, které odpovídají klauzulím (OR), je menší nebo roven 2^{n-1} a navíc potřebujeme jedno výstupní hradlo AND. □

Horní odhad velikosti obvodu ve větě 7.7 nelze pro alternující obvody hloubky 2 zlepšit.

Věta 7.8 (Lupanov [180]) *Každý alternující obvod C hloubky 2, který počítá funkci $XOR(x_1, \ldots, x_n)$, musí být velikosti aspoň $2^{n-1}+1$.*

Dále ukážeme konstrukci alternujícího obvodu pro součet dvou přirozených čísel.

Věta 7.6 (Wegener [282, 283]) *Součet dvou n-bitových přirozených čísel lze počítat alternujícím obvodem velikosti $O(n^2)$ a hloubky 4, resp. velikosti $O(n^3)$ a hloubky 3.*

Důkaz: Budeme počítat součet $z = x + y$ dvou přirozených čísel $x, y \in \mathbb{N}$. Označme $x_1 \ldots x_n$, $y_1 \ldots y_n \in \{0,1\}^n$ a $z_1 \ldots z_{n+1} \in \{0,1\}^{n+1}$ po řadě binární reprezentace čísel x, y a z. Nechť dále $c_i \in \{0,1\}$ $(i = 1, \ldots, n)$ je i-tý přenosový bit ve výsledku z_i.

Nejprve zkonstruujeme alternující obvod pro výpočet $c_1 \ldots c_n$. Pro tento účel definujme booleovské funkce $g_i = x_i \wedge y_i$ $(1 \leq i \leq n)$, tj. $g_i = 1$, právě když i-té bity sčítanců generují přenosový bit, a $p_i = x_i \vee y_i$ $(1 \leq i < n)$, tj. $p_i = 1$, právě když i-té bity sčítanců předávají přenosový bit dál. Pomocí nich definujeme funkce

$$f_{ij} = p_i \wedge p_{i+1} \wedge \ldots \wedge p_{j-1} \wedge g_j \quad 1 \leq i \leq n,\, i \leq j \leq n\,, \tag{7.5}$$

tj. $f_{ij} = 1$, právě když j-té bity sčítanců generují přenosový bit, který je dále šířen až k i-tému bitu. Z toho vyplývá, že i-tý přenosový bit lze počítat následujícím způsobem:

$$c_i = f_{i,i} \vee f_{i,i+1} \vee \ldots \vee f_{i,n} \quad 1 \leq i \leq n\,. \tag{7.6}$$

Tedy první vrstva příslušného alternujícího obvodu se skládá z $n-1$ hradel OR, které počítají p_i $(1 \leq i < n)$. Ve druhé vrstvě je $O(n^2)$ hradel AND počítajících f_{ij} $(1 \leq i \leq n,\, i \leq j \leq n)$ podle (7.5). Třetí výstupní vrstva počítá požadované $c_1 \ldots c_n$ podle (7.6).

Podobným způsobem lze zkonstruovat alternující obvod velikosti $O(n^2)$ a hloubky 3 pro výpočet $\bar{c}_1 \ldots \bar{c}_n$. K tomu se využijí následující booleovské funkce. Funkce $h_i = \bar{x}_i \wedge \bar{y}_i = 1$ $(1 \leq i \leq n)$, právě když přenosový bit je pohlcen v i-tém bitu sčítanců, a $q_i = \bar{x}_i \vee \bar{y}_i = 1$ $(1 \leq i < n)$, právě když i-té bity sčítanců negenerují přenosový bit. Potom $\bar{c}_i = e_{i,i} \vee e_{i,i+1} \vee \ldots \vee e_{i,n}$ $(1 \leq i \leq n)$, kde $e_{ij} = q_i \wedge q_{i+1} \wedge \ldots \wedge q_{j-1} \wedge h_j = 1$ $(1 \leq i \leq n,\, i \leq j \leq n)$, právě když j-tý bit sčítanců pohltí přenosový bit, který není dále generován až k i-tému bitu.

Nyní již budeme konstruovat alternující obvod pro výpočet součtu $z_1 \ldots z_{n+1} \in \{0,1\}^{n+1}$. Zřejmě $z_1 = c_1$, $z_{n+1} = x_n \oplus y_n$ a $z_{i+1} = x_i \oplus y_i \oplus c_{i+1}$ pro $1 \leq i < n$. Platí, že

$$x_i \oplus y_i = (x_i \vee y_i) \wedge (\bar{x}_i \vee \bar{y}_i) \tag{7.7}$$

$$\overline{x_i \oplus y_i} = (x_i \vee \bar{y}_i) \wedge (\bar{x}_i \vee y_i)\,. \tag{7.8}$$

Důkaz: Nejprve předpokládejme, že hradla v první vrstvě alternujícího obvodu C počítají AND a na výstupu je hradlo OR. Tedy pro každý vstup $(b_1,\ldots,b_n)\in\{0,1\}^n$ takový, že $XOR(b_1,\ldots,b_n)=1$, existuje v první vrstvě hradlo AND s výstupem 1. Toto hradlo závisí na všech vstupech

$$v_i=\begin{cases} x_i & b_i=1\\ \bar{x}_i & b_i=0\end{cases}\qquad i=1,\ldots,n\,, \tag{7.9}$$

protože, kdyby např. nezáviselo na vstupu v_j pro nějaké $1\leq j\leq n$, pak by

$$\begin{aligned} &XOR(x_1,\ldots,x_{j-1},\quad 0\quad ,x_{j+1},\ldots,x_n)=\\ &XOR(x_1,\ldots,x_{j-1},\quad 1\quad ,x_{j+1},\ldots,x_n)\,, \end{aligned} \tag{7.10}$$

což je spor s definicí funkce XOR. Navíc víme, že počet vstupů $(b_1,\ldots,b_n)\in\{0,1\}^n$, pro které $XOR(b_1,\ldots,b_n)=1$, je 2^{n-1}, tedy počet hradel AND v první vrstvě musí být 2^{n-1}, což spolu s výstupním hradlem určuje velikost $2^{n-1}+1$ alternujícího obvodu C.

Případ, kdy hradla v první vrstvě alternujícího obvodu C počítají OR a na výstupu je AND, je analogický. Např. z uvedeného obvodu vytvoříme alternující obvod $\bar{C}$ stejné velikosti, který počítá $\overline{XOR}(x_1,\ldots,x_n)$, tak, že podle de Morganových vzorců zaměníme všechna hradla AND a OR a vstupy x_i a $\bar{x}_i$ $(i=1,\ldots,n)$. Pro dolní odhad velikosti $\bar{C}$ aplikujeme stejný argument jako v předchozím případě. □

Dolní odhad velikosti alternujícího obvodu ve větě 7.8 platí jen pro hloubku obvodu 2. Následující věta zobecňuje tento odhad pro neomezenou hloubku.

Věta 7.9 *Existují booleovské funkce n proměnných, které pro výpočet funkční hodnoty vyžadují alternující obvod velikosti $\Omega(2^{\frac{n}{2}})$.*

Důkaz: Označme $C(s,n)$ počet alternujících obvodů velikosti s, které mají n vstupů. Nejprve odhadneme $C(s,n)$ shora. Počet orientovaných acyklických grafů na s vrcholech (tj. počet matic sousednosti minimálně s nulovou hlavní diagonálou) je menší než 2^{s^2-s}. Počet různých připojení n vstupů k takovému orientovanému grafu s s vrcholy je 3^{ns}, protože pro každý vstup $i=1,\ldots,n$ a pro každý vrchol v grafu buď x_i, nebo $\bar{x}_i$ je spojen s v nebo s ním i-tý vstup není spojen. Počet různých voleb výstupu v tomto grafu je s a počet možných přiřazení funkcí AND nebo OR jeho vrcholům je 2^s. Z toho dostáváme

$$C(s,n)<2^{s^2-s}3^{ns}s2^s=s2^{s^2}3^{ns}\,. \tag{7.11}$$

Na druhou stranu chceme najít nejmenší s, aby počet alternujících obvodů s n vstupy velikosti s byl větší nebo roven počtu všech booleovských funkcí n proměnných, tj.

$$C(s,n)\geq 2^{2^n}\,. \tag{7.12}$$

Nutně tedy $s\geq n$, protože, kdyby $s<n$, pak by $C(s,n)<2^{O(n^2)}$ podle (7.11), což je ve sporu s (7.12). Pro $s\geq n$ ze vztahu (7.11) vyplývá $C(s,n)<2^{O(s^2)}$ a porovnáním s (7.12) dostaneme $2^{O(s^2)}>2^{2^n}$, tj. $s=\Omega(2^{\frac{n}{2}})$. □

Techniku důkazu věty 7.9 lze použít pro podobný dolní odhad u klasických obvodů, který je znám již ze 40. let [236]. Avšak uvedený argument neposkytuje žádný příklad booleovské funkce, jejíž implementace by vyžadovala velký alternující obvod. Doposud se totiž např. nepodařilo najít booleovskou funkci, která by potřebovala k výpočtu své funkční hodnoty klasický obvod aspoň kvadratické velikosti. Na druhou stranu každou booleovskou funkci lze počítat alternujícím obvodem hloubky 3, jehož velikost se shoduje s uvedeným dolním odhadem ve větě 7.9. To znamená, že každou booleovskou funkci lze počítat třívrstvou neuronovou sítí s exponenciálním počtem neuronů, což je optimální velikost, pokud neurony mohou počítat pouze AND nebo OR.

Věta 7.10 (Redkin [233]) *Každou booleovskou funkci f s n proměnnými lze počítat alternujícím obvodem velikosti $O(2^{\frac{n}{2}})$ a hloubky 3.*

Důkaz: Pro jednoduchost předpokládejme, že n je sudé. Pro lichá n je důkaz podobný [260]. Pro konstrukci alternujícího obvodu pro výpočet f použijeme standardní techniku *rozděl a panuj*. Funkci f rozepíšeme jako disjunkci $2^{\frac{n}{2}}$ funkcí $h(b_1,\ldots,b_{\frac{n}{2}}):\{0,1\}^n\longrightarrow\{0,1\}$ s n proměnnými $x_1,\ldots,x_n$:

$$f(x_1,\ldots,x_n)=\bigvee_{(b_1,\ldots,b_{\frac{n}{2}})\in\{0,1\}^{\frac{n}{2}}} h(b_1,\ldots,b_{\frac{n}{2}})(x_1,\ldots,x_n)\,. \tag{7.13}$$

Funkce $h(b_1,\ldots,b_{\frac{n}{2}})$ se od sebe navzájem liší vektorem $\frac{n}{2}$ parametrů $(b_1,\ldots,b_{\frac{n}{2}})\in\{0,1\}^{\frac{n}{2}}$:

$$\begin{aligned} h(b_1,\ldots,b_{\frac{n}{2}})(x_1,\ldots,x_n) &= \bigwedge_{i=1}^{\frac{n}{2}}(x_i=b_i)\wedge\\ &\wedge\quad f(b_1,\ldots,b_{\frac{n}{2}},x_{\frac{n}{2}+1},\ldots,x_n)\,. \end{aligned} \tag{7.14}$$

Nyní popíšeme alternující obvod hloubky 3, který počítá f. Ve třetí vrstvě je výstupní hradlo OR, které bude realizovat disjunkci (7.13). Druhá vrstva se skládá z $2^{\frac{n}{2}}$ hradel AND, která vyhodnocují všechny funkce $h(b_1,\ldots,b_{\frac{n}{2}})$ $((b_1,\ldots,b_{\frac{n}{2}})\in\{0,1\}^{\frac{n}{2}})$ pomocí jejich konjunktivních normálních forem (viz důkaz věty 7.7). Za tímto účelem první vrstva obsahuje $2^{\frac{n}{2}}$ hradel OR, která odpovídají všem možným klauzulím nad $\frac{n}{2}$ proměnnými $x_{\frac{n}{2}+1},\ldots,x_n$ ze vstupní vrstvy. Každé hradlo ve druhé vrstvě je pak spojeno jen s relevantními hradly z první vrstvy, tj. počítá konjunkci těch klauzulí, které se vyskytují v normální formě odpovídající funkce $h(b_1,\ldots,b_{\frac{n}{2}})$. Dále je každé hradlo

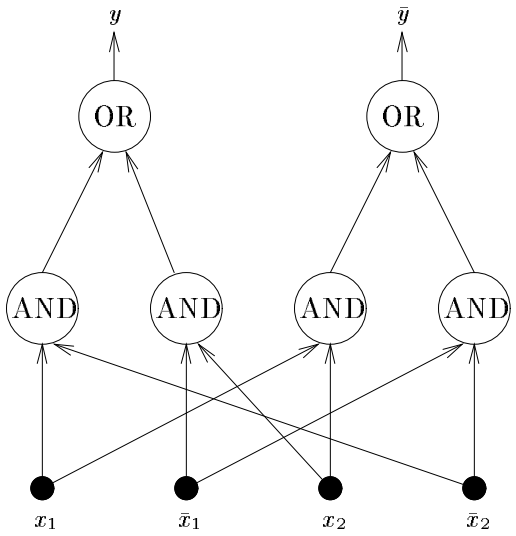
že u alternujících obvodů lze oproti klasickým obvodům nepatrně redukovat hloubku. Navíc také zkonstruujeme klasický obvod pro součet celých čísel. Nejprve však zformulujeme jednoduchou lemmu o výpočtu asociativní operace pomocí klasických obvodů.

Lemma 7.11 *Nechť* $\circ : \{0,1\}^2 \longrightarrow \{0,1\}$ *je asociativní binární operace, kterou lze počítat klasickým obvodem* C *velikosti* s *a hloubky* d*. Pak pro každé přirozené číslo* $n \geq 2$ *složenou* n*-ární operaci* $x_1 \circ x_2 \circ \cdots \circ x_n$ *lze počítat pomocí klasického obvodu velikosti* $(n-1)s$ *a hloubky* $d\lceil \log n \rceil$*.*

Důkaz: Tvrzení ukážeme indukcí dle n. Pro $n = 2$ dostaneme přímo předpoklad lemmy. Obecně pro $n > 2$ aplikujeme opět princip *rozděl a panuj*. Složenou n-ární operaci $\circ$ můžeme díky asociativitě rozložit:

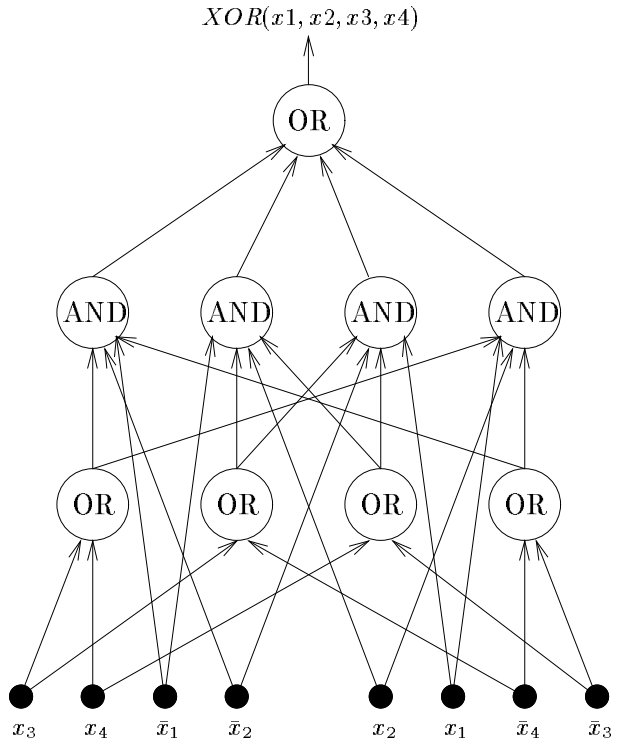
$$x_1 \circ \cdots \circ x_n = (x_1 \circ \cdots \circ x_{\lfloor \frac{n}{2} \rfloor}) \circ (x_{\lfloor \frac{n}{2} \rfloor + 1} \circ \cdots \circ x_n)\,. \tag{7.15}$$

Uzávorkované členy na pravé straně rovnice (7.15) lze dle indukčního předpokladu počítat dvěma klasickými obvody C_1 velikosti $s_1 = (\lfloor \frac{n}{2} \rfloor - 1)s$ s hloubkou $d_1 = d\lceil \log \lfloor \frac{n}{2} \rfloor \rceil$ a C_2 velikosti $s_2 = (\lceil \frac{n}{2} \rceil - 1)s$ s hloubkou $d_2 = d\lceil \log \lceil \frac{n}{2} \rceil \rceil$. Klasický obvod pro výpočet $x_1 \circ \cdots \circ x_n$ získáme podle (7.15) spojením C_1 a C_2 s kopií obvodu C tak, že výstupy C_1 a C_2 jsou vstupem C,



Obr. 7.5: Klasický (alternující) obvod pro výpočet funkcí XOR a $\overline{XOR}$ s hradly AND v první vrstvě.

AND z druhé vrstvy reprezentující $h(b_1, \ldots, b_{\frac{n}{2}})$ navíc podle (7.14) spojeno se vstupy x_i pro $b_i = 1$ a s negacemi vstupů $\bar{x}_i$ pro $b_i = 0$ $(1 \leq i \leq \frac{n}{2})$. Tím je zajištěno konzistentní vyhodnocení funkční hodnoty f podle (7.13) a (7.14). Velikost popsaného alternujícího obvodu je $2^{\frac{n}{2}+1} + 1 = O(2^{\frac{n}{2}})$. Příklad uvedené konstrukce pro $XOR(x_1, x_2, x_3, x_4)$ je na obrázku 7.4. □



Obr. 7.4: Alternující obvod hloubky 3 pro $XOR(x_1, x_2, x_3, x_4)$.

7.1.3 Klasické obvody

Jak jsme již v úvodu této podkapitoly konstatovali, v teorii booleovské složitosti se tradičně studují klasické obvody, které jsou speciálním případem logických obvodů (viz definici 7.3) a díky omezenému počtu vstupů u hradel jsou vhodné k hardwarové realizaci např. klasických počítačů. Na druhou stranu alternující obvody jsou kvůli velké hustotě propojení bližší modelům neuronových sítí. V tomto odstavci se pokusíme o jejich srovnání a uvidíme,

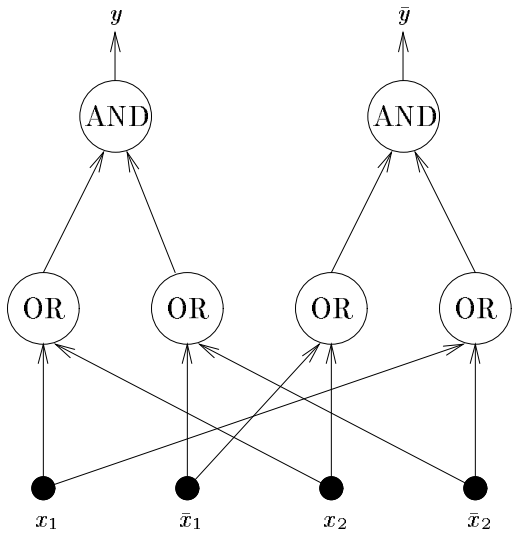
který počítá binární operaci $\circ$. Velikost výsledného obvodu je $s_1 + s_2 + s = (n-1)s$ a jeho hloubka je $\max(d_1, d_2) + d = d_2 + d = d\lceil \log \lceil \frac{n}{2} \rceil + \log 2 \rceil = d\lceil \log n \rceil$. □

Příklad použití lemmy 7.11 je konstrukce klasického (resp. alternujícího) obvodu pro výpočet parity.

Věta 7.12 *Funkci $XOR(x_1, \ldots, x_n)$ $(n \geq 2)$ lze počítat klasickým obvodem velikosti $6n-9$ a hloubky $2\lceil \log n \rceil$, resp. alternujícím obvodem velikosti $4n-5$ a hloubky $\lceil \log n \rceil + 1$.*

Důkaz: Na obrázku 7.5, resp. 7.6, je klasický obvod s 6 hradly hloubky 2, který počítá funkci $y = XOR(x_1, x_2)$ a její negaci $\bar{y} = \overline{XOR}(x_1, x_2)$. Funkce XOR je asociativní, a tedy dle lemmy 7.11 paritu $XOR(x_1, \ldots, x_n)$ lze počítat klasickým obvodem velikosti $6(n-1)$ a hloubky $2\lceil \log n \rceil$. Navíc tento obvod počítá i negaci $\overline{XOR}(x_1, \ldots, x_n)$, proto lze ušetřit odpovídající výstupní hradlo a dvě hradla s ním spojené. Tedy velikost výsledného klasického obvodu je $6n - 9$.

V případě alternujícího obvodu střídáme při konstrukci podle lemmy 7.11 obvod na obrázku 7.5 s obvodem na obrázku 7.6. Díky tomu můžeme využít



Obr. 7.6: Klasický (alternující) obvod pro výpočet funkcí XOR a $\overline{XOR}$ s hradly OR v první vrstvě.

kompresi dle (i) z lemmy 7.5 k redukci hloubky na $\lceil \log n \rceil + 1$ a k zmenšení velikosti na $4n-2$. Po odečtení tří ušetřených hradel určených k výstupu negace má výsledný alternující obvod požadovanou velikost $4n-5$. □

Nyní ukážeme konstrukci, jak z obecného logického obvodu vytvořit ekvivalentní klasický obvod kvadratické velikosti a hloubky zvětšené o logaritmický faktor.

Věta 7.13 *Pro každý logický obvod s n vstupy, velikosti s a hloubky d existuje ekvivalentní klasický obvod velikosti $s^2 + sn$ a hloubky $d\lceil \log(s+n) \rceil$.*

Důkaz: Každé hradlo AND, resp. OR s k vstupy můžeme díky asociativitě konjunkce a disjunkce podle lemmy 7.11 nahradit klasickým podobvodem velikosti $k-1$ a hloubky $\lceil \log k \rceil$. Zřejmě $k \leq s+n$, tedy velikost výsledného klasického obvodu je $s(k-1) \leq s^2 + sn$ a jeho hloubka je $d\lceil \log k \rceil \leq d\lceil \log(s+n) \rceil$. □

Na druhou stranu u klasického obvodu je možné redukovat hloubku, pokud uvolníme omezení na počet vstupů hradel.

Věta 7.14 *Pro každý klasický obvod velikosti s a hloubky d existuje ekvivalentní alternující obvod velikosti $2^{2^\delta}s + s$ a hloubky $\lceil \frac{d}{\delta} \rceil + 1$ pro libovolné přirozené číslo $\delta \in \mathbb{N}$.*

Důkaz: Daný klasický obvod rozdělíme horizontálně na $\lceil \frac{d}{\delta} \rceil$ pásů hloubky δ. Booleovská funkce, kterou počítá hradlo na výstupu pásu, závisí nejvýše na 2^δ hradlech z předchozích pásů, protože u klasického obvodu je počet vstupů u každého hradla nejvýše 2. Normální formu této funkce 2^δ proměnných však můžeme realizovat pomocí alternujícího podobvodu hloubky 2 a velikosti $2^{2^\delta} + 1$ (viz větu 7.7). Nahrazujeme postupně všechny pásy v obvodu pomocí těchto alternujících podobvodů tak, že střídáme implementace konjunktivní a disjunktivní normální formy, abychom mohli použít kompresi podle dle (i) v lemmě 7.5 k redukci hloubky na $\lceil \frac{d}{\delta} \rceil + 1$. Tomu odpovídá velikost výsledného alternujícího obvodu $2^{2^\delta}s + s$. □

Ačkoliv se může nárůst velikosti u alternujícího obvodu ve větě 7.14 zdát nepřípustně velký, pro vhodné δ dostaneme redukci hloubky v rámci polynomiální velikosti.

Důsledek 7.15 (Chandra, Stockmeyer, Vishkin [131]) *Pro každý klasický obvod velikosti s a hloubky d existuje ekvivalentní alternující obvod velikosti $2^{k\sqrt{\log s}}s + s$ a hloubky $\lceil \frac{kd}{\log \log s} \rceil + 1$ pro libovolné přirozené číslo $k \in \mathbb{N}$.*

Důkaz: Ve větě 7.14 položme $\delta = \lfloor \frac{\log \log s}{k} \rfloor$. □

Z důsledku 7.15 vyplývá, že modely diskrétních acyklických neuronových sítí mající neomezený počet vstupů neuronů mohou v aktivním režimu za cenu nejvýše kvadratického nárůstu velikosti $2\sqrt[k]{\log s}\,s + s = O(s^2)$ urychlit (paralelní) čas výpočtu o faktor $O(\log\log n)$ oproti obvodům, které využívají klasické počítače.

Ve větě 7.6 jsme ukázali konstrukci alternujícího obvodu polynomiální velikosti a konstantní hloubky pro součet dvou přirozených čísel. Tento výsledek zobecníme pro součet n celých čísel pomocí klasického obvodu polynomiální velikosti a logaritmické hloubky. Pro tento účel nejprve dokážeme následující lemmu:

Lemma 7.16 *Existuje klasický obvod velikosti* $16n$ *a hloubky 4, který pro vstup tří* n*-bitových přirozených čísel* a*,* b*,* c *vypočítá dvě přirozená čísla* d*,* e *taková, že* $a+b+c = d+e$*.*

Důkaz: Postupujeme podobně jako v důkazu věty 7.6. Zde $d_1 \ldots d_{n+1} \in \{0,1\}^{n+1}$ jsou přenosové bity a $e_1 \ldots e_n$ představuje výsledek součtu $a+b+c$ bez těchto bitů. Tedy $d_{n+1} = 0$ a

$$d_i = (a_i \vee b_i) \wedge (b_i \vee c_i) \wedge (a_i \vee c_i) \quad 1 \leq i \leq n\,, \tag{7.16}$$

kde a_i, b_i, c_i jsou i-té bity binární reprezentace čísel a, b, c. Klasický obvod hloubky 3 pro výpočet d podle (7.16) se skládá z $3n$ hradel OR a $2n$ hradel AND. Podobně $e_i = a_i \oplus b_i \oplus c_i$ $(1 \leq i \leq n)$ lze vyjádřit:

$$e_i = (a_i \wedge b_i \wedge c_i) \vee (a_i \wedge \bar{b}_i \wedge \bar{c}_i) \vee (\bar{a}_i \wedge b_i \wedge \bar{c}_i) \vee (\bar{a}_i \wedge \bar{b}_i \wedge c_i)\,. \tag{7.17}$$

To znamená, že klasický obvod hloubky 4 pro výpočet e podle (7.17) se skládá z $8n$ hradel AND a $3n$ hradel OR. Tedy požadovaný výsledný klasický obvod má velikost $16n$ a hloubku 4. □

Věta 7.17 (Hong [120, 121]) *Součet* n m*-bitových celých čísel lze počítat klasickým obvodem velikosti* $O(nm + m^3 + n\log n)$ *a hloubky* $O(\log n)$*.*

Důkaz: Ukážeme požadovaný výsledek pro přirozená čísla, zatímco důkaz pro celá čísla je podobný. Postupujeme podobně jako v lemmě 7.11. Začínáme se součtem n čísel, které rozdělíme na skupiny po třech číslech, pro jejichž součet vytvoříme klasický obvod C podle lemmy 7.16, a zredukujeme tak počet sčítanců na $\lceil \frac{2}{3}n \rceil$. Uvedený postup opakujeme pro tyto sčítance a jejich součty atd. Tedy po i-tém kroku zbývá $\lceil (\frac{2}{3})^i n \rceil$ sčítanců. Celý proces ukončíme se dvěma sčítanci, tj. po $\log_{\frac{3}{2}} n = O(\log n)$ krocích. To znamená, že hloubka vytvořeného klasického obvodu C_1 je $O(\log n)$, protože obvod C v lemmě 7.16 má konstantní hloubku. Dále obvod C_1 se skládá z $O(n)$ kopií

C s $O(m + \log n)$ vstupy, protože součet n m-bitových čísel lze reprezentovat pomocí $O(m + \log n)$ bitů, což dle lemmy 7.16 odpovídá $O(m + \log n)$ hradlům C. Z toho vyplývá, že velikost C_1 je $O(n(m + \log n))$.

Nakonec musíme ještě sečíst dvě čísla s reprezentací $O(m+\log n)$ bitů na výstupu C_1. Pro tento účel vytvoříme podle věty 7.6 alternující obvod C_2^A velikosti $O((m+\log n)^2)$ a hloubky 4, který převedeme na ekvivalentní klasický obvod C_2 hloubky $O(\log n)$ pomocí metody z důkazu věty 7.13. Obvod C_2 má v tomto případě velikost $O((m+\log n)^3)$, protože každé z $O((m+\log n)^2)$ hradel AND ve druhé vrstvě obvodu C_2^A může mít až $O(m + \log n)$ vstupů.

Výsledný klasický obvod hloubky $O(\log n)$ pro součet n m-bitových čísel vznikne spojením obvodů C_1 a C_2. Jestliže $m = \Omega(\log n)$, pak velikost C_1 je $O(nm)$ a velikost C_2 je $O(m^3)$. Pokud naopak $m = O(\log n)$, pak velikost C_1 je $O(n\log n)$ a velikost C_2 je $O(\log^3 n) = O(n\log n)$. Tedy velikost výsledného obvodu je $O(nm + m^3 + n\log n)$. □

Velikost obvodu ve větě 7.17 lze snížit na $O(nm + m^2\log^2 m + n\log n)$, pokud za C_2^A v jejím důkazu vezmeme alternující obvod velikosti $O(n\log n)$ a konstantní hloubky pro součet dvou n-bitových čísel popsaný v [259].

7.1.4 Posloupnosti logických obvodů

V předchozích odstavcích jsme se zabývali konečnými logickými obvody s n vstupy, které mohou realizovat booleovskou funkci n proměnných. Ty však vzhledem k pevnému počtu vstupů nelze považovat za obecné algoritmy, u nichž délka vstupu není předem omezená (např. vstupní páska Turingova stroje). Proto se uvažuje nekonečná *posloupnost logických obvodů*, kde každé délce vstupu odpovídá právě jeden logický obvod s příslušným počtem vstupů. Míry složitosti (velikost, hloubka) u tohoto výpočetního modelu jsou funkcí délky vstupu.

Definice 7.18 *Definujeme nekonečnou* posloupnost logických obvodů $\mathbf{C} = (C_0, C_1, C_2, \ldots)$*, v níž logický obvod* $C_n = (V_n, X_n, Y_n, E_n, \ell_n)$ *(*$n \geq 0$*) má* $|X_n| = n$ *vstupů.* Velikost *posloupnosti* $\mathbf{C}$ *je taková funkce* $S : \mathbb{N} \longrightarrow \mathbb{N}$ *na přirozených číslech, že velikost logického obvodu* C_n *v této posloupnosti je pro každé* $n \geq 0$ *menší nebo rovna* $S(n)$*. Podobně* hloubka *posloupnosti* $\mathbf{C}$ *je funkce* $D : \mathbb{N} \longrightarrow \mathbb{N}$*, pro níž hloubka* C_n *je menší nebo rovna* $D(n)$ *pro každé* $n \geq 0$*.*

Posloupnost logických obvodů budeme při studiu její složitosti podobně jako u Turingova stroje využívat k rozpoznávání jazyků nad binární abecedou $\{0,1\}$, které odpovídají problémům daným množinou slov s kladnou odpovědí. Avšak tento výpočetní model nemá podle definice 7.18 obecně konečný popis, proto posloupnosti logických obvodů rozhodují již všechny

(iii) Důkaz probíhá obdobně jako v části (*ii*), navíc uvažujeme výpočty Turingova stroje s omezenou časovou a prostorovou složitostí. Pro $L = L(M) \in P$ zkonstruujeme pro každé $n \geq 0$ s využitím jen logaritmického prostoru logický obvod C_n implementující funkci f_n. Konfigurace polynomiálního výpočtu Turingova stroje M kódujeme pomocí vrstev obvodu C_n, které jsou spojeny (stejnými) logickými podobvody realizujícími přechodovou funkci M. Na druhou stranu výpočet L-uniformní posloupnosti obvodů pro daný vstup lze realizovat v polynomiálním čase, protože odpovídající Turingův stroj $M_{\mathbf{C}}$ pro konstrukci příslušného obvodu C_n pracuje v logaritmickém prostoru, tj. nejvýše v polynomiálním čase. □

Věta 7.20 ukazuje, že posloupnost logických obvodů, tj. posloupnost diskrétních acyklických neuronových sítí, lze považovat za univerzální výpočetní prostředek. V tomto modelu lze také přirozeně definovat třídu problémů (jazyků), které jsou „prakticky" řešitelné, tj. rozhodnutelné v polynomiálním čase. Dále se budeme zabývat L-uniformními posloupnostmi logických (alternujících i klasických) obvodů s omezenou hloubkou. To je motivováno především praktickými potřebami, protože L-uniformita implikuje polynomiální velikost obvodu, která je nutnou podmínkou při hardwarové realizaci obvodu. Navíc omezená hloubka obvodu, která odpovídá paralelnímu času výpočtu, zajišťuje efektivitu jeho výpočtu. Tomu také odpovídá neurofyziologická skutečnost, protože nervová soustava se skládá z několika vrstev neuronů a počet neuronů není neomezený. Za tímto účelem definujeme následující třídy složitosti.

Definice 7.21 (Cook [48]) *Označme NC^k ($k \geq 1$), resp. AC^k ($k \geq 0$), třídu jazyků rozpoznatelných L-uniformní posloupností klasických, resp. alternujících, obvodů (polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta) polylogaritmické hloubky $D(n) = O(\log^k n)$. Dále definujeme tzv.* Nickovu třídu NC (**N***ick's* **C***lass), resp. třídu AC:*

$$NC = \bigcup_{k \geq 1} NC^k \tag{7.18}$$

$$AC = \bigcup_{k \geq 0} AC^k . \tag{7.19}$$

Nickovu třídu v definici 7.21 pojmenoval S. A. Cook podle Nicka Pippengera, který objevil vztah mezi NC a složitostí turingovských výpočtů [224]. Následující věta ukazuje vztah mezi uvedenými třídami složitosti a naznačuje možnou hierarchii těchto tříd, která je znázorněna na obrázku 7.7.

jazyky včetně nerekurzivních, tj. těch, které nejsou algoritmicky rozhodnutelné (např. Turingovým strojem). Z tohoto důvodu se navíc požaduje, aby posloupnost obvodů byla tzv. *uniformní.*

Definice 7.19 *Posloupnost logických obvodů* $\mathbf{C}$ *s jedním výstupem, tj.* $|Y_n| = 1$ *pro* $n \geq 0$, rozpoznává jazyk $L \subseteq \{0,1\}^\star$, *značíme* $L = L(\mathbf{C})$, *jestliže pro každý vstup* $\mathbf{x} \in \{0,1\}^n$ *výstup* $C_n(\mathbf{x}) = 1$, *právě když* $\mathbf{x} \in L$. *Dále řekneme, že posloupnost logických obvodů je* uniformní, *resp.* L-uniformní, *jestliže existuje algoritmus (např. Turingův stroj), resp. algoritmus pracující v logaritmickém prostoru, který pro každé* $n \geq 0$ *zkonstruuje logický obvod* C_n *z této posloupnosti.*

Věta 7.20

(i) *Pro každý jazyk $L \subseteq \{0,1\}^\star$ existuje (obecně neuniformní) posloupnost $\mathbf{C}$ logických obvodů exponenciální velikosti $S(n) = O(2^n)$ a konstantní hloubky $D(n) = O(1)$, která rozpoznává $L = L(\mathbf{C})$.*

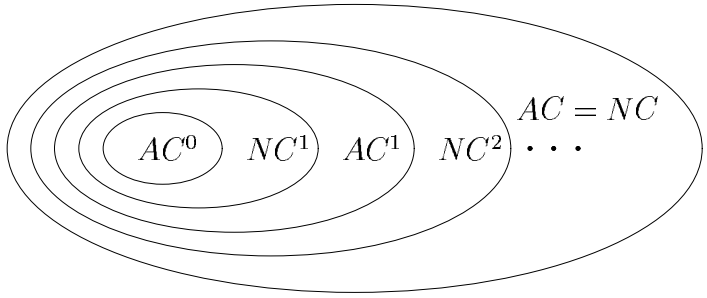
(ii) *Jazyk $L \subseteq \{0,1\}^\star$ je rekurzivní, právě když existuje uniformní posloupnost $\mathbf{C}$ logických obvodů, která rozpoznává $L = L(\mathbf{C})$.*

(iii) *Jazyk $L \subseteq \{0,1\}^\star$ je algoritmicky rozhodnutelný (Turingovým strojem) v polynomiálním čase, tj. $L \in P$, právě když existuje L-uniformní posloupnost $\mathbf{C}$ logických obvodů, která rozpoznává $L = L(\mathbf{C})$.*

Důkaz:

(i) Pro každou délku slova $n \geq 0$ uvažujeme booleovskou funkci $f_n : \{0,1\}^n \longrightarrow \{0,1\}$ s n proměnnými takovou, že pro každé slovo $\mathbf{x} \in \{0,1\}^n$ délky n je $f_n(\mathbf{x}) = 1$, právě když $\mathbf{x} \in L$. Funkce f_n ($n \geq 0$) lze dle věty 7.7 realizovat logickými obvody exponenciální velikosti a konstantní hloubky, které tvoří výslednou posloupnost rozpoznávající L.

(ii) Nechť nejprve L je rekurzivní jazyk, tj. existuje Turingův stroj M, který rozhoduje $L = L(M)$. Logický obvod C_n pro vstup délky n v posloupnosti $\mathbf{C}$, která rozpoznává $L = L(\mathbf{C})$, lze algoritmicky zkonstruovat tak, že podobně jako v důkazu (*i*) realizujeme funkci f_n, jejíž hodnoty získáme výpočtem Turingova stroje M pro všechny vstupy délky n. Obráceně předpokládejme, že $L = L(\mathbf{C})$ je rozpoznáván uniformní posloupností $\mathbf{C}$ alternujících obvodů. Tedy existuje Turingův stroj $M_{\mathbf{C}}$, který pro každé $n \geq 0$ zkonstruuje logický obvod C_n z posloupnosti $\mathbf{C}$. Algoritmický test, zda $\mathbf{x} \in L$, probíhá tak, že pomocí $M_{\mathbf{C}}$ je nejprve sestrojen obvod $C_{|\mathbf{x}|}$, kde $|\mathbf{x}|$ je délka slova $\mathbf{x}$, a pak je podle definice 7.2 vypočten výstup $C_{|\mathbf{x}|}$ pro vstup $\mathbf{x}$, který je 1, právě když $\mathbf{x} \in L$.



Obr. 7.7: Hierarchie tříd složitosti AC a NC.

Věta 7.22

(i) *Pro každé $k \geq 1$ platí inkluze $NC^k \subseteq AC^k$.*

(ii) *Pro každé $k \geq 0$ platí inkluze $AC^k \subseteq NC^{k+1}$.*

(iii) $NC = AC$.

(iv) $NC \subseteq P$.

Důkaz:

(i) Klasický obvod je speciálním případem obecného logického obvodu, který lze podle věty 7.4 převést na ekvivalentní alternující obvod téměř stejné hloubky.

(ii) Podle věty 7.13 můžeme z obecného logického obvodu C_n polynomiální velikosti $S(n)$ a hloubky $D(n) = O(\log^k n)$ vytvořit ekvivalentní klasický obvod C'_n polynomiální velikosti $O(S^2(n) + nS(n))$ a hloubky $O(\log^{k+1} n)$, která je zvětšená jen o logaritmický faktor $\lceil \log(S(n)+n) \rceil = O(\log n)$, protože $S(n)$ je nejvýš polynom.

(iii) Tvrzení je přímým důsledkem (*i*) a (*ii*).

(iv) Tvrzení plyne z (*iii*) věty 7.20, v níž chybí omezení na hloubku. □

Platnost či neplatnost obrácené inkluze v tvrzení (*iv*) věty 7.22 je jedním z hlavních otevřených problémů paralelní teorie složitosti. Rovnost $NC = P$ by znamenala, že každý prakticky řešitelný problém by se také dal pouze s polynomiálním hardwarem počítat v polylogaritmickém paralelním čase. Jinými slovy každý polynomiální algoritmus by bylo možné efektivně zparalelnit. Všeobecně přijímaná hypotéza $NC \neq P$ tuto možnost vylučuje.

Vzhledem k tomu, že $NC = AC$ (viz (*iii*) ve větě 7.22), podobné úvahy o efektivních paralelních obvodech lze přenést i na diskrétní acyklické neuronové sítě. Při studiu problému NC–P se také uvažují úplné problémy vzhledem k tzv. *AC-redukci.*

Definice 7.23 *Řekneme, že problém (jazyk) $B \subseteq \{0,1\}^\star$ je AC-redukovatelný na problém $A \subseteq \{0,1\}^\star$ a značíme $B \leq_c A$, jestliže existuje zobrazení $f : \{0,1\}^\star \longrightarrow \{0,1\}^\star$, které lze počítat L-uniformní posloupností alternujících obvodů (polynomiální velikosti) polylogaritmické hloubky, takové, že pro každé $\mathbf{x} \in \{0,1\}^\star$ platí $\mathbf{x} \in B$, právě když $f(\mathbf{x}) \in A$. Problém A je P-těžký, jestliže pro každý problém $B \in P$ platí $B \leq_c A$. Problém A je P-úplný, jestliže je $P -$ těžký a $A \in P$.*

Věta 7.24 *Jestliže A je P-úplný problém a $A \in NC$, pak $NC = P$.*

Důkaz: Nechť problém $B \in P$. Tedy B lze AC-redukovat na A, protože A je P-úplný problém. To znamená, že existuje L-uniformní posloupnost $\mathbf{C} = (C_0, C_1, C_2, \ldots)$ alternujících obvodů C_n polylogaritmické hloubky, která každý vstup $\mathbf{x} \in \{0,1\}^\star$ problému B převede pomocí $C_{|\mathbf{x}|}$ na vstup $f(\mathbf{x}) \in \{0,1\}^\star$ pro A tak, že $\mathbf{x} \in B$, právě když $f(\mathbf{x}) \in A$. Díky tomu, že $A \in NC$, existuje L-uniformní posloupnost $\mathbf{C}^A = (C_0^A, C_1^A, C_2^A, \ldots)$ klasických, a tedy i alternujících obvodů C_n^A polylogaritmické hloubky, která rozpoznává $A = L(\mathbf{C}^A)$. Tedy výsledná L-uniformní posloupnost $\mathbf{C}^B = (C_0^B, C_1^B, C_2^B, \ldots)$ alternujících obvodů C_n^B polylogaritmické hloubky, která rozpoznává $B = L(\mathbf{C}^B)$, vznikne spojením obvodů C_n a C_n^A tak, že výstupy C_n jsou vstupem C_n^A. Z toho plyne $NC = AC = P$. □

Nyní ukážeme příklad P-úplného problému:

Problém vyhodnocení obvodu CVP (**C**ircuit **V**alue **P**roblem):
instance: Alternující obvod C s n vstupy a jedním výstupem; n bitů $a_1, \ldots, a_n \in \{0,1\}$.
otázka: Jaký je výstup C pro vstup $a_1, \ldots, a_n$?

Věta 7.25 (Ladner [167]) *CVP je P-úplný problém.*

Důkaz: Zřejmě $CVP \in P$, protože alternující obvod lze vyhodnotit v polynomiálním čase. Při důkazu, že CVP je P-těžký problém, předpokládejme, že B je libovolný problém z P. Tedy dle (*iii*) z věty 7.20 a věty 7.4 existuje L-uniformní posloupnost alternujících obvodů $\mathbf{C} = (C_0, C_1, C_2, \ldots)$ polynomiální velikosti, která rozpoznává $B = L(\mathbf{C})$. Definujme zobrazení $f : \{0,1\}^\star \longrightarrow \{0,1\}^\star$ tak, že $f(x_1 \ldots x_n) = (code(C_n), x_1 \ldots x_n)$, kde $code(C_n) \in \{0,1\}^\star$ je nějaký binární kód alternujícího obvodu C_n. Zobrazení f lze realizovat posloupností $\mathbf{C}_f$ alternujících obvodů konstantní hloubky

tak, že konstantní výstupní hradla reprezentují kód C_n a zbylá hradla zkopírují vstup. Posloupnost $\mathbf{C}_f$ je díky L-uniformitě posloupnosti $\mathbf{C}$ také L-uniformní. Navíc $\mathbf{x} \in B$, právě když $f(\mathbf{x}) \in CVP$, a tedy $B \leq_c CVP$. □

V závěru této podkapitoly ještě zformulujeme bez důkazu tvrzení, která se týkají separace hierarchie tříd složitosti pro L-uniformní posloupnosti alternujících obvodů s konstantní hloubkou. Tyto výsledky jsou relevantní pro modely diskrétních acyklických neuronových sítí, které většinou předpokládají konstantní hloubku.

Definice 7.26 *Označme AC^0_k ($k \geq 0$), třídu jazyků rozpoznatelných L-uniformní posloupností alternujících obvodů (polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta) konstantní hloubky $D(n) = k$. Zřejmě*

$$AC^0 = \bigcup_{k \geq 0} AC^0_k \tag{7.20}$$

a $AC^0_1 \subseteq AC^0_2 \subseteq AC^0_3 \subseteq \ldots$ je tzv. hierarchie AC^0.

Je zřejmé, že $AC^0_1 \subsetneq AC^0_2$, protože AC^0_1 se skládá jen z konjunkcí, popř. jen disjunkcí vybraných vstupů nebo jejich negací. Podle věty 7.6 víme, že součet dvou n-bitových čísel lze realizovat pomocí alternujícího obvodu hloubky 3 a polynomiální velikosti. Na druhou stranu každý alternující obvod hloubky 2 vyžaduje pro výpočet (prvního bitu) takového součtu aspoň 2^{n-1} hradel [259], což lze ukázat pomocí podobného argumentu jako v důkazu věty 7.8. Z toho vyplývá, že $AC^0_2 \subsetneq AC^0_3$. Ve skutečnosti třídy celé hierarchie AC^0 lze oddělit.

Věta 7.27 (Sipser [257]) *Pro každé $k \geq 0$ platí nerovnost $AC^0_k \subsetneq AC^0_{k+1}$.*

Podle věty 7.12 víme, že parita je v NC^1, ale podle věty 7.8 není v AC^0_2, což lze zobecnit pro AC^0. Tento výsledek implikuje jedinou dosud známou separaci tříd v hierarchii na obrázku 7.7.

Věta 7.28 (Furst, Saxe, Sipser [70, 282]) $XOR \in NC^1 \setminus AC^0$.

7.2 Prahové obvody

V předchozí podkapitole jsme se zabývali logickými obvody, které lze chápat jako diskrétní acyklické neuronové sítě s funkcí neuronu omezenou jen na konjunkci a disjunkci, tj. na jednoduché booleovské prahové funkce (viz větu 6.26). V této podkapitole zobecníme tento výpočetní model tak, že hradla v tzv. *prahovém obvodu* budou počítat obecnou booleovskou prahovou funkci (viz podkapitolu 6.4). Tím dosáhneme věrnější model diskrétní acyklické neuronové sítě.

Definice 7.29 Prahový obvod $C = (V, X, Y, E, \ell)$ *je obvod (viz definici 7.1), jehož třídu $\mathcal{F}$ hradlových funkcí tvoří obecně lineární prahové funkce. Pokud neuvedeme jinak, budeme vždy uvažovat jen booleovské prahové funkce. To znamená, že zobrazení ℓ přiřadí každému hradlu s k vstupy celočíselnou (obecně reálnou) reprezentaci $(w_1, \ldots, w_k; h)$ příslušné booleovské prahové funkce.*

V prahovém obvodu podobně jako u logického obvodu studujeme míry deskriptivní složitosti jako je velikost, hloubka a navíc maximální váha a váha obvodu. Na maximální váhu můžeme klást jistá omezení a získáme tak speciální podtřídy prahových obvodů. To odpovídá paměťové náročnosti při implementaci diskrétní acyklické neuronové sítě, kdy na reprezentaci každé váhy v síti máme jen omezený počet bitů.

Definice 7.30 Maximální váha w_{max} *v prahovém obvodu je hodnota největší váhy booleovské prahové funkce, kterou hradlo v obvodu počítá.* Váha *prahového obvodu $|w|$ je součet všech absolutních hodnot váhových parametrů v obvodu. Prahové obvody s* jednotkovými váhami *mají maximální váhu 1 a prahové obvody s* malými váhami *mají maximální váhu omezenou polynomem vzhledem k počtu vstupů obvodu, tj. na reprezentaci jedné váhy stačí $O(\log n)$ bitů.*

Je zřejmé, že $1 \leq w_{max} \leq |w|$. Navíc pro obecný prahový obvod velikosti s se souvislou topologií platí, že $s - 1 \leq |w| \leq s^2 w_{max}$.

Podobně jako u alternujících obvodů (viz definici 7.3) budeme někdy předpokládat, že každý vstup x_i ($i = 1, \ldots, n$) prahového obvodu je navíc sdružen s přidaným vstupem $\bar{x}_i$ odpovídajícím jeho negaci. To nám umožní omezit se na obvody s kladnými váhami pomocí analogie věty 7.4.

Věta 7.31 *Pro každý prahový obvod (s jednotkovými, resp. malými váhami) velikosti s, hloubky d a maximální váhy w_{max} existuje ekvivalentní prahový obvod (s jednotkovými, resp. malými váhami) velikosti $2s$, hloubky d a maximální váhy w_{max}, který má jen kladné váhy.*

Důkaz: Pomocí (*ii*) v lemmě 6.27 lze záporné váhy nahradit negací odpovídajících vstupů hradel a ty lze přesunout do vstupní vrstvy podobným způsobem jako v důkazu věty 7.4 s využitím (*iii*) z lemmy 6.27. □

7.2.1 Implementace funkcí

Pomocí prahových obvodů je možno efektivně počítat symetrickou booleovskou funkci, jejíž hodnota nezávisí na pořadí vstupů.

Věta 7.32 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Každou symetrickou booleovskou funkci f s n proměnnými lze počítat prahovým obvodem (s jednotkovými váhami) velikosti $2n + 1$ a hloubky 2.*

Důkaz: Hodnota symetrické booleovské funkce závisí jen na počtu proměnných s hodnotou 1. Uvažujme množinu $N_f = \{n_1, \ldots, n_k\}$ $(0 \leq k = |N_f| \leq n)$ všech takových počtů jednotkových vstupů, pro které hodnota f je 1, tj.

$$N_f = \left\{ n' \;\middle|\; f(x_1, \ldots, x_n) = 1 \leftrightarrow \sum_{i=1}^{n} x_i = n',\ 0 \leq n' \leq n \right\} . \qquad (7.21)$$

První vrstva prahového obvodu pro výpočet f se skládá z $k = |N_f|$ párů hradel. V j-tém páru $(j = 1, \ldots, k)$ je první hradlo aktivní, tj. jeho výstup je 1, právě když $\sum_{i=1}^{n} x_i \geq n_j$, a druhé hradlo je aktivní, právě když $\sum_{i=1}^{n} x_i \leq n_j$, tj. $\sum_{i=1}^{n} -x_i \geq -n_j$ (resp. $\sum_{i=1}^{n} \bar{x}_i \geq n - n_j$). Z toho vyplývá, že v první vrstvě je vždy aspoň k hradel aktivních a to z každého páru aspoň jedno. Pokud navíc $f(x_1, \ldots, x_n) = 1$, tj. $\sum_{i=1}^{n} x_i = n_j$ pro nějaké $n_j \in N_f$, pak v j-tém páru jsou obě hradla aktivní, tedy celkově v první vrstvě je právě $k+1$ hradel aktivních. Stačí tedy spojit všechna hradla v první vrstvě s jedním výstupním hradlem ve druhé vrstvě, které je aktivní, právě když je aktivní aspoň $k+1$ hradel z první vrstvy, tj. má jednotkové váhy a práh $k+1$. □

Velikost (obecného) prahového obvodu pro výpočet symetrické booleovské funkce ve větě 7.32 lze snížit na $O(\sqrt{n})$ [259], pokud uvažujeme hloubku 3, pro níž je tato velikost již téměř optimální (dolní odhad pro hloubku 3 je $\Omega(n^{\frac{1}{2}-\varepsilon})$ pro $\varepsilon > 0$ [258]).

Podobnou technikou jako v důkazu věty 7.32 ukážeme analogii věty 7.17 o součtu čísel pro prahové obvody nejprve s neomezenými váhami. Díky tomu, že hradla počítají booleovskou prahovou funkci a mají neomezený počet vstupů, součet čísel lze realizovat s konstantní hloubkou obvodu.

Věta 7.33 (Hofmeister, Hohberg, Köhling [117]) *Součet n m-bitových přirozených čísel lze počítat prahovým obvodem hloubky 2, velikosti $O(nm + n \log n)$ a maximální váhy 2^{m-1}.*

Důkaz: Označme $x_{i1}x_{i2}\ldots x_{im} \in \{0,1\}^m$ $(i = 1, \ldots, n)$ binární reprezentaci i-tého čísla na vstupu a $y_1 \ldots y_{p+1}y_{p+2}\ldots y_{m+p+1} \in \{0,1\}^{m+p+1}$ binární reprezentaci výsledného součtu těchto čísel na výstupu, kde $p = \lfloor \log n \rfloor$. Nejprve zkonstruujeme prahový obvod pro výpočet y_k pro $k \geq p+2$. Hodnota y_k závisí na $x_{i,k-p-1}x_{i,k-p}\ldots x_{i,m} \in \{0,1\}^{m+p+2-k}$ pro $i = 1, \ldots, n$, jejichž součet je $c_1 \ldots c_{p+1}y_k y_{k+1}\ldots y_{m+p+1} \in \{0,1\}^{m+2p+3-k}$, kde $c_1 \ldots c_{p+1} \in \{0,1\}^{p+1}$ jsou přenosové bity tohoto součtu. Každé hradlo z první vrstvy prahového obvodu bude spojeno s x_{ij} pomocí váhy 2^{m-j} pro $1 \leq i \leq n$, $k-p-1 \leq j \leq m$. Pro každou možnou hodnotu přenosových bitů $c_1 \ldots c_{p+1} \in \{0,1\}^{p+1}$ uvažujeme pár hradel. První hradlo v tomto páru bude aktivní,

právě když zvážená suma vstupů bude větší nebo rovna číslu, které má následující binární reprezentaci:

$$c_1 c_2 \ldots c_{p+1} 1 \underbrace{0 \ldots 0}_{m+p+1-k} .$$

Podobně druhé hradlo v tomto páru bude aktivní, právě když zvážená suma vstupů bude menší nebo rovna (tj. záporná zvážená suma větší nebo rovna) číslu s následující binární reprezentací:

$$c_1 c_2 \ldots c_{p+1} 1 \underbrace{1 \ldots 1}_{m+p+1-k} .$$

Z toho vyplývá, že v první vrstvě je vždy aspoň 2^{p+1} hradel aktivních a to z každého páru aspoň jedno. Pokud navíc $y_k = 1$ s přenosovými bity $c_1 \ldots c_{p+1} \in \{0,1\}^{p+1}$, pak v odpovídajícím páru jsou obě hradla aktivní, tedy celkově v první vrstvě je právě $2^{p+1}+1$ hradel aktivních. Pro výpočet y_k stačí spojit všechna hradla z první vrstvy s jedním výstupním hradlem ve druhé vrstvě, které je aktivní, právě když je aktivní aspoň $2^{p+1}+1$ hradel v první vrstvě, tj. má jednotkové váhy a práh $2^{p+1}+1$. Tedy máme prahový obvod pro výpočet y_k velikosti $2^{p+2}+1 = O(n)$, hloubky 2 a maximální váhy $2^{m+p+1-k} \leq 2^{m-1}$ pro $k \geq p+2$. Podobným způsobem lze sestrojit prahové obvody stejných parametrů pro výpočet $y_1, \ldots, y_{p+1}$. Výsledný prahový obvod pro součet n m-bitových čísel má velikost $O(nm + n \log n)$, hloubku 2 a maximální váhu 2^{m-1}. □

Analogii věty 7.33 lze dokázat i pro malé váhy za cenu nepatrného nárůstu velikosti a hloubky prahového obvodu.

Věta 7.34 (Hofmeister, Hohberg, Köhling [117]) *Součet n m-bitových celých čísel lze počítat prahovým obvodem velikosti $O(nm+m^2+n \log n)$, hloubky 5 a maximální váhy $O(n)$.*

Důkaz: Požadovaný výsledek ukážeme pro přirozená čísla, zatímco důkaz pro celá čísla je podobný. Sčítance nejprve uspořádáme do $n \times m$ bitového pole, které rozdělíme na $\lceil m/\lfloor \log n + 1 \rfloor \rceil$ bloků s n řádkami a $\lfloor \log n + 1 \rfloor$ sloupci. Součet každého bloku, příp. zleva doplněný nulami, se skládá z $2\lfloor \log n + 1 \rfloor$ bitů, z nichž první polovina jsou přenosové bity a druhá polovina tvoří výsledek součtu bloku. Spojením přenosových, resp. součtových, bitů od všech těchto bloků dostaneme přenosové, resp. součtové, bity hledaného součtu. Součtem přenosových a součtových bitů získáme výsledný součet.

Uvedený součet jednoho bloku lze podle věty 7.33 realizovat pomocí prahového obvodu velikosti $O(n \log n)$, hloubky 2 a maximální váhy $O(n)$. Tedy

pro součet $O(m/\log n)$ bloků je potřeba prahový obvod velikosti $O(nm)$. Pro součet $O(m+\log n)$ přenosových a součtových bitů použijeme alternující obvod z věty 7.6, tj. prahový obvod s jednotkovými váhami velikosti $O((m+\log n)^2)$ a hloubky 4. Spojením uvedených obvodů dostaneme prahový obvod velikosti $O(mn+(m+\log n)^2)$, hloubky 6 a maximální váhy $O(n)$. Jestliže $m = O(\log n)$, velikost obvodu je $O(n\log n + \log^2 n) = O(n\log n)$, a pro $m = \Omega(\log n)$ ji lze vyjádřit $O(nm+m^2)$. Tedy celkově má uvedený prahový obvod požadovanou velikost $O(nm+m^2+n\log n)$.

Navíc z důkazu věty 7.33 víme, že zvážená suma prahových hradel ve druhé vrstvě je buď rovna prahu, nebo je o 1 menší. Proto můžeme konjunkci g_i (podobně disjunkci p_i) z důkazu věty 7.6 počítat již ve druhé vrstvě jedním hradlem spojeným se vstupy obou operandů této konjunkce. Práh tohoto hradla je součtem prahů příslušných dvou hradel, které původně ve druhé vrstvě realizovaly operandy g_i. Tím ušetříme třetí vrstvu a získáme výsledný prahový obvod hloubky 5. □

Jako důsledek věty 7.34 dostaneme efektivní prahový obvod s malými vahami pro násobení dvou čísel.

Důsledek 7.35 *Součin dvou n-bitových přirozených čísel lze počítat prahovým obvodem velikosti $O(n^2)$, hloubky 6 a maximální váhy $O(n)$.*

Důkaz: Počítáme součin z dvou přirozených čísel $x, y \in \mathbb{N}$. Nechť $y_1, \ldots, y_n \in \{0,1\}^n$ je binární reprezentace čísla y. Tj. můžeme psát

$$z = xy = x\sum_{i=1}^{n} 2^{n-i}y_i = \sum_{i=1}^{n} 2^{n-i}(x \cdot y_i)\,. \tag{7.22}$$

Součiny $(x \cdot y_i)$ v součtu (7.22) pro $i = 1, \ldots, n$ lze počítat jednou vrstvou $O(n^2)$ hradel AND, tj. prahových hradel s jednotkovými váhami. Součet (7.22) n n-bitových čísel lze pak podle věty 7.34 realizovat pomocí prahového obvodu velikosti $O(n^2)$, hloubky 5 a maximální váhy $O(n)$. Tedy výsledný prahový obvod s malými váhami pro požadovaný součin má velikost $O(n^2)$ a hloubku 6. □

Ještě bez důkazu uvedeme analogii věty 7.34 a důsledku 7.35 pro prahové obvody s jednotkovými váhami.

Věta 7.36 (Parberry, Pippenger, Paterson [212, 213]) *Součet n m-bitových celých čísel lze počítat prahovým obvodem s jednotkovými váhami velikosti $O(nm+m^2\log\log\log n/\log\log n+n\log n)$ a hloubky 7. Součin dvou n-bitových přirozených čísel lze počítat prahovým obvodem s jednotkovými váhami velikosti $O(n^2)$ a hloubky 8.*

Předchozí výsledky naznačují, že prahové obvody polynomiální velikosti a konstantní hloubky představují silný výpočetní prostředek. Následující věta, kterou uvedeme bez důkazu, ukazuje, že pomocí nich lze počítat analytické funkce, tj. např. odmocninu, sinus, kosinus, logaritmus atd. To ukazuje na velkou výpočetní sílu diskrétních vícevrstvých neuronových sítí s polynomiálním počtem neuronů a konstantním počtem vrstev, které odpovídají neurofyziologické skutečnosti.

Věta 7.37 (Reif, Tate [234]) *Nechť analytická funkce má konvergentní rozvoj do Taylorovy řady $f(x) = \sum_{n=0}^{\infty} c_n(x-x_0)^n$ v intervalu $|x-x_0| \le \varepsilon$, kde $0 < \varepsilon < 1$ a koeficienty $c_n = \frac{a_n}{b_n} \in \mathbb{Q}$ jsou racionální čísla, tj. $a_n, b_n \in \mathbb{Z}$ jsou celá čísla, a navíc $|a_n|, |b_n| \le 2^{n^{O(1)}}$. Pak existuje algoritmus, který pro každé $n \ge 0$ zkonstruuje v polynomiálním čase prahový obvod s jednotkovými váhami polynomiální velikosti a konstantní hloubky, který pro binární reprezentaci $x_1 \ldots x_n \in \{0,1\}^n$ čísla x počítá funkční hodnotu $f(x)$ s přesností 2^{-n^c} pro libovolnou konstantu $c \ge 1$.*

Hradlo prahového obvodu s neomezenými váhami počítá booleovskou prahovou funkci. Pokud však uvažujeme malé, popř. jednotkové váhy, pak podle věty 6.38 nelze pomocí jednoho hradla s omezenými váhami realizovat všechny booleovské prahové funkce. Ty však lze počítat pomocí prahového obvodu s omezenými váhami polynomiální velikosti a konstantní hloubky.

Věta 7.38 *Booleovskou prahovou funkci f s n proměnnými lze počítat prahovým obvodem velikosti $O(n^2\log^2 n)$, s malými váhami a hloubkou 6, resp. s jednotkovými váhami a hloubkou 8.*

Důkaz: Nechť $(w_1, \ldots w_n; h)$ je reprezentace booleovské prahové funkce f. Podle lemmy 6.7 můžeme předpokládat, že $h = 0$, a z věty 6.30 vyplývá, že na reprezentaci jedné váhy w_i $(i = 1, \ldots, n)$ stačí $O(n\log n)$ bitů. První vrstva prahového obvodu bude realizovat součiny konstantních bitů reprezentace váhy w_i s odpovídajícím binárním vstupem $x_i \in \{0,1\}$ pro $i = 1, \ldots, n$ tak, že pro nulový bit reprezentace váhy zavedeme hradlo s konstantním nulovým výstupem a pro jednotkový bit reprezentace váhy w_i uvažujeme hradlo s výstupem x_i, které jen kopíruje vstup x_i, tj. je s ním spojeno jednotkovou váhou a má jednotkový práh. Tomu odpovídá $O(n^2\log n)$ hradel v první vrstvě prahového obvodu, jejichž výstupy představují n binárních reprezentací součinů w_ix_i pro $i = 1, \ldots, n$, z nichž každá se skládá z $O(n\log n)$ bitů. Pro součet těchto n $O(n\log n)$-bitových čísel využijeme prahový obvod velikosti $O(n^2\log^2 n)$, hloubky 5 a maximální váhy $O(n)$ z věty 7.34, resp. prahový obvod s jednotkovými váhami velikosti $O(n^2\log^2 n)$ a hloubky 7 z věty 7.36. Výstup prahového obvodu je realizován hradlem, které odpovídá znaménkovému bitu uvedeného součtu, protože ten pro $h = 0$ představuje

7.2.2 Analogové prahové obvody

Doposud jsme se zabývali prahovými obvody, které odpovídají diskrétním neuronovým sítím. V tomto odstavci zobecníme tento model na tzv. *analogové prahové obvody*, jejichž hradla počítají spojitou funkci složenou z lineární a aktivační (přenosové) funkce. Ve shodě s aplikacemi modelů neuronových sítí můžeme uvažovat různé aktivační funkce. V této kapitole se např. omezíme na standardní sigmoidu (1.9), kterou využívá učící algoritmus backpropagation (viz podkapitolu 2.2). Ačkoliv hradla analogového prahového obvodu počítají reálné funkce, výstupy obvodu zůstávají binární, a proto je potřeba specifikovat, jakým způsobem je zaokrouhlíme.

Definice 7.42 Analogový prahový obvod $C = (V, X, Y, E, \ell)$ *je obvod (viz definici 7.1), kde třída $\mathcal{F}$ hradlových funkcí obsahuje funkce složené z lineární a aktivační (přenosové) funkce. Přesněji zobrazení ℓ přiřadí každému hradlu $j \in V$ s k vstupy $x_1, \ldots, x_k$ obecně reálné váhy a práh $\ell(j) = (w_1, \ldots, w_k; h)$ lineární funkce $\xi_j = \sum_{i=1}^{k} w_i x_i - h$. Pokud neuvedeme jinak, omezíme se vždy na celočíselné hodnoty $w_1, \ldots, w_k, h \in \mathbb{Z}$. Každé nevýstupní hradlo s k vstupy pak počítá reálnou funkci*

$$y = \sigma(\sum_{i=1}^{k} w_i x_i - h)\,, \tag{7.24}$$

kde σ je aktivační funkce. Můžeme např. uvažovat standardní sigmoidu:

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}}\,. \tag{7.25}$$

Výstupní hradlo má binární výstup se separací *$\delta \geq 0$:*

$$y = \begin{cases} 1 & \sum_{i=1}^{k} w_i x_i \geq h + \delta \\ 0 & \sum_{i=1}^{k} w_i x_i < h - \delta\,. \end{cases} \tag{7.26}$$

Zformulujeme bez důkazu větu, která ukazuje, že analogové prahové obvody konstantní velikosti a hloubky 2 počítají více booleovských funkcí než diskrétní prahové obvody stejných parametrů. To znamená, že analogová dvouvrstvá síť s malým konstantním počtem neuronů, která se obvykle používá v praxi, může být silnějším výpočetním prostředkem než její diskrétní verze.

Věta 7.43 (Maass, Schnitger, Sontag [183]) *Booleovskou funkci $F_n : \{0,1\}^{2n} \longrightarrow \{0,1\}$, definovanou jako paritu dvou konsenzuálních funkcí:*

$$\begin{aligned} &F_n(x_1, \ldots, x_n, y_1, \ldots, y_n) \quad = \\ &MAJORITY(x_1, \ldots, x_n) \quad \oplus \quad MAJORITY(y_1, \ldots, y_n)\,, \end{aligned} \tag{7.27}$$

hodnotu booleovské prahové funkce f. Výsledný prahový obvod má tedy velikost $O(n^2 \log^2 n)$, hloubku 6 v případě malých vah, resp. hloubku 8 pro jednotkové váhy. □

Dále budeme implementovat obecnou booleovskou funkci pomocí prahového obvodu. Podobně jako pro alternující obvody (viz větu 7.9) ukážeme nejprve dolní odhad velikosti prahového obvodu.

Věta 7.39 (Siu, Roychowdhury, Kailath [259]) *Existují booleovské funkce n proměnných, které pro výpočet funkční hodnoty vyžadují prahový obvod velikosti $\Omega(2^{\frac{n}{3}})$.*

Důkaz: Technika důkazu je stejná jako ve větě 7.9. Počet různých prahových funkcí $s + n$ proměnných je dle věty 6.32 nejvýše $2^{(s+n+1)^2}$, a tedy počet možných přiřazení těchto funkcí s hradlům lze shora odhadnout $2^{s(s+n+1)^2}$. Potom horní odhad (7.11) pro počet $C(s,n)$ obvodů velikosti s s n vstupy můžeme pro prahové obvody přepsat následujícím způsobem:

$$C(s,n) < 2^{s^2-s} 3^{ns} s 2^{s(s+n+1)^2} = s 2^{s(s+n+1)^2+s^2-s} 3^{ns}\,. \tag{7.23}$$

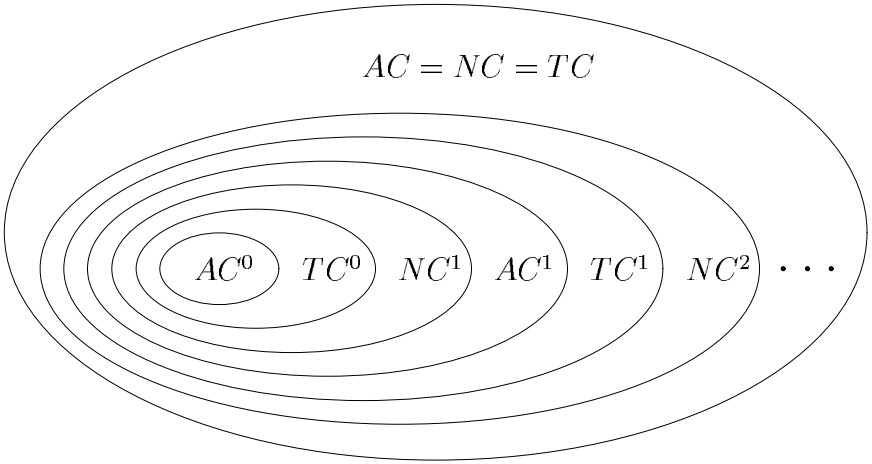
Navíc podmínka (7.12) implikuje $s \geq n$. Tedy (7.23) lze upravit na $C(s,n) < 2^{O(s^3)}$ a porovnáním se vztahem (7.12) dostaneme $2^{O(s^3)} > 2^{2^n}$, tj. $s = \Omega(2^{\frac{n}{3}})$. □

Dolní odhad ve větě 7.39 lze vylepšit na $\Omega(2^{\frac{n}{2}}/\sqrt{n})$. Tvrzení zformulujeme bez důkazu obecně pro vektorovou booleovskou funkci.

Věta 7.40 (Horne, Hush [126]) *Existují vektorové booleovské funkce $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ s n proměnnými a m výstupy, které pro výpočet funkční hodnoty vyžadují prahový obvod velikosti $\Omega(m 2^{\frac{n}{2}}/\sqrt{n - \log m})$.*

Na druhou stranu podle věty 7.10 lze booleovskou funkci n proměnných počítat prahovým obvodem velikosti $O(2^{\frac{n}{2}})$ a hloubky 3, dokonce s jednotkovými váhami (viz větu 6.26), což je v případě jednotkových vah optimální velikost [126]. Následující věta, kterou zformulujeme bez důkazu, ukazuje, že uvedenou velikost prahového obvodu pro výpočet booleovské funkce lze pro neomezené váhy a hloubku 4 vylepšit tak, že je podle věty 7.40 také optimální. V práci [181] je popsána konstrukce prahového obvodu, která je optimální pro vektorové booleovské funkce.

Věta 7.41 (Lupanov [180]) *Booleovskou funkci n proměnných lze počítat prahovým obvodem velikosti $O(2^{\frac{n}{2}}/\sqrt{n})$ a hloubky 4.*



Obr. 7.8: Hierarchie tříd složitosti AC, NC a TC.

Třídy složitosti TC^k ($k \geq 0$) pro prahové obvody můžeme zařadit do hierarchie tříd složitosti pro alternující a klasické obvody. Obrázek 7.8 je rozšířením hierarchie těchto tříd z obrázku 7.7. Např. třídy $TC = NC$ splývají, tedy pro TC můžeme uplatnit stejné výsledky jako pro NC (viz odstavec 7.1.4).

Věta 7.46

(i) $TC^0 \neq AC^0$.

(ii) *Pro každé* $k \geq 0$ *platí inkluze* $AC^k \subseteq TC^k$.

(iii) *Pro každé* $k \geq 0$ *platí inkluze* $TC^k \subseteq NC^{k+1}$.

(iv) $NC = AC = TC$.

Důkaz:

(i) Podle věty 7.32 je $XOR \in TC^0$ a na druhou stranu $XOR \notin AC^0$ podle věty 7.27. Proto $TC^0 \neq AC^0$.

(ii) Alternující obvody jsou podle věty 6.26 speciálním případem prahových obvodů.

lze počítat analogovým prahovým obvodem velikosti 5, hloubky 2 a maximální váhy $O(1)$, se separací $\Omega(1/n^2)$ na výstupu. Na druhou stranu funkci F_n nelze počítat prahovým obvodem (s neomezenými váhami) konstantní velikosti a hloubky 2.

Výsledek ve větě 7.43 lze zobecnit pro neomezenou hloubku [56]. Dolní odhad na velikost prahového obvodu, který počítá tzv. *unární mocninu*

$$F_n(x,y) = \begin{cases} 1 & (\sum_{i=1}^{n} x_i)^2 \geq \sum_{i=1}^{n^2} y_i,\ x \in \{0,1\}^n,\ y \in \{0,1\}^{n^2} \\ 0 & \text{jinak}, \end{cases} \tag{7.28}$$

je $\Omega(\log n)$, zatímco pomocí analogového prahového obvodu ji lze počítat s $\Omega(1)$ separací jen se dvěma hradly.

7.2.3 Třídy složitosti a jejich hierarchie

Podobně jako v případě logických obvodů uvažujeme nekonečné *posloupnosti prahových obvodů* pro různé délky vstupu. Pomocí nich pak definujeme třídy složitosti pro prahové obvody.

Definice 7.44 *Označme UC^k, resp. TC^k, resp. WC^k, ($k \geq 0$) třídu jazyků rozpoznatelných L-uniformní posloupností prahových obvodů (polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta) polylogaritmické hloubky $D(n) = O(\log^k n)$ s jednotkovými, resp. malými, resp. neomezenými, váhami. Dále definujeme třídy složitosti:*

$$UC = \bigcup_{k \geq 0} UC^k \tag{7.29}$$

$$TC = \bigcup_{k \geq 0} TC^k \tag{7.30}$$

$$WC = \bigcup_{k \geq 0} WC^k . \tag{7.31}$$

Následující věta ukazuje, že tři typy tříd složitosti prahových obvodů v definici 7.44 podle velikosti vah ve skutečnosti splývají. Proto se dále můžeme omezit např. na prahové obvody s malými váhami.

Věta 7.45 (Parberry, Schnitger [214]) *Pro každé $k \geq 0$ platí rovnost $UC^k = TC^k = WC^k$.*

Důkaz: Každé prahové hradlo s neomezenými váhami lze podle věty 7.38 nahradit prahovým podobvodem polynomiální velikosti a konstantní hloubky, s jednotkovými, resp. malými váhami. □

Věta 7.50 (Maass, Schnitger, Sontag [183]) *Pro každé $k \geq 1$ platí rovnost $TC^0_k(\sigma) = TC^0_k$.*

Pokud navíc připustíme zvýšení hloubky prahového obvodu o konstantní faktor, pak ekvivalence mezi prahovými a analogovými obvody v rámci polynomiální velikosti platí i pro neomezené váhy a neomezenou hloubku [55].

V hierarchii TC^0 je první inkluze ostrá, tj. $TC^0_1 \subsetneq TC^0_2$. To odpovídá známému argumentu Minského a Paperta (viz podkapitolu 1.1), že jednovrstvá neuronová síť nemůže počítat složitější funkce. Méně známým faktem je novější a hlubší analogický výsledek pro dvouvrstvé neuronové sítě, pomocí kterých nelze s polynomiálním počtem neuronů a s polynomiálními váhami (vzhledem k počtu vstupů) počítat některé funkce (např. booleovský skalární součin), které lze počítat třívrstvou neuronovou sítí s polynomiálním počtem neuronů a s polynomiální velikostí vah. To odpovídá separaci $TC^0_2 \subsetneq TC^0_3$, kterou v závěru této kapitoly dokážeme. Za tímto účelem zformulujeme nejprve několik pomocných vět.

Věta 7.51 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Definujme* booleovský skalární součin *jako funkci $IP : \{0,1\}^{2n} \longrightarrow \{0,1\}$:*

$$IP(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i) \tag{7.33}$$

a označme odpovídající jazyk $IP = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \mid IP(\mathbf{x}, \mathbf{y}) = 1,\ n \geq 1\}$. Pak $IP \in TC^0_3$.

Důkaz: V první vrstvě prahové obvodu je n hradel počítajících konjunkci $x_i \wedge y_i$ pro $i = 1, \ldots, n$. Další dvě vrstvy počítají jejich paritu podle věty 7.32. Tedy výsledný prahový obvod (s jednotkovými váhami) pro IP má velikost $O(n)$ a hloubku 3. □

Definice 7.52 *Nechť $A, B \subseteq \{0,1\}^n$ jsou dvě disjunktní množiny, tj. $A \cap B = \emptyset$. Množinu Q nazveme* ε-diskriminátorem *A nad B, jestliže Q protíná větší procento A než B, tj.*

$$\frac{|Q \cap A|}{|A|} - \frac{|Q \cap B|}{|B|} \geq \varepsilon\,. \tag{7.34}$$

Dále označme doplněk $\bar{A} = \{0,1\}^n \setminus A$. Řekneme, že množina Q je ε-diskriminátorem *A, jestliže je ε-diskriminátorem A nad $\bar{A}$. Navíc nechť C je prahový obvod s n vstupy a jedním výstupem, pak definujeme jazyk $L(C) = \{\mathbf{x} \in \{0,1\}^n \mid C(\mathbf{x}) = 1\}$.*

(iii) Booleovskou prahovou funkci lze počítat pomocí klasického obvodu velikosti $O(n^3 \log^3 n)$ a hloubky $O(\log n)$, což lze ukázat stejným způsobem jako v důkazu věty 7.38, kdy pro součet n $O(n \log n)$-bitových čísel použijeme klasický obvod z věty 7.17. Tedy každé hradlo v prahovém obvodu lze nahradit klasickým podobvodem polynomiální velikosti a logaritmické hloubky, čímž nám vzroste hloubka obvodu o logaritmický faktor.

(iv) Tvrzení je přímým důsledkem (*iii*). □

V dalším výkladu se zaměříme na prahové obvody polynomiální velikosti a konstantní hloubky. Ty odpovídají věrně modelům vícevrstvých neuronových sítí, které při praktických aplikacích obvykle mají dvě až tři vrstvy s přiměřeným počtem neuronů. Nejprve opět zavedeme třídy složitosti, které vytváří možnou hierarchii prahových obvodů s konstantní hloubkou a uvedeme bez důkazu jejich vztah pro různé velikosti vah.

Definice 7.47 *Označme UC^0_k, resp. TC^0_k, resp. WC^0_k, ($k \geq 1$), třídu jazyků rozpoznatelných L-uniformní posloupností prahových obvodů (polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta) konstantní hloubky $D(n) = k$ s jednotkovými, resp. malými, resp. neomezenými, váhami. Zřejmě*

$$TC^0 = \bigcup_{k \geq 1} TC^0_k = UC^0 = WC^0 \tag{7.32}$$

a $TC^0_1 \subseteq TC^0_2 \subseteq TC^0_3 \subseteq \ldots$ je tzv. hierarchie *TC^0.*

Věta 7.48

(i) (Goldmann, Håstad, Razborov [80])
Pro každé $k \geq 1$ platí inkluze $WC^0_k \subseteq TC^0_{k+1}$.

(ii) *Pro každé $k \geq 1$ platí inkluze $TC^0_k \subseteq UC^0_{k+1}$.*

Zavedeme také analogickou hierarchii tříd pro analogové prahové obvody.

Definice 7.49 *Označme $TC^0_k(\sigma)$ ($k \geq 1$) třídu jazyků rozpoznatelných L-uniformní posloupností analogových prahových obvodů (polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta) konstantní hloubky $D(n) = k$ s malými váhami.*

Ačkoliv podle věty 7.43 jsou analogové prahové obvody silnějším výpočetním prostředkem než prahové obvody, následující věta, kterou uvedeme bez důkazu, ukazuje, že v rámci polynomiální velikosti obvodu mezi nimi není rozdílu dokonce při stejné hloubce. Příslušné hierarchie z definic 7.47 a 7.49 splývají.

Věta 7.56 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Každý prahový obvod hloubky 2 a váhy $|w|$, který počítá $IP : \{0,1\}^{2n} \longrightarrow \{0,1\}$, má velikost aspoň $\Omega(2^{\frac{n}{2}}/|w|^2)$.*

Důkaz: Nechť C_1 je prahový obvod velikosti s, hloubky 2 a váhy $|w|$, který počítá $IP \cap \{0,1\}^{2n}$. Podle (i) a (ii) z lemmy 6.27 existuje prahový obvod C_2 velikosti s, hloubky 2 a váhy $|w|$, který počítá $\overline{IP}$ a jehož výstupní hradlo má kladné váhy. Z něho dále vytvoříme ekvivalentní prahový obvod C, kde uvedené výstupní hradlo ve druhé vrstvě má jednotkové váhy. Jeho spoj s kladnou celou váhou $v > 1$ nahradíme v spoji s jednotkovými váhami, které vedou od v kopií odpovídajícího hradla, přidaných do první vrstvy. Prahový obvod C má pak velikost $s|w|$. Výstupní hradlo v C je spojeno s $m = O(s|w|)$ prahovými podobvody C_j $(j = 1, \ldots, m)$, resp. jednotlivými hradly C_j v první vrstvě obvodu C, která počítají booleovské prahové funkce. Podle lemmy 7.53 existuje $1 \leq k \leq m$ takové, že $L(C_k)$ je $1/m$-diskriminátor $L(C) = \overline{IP} \cap \{0,1\}^{2n}$. Nechť $(w_1, \ldots, w_{2n}; h)$ je příslušná celočíselná reprezentace C_k. Označme

$$X_r = \left\{ x_1 \ldots x_n \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} w_i x_i = r \right\} \tag{7.40}$$

$$Y_r = \left\{ y_1 \ldots y_n \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} w_{n+i} y_i \geq r \right\} \tag{7.41}$$

pro celé číslo $r \in \mathbb{Z}$. Zřejmě platí

$$L(C_k) = \bigcup_{p=-|w|}^{|w|} (X_p \times Y_{h-p}) \; . \tag{7.42}$$

Počítejme

$$\begin{aligned}
& \Big| |L(C_k) \cap L(C)| - \left|L(C_k) \cap \overline{L(C)}\right| \Big| \leq \\
\leq \; & \left| \sum_{p=-|w|}^{|w|} \left|(X_p \times Y_{h-p}) \cap \overline{IP}\right| - |L(C_k) \cap IP| \right| = \qquad (7.43) \\
= \; & \left| |L(C_k) \cap IP| - \sum_{p=-|w|}^{|w|} \left|(X_p \times Y_{h-p}) \cap \overline{IP}\right| \right| \leq \\
\leq \; & \left| \sum_{p=-|w|}^{|w|} |(X_p \times Y_{h-p}) \cap IP| - \sum_{p=-|w|}^{|w|} \left|(X_p \times Y_{h-p}) \cap \overline{IP}\right| \right| \leq \qquad (7.44)
\end{aligned}$$

Lemma 7.53 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Nechť C je prahový obvod s jedním výstupním hradlem, které je spojeno přes jednotkové váhy s m prahovými podobvody $C_1, \ldots, C_m$. Pak existuje $1 \leq k \leq m$ takové, že $L(C_k)$ je $1/m$-diskriminátor $L(C)$.*

Důkaz: Nechť C má n vstupů a jeho výstupní hradlo má práh h. Tedy pro každé $\mathbf{x} \in L(C)$ aspoň h podobvodů C_j $(1 \leq j \leq m)$ má pro $\mathbf{x}$ výstup 1, tj. $\mathbf{x} \in L(C_j)$. Z toho vyplývá, že

$$\sum_{j=1}^{m} |L(C_j) \cap L(C)| \geq h|L(C)| \, . \tag{7.35}$$

Podobně, pro každé $\mathbf{x} \in \overline{L(C)}$ nejvýše $h-1$ podobvodů C_j $(1 \leq j \leq m)$ má pro $\mathbf{x}$ výstup 1, tj. $\mathbf{x} \in L(C_j)$, a proto

$$\sum_{j=1}^{m} \left|L(C_j) \cap \overline{L(C)}\right| \leq (h-1)|\overline{L(C)}| \, . \tag{7.36}$$

K nerovnosti (7.35) přičteme nerovnost (7.36) vynásobenou -1 a dostáváme:

$$\sum_{j=1}^{m} \left(\frac{|L(C_j) \cap L(C)|}{|L(C)|} - \frac{\left|L(C_j) \cap \overline{L(C)}\right|}{|\overline{L(C)}|} \right) \geq 1 \, . \tag{7.37}$$

Podle (7.37) tedy musí existovat podobvod C_k $(1 \leq k \leq m)$ takový, že

$$\frac{|L(C_k) \cap L(C)|}{|L(C)|} - \frac{\left|L(C_k) \cap \overline{L(C)}\right|}{|\overline{L(C)}|} \geq 1/m \, , \tag{7.38}$$

tj. $L(C_k)$ je $1/m$-diskriminátor $L(C)$. □

Lemma 7.54 (Lindsey [25]) *Pro všechna $X, Y \subseteq \{0,1\}^n$ platí*

$$\Big| |(X \times Y) \cap IP| - \left|(X \times Y) \cap \overline{IP}\right| \Big| \leq \sqrt{2^n |X| \cdot |Y|} \, . \tag{7.39}$$

Lemma 7.55 $\left|IP \cap \{0,1\}^{2n}\right| = 2^{n-1}(2^n - 1)$.

Důkaz: Označme $T(n) = |IP \cap \{0,1\}^{2n}|$ a $F(n) = |\overline{IP} \cap \{0,1\}^{2n}|$. Zřejmě $T(1) = 1$ $(IP(1,1) = 1)$ a $F(1) = 3$ $(IP(1,0) = IP(0,1) = IP(0,0) = 0)$. Obecně $T(n) = 3T(n-1) + F(n-1)$ a $F(n) = 3F(n-1) + T(n-1)$ pro $n > 1$, protože $IP(\mathbf{x}', 1, \mathbf{y}', 0) = IP(\mathbf{x}', 0, \mathbf{y}', 1) = IP(\mathbf{x}', 0, \mathbf{y}', 0) = IP(\mathbf{x}', \mathbf{y}')$ a $IP(\mathbf{x}', 1, \mathbf{y}', 1) = \overline{IP}(\mathbf{x}', \mathbf{y}')$ pro $\mathbf{x}', \mathbf{y}' \in \{0,1\}^{n-1}$. Dokážeme indukcí dle n, že $T(n) = 2^{n-1}(2^n - 1)$ a $F(n) = 2^{n-1}(2^n + 1)$. Víme, že $T(1) = 1$ a $F(1) = 3$. Nechť tedy tvrzení platí pro $n-1$ $(n > 1)$. Potom dle indukčního předpokladu $T(n) = 3T(n-1) + F(n-1) = 3 \cdot 2^{n-2}(2^{n-1} - 1) + 2^{n-2}(2^{n-1} + 1) = 2^{n-1}(2^n - 1)$ a podobně pro $F(n)$. □

Věta 7.56 implikuje podobnou separaci pro nepolynomiální váhy. Např. pro všechna $0 < \varepsilon < \frac{1}{4}$ každý prahový obvod hloubky 2 a váhy $2^{\varepsilon n}$, který počítá IP, má velikost aspoň $\Omega(2^{\frac{(1-4\varepsilon)n}{2}})$. Další separace hierarchie TC^0 není známa, ani zda dojde ke zhroucení této hierarchie v $TC^0_3 = TC^0$. Tento kolaps by znamenal, že třívrstvé neuronové sítě jsou v rámci polynomiálního počtu neuronů schopny již počítat vše, co lze realizovat pomocí vícevrstvých neuronových sítí s konstantním počtem vrstev. Uvažuje se i slabší hypotéza, že $AC^0 \subseteq TC^0_3$ [138], avšak je známo jen to, že každou funkci z AC^0 lze počítat prahovým obvodem hloubky 3 a velikosti $n^{\log^c n}$ [5].

$$\leq \sum_{p=-|w|}^{|w|} \Big| |(X_p \times Y_{h-p}) \cap IP| - \left|(X_p \times Y_{h-p}) \cap \overline{IP}\right| \Big| \leq$$

$$\leq \sum_{p=-|w|}^{|w|} \sqrt{2^n |X_p| \cdot |Y_{h-p}|} \leq (2|w|+1)2^{\frac{3n}{2}} . \qquad (7.45)$$

V řádcích (7.43), (7.44) jsme za $L(C_k)$ dosadili (7.42). V řádku (7.45) jsme aplikovali lemmu 7.54 a využili faktu, že $|X_p| \leq 2^n$ a $|Y_{h-p}| \leq 2^n$ pro $-|w| \leq p \leq |w|$.

Navíc $L(C_k)$ je $1/m$-diskriminátor $L(C)$, tj.

$$1/m \leq \frac{|L(C_k) \cap L(C)|}{|L(C)|} - \frac{\left|L(C_k) \cap \overline{L(C)}\right|}{|\overline{L(C)}|} , \qquad (7.46)$$

a pomocí lemmy 7.55 obdržíme

$$\begin{aligned} 1/m &\leq \frac{|L(C_k) \cap L(C)|}{2^{n-1}(2^n+1)} - \frac{\left|L(C_k) \cap \overline{L(C)}\right|}{2^{n-1}(2^n-1)} \leq \\ &\leq \frac{|L(C_k) \cap L(C)| - \left|L(C_k) \cap \overline{L(C)}\right|}{2^{2n-1}} . \qquad (7.47) \end{aligned}$$

Čitatel ve výrazu (7.47) je díky lemmě 7.53 kladný, a proto pomocí vztahu (7.45) dostáváme

$$1/m \leq \frac{(2|w|+1)2^{\frac{3n}{2}}}{2^{2n-1}} = \frac{2|w|+1}{2^{\frac{n}{2}-1}} . \qquad (7.48)$$

Z toho vyplývá, že $O(s|w|) = m \geq 2^{\frac{n}{2}-1}/(2|w|+1) = \Omega(2^{\frac{n}{2}}/|w|)$. Tedy velikost prahového obvodu C_1, který počítá $IP \cap \{0,1\}^{2n}$, je $s = \Omega(2^{\frac{n}{2}}/|w|^2)$. □

Důsledek 7.57 $TC^0_1 \subsetneq TC^0_2 \subsetneq TC^0_3$.

Důkaz: Funkci $XOR \notin TC^0_1$ nelze podle věty 6.26 počítat jednou vrstvou prahových hradel, ale můžeme ji podle věty 7.32 počítat pomocí prahového obvodu (s jednotkovými váhami) polynomiální velikosti a hloubky 2, tj. $XOR \in TC^0_2$. Tedy $TC^0_1 \subsetneq TC^0_2$.

Podobně podle věty 7.51 je $IP \in TC^0_3$, ale na druhou stranu podle věty 7.56 každý prahový obvod hloubky 2 a polynomiální váhy n^c (c je konstanta), který počítá IP, má velikost $\Omega(2^{\frac{n}{2}}/n^{2c})$ větší než polynomiální, tj. $IP \notin TC^0_2$. Tedy $TC^0_2 \subsetneq TC^0_3$. □

Dále definujeme deskriptivní míry složitosti neuronové sítě. Počet neuronů $s = |V|$ *v síti se nazývá* velikost *a součet absolutních hodnot všech vah*

$$|w| = \sum_{i,j \in V} |w(i,j)| \tag{8.1}$$

určuje váhu *neuronové sítě* N. Maximální váha w_{max} *je v absolutní hodnotě největší váha v neuronové síti.*

Je zřejmé, že $1 \leq w_{max} \leq |w|$ a navíc díky souvislosti topologie platí, že $s - 1 \leq |w| \leq s^2 w_{max}$. Následující definice formalizuje výpočet cyklické neuronové sítě, při kterém neurony počítají booleovské prahové funkce (viz podkapitolu 6.4), tj. reprezentují perceptrony.

Definice 8.2 *Nechť* $N = (V, X, Y, A, w, h)$ *je cyklická neuronová síť a označme* $X = \{1, \ldots, n\}$ *množinu* n *vstupních neuronů. Dále popíšeme výpočet* N *pro* vstup $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$. *Za tímto účelem definujeme binární* stav $y_j^{(t)} \in \{0,1\}$ *a celočíselný* vnitřní potenciál $\xi_j^{(t)} \in \mathbb{Z}$ *neuronu* $j \in V$ *v diskrétním čase výpočtu* $t = 0, 1, 2, \ldots$. *V počátečním čase 0 inicializujeme stavy vstupních neuronů na vstup sítě a zbylé neurony podle množiny iniciálně aktivních neuronů:*

$$y_j^{(0)} = \begin{cases} x_j & j \in X \\ 1 & j \in A \\ 0 & j \in V \setminus (X \cup A)\,. \end{cases} \tag{8.2}$$

V čase $t \geq 0$ *je určen vnitřní potenciál každého neuronu podle stavů neuronů v síti:*

$$\xi_j^{(t)} = \sum_{i \in V} w_{ji} y_i^{(t)}\,. \tag{8.3}$$

V čase $t \geq 1$ *jsou pak aktualizovány stavy neuronů* $j \in \alpha_t$ *z množiny* $\alpha_t \subseteq V$*:*

$$y_j^{(t)} = \begin{cases} 1 & \xi_j^{(t-1)} \geq h_j \\ 0 & \xi_j^{(t-1)} < h_j \end{cases} \qquad j \in \alpha_t \tag{8.4}$$

a ostatní neurony svůj stav nemění, tj. $y_j^{(t)} = y_j^{(t-1)}$ *pro* $j \notin \alpha_t$. *Řekneme, že neuron* $j \in V$ *je v čase* t aktivní, *pokud* $y_j^{(t)} = 1$, *v opačném případě je* pasivní. *Podle volby* α_t *(*$t \geq 1$*) rozlišujeme režimy výpočtu* N. *Pokud volba* α_t *je deterministická a centrálně řízená, hovoříme o* synchronním *výpočtu. Naopak v případě náhodné distribuované aktualizace neuronů se jedná o* asynchronní *výpočet. Neuronová síť pracuje v* sekvenčním *režimu, jestliže* $|\alpha_t| \leq 1$ *pro všechna* $t \geq 1$ *a nachází se v* paralelním *režimu, pokud* $|\alpha_t| \geq 2$ *pro nějaké* $t \geq 1$. *Dále uvažujeme* plně paralelní *režim, kdy* $\alpha_t = V$ *pro*

Kapitola 8

Cyklické neuronové sítě

8.1 Formální model

V této podkapitole budeme formalizovat výpočetní model diskrétní cyklické neuronové sítě (viz odstavec 1.3.2), který odpovídá jejímu aktivnímu režimu. Na rozdíl od prahových obvodů v kapitole 7, které modelovaly výpočet diskrétní acyklické neuronové sítě, připouštíme v topologii tohoto modelu existenci cyklu stejně jako u Hopfieldovy sítě (viz podkapitolu 3.2) nebo u Boltzmannova stroje (viz podkapitolu 3.4). Tedy výpočet nemusí obecně skončit.

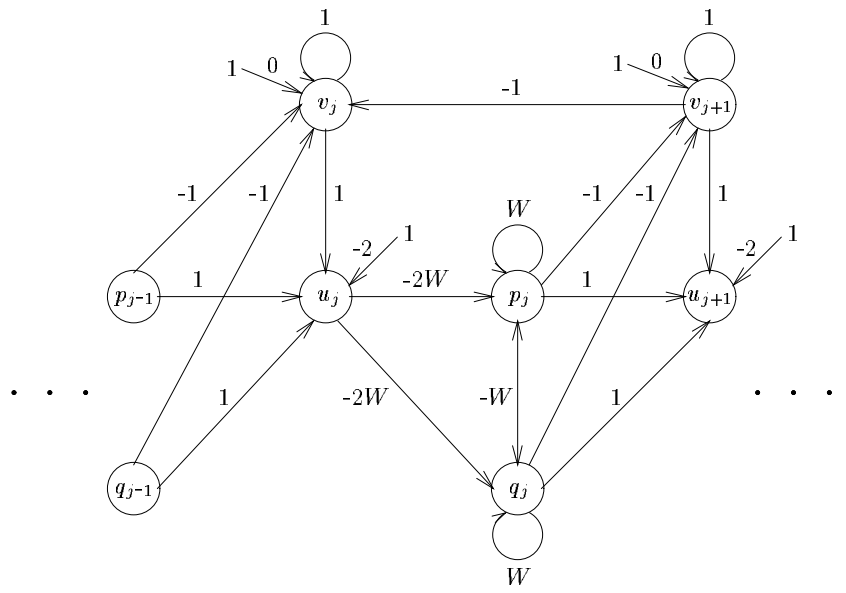
Formální definice cyklické sítě a jejího výpočtu jsou podobně jako u obvodů (srovnejte s definicemi 7.1, 7.2) relativně nepřehledné, i když principy modelu jsou celkem jednoduché a přirozené. V některých důkazech se na ni budeme odvolávat, abychom upřesnili neformální postupy.

Definice 8.1 Cyklická neuronová síť *je šestice* $N = (V, X, Y, A, w, h)$, *kde* V *je konečná množina* neuronů, $X \subseteq V$ *je množina* $n = |X|$ vstupních *neuronů,* $Y \subseteq V$ *je množina* výstupních *neuronů,* $A \subseteq V \setminus X$ *je množina* iniciálně aktivních *(nevstupních) neuronů, zobrazení* $w : V \times V \longrightarrow \mathbb{Z}$ *přiřazuje každé uspořádané dvojici neuronů* $(i,j) \in V \times V$ *celočíselnou* váhu $w(i,j) = w_{ji} \in \mathbb{Z}$ *a zobrazení* $h : V \longrightarrow \mathbb{Z}$ *určuje u každého neuronu* $j \in V$ *jeho celočíselný* práh $h(j) = h_j \in \mathbb{Z}$. *Označme navíc* $E = \{(i,j) \in V \times V \mid w(i,j) \neq 0\}$ *množinu* spojů *a obecně cyklický orientovaný graf* (V, E) *pak tvoří* architekturu *(*topologii*) neuronové sítě* N, *u které předpokládáme souvislost. Většinou také předpokládáme, že váhy, které explicitně nezmiňujeme, jsou nulové a odpovídající neurony nejsou v architektuře sítě spojeny. Neuronová síť je* jednoduchá, *pokud* $w(j,j) = 0$ *pro všechna* $j \in V$, *tj. má nulové zpětné vazby.*

lemmy 6.29 můžeme navíc bez újmy na obecnosti předpokládat, že booleovské prahové funkce neuronů v N mají rozhodující reprezentace. Tomu odpovídají následující váhy a prahy těchto neuronů:

$$\begin{aligned} w'(p_i,p_j) = w'(q_i,q_j) &= w(i,j) \qquad j,i \in V \\ w'(q_i,p_j) = w'(p_i,q_j) &= -w(i,j) \\ h'(p_j) = -h'(q_j) &= 2h(j) - \sum_{i \in V} w(i,j) \end{aligned} \tag{8.6}$$

a také množina iniciálně aktivních (podobně vstupních a výstupních) neuronů $A' = \{p_j \mid j \in A\} \cup \{q_j \mid j \in V \setminus A\} \cup \{v_1\}$. Pro potřeby synchronizace výpočtu budeme stavy neuronů p_j, q_j ($j \in V$) fixovat silnými zpětnými a vzájemnými vazbami, tj. $w'(p_j,p_j) = w'(q_j,q_j) = W$ a $w'(p_j,q_j) = w'(q_j,p_j) = -W$, kde W volíme tak, aby $W > \max_{j \in V} 2(\sum_{i \in V} |w(i,j)| + |h(j)|)$. Aktualizaci takto fixovaných neuronů budeme postupně uvolňovat v pořadí odpovídajícím systematickému synchronnímu výpočtu.



Obr. 8.1: Synchronizace asynchronního výpočtu.

Neurony v_j slouží k zachování správného pořadí aktualizace a jsou tedy v tomto pořadí zapojeny do cyklu, tj. $w'(v_{j+1},v_j) = -1$ pro $j = 1,\ldots,s$

všechna $t \geq 1$, produktivní *výpočet, při kterém α_t obsahuje aspoň jeden* nestabilní neuron*, tj. pro všechna $t \geq 1$ existuje $j \in \alpha_t$ takové, že $y_j^{(t)} \neq y_j^{(t-1)}$, výpočet s* fixovanými vstupy*, tj. $\alpha_t \cap X = \emptyset$ pro všechna $t \geq 1$, a* systematický *výpočet, kdy je po řadě aktualizováno všech $s = |V|$ neuronů, tj. $\alpha_t = \{j\}$ pro $j \in V$, v* makroskopickém čase *$\tau \geq 0$ ($\tau s < t \leq (\tau+1)s$).*

V předchozí definici se uvažuje obecně nekonečný výpočet, avšak smysluplné výpočty cyklické neuronové sítě končí a stavy výstupních neuronů určují jejich výsledek.

Definice 8.3 *Nechť $N = (V,X,Y,A,w,h)$ je cyklická neuronová síť, u níž probíhá výpočet na vstup $\mathbf{x} \in \{0,1\}^n$. Definujeme* stav $\mathbf{y}_J^{(t)} \in \{0,1\}^m$ *podmnožiny $J = \{j_1,\ldots,j_m\} \subseteq V$ jejích neuronů v čase $t \geq 0$:*

$$\mathbf{y}_J^{(t)} = (y_{j_1}^{(t)},\ldots,y_{j_m}^{(t)}) \tag{8.5}$$

a označíme $\mathbf{y}^{(t)} = \mathbf{y}_V^{(t)}$. Produktivní výpočet N je ukončen *(*zastaví se, dosáhl stabilního stavu, konverguje*) v čase $t^\star \geq 0$, jestliže $\mathbf{y}^{(t^\star)} = \mathbf{y}^{(t^\star+c)}$ pro všechna $c \geq 1$. Nejmenší takové $t^\star$ se nazývá* čas výpočtu *a $\mathbf{y}_Y^{(t^\star)}$ je* výstup *N pro vstup $\mathbf{x}$.*

Definice 8.4 *Cyklická neuronová síť N_2 je $f(t)$-*ekvivalentní *se sítí N_1, jestliže pro všechny vstupy $\mathbf{x}$ a každý výpočet N_1 na $\mathbf{x}$, který skončí v čase t, existuje výpočet N_2 na vstup $\mathbf{x}$, který skončí v čase $f(t)$ se stejným výstupem. Neuronová síť N_2 je* ekvivalentní *se sítí N_1, jestliže je s ní t-ekvivalentní.*

Asynchronní model neuronové sítě, v němž neurony aktualizují své stavy v náhodném pořadí, je výpočetně ekvivalentní se zdánlivě silnějšími modely pracujícími v systematickém sekvenčním či plně paralelním režimu. To nám umožní se v následujícím výkladu většinou omezit na synchronní výpočty.

Věta 8.5 (Orponen [211]) *Nechť $N = (V,X,Y,A,w,h)$ je jednoduchá cyklická neuronová síť velikosti s, která v sekvenčním režimu realizuje synchronní systematické výpočty, jejichž ukončení je signalizováno aktivitou speciálního neuronu. Pak existuje asynchronní sekvenční neuronová síť $N' = (V',X',Y',A',w',h')$ velikosti $4s+1$, která je $(6t+2)$-ekvivalentní s N.*

Důkaz: Nechť očíslování neuronů $V = \{1,\ldots,s\}$ určuje pořadí aktualizace synchronního systematického sekvenčního výpočtu N. Každému neuronu z N budou odpovídat čtyři neurony v N', tj. $V' = \{p_j,q_j,u_j,v_j \mid j \in V\} \cup \{z\}$. Stavy neuronů p_j budou během asynchronního výpočtu reprezentovat stavy původních neuronů $j \in V$ a stavy neuronů q_j jsou jejich negacemi. Podle

a $h_2(i) = 1$ pro $i \in X$. Zbylé váhy a prahy se nemění, tj. $w_2(i,j) = w_1(i,j)$ a $h_2(j) = h_1(j)$ pro $i \in V$ a $j \in V \setminus X$. □

Nyní ještě porovnáme výpočetní sílu konvergentních cyklických neuronových sítí a acyklických sítí (prahových obvodů). Standardní technika použitá při simulaci cyklické neuronové sítě pomocí prahového obvodu pochází od Savage [247].

Věta 8.7 (Goldschlager, Parberry [81, 215]) *Pro každou cyklickou neuronovou síť $N = (V, X, Y, A, w, h)$ velikosti s a váhy $|w|$, která se v plně paralelním režimu zastaví v čase $t^\star$ pro všechny vstupy, existuje ekvivalentní prahový obvod $C = (V', X', Y', E, \ell)$ velikosti $st^\star$, váhy $|w|t^\star$ a hloubky $t^\star$.*

Důkaz: Prahový obvod C simuluje výpočet neuronové sítě N tak, že každému časovému kroku výpočtu N odpovídá jedna vrstva C, která se skládá z neuronů V. Stav neuronů sítě N v čase t je roven stavům neuronů v t-té vrstvě. Architektura N určuje spojení sousedních vrstev. Následuje formální popis konstrukce C. Označme $X = \{1, \ldots, n\}$.

$$\begin{aligned} V' &= \{(j,t) \mid j \in V, 1 \le t \le t^\star\} \\ X' &= \{x_1, \ldots, x_n\} \\ Y' &= \{(j,t^\star) \mid j \in Y\} \\ E &= \{((i,t),(j,t+1)) \mid w(i,j) \neq 0;\, i,j \in V;\, 1 \le t < t^\star\} \cup \\ &\quad \cup \{(x_i,(j,1)) \mid w(i,j) \neq 0;\, j \in V;\, i = 1, \ldots, n\}\,. \end{aligned} \tag{8.7}$$

Pro definici ℓ nejprve předpokládejme, že $i_1, \ldots, i_k \in X$ jsou všechny vstupní neurony N spojené s neuronem $j \in V$, tj. $w(i_p, j) \neq 0$ $(p = 1, \ldots, k)$. Definujeme reprezentaci booleovské prahové funkce přiřazené hradlům $(j,1) \in V'$:

$$\ell(j,1) = \left(w(i_1,j), \ldots, w(i_k,j); h(j) - \sum_{i \in A} w(i,j) \right)\,. \tag{8.8}$$

Podobně předpokládejme, že $i_1, \ldots, i_k \in V$ jsou všechny neurony N spojené s neuronem $j \in V$. Definujeme reprezentaci

$$\ell(j,t) = (w(i_1,j), \ldots, w(i_k,j); h(j)) \tag{8.9}$$

booleovské prahové funkce přiřazené hradlům $(j,t) \in V'$ pro $t = 2, \ldots, t^\star$. Indukcí dle t lze ukázat ekvivalenci N a C. □

8.2 Zastavení cyklických sítí

Podobně jako u turingovských výpočtů nemusí výpočet cyklické neuronové sítě skončit. Avšak vzhledem k tomu, že síť velikosti s se může nacházet jen

(kvůli jednoduchosti zápisu v_{s+1} značí také neuron $v_1 \in V'$ a podobně dále pro u_{s+1}). K neuronu v_j $(j = 1, \ldots, s)$, který má nulový práh $h'(v_j) = 0$ a jednotkovou zpětnou vazbu $w'(v_j, v_j) = 1$, jsou dále připojeny neurony p_{j-1}, q_{j-1} s váhou $w'(p_{j-1}, v_j) = w'(q_{j-1}, v_j) = -1$ (opět kvůli jednoduchosti zápisu p_{-1} značí také neuron $p_s \in V'$ a podobně pro q_{-1}). Aktivita neuronu v_j signalizuje začátek simulace aktualizace původního neuronu j. Neurony v_j, p_{j-1}, q_{j-1} jsou spojeny s neuronem u_j $(j = 1, \ldots, s)$ jednotkovou váhou, tj. $w'(v_j, u_j) = w'(p_{j-1}, u_j) = w'(q_{j-1}, u_j) = 1$, jehož práh $h'(u_j) = 2$ zajistí, že aktivita neuronu v_j způsobí aktivitu u_j, protože také aspoň jeden z neuronů p_{j-1}, q_{j-1} je vždy aktivní. Aktivita neuronu u_j uvolní fixaci neuronů p_j, q_j tak, že nejprve jsou oba pasivní (jediný případ, kdy q_j není negací p_j). Tomu odpovídají váhy $w'(u_j, p_j) = w'(u_j, q_j) = -2W$ $(j = 1, \ldots, s)$. Pasivita těchto neuronů způsobí aktivitu neuronu v_{j+1}, která implikuje pasivitu v_j a posléze i pasivitu u_j. Neuron u_{j+1} zatím díky pasivitě p_j, q_j zůstává pasivní. V tomto okamžiku jsou správně aktualizovány neurony p_j, q_j a výpočet dále pokračuje aktivitou u_{j+1}. Část sítě, která odpovídá za synchronizaci aktualizace neuronu $j \in V$ je znázorněna na obrázku 8.1.

Neuronová síť N' je navržena tak, aby v každém okamžiku byl nestabilní právě jeden neuron, který je v produktivním asynchronním výpočtu aktualizován. Simulace jednoho kroku N vyžaduje 6 kroků N'. Navíc neuron p_e odpovídající speciálnímu neuronu $e \in V$, který signalizuje ukončení výpočtu N, aktivuje neuron $z \in V'$, který je spojen s příslušnými neurony z V' dostatečně malou zápornou váhou tak, aby zmrazil také výpočet N'. □

Ve větě 8.5 lze odstranit předpoklad o jednoduchosti sítě při zachování lineárního času simulace a lineární velikosti simulující sítě. Využitím konstrukce z důkazu věty 8.16 lze ukázat obdobnou větu pro paralelní režim [211].

Výpočet neuronové sítě s fixovanými vstupy v definici 8.2 odpovídá např. aktivnímu režimu Boltzmannova stroje (viz odstavec 3.4.1). Ukážeme, že existuje ekvivalentní neuronová síť s nepatrně větší váhou bez fixovaných vstupů.

Věta 8.6 *Pro každou cyklickou neuronovou síť N_1 s n fixovanými vstupy, velikosti s a váhy $|w|$ existuje ekvivalentní neuronová síť N_2 bez fixovaných vstupů, velikosti s a váhy $|w| + n$.*

Důkaz: Nechť cyklická neuronová síť $N_1 = (V, X, Y, A, w_1, h_1)$ má fixovány vstupy. Definujeme ekvivalentní neuronovou síť $N_2 = (V, X, Y, A, w_2, h_2)$ bez fixovaných vstupů tak, že upravíme váhy a prahy sítě N_1. Zrušíme spoje vedoucí ke vstupům, které by mohly během výpočtu způsobit jejich změnu, tj. $w_2(j,i) = 0$ pro $j \in V$, $i \in X$ a $j \neq i$. Dále přidáme zpětnou vazbu ke vstupním neuronům, která spolu s prahem fixuje stav vstupů, tj. $w_2(i,i) = 1$

v 2^s různých stavech, její výpočet se po nějakém čase začne opakovat. Posloupnost různých stavů výpočtu, které se opakují, se nazývá *cyklus*. Stejný termín se používá pro grafový cyklus v topologii neuronové sítě, avšak z kontextu by mělo být snadné rozlišit, co máme v danou chvíli na mysli. Ukážeme příklad neuronové sítě, která se dostane do cyklu dané délky.

Věta 8.8 *Pro každé přirozené číslo $s \in \mathbb{N}$ existuje cyklická neuronová síť, která při libovolném režimu produktivního výpočtu vstoupí do cyklu délky $2s$ stavů.*

Důkaz: Definujme neuronovou síť $N = (V, X, Y, A, w, h)$, jejíž architektura je cyklus délky s, tj. $V = \{1, \ldots, s\}$, $X = Y = A = \emptyset$, $w(j, j+1) = 1$ pro $1 \leq j < s$ a $w(s, 1) = -1$, $h(j) = 1$ pro $1 < j \leq s$ a $h(1) = 0$. Zřejmě neurony N jsou všechny na začátku pasivní, v čase 1 je aktivní neuron 1, který způsobí aktivitu neuronu 2 v čase 2 atd. V čase i jsou tedy aktivní neurony $1, \ldots, i$. Nakonec v čase s jsou všechny neurony aktivní. Dále výpočet pokračuje podobně tak, že neuron 1 je v čase $s+1$ pasivní a způsobí pasivitu neuronu 2 v čase $s+2$ atd. V čase $s+i$ jsou tedy aktivní neurony $i+1, \ldots, s$. Nakonec v čase $2s$ jsou všechny neurony pasivní a neuronová síť N se nachází v počátečním stavu, tedy výpočet se po $2s$ stavech opakuje. Vzhledem k tomu, že během popsaného výpočtu je vždy nestabilní právě jeden neuron, tvrzení platí pro libovolný režim produktivního výpočtu. □

Pro většinu praktických aplikací má smysl uvažovat jen výpočty, které končí ve stabilním stavu. Proto je užitečné se ptát, zda daná cyklická neuronová síť má stabilní stavy. Zformulujeme problém existence stabilního stavu neuronové sítě a ukážeme, že pravděpodobně (tj. pokud $NP \neq P$) neexistuje efektivní algoritmus pro jeho řešení.

Problém stabilního stavu neuronové sítě SNN (**S**table Configuration Problem in **N**eural **N**etworks):
instance: Cyklická neuronová síť N.
otázka: Existuje stabilní stav N?

Věta 8.9 (Alon [8], Godbeer, Lipscomb [77], Porat [229]) *SNN je NP-úplný problém.*

Důkaz: Zřejmě $SNN \in NP$, protože nedeterministický algoritmus v polynomiálním čase uhádne a ověří stabilní stav zadané neuronové sítě. Důkaz, že SNN je NP-těžký problém, probíhá standardním postupem [72], tj. polynomiální redukcí známého NP-úplného problému, v našem případě *problému splnitelnosti booleovské formule* SAT, na problém SNN.

Problém splnitelnosti booleovské formule SAT (**SAT**isfiability Problem):
instance: Booleovská formule f v konjunktivní normální formě.
otázka: Je f splnitelná, tj. existuje ohodnocení booleovských proměnných takové, že hodnota f je 1?

Je všeobecně známo, že SAT je NP-úplný problém [47].

Nechť tedy f je instance SAT. V polynomiálním čase zkonstruujeme odpovídající instanci N_f problému SNN takovou, že f je splnitelná, právě když N_f má stabilní stav. Nejprve podle důkazu věty 7.7 vytvoříme prahový obvod C_f lineární velikosti vzhledem k délce f, který ji vyhodnocuje. Obvod C_f lze chápat jako neuronovou síť, když jeho vstupy odpovídající booleovským proměnným f budeme považovat za vstupní neurony, které navíc během výpočtu sítě fixujeme pomocí věty 8.6. V neuronové síti N_f výstupní neuron v obvodu C_f ještě připojíme přes jednotkovou váhu $w(v, c) = 1$ k dalšímu neuronu c, který má nulový práh $h(c) = 0$ a zápornou jednotkovou zpětnou vazbu $w(c, c) = -1$. Z toho plyne, že c je stabilní (aktivní), právě když v je aktivní. Tedy N_f je stabilní, právě když neuron v je aktivní, tj. existuje vstup, pro který výstup C_f je 1, a to je, právě když f je splnitelná. □

Důsledek 8.10 *Problém SNN je NP-úplný, i když se omezíme na cyklické neuronové sítě s maximální váhou 1 a*

(i) *buď počet vstupů každého neuronu je nejvýše 2 a všechny končící výpočty se zastaví v polynomiálním čase,*

(ii) *nebo všechny končící výpočty se v plně paralelním režimu zastaví v konstantním čase.*

Důkaz: V důkazu věty 8.9 jsou při konstrukci N_f použity jen jednotkové váhy.

(i) Navíc obvod C_f je logický obvod, ke kterému existuje podle věty 7.13 ekvivalentní klasický obvod polynomiální velikosti, a všechny končící výpočty N_f mění stav každého neuronu nejvýše jednou.

(ii) Obvod C_f má konstantní hloubku, která odpovídá paralelnímu času výpočtu N_f v plně paralelním režimu. □

Možná zajímavější otázka než existence stabilního stavu je problém, zda daná cyklická neuronová síť skončí svůj výpočet pro daný vstup. Avšak tento problém je ještě těžší.

metrické, tj. pro každé $i, j \in V$ platí $w(i,j) = w(j,i)$ (při výpočtu váhy sítě uvažujeme váhu každého symetrického spoje jen jednou) a architekturu (V, E) tvoří neorientovaný graf. Dále N se nazývá semijednoduchá *(zobecnění pojmu jednoduché sítě z definice 8.1), jestliže má jen nezáporné zpětné vazby, tj. pro každý neuron $j \in V$ platí $w(j,j) \geq 0$. Neuronová síť N je v* normálním tvaru, *jestliže reprezentace všech neuronů (tj. odpovídajících booleovských prahových funkcí) jsou rozhodující (viz definici 6.28) a mají nulový práh.*

V definice 8.12 je symetrická síť zavedena jako speciální případ cyklické neuronové sítě, která má podle definice 8.2 binární stavy neuronů z množiny $\{0,1\}$, zatímco v popisu Hopfieldovy sítě v podkapitole 3.2 předpokládáme bipolární stavy neuronů z množiny $\{-1,1\}$. Poznámka k definici 6.24 však ukazuje, že oba přístupy jsou ekvivalentní. Navíc pro symetrické neuronové sítě platí analogie věty 8.5 o asynchronní simulaci synchronního výpočtu v sekvenčním režimu, kterou zformulujeme bez důkazu.

Věta 8.13 (Orponen [211]) *Nechť $N = (V, X, Y, A, w, h)$ je jednoduchá Hopfieldova síť velikosti s, která v sekvenčním režimu realizuje synchronní systematické výpočty v čase $t^\star$. Pak existuje asynchronní sekvenční Hopfieldova síť $N' = (V', X', Y', A', w', h')$ velikosti $8 \log t^\star + 6s + 8 \log s + 10$, která je $38t$-ekvivalentní s N.*

Také pro symetrické sítě lze ve větě 8.13 odstranit předpoklad o jednoduchosti sítě a využít konstrukci z důkazu věty 8.16 pro paralelní verzi této věty [211].

Věta 8.14 (Hopfield [122]) *Každý produktivní sekvenční výpočet jednoduché Hopfieldovy sítě N v normálním tvaru velikosti s, s váhou $|w|$ a maximální váhou w_{max}, skončí v čase $2|w| = O(s^2 w_{max}) = O(2^s)$.*

Důkaz: Nechť $N = (V, X, Y, A, w, h)$ je Hopfieldova síť a uvažujeme její produktivní sekvenční výpočet. Pro něj definujeme *stabilitu* $\beta(t)$ sítě N v čase t, jejíž hodnota s opačným znaménkem odpovídá energii Hopfieldovy sítě z odstavce 3.2.2:

$$\beta(t) = \beta(\mathbf{y}^{(t)}) = \frac{1}{2} \sum_{j \in V} y_j^{(t)} \xi_j^{(t)} = \frac{1}{2} \sum_{j \in V} \sum_{i \in V} w_{ji} y_i^{(t)} y_j^{(t)} . \tag{8.10}$$

Dále uvažujeme neuron $v \in V$ a určíme jeho příspěvek ke stabilitě $\beta(t)$ v čase t, tj. členy součtu (8.10) pro $j = v$ a $i = v$:

$$\frac{1}{2} \sum_{i \in V} w_{vi} y_i^{(t)} y_v^{(t)} + \frac{1}{2} \sum_{j \in V} w_{jv} y_v^{(t)} y_j^{(t)} . \tag{8.11}$$

Problém zastavení neuronové sítě HNN (**H**alting Problem in **N**eural **N**etworks):
instance: Cyklická neuronová síť N s n vstupy; vstup $\mathbf{x} \in \{0,1\}^n$.
otázka: Zastaví se N v plně paralelním režimu na vstup $\mathbf{x}$?

Věta 8.11 (Lepley, Miller [169]) *HNN je $PSPACE$-úplný problém.*

Důkaz: Zřejmě $HNN \in PSPACE$, protože výpočet cyklické neuronové sítě pro daný vstup lze pomocí Turingova stroje simulovat v polynomiálním prostoru. Tvrzení, že HNN je $PSPACE$-těžký problém, dokážeme tak, že každý problém v $PSPACE$ redukujeme na HNN v polynomiálním čase [72]. Nechť tedy $A \in PSPACE$ je problém řešitelný Turingovým strojem v polynomiálním prostoru. Potom také komplement $\bar{A} \in co - PSPACE = PSPACE$ je řešitelný Turingovým strojem M v polynomiálním prostoru, tj. $\bar{A} = L(M)$, protože třída $PSPACE$ je uzavřená na doplňky. Bez újmy na obecnosti předpokládejme, že M má jednu pásku omezenou polynomem $p(n)$ vzhledem k délce vstupu n. Dále M se zastaví na všechny vstupy a v prvním bitu pásky píše 1, právě když přijímá vstup.

Nechť $\mathbf{x} \in \{0,1\}^n$ je instance $\bar{A}$, tj. vstup pro M. Zkonstruujeme v polynomiálním čase odpovídající cyklickou neuronovou síť $N_\mathbf{x}$ se vstupem $\mathbf{x}$, která simuluje výpočet M nad $\mathbf{x}$ tak, že $\mathbf{x} \in A$, právě když $N_\mathbf{x}$ se zastaví na vstup $\mathbf{x}$. Neuronová síť $N_\mathbf{x}$ se skládá z $p(|\mathbf{x}|)$ podsítí konstantní velikosti, jejichž stavy reprezentují konfiguraci M, tj. každá taková podsíť odpovídá jednomu bitu pásky, popř. kóduje polohu hlavy a stav M. Propojení těchto bloků implementuje lokálně přechodovou funkci M. Neurony odpovídající bitům pásky, na nichž je na začátku výpočtu M vstup $\mathbf{x}$, jsou vstupní. Neuron v odpovídající prvnímu bitu pásky, jehož aktivita signalizuje $\mathbf{x} \in \bar{A}$, je spojen s další podsítí, která vstoupí do cyklu, právě když neuron v je aktivní. Tedy $\mathbf{x} \in A$, právě když při výpočtu M nad $\mathbf{x}$ je v prvním bitu pásky 0, tj. při výpočtu $N_\mathbf{x}$ nad $\mathbf{x}$ je neuron v pasivní a to je, právě když $N_\mathbf{x}$ se zastaví na vstup $\mathbf{x}$. □

8.3 Symetrické neuronové sítě

Jedním z nejznámější modelů neuronových sítí je Hopfieldova síť (viz v podkapitolu 3.2), která se také díky symetrickým spojům často nazývá *symetrická* neuronová síť. Její popularita je způsobena faktem, že za určitých předpokladů se její sekvenční výpočet, na rozdíl od obecné cyklické sítě, zastaví pro každý vstup. V této podkapitole nejprve toto tvrzení i analogickou větu pro paralelní režim formálně dokážeme.

Definice 8.12 *Nechť $N = (V, X, Y, A, w, h)$ je cyklická neuronová síť. Řekneme, že N je* symetrická *(*Hopfieldova*) síť, jestliže má všechny spoje sy-*

Výraz (8.11) lze díky symetrii a jednoduchosti N sečíst:

$$\sum_{i\in V} w_{vi} y_i^{(t)} y_v^{(t)} = y_v^{(t)} \xi_v^{(t)} . \tag{8.12}$$

Tedy příspěvek neuronu $v \in V$ ke stabilitě $\beta(t)$ v čase t je $y_v^{(t)} \xi_v^{(t)}$.

Nechť v čase $t \geq 1$ produktivního výpočtu je aktualizován nestabilní neuron $v \in V$. Dokážeme, že se tím zvýší stabilita sítě aspoň o 1, tj. $\beta(t) \geq \beta(t-1)+1$. Nejprve uvažujme případ, kdy $y_v^{(t-1)} = 0$, a tedy $\xi_v^{(t-1)} \geq h_v$, resp. $\xi_v^{(t-1)} \geq 1$ díky normálnímu tvaru N. Příspěvek neuronu v ke stabilitě $\beta(t-1)$ v čase $t-1$ je tedy 0. Avšak v čase t je $y_v^{(t)} = 1$ a odpovídající příspěvek ke stabilitě je tedy $\xi_v^{(t)} = \xi_v^{(t-1)}$. Proto $\beta(t) = \beta(t-1) + \xi_v^{(t-1)} \geq \beta(t-1)+1$. Obdobně pro případ, kdy $y_v^{(t-1)} = 1$, a tedy $\xi_v^{(t-1)} < h_v$, resp. $\xi_v^{(t-1)} \leq -1$ díky normálnímu tvaru N. Příspěvek neuronu v ke stabilitě $\beta(t-1)$ v čase $t-1$ je tedy $\xi_v^{(t-1)}$. Avšak v čase t je $y_v^{(t)} = 0$ a odpovídající příspěvek ke stabilitě je tedy 0. Proto $\beta(t) = \beta(t-1) - \xi_v^{(t-1)} \geq \beta(t-1)+1$.

Z definice stability (8.10) je zřejmé, že pro každé $t \geq 1$ je $-|w| \leq \beta(t) \leq |w|$ omezená. Z $\beta(t) \geq \beta(t-1)+1$ pak vyplývá, že produktivní výpočet musí skončit v čase nejvýše $2|w|$, což lze dle poznámky k definici 8.1 shora odhadnout pomocí $O(s^2 w_{max})$. Navíc dle věty 6.30 platí $w_{max} \leq (s+1)^{\frac{s+1}{2}}/2^s$ a z toho vyplývá $O(s^2 w_{max}) = O(2^s)$, což také odpovídá počtu všech různých stavů sítě N. □

Ve větě 8.14 lze předpoklad jednoduchosti sítě zeslabit a předpoklad o normálním tvaru sítě lze odstranit úplně za cenu lineárního nárůstu času.

Důsledek 8.15 *Každý produktivní sekvenční výpočet semijednoduché Hopfieldovy sítě velikosti s, s váhou $|w|$ a maximální váhou w_{max}, skončí v čase $20|w| = O(s^2 w_{max}) = O(2^s)$.*

Důkaz: Nejprve převedeme Hopfieldovu síť do normálního tvaru. Podle lemmy 6.29 lze reprezentaci každého neuronu sítě nahradit rozhodující reprezentací dvojnásobné váhy, a tedy vzniklá síť $N = (V, X, Y, A, w, h)$ má váhu $2|w|$. Podobně jako v lemmě 6.7 upravíme síť tak, aby její prahy byly nulové. Za tímto účelem přidáme k síti další iniciálně aktivní neuron v s velkou zpětnou vazbou, která fixuje jeho aktivitu. Neuron v spojíme se všemi neurony $j \in V$ pomocí váhy $w_{jv} = -h_j$, která odpovídá příslušnému prahu s opačným znaménkem, a prahy všech neuronů v síti položíme rovny 0. Bez újmy na obecnosti můžeme dále předpokládat, že $\sum_{i\in V} |w_{ji}| \geq |h_j|$ pro každé $j \in V$, protože v opačném případě by stav neuronu j byl konstantní. Z toho vyplývá, že největší možný vliv sítě na neuron v lze shora omezit

$$\sum_{j\in V} |h_j| \leq \sum_{j\in V} \sum_{i\in V} |w_{ji}| \leq 2(2|w|) = 4|w| , \tag{8.13}$$

a tedy stačí volit váhu $w_{vv} = 4|w|$. Výsledná ekvivalentní Hopfieldova síť $N' = (V', X, Y, A', w', h')$ v normálním tvaru má pak váhu $2|w| + 4|w| + 4|w| = 10|w|$.

Dále důkaz probíhá podobně jako důkaz věty 8.14, avšak pro semijednoduchou síť N' musíme nepatrně upravit definici stability (8.10):

$$\beta'(t) = \sum_{j\in V'} w_{jj} \left(y_j^{(t)}\right)^2 + \frac{1}{2} \sum_{j\in V'} \sum_{i\in V'\setminus\{j\}} w_{ji} y_i^{(t)} y_j^{(t)} . \tag{8.14}$$

Opět lze ukázat, že příspěvek neuronu $v \in V'$ ke stabilitě $\beta'(t)$ v čase t je $y_v^{(t)} \xi_v^{(t)}$. Navíc při důkazu $\beta'(t) \geq \beta'(t-1)+1$ pro případ, kdy $y_v^{(t-1)} = 0$ pro nestabilní neuron v, platí, že $\xi_v^{(t)} = \xi_v^{(t-1)} + w_{vv} \geq \xi_v^{(t-1)}$, protože $w_{vv} \geq 0$ díky semijednoduchosti sítě, a proto $\beta'(t) \geq \beta'(t-1) + \xi_v^{(t-1)} \geq \beta'(t-1)+1$.

Z definice stability (8.14) pro N' s váhou $10|w|$ je také zřejmé, že pro každé $t \geq 1$ je $-10|w| \leq \beta'(t) \leq 10|w|$ omezená, a proto každý produktivní výpočet musí skončit v čase nejvýše $20|w| = O(s^2 w_{max}) = O(2^s)$. □

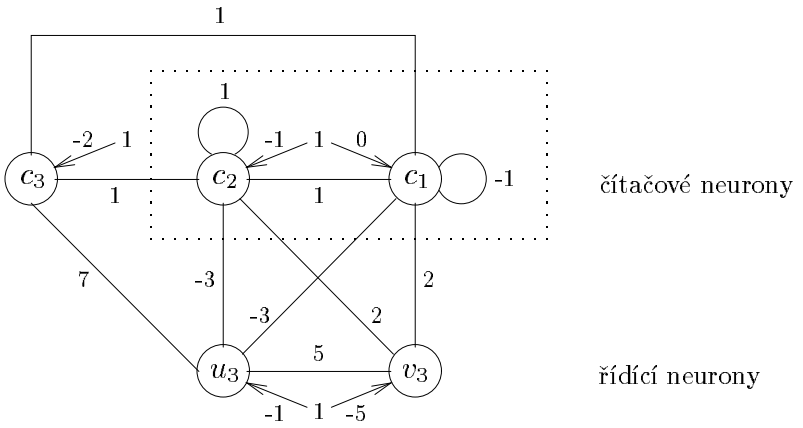
Předpoklady v důsledku 8.15 nelze již dále zeslabit. Příkladem nesemijednoduché symetrické sítě, která se nezastaví v sekvenčním režimu, je např. jeden neuron s nulovým prahem a se zápornou jednotkovou zpětnou vazbou. Naopak např. jednoduchá neuronová síť, která se skládá ze dvou neuronů s jednotkovými prahy, které jsou spojeny jednotkovou váhou, a v níž je právě jeden z těchto neuronů iniciálně aktivní, je příkladem sítě, která se nezastaví v plně paralelním režimu. Avšak pro plně paralelní režim výpočtu lze ukázat analogickou slabší větu, která tvrdí, že Hopfieldova síť (která nemusí být semijednoduchá) se může dostat do cyklu délky nejvýše 2.

Věta 8.16 (Poljak, Sura [43, 228]) *Každý plně paralelní výpočet Hopfieldovy sítě N velikosti s, váhy $|w|$ a maximální váhy w_{max}, se v paralelním čase $40|w|/s = O(s w_{max}) = O(2^s)$ buď zastaví, nebo bude případně střídat dva různé stavy.*

Důkaz: Nechť $N = (V, X, Y, A, w, h)$ je Hopfieldova síť váhy $|w|$ pracující v plně paralelním režimu. Zkonstruujeme symetrickou neuronovou síť $N' = (V', X', Y', A', w', h')$, která v sekvenčním systematickém výpočtu simuluje výpočet N. Architekturu sítě N' tvoří bipartitní graf se dvěma disjunktními množinami vrcholů $V' = V_1 \cup V_2$ ($V_1 \cap V_2 = \emptyset$), které jsou kopiemi V, tj. $V_1 = \{(j,1) \mid j \in V\}$ a $V_2 = \{(j,2) \mid j \in V\}$. Množiny vstupů $X' = \{(j,1) \mid j \in X\}$ a iniciálně aktivních neuronů $A' = \{(j,1) \mid j \in A\}$ odpovídají příslušným neuronům ve V_1, zatímco z hlediska konvergence sítě nás výstupy nezajímají, proto $Y' = \emptyset$. Propojení neuronů mezi V_1 a V_2 je stejné jako propojení v rámci V a také prahy odpovídají:

$$\begin{aligned} w'((i,1),(j,2)) = w'((j,2),(i,1)) &= w(i,j) \quad j,i \in V \\ h'((j,1)) = h'((j,2)) &= h(j) \quad j \in V . \end{aligned} \tag{8.15}$$

přidávají čítačové neurony c_k řádu k $(3 \leq k \leq p)$, které reprezentují odpovídající bity čítače. Ke každému čítačovému neuronu c_k přísluší dva řídící neurony u_k, v_k řádu k $(3 \leq k \leq p)$, které slouží k inicializaci (čítačových i řídících) neuronů nižších řádů ve chvíli, kdy se aktivuje neuron c_k. To znamená, že $V = \{c_1, c_2\} \cup \{c_k, u_k, v_k \mid 3 \leq k \leq p\}$ a velikost čítače bude tedy $3p - 4 = O(p)$. Navíc položme $X = Y = A = \emptyset$.



Obr. 8.2: Symetrický 3-bitový čítač.

Každý čítačový neuron c_k $(3 \leq k \leq p)$ je spojen (symetrickými spoji) se všemi neurony nižšího řádu pomocí jednotkové váhy, tj. $w(c_k, c_j) = 1$ pro $j = 1, \ldots, k-1$ a $w(c_k, u_j) = w(c_k, v_j) = 1$ pro $j = 3, \ldots, k-1$, a jeho práh odpovídá počtu těchto spojů, tj. $h(c_k) = 3(k-1) - 4 = 3k - 7$. Také řídící neuron u_k $(3 \leq k \leq p)$ je spojen se všemi neurony nižšího řádu pomocí záporné váhy s dostatečně velkou absolutní hodnotou, která u každého takového neuronu vyruší všechny kladné váhy jejich vzájemných spojů, včetně spojů s čítačovým neuronem c_k, tj. např.

$$w(u_k, c_j) = -\sum_{i=1}^{k} |w(c_j, c_i)| - \sum_{i=3}^{k-1} |w(c_j, u_i)| - \sum_{i=3}^{k-1} |w(c_j, v_i)| \qquad (8.17)$$

pro $j = 1, \ldots, k-1$ a

$$w(u_k, u_j) \;=\; -\sum_{i=1}^{k} |w(u_j, c_i)| - \sum_{i=3}^{k-1} |w(u_j, u_i)| - \sum_{i=3}^{k-1} |w(u_j, v_i)| \qquad (8.18)$$

Symetrická síť N' má váhu $2|w|$, protože každá váha v N odpovídá nejvýše dvěma stejným váhám v N'.

Hopfieldova síť N' simuluje síť N následujícím způsobem. Na začátku jsou nastaveny vstupní a iniciálně aktivní neurony z V_1. V průběhu výpočtu jsou nejprve systematicky aktualizovány neurony z V_2 a pak neurony z V_1 a potom opět neurony z V_2 atd. Během sekvenčního výpočtu se tedy střídá aktualizace neuronů z V_1 a V_2. Z definice N' je zřejmé, že

$$\mathbf{y}_V^{(t)} = \begin{cases} \mathbf{y}_{V_1}^{(st)} & t \text{ sudé} \\ \mathbf{y}_{V_2}^{(st)} & t \text{ liché} \end{cases} \qquad t \geq 0\,. \qquad (8.16)$$

Navíc N' je jednoduchá, protože případné nenulové zpětné vazby $w(j,j) \neq 0$ $(j \in V)$ v N odpovídají v N' spojům s váhami $w'((j,1),(j,2)) = w(j,j)$. Tedy dle důsledku 8.15 popsaná sekvenční simulace sítě N (přesněji její produktivní verze) se zastaví v čase $st^\star \leq 20(2|w|) = 40|w|$. Kvůli jednoduchosti zápisu nechť např. $t^\star$ je sudé. Potom dle (8.16) platí $\mathbf{y}_V^{(t^\star)} = \mathbf{y}_{V_1}^{(st^\star)} = \mathbf{y}_V^{(t^\star+2t)}$ a $\mathbf{y}_V^{(t^\star+1)} = \mathbf{y}_{V_2}^{(s(t^\star+1))} = \mathbf{y}_V^{(t^\star+2t+1)}$ pro paralelní čas $t \geq 0$. To znamená, že Hopfieldova síť N v plně paralelním režimu v čase $t^\star = 40|w|/s = O(sw_{max})$ vstoupí do cyklu délky 2, nebo se popř. zastaví, pokud $\mathbf{y}_{V_1}^{(st^\star)} = \mathbf{y}_{V_2}^{(st^\star)}$. □

Ještě zformulujeme bez důkazu postačující podmínku, kdy Hopfieldova síť konverguje v plně paralelním režimu ke stabilnímu stavu.

Věta 8.17 (Goles-Chacc, Fogelman-Soulié, Pellegrin [82]) *Nechť $N = (V, X, Y, A, w, h)$ je Hopfieldova síť velikosti s, s jednotkovými váhami, tj. pro každé $j, i \in V$ je $w_{ji} \in \{-1, 0, 1\}$, jejíž stabilita (viz (8.10)) je pozitivně definitní kvadratická forma, tj. pro každý nenulový stav $\mathbf{y}$ sítě N je $\beta(\mathbf{y}) > 0$. Potom každý výpočet Hopfieldovy sítě N v plně paralelním režimu konverguje ke stabilnímu stavu.*

Věty 8.14, 8.16 a důsledek 8.15 poskytují exponenciální horní odhad $O(2^s)$ času výpočtu Hopfieldovy sítě velikosti s v sekvenčním nebo paralelním režimu, který odpovídá počtu všech různých stavů této sítě. Následující věta ukazuje pro plně paralelní režim výpočtu Hopfieldovy sítě, že v nejhorším případě nelze tento čas zlepšit.

Věta 8.18 (Goles-Chacc, Olivos [84, 83]) *Pro každé přirozené číslo $p \in \mathbb{N}$ existuje Hopfieldova síť velikosti $O(p)$ a maximální váhy $2^{O(p)}$, která se zastaví v plně paralelním režimu v čase $\Omega(2^p)$.*

Důkaz: Idea důkazu spočívá v konstrukci p-bitového čítače realizovaného pomocí symetrické neuronové sítě $N = (V, X, Y, A, w, h)$, který postupně počítá od 0 do 2^p. Na obrázku 8.2 je ve vyznačeném rámečku načrtnut základní 2-bitový čítač s čítačovými neurony c_1 a c_2. K němu se postupně

konstrukcí čítače, který navíc počítá ve všech možných pořadích aktualizace neuronů. Uvedené výsledky o exponenciálním času výpočtu Hopfieldovy sítě uvažují nejhorší případ. Avšak v praxi symetrická neuronová síť obvykle konverguje velmi rychle ke stabilnímu stavu. Dá se ukázat [161], že pokud Hopfieldova síť velikosti s vznikla adaptací podle Hebbova zákona (3.26) pomocí $(s/4)\ln s$ náhodných vzorů, pak tyto vzory jsou s velkou pravděpodobností obklopeny oblastmi atrakce (viz odstavec 3.2.2), které zahrnují i stavy sítě lišící se od těchto vzorů více než v $0.024s$ stavech neuronů. V uvedených oblastech atrakce pak konverguje Hopfieldova síť v čase $O(\log\log s)$ produktivního sekvenčního výpočtu ke stabilnímu stavu.

V závěru této podkapitoly porovnáme ještě výpočetní sílu Hopfieldových sítí a obecných cyklických neuronových sítí. Je zřejmé, že Hopfieldovy sítě jsou obecně díky odlišným konvergenčním vlastnostem slabším výpočetním modelem. Proto se omezíme na neuronové sítě, které se zastaví pro každý vstup. Nejprve uvažujeme nejjednodušší příklad takového modelu, kterým jsou acyklické neuronové sítě, tj. prahové obvody.

Věta 8.19 (Parberry [212], Wiedermann [289]) *Pro každý prahový obvod C velikosti s, hloubky d a maximální váhy w_{max}, který má n vstupů, existuje ekvivalentní Hopfieldova síť velikosti $s+n$ a maximální váhy $(2w_{max}s)^d+1$, která počítá stejnou funkci v plně paralelním režimu v čase d.*

Důkaz: Idea důkazu spočívá v tom, že u prahového obvodu C, který považujeme za speciální případ neuronové sítě (vstupy obvodu formálně chápeme jako vstupní neurony sítě, které podle věty 8.6 fixujeme), každou orientovanou hranu nahradíme neorientovanou (symetrickou) hranou. Navíc váhy na cestách směrem od vstupů k výstupům upravíme tak, aby šíření informace ve vzniklé Hopfieldově síti probíhalo stejně jako v původním prahovém obvodu C jen jedním směrem.

Za tímto účelem nejprve podle lemmy 6.29 upravíme váhy v obvodu tak, aby reprezentace booleovských prahových funkcí u všech hradel byly rozhodující. Maximální váha obvodu je pak $2w_{max}$. Dále váhy a prahy neuronů v j-té vrstvě obvodu vynásobíme $(2w_{max}s)^{d-j} > 0$ pro $j = 1,\ldots,d$. Podle věty 6.6 tak nezměníme funkci hradel. Navíc pro reprezentaci $(w_1,\ldots,w_k;h)$ booleovské prahové funkce u hradla v j-té vrstvě $(1 \le j < d)$, které má $k \le s$ vstupů, platí

$$\left|\sum_{i=1}^{k} w_i x_i - h\right| \ge (2w_{max}s)^{d-j} \tag{8.22}$$

pro každý vstup $(x_1,\ldots,x_k) \in \{0,1\}^k$. Na druhou stranu součet absolutních hodnot vah spojů, které v obvodu vychází z tohoto hradla, je menší než $2w_{max}s(2w_{max}s)^{d-j-1} = (2w_{max}s)^{d-j}$. Nyní nahradíme orientované hrany

$$w(u_k,v_j) = -\sum_{i=1}^{k}|w(v_j,c_i)| - \sum_{i=3}^{k-1}|w(v_j,u_i)| - \sum_{i=3}^{k-1}|w(v_j,v_i)| \tag{8.19}$$

pro $j = 3,\ldots,k-1$. Podobně řídící neuron v_k $(3 \le k \le p)$ je spojen se všemi neurony nižšího řádu pomocí kladných vah, které jsou o 1 menší než absolutní hodnota vah u odpovídajících spojů s u_k, tj. $w(v_k,c_j) = |w(u_k,c_j)| - 1$ pro $j = 1,\ldots,k-1$, $w(v_k,u_j) = |w(u_k,u_j)| - 1$ a $w(v_k,v_j) = |w(u_k,v_j)| - 1$ pro $j = 3,\ldots,k-1$. Navíc čítačový neuron c_k je spojen s řídícím neuronem u_k tak, aby aktivace c_k způsobila aktivaci u_k, tj.

$$w(c_k,u_k) = 1 + \sum_{i=1}^{k-1}|w(u_k,c_i)| + \sum_{i=3}^{k-1}|w(u_k,u_i)| + \sum_{i=3}^{k-1}|w(u_k,v_i)| \tag{8.20}$$

a $h(u_k) = 1$. Podobně řídící neuron u_k je spojen s řídícím neuronem v_k tak, aby aktivace u_k způsobila aktivaci v_k, ale na druhou stranu, aby neurony nižšího řádu neovlivnily stav neuronu v_k, tj. např.

$$\begin{aligned} h(v_k) &= w(u_k,v_k) = \\ &= 1 + \sum_{i=1}^{k-1}|w(v_k,c_i)| + \sum_{i=3}^{k-1}|w(v_k,u_i)| + \sum_{i=3}^{k-1}|w(v_k,v_i)| \,. \end{aligned} \tag{8.21}$$

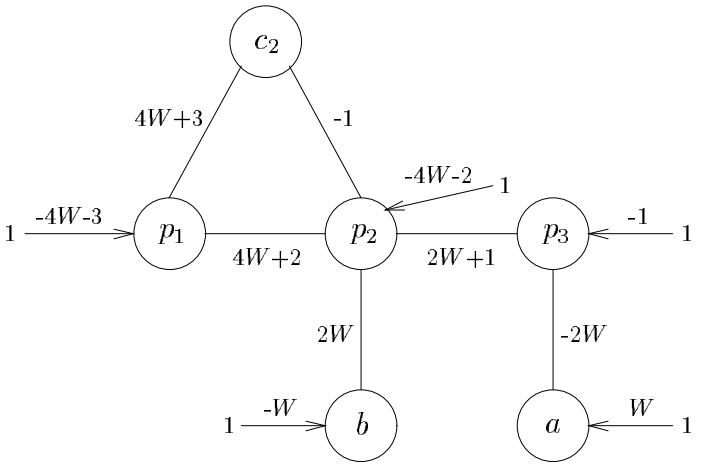
Nyní je lehké nahlédnout, že základní 2-bitový čítač postupně počítá 3 kroky od 00 do 11, tj. čas jeho výpočtu je $T(2) = 3$. V obecném případě, když jsou aktivovány všechny (čítačové i řídící) neurony řádu menšího než k, tj. proběhl výpočet od 0 do 2^{k-1}, dochází k aktivaci čítačového neuronu c_k, který v dalším kroku aktivuje řídící neuron u_k. Ten v příštím kroku způsobí pasivitu všech neuronů řádu nižšího než k a aktivuje řídící neuron v_k, který kompenzuje vliv u_k na neurony nižších řádů a umožní tak (po přerušení trvajícím 3 kroky) jejich opětovný výpočet od 0 do 2^{k-1}. To znamená, že čas výpočtu čítače řádu k je $T(k) = 2T(k-1)+3$. Nakonec v p-bitovém čítači budou aktivní všechny neurony a Hopfieldova síť N se nachází ve stabilním stavu. Indukcí lze ukázat že celková doba výpočtu N je $2^p+2^{p-1}-3 = \Omega(2^p)$.

Váha základního 2-bitového čítače je $W(2) = 3$. Pro $k \ge 3$ součet absolutních hodnot vah (8.17), (8.18), (8.19) spojů neuronu u_k (podobně v_k) s neurony nižšího řádu odpovídá řádově $O(W(k-1))$, kde $W(k-1)$ je váha $(k-1)$-bitového čítače. To znamená, že také $w(c_k,u_k) = O(W(k-1))$ podle (8.20) a $w(u_k,v_k) = O(W(k-1))$ podle (8.21). Tedy váha k-bitového čítače je $W(k) = O(W(k-1))$ a výsledná váha N je $W(p) = 2^{O(p)}$. Zřejmě maximální váha Hopfieldovy sítě N je $2^{O(p)}$. □

Exponenciální dolní odhad $\Omega(2^p)$ času výpočtu Hopfieldovy sítě velikosti $O(p)$ v sekvenčním režimu [92] lze ukázat podobně jako v důkazu věty 8.18

Každý další krok výpočtu N bude simulován pomocí 6 kroků N_H (oscilátor generuje posloupnost stavů 110011). Z uvedeného vyplývá požadovaný čas simulace.

Pomocí oscilátoru vytvoříme hodiny, které mají dva synchronizační neurony a, b pro řízení výpočtu N_H. Neuron a během výpočtu generuje posloupnost stavů $01^80(111110)^{s-1}$ a neuron b sekvenci $0^810(000010)^{s-1}$. K převodu signálů oscilátoru na stavy neuronů a, b slouží symetrická podsíť N_T načrtnutá na obrázku 8.3, kde parametr W je např. třikrát větší než celková váha N. Navíc váhy a prahy čítače je potřeba uzpůsobit tak (např. vynásobit $6W$), aby neurony N_T díky symetrickým spojům nemohly zpětně ovlivnit jeho výpočet.



Obr. 8.3: Synchronizační hodiny N_T.

Na začátku jsou všechny neurony N_T pasivní. V prvním kroku je aktivován neuron a, který zůstává aktivní, dokud pomocný neuron p_3 není aktivován. Ve druhém kroku je aktivován oscilátor c_2, který v dalším kroku způsobí aktivaci pomocného neuronu p_1. Po krátké dočasné stabilitě N_T proběhne další aktualizace až v šestém kroku, kdy je oscilátor pasivní, což v sedmém kroku umožní předání aktivace z pomocného neuronu p_1 na neuron p_2. To způsobí aktivaci synchronizačního neuronu b v osmém kroku a další předání aktivity z neuronu p_2 na neuron p_3. Pasivita p_2 a aktivita p_3 implikují pasivitu obou synchronizačních neuronů a, b v devátém kroku včetně pasivity všech pomocných neuronů p_1, p_2, p_3. Tím se podsíť N_T dostane opět do počátečního stavu a její výpočet se opakuje s tím, že vzhledem k jinému stavu

obvodu neorientovanými spoji Hopfieldovy sítě. Ze vztahu (8.22) pak vyplývá, že spoje ve vzniklé Hopfieldově síti, které původně vystupovaly z prahových hradel, nemohou díky malým váhám ovlivnit booleovskou prahovou funkci odpovídajících neuronů, která závisí jen na spojích, které původně vstupovaly do těchto hradel. Nakonec fixujeme vstupní neurony pomocí zpětné vazby s váhou a prahem velikosti $(2w_{max}s)^d + 1$, což je maximální váha této sítě. Vlastní výpočet Hopfieldovy sítě potom v d krocích simuluje po vrstvách výpočet původního prahového obvodu C. □

Důsledek 8.20 *Hopfieldovy sítě mají stejnou výpočetní sílu jako obecné cyklické neuronové sítě, které se zastaví pro každý vstup.*

Důkaz: Hopfieldovy sítě jsou speciálním případem obecných cyklických sítí. Na druhou stranu podle věty 8.7 ke každé cyklické síti, která se zastaví na každý vstup, existuje ekvivalentní prahový obvod a podle věty 8.19 k němu dále existuje ekvivalentní Hopfieldova síť. □

Konstrukce Hopfieldovy sítě, která je ekvivalentní s obecnou konvergentní cyklickou neuronovou sítí, uvedená v důkazu důsledku 8.20, využívá přechod k ekvivalentnímu prahovému obvodu, který nemusí být efektivní. Např. velikost Hopfieldovy sítě závisí na délce výpočtu původní obecné neuronové sítě, která může být dle věty 8.18 (platí tím spíš pro obecné sítě) exponenciální. Proto v závěru této podkapitoly ukážeme analogickou přímou konstrukci, která zohledňuje deskriptivní složitost výsledné Hopfieldovy sítě.

Věta 8.21 (Orponen [209]) *Pro každou cyklickou neuronovou síť N velikosti s, která se v plně paralelním režimu zastaví pro každý vstup, existuje $(6t+3)$-ekvivalentní Hopfieldova síť N_H velikosti $O(s^2)$ pracující v plně paralelním režimu.*

Důkaz: Konstrukce ekvivalentní Hopfieldovy sítě N_H je založena na realizaci orientovaných spojů pomocí symetrických podsítí [94], jejichž činnost je synchronizována hodinami, které využívají Hopfieldova čítače z důkazu věty 8.18. Nejprve tedy upravíme tento čítač pro naše potřeby.

Vzhledem k tomu, že daná obecná cyklická neuronová síť N vždy konverguje ke stabilnímu stavu, lze délku jejího výpočtu omezit počtem všech jejích možných stavů $2^s - 1$ (vyjma počátečního), protože jinak by se zacyklila. Ke zpomalení jejího výpočtu budeme využívat čítačový neuron c_2 řádu 2 (viz důkaz věty 8.18) jako oscilátor, a proto k realizaci hodin pro celý její výpočet potřebujeme $(s+1)$-bitový čítač skládající se z $O(s)$ neuronů, který počítá od 0 do 2^{s+1}. Je snadné nahlédnout, že oscilátor c_2 v průběhu výpočtu tohoto čítače generuje své stavy $0011110011(110011)^{s-1}$, kde prvních 9 kroků N_H (oscilátor generuje posloupnost stavů 0011110011 včetně počátečního stavu) slouží k realizaci prvního kroku N a k počáteční inicializaci hodin.

8.4 Stabilní stavy Hopfieldovy sítě

Podobně jako u obecných neuronových sítí (viz větu 8.9) se budeme zabývat problémem existence stabilních stavů v Hopfieldově síti. Při využití Hopfieldovy sítě jako asociativní paměti odpovídají některé stabilní stavy sítě naučeným tréninkovým vzorům (viz odstavec 3.2.2), a proto je otázka počtu stabilních stavů důležitá. Věty 8.14, 8.16 a 8.17 formulují velmi obecné předpoklady, za kterých má symetrická síť aspoň jeden stabilní stav. Analogie věty 8.9 pro nesemijednoduché Hopfieldovy sítě ukazuje, že problém existence stabilního stavu v obecné symetrické síti je NP-úplný [64]. Také otázka počtu stabilních stavů v Hopfieldově síti je obecně těžký problém i pro síť s malými váhami.

Problém dvou stabilních stavů Hopfieldovy sítě SHN (Two **S**table Configurations Problem in **H**opfield **N**etworks):
instance: Hopfieldova síť N s malými váhami.
otázka: Existují aspoň dva různé stabilní stavy N?

Věta 8.22 (Lipscomb [77]) *SHN je NP-úplný problém.*

Důkaz: Díky poznámce k definici 8.12 můžeme v tomto důkazu u Hopfieldovy sítě předpokládat bipolární stavy neuronů z množiny $\{-1, 1\}$. Zřejmě $SHN \in NP$, protože nedeterministický algoritmus v polynomiálním čase uhádne a ověří dva stabilní stavy zadané Hopfieldovy sítě. Při důkazu, že SHN je NP-těžký problém, redukujeme v polynomiálním čase NP-úplný *problém nezávislé množiny grafu* $INDSET$ [146] na problém SHN.

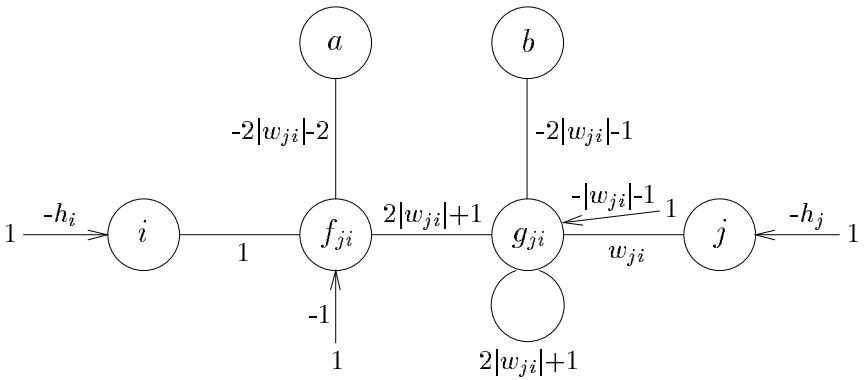
Problém nezávislé množiny grafu $INDSET$ (Graph **IND**ependent **SET** Problem):
instance: Neorientovaný graf $G = (V, E)$; přirozené číslo $k \in \mathbb{N}$.
otázka: Obsahuje G nezávislou množinu velikosti k, tj. množinu k vrcholů takovou, že žádné dva z nich nejsou spojené hranou?

Nechť tedy $(G = (V, E); k)$ je instance $INDSET$. V polynomiálním čase zkonstruujeme odpovídající instanci $N_G = (V', X, Y, A, w, h)$ problému SHN takovou, že G obsahuje nezávislou množinu velikosti k, právě když N_G má aspoň dva různé stabilní stavy. Neurony $V' = V \cup V_2$ odpovídají vrcholům V a jejich neuspořádaným dvojicím $V_2 = \{\{i, j\} \mid i \neq j;\ i, j \in V\}$. Dále formálně položme $X = Y = A = \emptyset$. Nechť neorientovaný graf G má $p = |V|$ vrcholů a označme $d_G(j) = |\{i \in V \mid \{i, j\} \in E\}|$ počet hran spojených s vrcholem $j \in V$ (tj. stupeň vrcholu). Volme celá čísla $\alpha, \beta, \gamma, \delta \in \mathbb{Z}$ tak, aby

$$0 < \quad \alpha \quad < 2\beta \tag{8.23}$$

oscilátoru bude dočasná stabilita N_T během dalšího výpočtu kratší a celý cyklus hodin bude dále trvat jen 6 kroků.

Nyní se již budeme zabývat vlastní simulací výpočtu neuronové sítě N. Každou orientovanou hranu (i, j) sítě N s váhou w_{ji} nahradíme v N_H cestou neuronů i, f_{ji}, g_{ji}, j, kde neuron f_{ji} je navíc spojen se synchronizačním neuronem a u hodin N_T a neuron g_{ji} se zpětnou vazbou je spojen s neuronem b. Váhy a prahy těchto spojů jsou vyznačeny na obrázku 8.4. Díky volbě W neurony f_{ji}, g_{ji} nemají na synchronizační neurony a, b vliv.



Obr. 8.4: Symetrická implementace orientovaného spoje s váhou w_{ji}.

Simulace přenosu signálu z neuronu i do neuronu j trvá v obecném případě 6 kroků (na začátku díky inicializaci hodin 9 kroků) výpočtu N_H. Na začátku tohoto cyklu jsou neurony a, b, f_{ji}, g_{ji} pasivní. V prvním kroku dojde k přesunu stavu y_i neuronu i do neuronu f_{ji} a také se aktivizuje neuron a. V dalším kroku se přesune stav y_i z neuronu f_{ji} do neuronu g_{ji} a neuron f_{ji} je díky aktivitě a pasivní. Další dva kroky se stavy synchronizačních neuronů a, b kvůli dočasné stabilitě N_T nemění (tj. a je aktivní a b je pasivní), neuron f_{ji} je pasivní a neuron g_{ji} díky zpětné vazbě uchovává stav y_i. V pátém kroku cyklu je aktivován neuron b, který v posledním šestém kroku cyklu umožní aktualizaci neuronu j pomocí vstupu y_i (původně z neuronu i) s váhou w_{ji}. Zároveň jsou neurony a, b, f_{ji}, g_{ji} pasivní a celý cyklus se opakuje pro přesun nového signálu od neuronu i k neuronu j.

Ke každé hraně (i, j) sítě N, kterých je nejvýše $O(s^2)$, budou v Hopfieldově síti N_H navíc 2 neurony f_{ji}, g_{ji}, tedy výsledná velikost N_H bude včetně hodin s čítačem nejvýše $O(s^2)$. □

$j \in V'$. Z uvedené konstrukce N_G vyplývá, že

$$d_V(j) = \begin{cases} d_G(j) & j \in V \\ 2 & j \in V_2 \end{cases} \qquad (8.27)$$
$$d_{V_2}(j) = \begin{cases} p-1 & j \in V \\ 0 & j \in V_2 . \end{cases}$$

Ukážeme, že stavy $y_j^S = -1$ pro $j \in V$ a $y_j^S = 1$ pro $j \in V_2$ tvoří *triviální* stabilní stav $\mathbf{y}_{V'}^S$ Hopfieldovy sítě N_G. Nejprve uvažujme neuron $j \in V$ a jeho stav $y_j^S = -1$, který je stabilní, právě když vnitřní potenciál $\xi_j^S < h(j)$ je menší než práh, tj. podle (8.26) a (8.27) to znamená, že $d_G(j)(-\beta)(-1)+(p-1)(-\delta)1 < -\alpha+\beta d_G(j)+\delta(p-1)$. Po úpravě získáme nerovnost $\alpha < 2(p-1)\delta$, pro jejíž platnost podle (8.25) stačí, aby $2(p-k)\delta+1 \le 2(p-1)\delta$, což je ekvivalentní s $1 \le 2(k-1)\delta$ a to pro $k \ge 2$ platí, protože $1 \le 2\delta$ podle (8.24). Na druhou stranu stav $y_j^S = 1$ neuronu $j \in V_2$ je stabilní, právě když vnitřní potenciál $\xi_j^S \ge h(j)$ je větší nebo roven prahu, tj. podle (8.26) a (8.27) to znamená, že $2(-\delta)(-1) \ge -\gamma$, což platí, protože podle (8.24) je $2\delta > 0$ a $-\gamma < 0$.

Dále nahlédneme, že stabilní neuron $j \in V_2$ je pasivní, právě když jsou jeho sousední neurony $i_1, i_2 \in V$ aktivní. Pro pasivní neurony i_1, i_2 je podle (8.24) vnitřní potenciál $\xi_j = 2(-\delta)(-1) > -\gamma$, což implikuje nestabilitu pasivního stavu neuronu j. Podobně, pokud je jeden z neuronů i_1, i_2 aktivní a druhý pasivní, pak podle (8.24) je vnitřní potenciál $\xi_j = (-\delta)(-1)+(-\delta)1 = 0 > -\gamma$ a pasivní neuron j je nestabilní. Konečně pro aktivní neurony i_1, i_2 je podle (8.24) vnitřní potenciál $\xi_j = 2(-\delta)1 < -\gamma$, což znamená stabilitu pasivního neuronu j.

Nyní již předpokládejme nejprve, že N_G má aspoň dva různé stabilní stavy. To znamená že existuje stabilní stav $\mathbf{y}_{V'}^\star \ne \mathbf{y}_{V'}^S$ Hopfieldovy sítě N_G, který se liší od triviálního stabilního stavu, tj. $y_j^\star = 1$ pro $j \in V$ nebo $y_j^\star = -1$ pro $j \in V_2$. Určitě existuje aktivní neuron $v \in V$ (tj. $y_v^\star = 1$), protože v opačném případě $y_j^\star = -1$ pro nějaký neuron $j \in V_2$ a ten je stabilní, právě když jeho sousední neurony $i \in V$ jsou aktivní. Pro tento neuron v označme $a = |\{i \in V_2 \,|\, w(i,v) \ne 0;\ y_i^\star = 1\}|$ a $b = |\{i \in V \,|\, w(i,v) \ne 0;\ y_i^\star = 1\}|$ počty s ním spojených aktivních neuronů po řadě z V_2 a z V. Víme, že vnitřní potenciál $\xi_v^\star \ge h(v)$ neuronu v je větší nebo roven jeho prahu, protože je aktivní, což lze psát jako

$$b(-\beta)1 + (d_G(v)-b)(-\beta)(-1) + a(-\delta)1 + (p-1-a)(-\delta)(-1) \ge$$
$$\ge -\alpha + \beta d_G(v) + \delta(p-1) . \qquad (8.28)$$

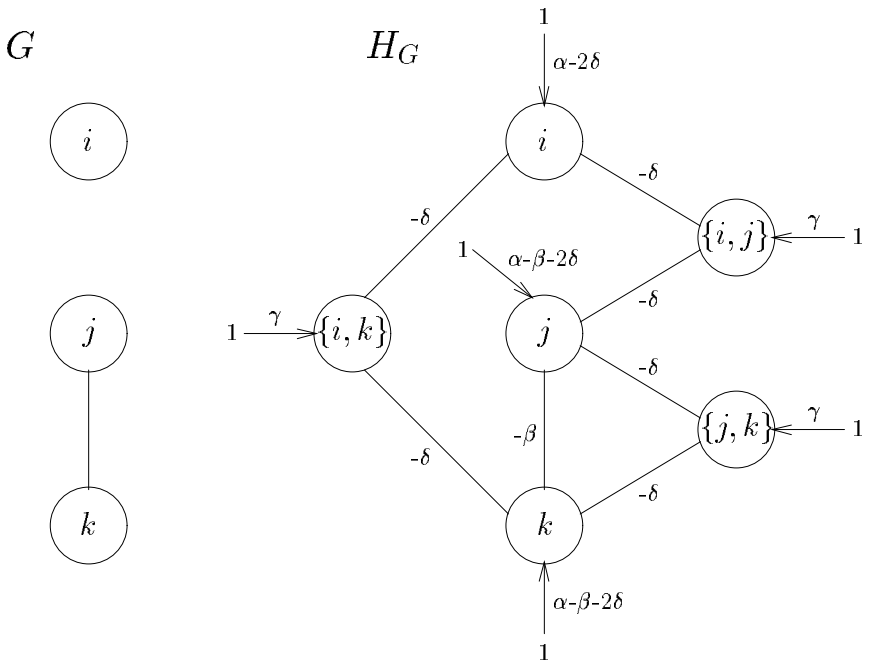
Po úpravě nerovnosti (8.28) obdržíme $\alpha \ge 2a\delta + 2b\beta$. Ze vztahu (8.24) je $2a\delta \ge 0$, a proto podle (8.23) z předchozího vyplývá, že $b = 0$. Tedy

$$0 < \gamma < 2\delta \qquad (8.24)$$
$$2(p-k)\delta \le \alpha < 2(p-k)\delta + 1 , \qquad (8.25)$$

tj. např. $\alpha = 2(p-k)$, $\beta = p-k+1$, $\gamma = \delta = 1$. Potom definujeme následujícím způsobem malé váhy a prahy Hopfieldovy sítě N_G:

$$w(i,j) = w(j,i) = \begin{cases} -\beta & \{i,j\} \in E;\ i,j \in V \\ -\delta & j = \{i,u\} \in V_2;\ i,u \in V \\ 0 & \text{jinak} \end{cases} \qquad (8.26)$$
$$h(j) = \begin{cases} -\alpha + \beta d_G(j) + \delta(p-1) & j \in V \\ -\gamma & j \in V_2 . \end{cases}$$

Na obrázku 8.5 je příklad Hopfieldovy sítě N_G odpovídající neorientovanému grafu G skládajícímu se z jedné hrany a jednoho izolovaného vrcholu.



Obr. 8.5: Příklad Hopfieldovy sítě H_G odpovídající grafu G.

Označme $d_V(j) = |\{i \in V \,|\, w(i,j) \ne 0\}|$ a $d_{V_2}(j) = |\{i \in V_2 \,|\, w(i,j) \ne 0\}|$ počet neuronů po řadě z V a z V_2, které jsou v síti N_G spojeny s neuronem

dostáváme $\alpha \geq 2a\delta$, což podle (8.25) implikuje $a \leq p-k$, protože α je celé číslo. Potom počet stabilních pasivních neuronů z V_2 spojených s neuronem $v \in V$ lze zdola odhadnout $p-1-a \geq p-1-(p-k) = k-1$. Ty mají aspoň $k-1$ aktivních sousedních neuronů z V, což spolu s aktivním neuronem $v \in V$ činí aspoň k aktivních neuronů z V. Označme $I \subseteq V$ množinu těchto aktivních neuronů, tj. $|I| \geq k$. Tvrdíme, že I je nezávislá množina vrcholů grafu G. Nechť naopak existují dva vrcholy $i,j \in I$, které jsou v G spojené hranou, tj. $\{i,j\} \in E$. Víme, že odpovídající neurony i,j v N_G jsou aktivní, tj. $y_j^\star = y_i^\star = 1$, a stabilní, což např. pro neuron j znamená, že jeho vnitřní potenciál $\xi_j^\star \geq h(j)$. Vnitřní potenciál kromě příspěvků od pasivních neuronů z V_2 a V obsahuje aspoň jeden příspěvek od aktivního neuronu $i \in I \subseteq V$, tedy po dosazení

$$\begin{aligned}&(p-1)(-\delta)(-1) + (d_G(j)-1)(-\beta)(-1)) + (-\beta)1 \geq \\ &\geq -\alpha + \beta d_G(j) + \delta(p-1)\,. \end{aligned} \tag{8.29}$$

Po úpravě (8.29) dostáváme $\alpha \geq 2\beta$, což je ve sporu s (8.23). Tedy I je nezávislá množina grafu G velikosti aspoň k.

Nechť naopak graf G obsahuje nezávislou množinu velikosti aspoň k a označme $I \subseteq V$ takovou maximální nezávislou množinu G. Definujeme stabilní stav $\mathbf{y}_{V'}^\star \neq \mathbf{y}_{V'}^S$ Hopfieldovy sítě N_G, který se liší od triviálního stabilního stavu $\mathbf{y}_{V'}^S$:

$$y_j^\star = \begin{cases} 1 & j \in I \text{ nebo } j \in V_2 \setminus \{\{i,u\} \mid i,u \in I;\, i \neq u\} \\ -1 & \text{jinak}\,. \end{cases} \tag{8.30}$$

Tedy Hopfieldova síť má aspoň dva stabilní stavy $\mathbf{y}_{V'}^\star$, $\mathbf{y}_{V'}^S$. □

V důkazu věty 8.22 měla Hopfieldova síť N_G nekladné váhy, a proto tvrzení této věty platí i pro třídu symetrických sítí s nekladnými malými váhami. Pro Hopfieldovy sítě s nezápornými váhami je problém *SHN* P-úplný (viz definici 7.23). Také se dá ukázat [64], že problém existence tří stabilních stavů v Hopfieldově síti s nulovými prahy (bez omezení na váhy) je NP-úplný. Přestože určit počet stabilních stavů pro konkrétní danou symetrickou síť je v nejhorším případě těžké, existuje asymptotický odhad $1.05 \times 2^{0.2874s}$ pro počet stabilních stavů jednoduché Hopfieldovy sítě s nulovými prahy, jejíž váhy jsou nezávislé náhodné veličiny distribuované kolem nuly s Gaussovým rozdělením pravděpodobností [190, 269].

Další důležitou otázkou u Hopfieldových asociativních pamětí je velikost oblastí atrakce stabilního stavu (viz odstavec 3.2.2). Velikost těchto oblastí měříme pomocí tzv. *poloměru atrakce.*

Definice 8.23 *Nechť $N = (V, X, Y, A, w, h)$ je Hopfieldova síť.* Hammingova vzdálenost $H(\mathbf{y}_V, \mathbf{y}'_V)$ *dvou stavů $\mathbf{y}_V$, $\mathbf{y}'_V$ sítě N je počet různých stavů*

jednotlivých neuronů, tj.

$$H(\mathbf{y}_V, \mathbf{y}'_V) = \left|\left\{j \in V \mid y_j \neq y'_j\right\}\right|\,. \tag{8.31}$$

Poloměr atrakce $\varrho(\mathbf{y}_V^\star)$ *stabilního stavu $\mathbf{y}_V^\star$ je největší Hammingova vzdálenost taková, že z každého stavu $\mathbf{y}_V$ sítě N, pro který $H(\mathbf{y}_V^\star, \mathbf{y}_V) \leq \varrho(\mathbf{y}_V^\star)$, je vždy (v paralelním, resp. sekvenčním režimu) zaručena konvergence k $\mathbf{y}_V^\star$.*

Zformulujeme bez důkazu větu o složitosti problému určení poloměru atrakce daného stabilního stavu v Hopfieldově síti.

Problém poloměru atrakce Hopfieldovy sítě *ARHN* (**A**ttraction **R**adius Problem in **H**opfield **N**etworks):
instance: Hopfieldova síť N; její stabilní stav $\mathbf{y}^\star$; přirozené číslo $k \in \mathbb{N}$.
otázka: Je poloměr atrakce $\varrho(\mathbf{y}^\star)$ v N menší než k?

Věta 8.24 (Floréen, Orponen [65]) *$ARHN$ je NP-těžký problém.*

Dokonce se dá ukázat [65], že (za předpokladu $P \neq NP$) neexistuje polynomiální algoritmus, který aproximuje poloměr atrakce stabilního stavu Hopfieldovy sítě velikosti s s přesností $s^{1-\varepsilon}$ pro libovolné pevné $0 < \varepsilon \leq 1$.

Hopfieldovy sítě se kromě asociativních pamětí využívají k heuristickému řešení těžkých optimalizačních úloh jako např. problém obchodního cestujícího (viz odstavec 3.3.2), který je NP-úplný. Cílem výpočtu je minimalizace energetické funkce (3.29), která s opačným znaménkem odpovídá stabilitě Hopfieldovy sítě (8.10). Úspěch tohoto úsilí závisí podstatně na volbě počátečního stavu, který by měl ležet v oblasti atrakce stabilního stavu s co nejmenší energií, resp. největší stabilitou. Ukážeme však, že volba takového počátečního stavu je těžký problém.

Problém minimalizace energie Hopfieldovy sítě *MEHN* (**M**inimum **E**nergy Problem in **H**opfield **N**etworks):
instance: Hopfieldova síť N; celé číslo $k \in \mathbb{Z}$.
otázka: Existuje počáteční stav $\mathbf{y}^{(0)}$ sítě N, ze kterého N konverguje ke stabilnímu stavu $\mathbf{y}^{(t^\star)}$ s energií menší než k, tj. se stabilitou $\beta(\mathbf{y}^{(t^\star)}) > k$?

Věta 8.25 (Wiedermann [289]) *$MEHN$ je NP-úplný problém.*

Důkaz: Zřejmě $MEHN \in NP$, protože nedeterministický algoritmus v polynomiálním čase uhádne stabilní stav zadané Hopfieldovy sítě a ověří, zda jeho stabilita je větší než dané k. Důkaz NP-těžkosti $MEHN$ probíhá podobně jako v důkazu věty 8.9 polynomiální redukcí instance f problému SAT na instanci $(N; k)$ problému $MEHN$. K výstupnímu neuronu v příslušného obvodu C_f je připojen neuron c, v tomto případě pomocí dostatečně velké

váhy $w(v,c) = W$ s odpovídajícím prahem $h(c) = W$. Příslušnou Hopfieldovu síť N_f s váhou $|w|$ vytvoříme z tohoto obvodu pomocí věty 8.19. Potom f je splnitelná, právě když existuje počáteční stav sítě N_f, ze kterého síť konverguje ke stabilnímu stavu, v němž jsou oba neurony v a c aktivní, tj. stabilita tohoto stavu je díky volbě např. $w(v,c) = W = k + |w| + 1$ větší než k. □

pro vstup $\mathbf{x}$*, je aspoň* $1-\varepsilon$*, a naopak pro každé* $\mathbf{x} \notin L$ *pravděpodobnost, že výstup obvodu* C^p *je aktivní pro vstup* $\mathbf{x}$*, je nejvýše* ε*.*

Pro pravděpodobnostní analýzu prahových obvodů budeme používat následující lemmu, kterou uvádíme bez důkazu.

Lemma 9.2 *Označme* $B(m,N,p)$ *pravděpodobnost, že z* N *nezávislých náhodných Bernoulliho pokusů je nejméně* m *neúspěšných, když pravděpodobnost neúspěchu v jednom pokusu je* p*.*

(i) (Chernoff [136])
Je-li $m \geq Np$*, pak*

$$B(m,N,p) \leq \left(\frac{Np}{m}\right)^m \left(\frac{N-Np}{N-m}\right)^{N-m} . \tag{9.1}$$

(ii) (Angluin, Valiant [21])
Pro každé $0 \leq \beta \leq 1$ *platí*

$$B(Np(1+\beta),N,p) \leq e^{-\frac{1}{2}\beta^2 Np} . \tag{9.2}$$

(iii) (Hajnal, Maass, Pudlák, Szegedy, Turán [91])
Pro každé přirozené číslo $k \in \mathbb{N}$ *platí*

$$B\left(\frac{k}{2},k,\frac{1}{2}\right) = \begin{cases} \frac{1}{2} & \text{pro } k \text{ liché} \\ \frac{1}{2}+\sqrt{\frac{1}{2\pi k}} & \text{pro } k \text{ sudé}. \end{cases} \tag{9.3}$$

Rozpoznávání jazyka pomocí prahových obvodů s velkou pravděpodobností chyby, např. $\varepsilon = 0.4$, není příliš spolehlivé, avšak opakováním výpočtu takového obvodu lze dosáhnout libovolně malé pravděpodobnosti chyby.

Věta 9.3 *Nechť* $0 < \lambda < \varepsilon < \frac{1}{2}$ *jsou dvě libovolná reálná čísla. Potom každý jazyk* $L = L(C^p,\varepsilon)$*,* ε*-rozpoznávaný pravděpodobnostním prahovým obvodem* C^p *hloubky* d *a velikosti* s*, lze* λ*-rozpoznat pravděpodobnostním prahovým obvodem* $C^{p'}$ *hloubky* $d+1$ *a velikosti*

$$\left\lceil \frac{2\log\lambda}{\log(4\varepsilon(1-\varepsilon))} \right\rceil s + 1 , \tag{9.4}$$

tj. $L = L(C^{p'},\lambda)$*.*

Důkaz: Pravděpodobnostní prahový obvod $C^{p'}$ se bude skládat z N kopií obvodu C^p, které mají stejný deterministický vstup a jejich výstupní hradla budou spojena s výstupním prahovým hradlem obvodu $C^{p'}$, které

Kapitola 9

Pravděpodobnostní neuronové sítě

Je známo, že biologické neurony jsou relativně nespolehlivé výpočetní jednotky, např. ve srovnání s elektrickými prvky klasických obvodů. Přesto náš mozek je schopen celkem spolehlivě vykonávat velmi přesné činnosti. Tato kapitola se snaží naznačit příčiny tohoto jevu studiem (acyklických) *pravděpodobnostních neuronových sítí*. V následujících podkapitolách budeme uvažovat tři různé typy pravděpodobnostního chování.

9.1 Pravděpodobnostní prahové obvody

Nejprve definujeme tzv. *pravděpodobnostní prahové obvody*, které se liší od deterministických prahových obvodů z podkapitoly 7.2 tak, že navíc mají náhodné vstupy, které jsou s danou různou pravděpodobností aktivní. Pravděpodobnostní prahové obvody s n deterministickými vstupy budeme také využívat k rozpoznávání jazyků $L \subseteq \{0,1\}^n$ binárních slov délky n. V tomto případě výstup obvodu s určitou pravděpodobností chyby signalizuje, zda deterministická část vstupu patří do příslušného jazyka.

Definice 9.1 *Řekneme, že obvod* $C^p = (V \cup X^p, X, Y, E, \ell^p)$ *je* pravděpodobnostní prahový obvod*, jestliže každé* prahové *hradlo* $v \in V$ *počítá booleovskou prahovou funkci s reprezentací* $\ell^p(v)$ *a každé tzv.* náhodné vstupní *hradlo* $x^p \in X^p$*, které se nachází ve vstupní vrstvě, je aktivní s pravděpodobností* $0 < \ell^p(x^p) < 1$*. Řekneme, že pravděpodobnostní prahový obvod* C^p *s* $n = |X|$ *deterministickými vstupy a jedním výstupním hradlem* ε-rozpoznává jazyk $L \subseteq \{0,1\}^n$ *a značíme* $L = L(C^p,\varepsilon)$*, kde* $0 < \varepsilon < \frac{1}{2}$*, jestliže pro každé* $\mathbf{x} \in L$ *pravděpodobnost, že výstup obvodu* C^p *je aktivní*

bude podle věty 6.26 počítat konsenzuální funkci $MAJORITY$. Je zřejmé, že výpočet obvodu C^p pro daný deterministický vstup proběhne N-krát a výstupní hradlo obvodu $C^{p\prime}$ bude aktivní, právě když aspoň $N/2$ těchto výpočtů skončí s výstupem 1. Využijeme (i) z lemmy 9.2 tak, že za neúspěšný pokus považujeme chybný výpočet jedné z N kopií obvodu C^p v obvodu $C^{p\prime}$, který nastane s pravděpodobností $p = \varepsilon$. Potom podle (9.1) pravděpodobnost $B(N/2, N, \varepsilon)$, že aspoň $N/2$ těchto výpočtů je chybných, lze shora odhadnout následujícím způsobem:

$$B\left(\frac{N}{2}, N, \varepsilon\right) \leq (4\varepsilon(1-\varepsilon))^{\frac{N}{2}} \ . \tag{9.5}$$

Tedy podle (9.5) volíme N tak, aby $(4\varepsilon(1-\varepsilon))^{N/2} \leq \lambda$, tj. pomocí $0 < \varepsilon < \frac{1}{2}$ dostáváme:

$$N \geq \frac{2\log\lambda}{\log(4\varepsilon(1-\varepsilon))}\,, \tag{9.6}$$

což zajistí to, že pravděpodobnostní prahový obvod $C^{p\prime}$ λ-rozpoznává stejný jazyk $L = L(C^{p\prime}, \lambda)$ jako obvod C^p. Navíc $C^{p\prime}$ má zřejmě hloubku $d+1$ a velikost $Ns+1$. □

Také je možné uvažovat následující obecnější definici rozpoznávání jazyka pomocí pravděpodobnostního prahového obvodu, kde pravděpodobnost správného přijetí a zamítnutí vstupního slova není obecně symetrická.

Definice 9.4 *Řekneme, že pravděpodobnostní prahový obvod C^p s n deterministickými vstupy a jedním výstupním hradlem (α,β)*-separuje jazyk *$L \subseteq \{0,1\}^n$ a značíme $L = L(C^p, \alpha, \beta)$, kde $0 < \alpha < \beta < 1$, jestliže pro každé $\mathbf{x} \in L$ pravděpodobnost, že výstup obvodu C^p je aktivní pro vstup $\mathbf{x}$, je aspoň β, a naopak pro každé $\mathbf{x} \notin L$ pravděpodobnost, že výstup obvodu C^p je aktivní pro vstup $\mathbf{x}$, je nejvýše α.*

Zřejmě pravděpodobnostní prahový obvod ε-rozpoznává jazyk L, právě když ho $(\varepsilon, 1-\varepsilon)$-separuje. Platí i opak, totiž že každý separovaný jazyk lze také rozpoznat za cenu nepatrného zvýšení velikosti a váhy pravděpodobnostního prahového obvodu.

Věta 9.5 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Pro každý jazyk $L = L(C^p, \alpha, \beta) \subseteq \{0,1\}^n$ (α,β)-separovaný $(0 < \alpha < \beta < 1)$ pravděpodobnostním prahovým obvodem C^p velikosti s, váhy $|w|$ a hloubky d existuje pravděpodobnostní prahový obvod $C^{p\prime}$ velikosti $s+2$, váhy $4|w|$ a hloubky d, který $(1-\alpha-p(\beta-\alpha))/(2-\beta-\alpha)$-rozpoznává L pro každé p splňující*

$$\frac{1}{2} < p < \min\left(1, \frac{1}{\alpha+\beta}\right) \ . \tag{9.7}$$

Důkaz: Pravděpodobnostní prahový obvod $C^{p\prime}$ je zkonstruován z obvodu C^p tak, že přidáme dvě nová náhodná vstupní hradla x_1^p, x_2^p, která jsou aktivní s pravděpodobností po řadě $p_1 = p$ a

$$p_2 = \frac{\frac{1}{p} - \alpha - \beta}{2 - \beta - \alpha}\,. \tag{9.8}$$

Zřejmě díky (9.7) platí $0 < p_1, p_2 < 1$. Obě hradla x_1^p, x_2^p připojíme k výstupnímu prahovému hradlu v obvodu C^p tak, aby výstup vzniklého obvodu $C^{p\prime}$ byl aktivní, právě když x_1^p je aktivní a současně v nebo x_2^p je aktivní. Bez újmy na obecnosti předpokládáme $|h(v)| \leq |w|$, protože v opačném případě by hradlo v realizovalo konstantní funkci. Potom můžeme definovat odpovídající váhy nových spojů $w(x_1^p, v) = 2|w|$, $w(x_2^p, v) = |w|$ a práh výstupního hradla v obvodu $C^{p\prime}$ navíc zvýšíme na $h(v) + 2|w|$. Tím zajistíme uvedenou funkci obvodu a $C^{p\prime}$ má požadovanou velikost $s+2$, váhu $4|w|$ a hloubku d.

Nyní tedy výstup obvodu $C^{p\prime}$ je aktivní, právě když x_1^p je aktivní a zároveň výstup C^p nebo x_2^p je aktivní. Tedy pro $\mathbf{x} \in L$ bude díky nezávislosti aktivity x_1^p a podle principu inkluze a exkluze výstup $C^{p\prime}$ aktivní s pravděpodobností aspoň

$$p_1(p_2 + \beta - p_2\beta) = \frac{1 - \beta + p(\beta - \alpha)}{2 - \beta - \alpha}\,. \tag{9.9}$$

Podobně pro $\mathbf{x} \notin L$ bude výstup $C^{p\prime}$ aktivní s pravděpodobností nejvýše

$$p_1(p_2 + \alpha - p_2\alpha) = \frac{1 - \alpha - p(\beta - \alpha)}{2 - \beta - \alpha}\,. \tag{9.10}$$

Položme

$$\varepsilon = \frac{1 - \alpha - p(\beta - \alpha)}{2 - \beta - \alpha}\,. \tag{9.11}$$

Opět díky (9.7) platí $0 < \varepsilon < \frac{1}{2}$, protože z $p > \frac{1}{2}$ plyne $\varepsilon < \frac{1}{2}$ a

$$p < \frac{1}{\alpha + \beta} < \frac{1 - \alpha}{\beta - \alpha} \tag{9.12}$$

implikuje $\varepsilon > 0$. Nakonec podle (9.9) a (9.10) pravděpodobnostní prahový obvod $C^{p\prime}$ $(\varepsilon, 1-\varepsilon)$-separuje L, tj. obvod $C^{p\prime}$ ε-rozpoznává L. □

Dále ukážeme, že pravděpodobnostní výpočet prahového obvodu lze nahradit deterministickým za cenu zvýšení velikosti a hloubky obvodu.

Věta 9.6 (Parberry, Schnitger [215]) *Pro každé $\frac{1}{4} < \varepsilon < \frac{1}{2}$ každý jazyk $L = L(C^p, \varepsilon) \subseteq \{0,1\}^n$ ε-rozpoznávaný pravděpodobnostním prahovým obvodem velikosti s a hloubky d lze rozpoznat prahovým obvodem C velikosti*

9.2 Pravděpodobnostní třídy složitosti

V této podkapitole se budeme zabývat pravděpodobnostní verzí hierarchie TC^0 (viz podkapitolu 7.2.3). Za tímto účelem nejprve definujeme analogicky posloupnosti pravděpodobnostních prahových obvodů, u kterých pravděpodobnost chyby výstupu roste pomalu s počtem vstupů. Takové posloupnosti pak lze využít k rozpoznávání jazyků $L \subseteq \{0,1\}^\star$.

Definice 9.7 *Nechť $\mathbf{C}^p = (C_0^p, C_1^p, C_2^p, \ldots)$ je nekonečná posloupnost pravděpodobnostních prahových obvodů, kde obvod $C_n^p = (V_n \cup X_n^p, X_n, Y_n, E_n, \ell_n^p)$ $(n \geq 0)$ má $|X_n| = n$ deterministických vstupů a jeden výstup, tj. $|Y_n| = 1$. Řekneme, že posloupnost $\mathbf{C}^p$ $\varepsilon(n)$-rozpoznává jazyk $L \subseteq \{0,1\}^\star$, a značíme $L = L(\mathbf{C}^p, \varepsilon(n))$, kde $\varepsilon : \mathbb{N} \longrightarrow (0, \frac{1}{2})$ je reálná posloupnost, jestliže obvod C_n^p $\varepsilon(n)$-rozpoznává jazyk $L \cap \{0,1\}^n$ pro každé $n \geq 0$. Definujeme RTC_k^0 $(k \geq 1)$ třídu jazyků $L \subseteq \{0,1\}^\star$ $\varepsilon(n)$-rozpoznatelných L-uniformní posloupností pravděpodobnostních prahových obvodů (polynomiální velikosti $S(n) = n^{O(1)}$) konstantní hloubky $D(n) = k$ s malými váhami a chybou*

$$\varepsilon(n) = \frac{1}{2} - \frac{1}{n^{O(1)}}\,. \tag{9.17}$$

Nakonec definujeme třídu složitosti

$$RTC^0 = \bigcup_{k \geq 0} RTC_k^0\,. \tag{9.18}$$

Ukážeme, že pravděpodobnostní prahové obvody počítají např. booleovský skalární součin (viz větu 7.51) s menší hloubkou než deterministické prahové obvody.

Věta 9.8 (Hajnal, Maass, Pudlák, Szegedy, Turán [91])

$$IP \in RTC_2^0\,.$$

Důkaz: Konstruujeme pravděpodobnostní prahový obvod C_n^p, který má v nulté vrstvě n náhodných vstupních hradel $x_1^p, \ldots, x_n^p$, které jsou s pravděpodobností $\frac{1}{2}$ aktivní, a $2n$ deterministických vstupů $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ spolu s jejich negacemi $\bar{x}_1, \ldots, \bar{x}_n$, $\bar{y}_1, \ldots, \bar{y}_n$, které lze příp. podle (ii) z lemmy 6.27 odstranit. První vrstva se skládá z $2n$ prahových hradel $u_1, \ldots, u_n$, $v_1, \ldots, v_n$, kde podle věty 6.26 u_i $(1 \leq i \leq n)$ počítá konjunkci $x_i \wedge y_i \wedge x_i^p$ a v_i $(1 \leq i \leq n)$ počítá disjunkci $\bar{x}_i \vee \bar{y}_i \vee x_i^p$. Všechna hradla v první vrstvě jsou připojena pomocí jednotkových vah $w(u_i, v) = w(v_i, v) = 1$ k výstupnímu prahovému hradlu v s prahem $h(v) = n$.

Pro daný vstup obvodu C_n^p označme $k = |I|$, kde $I = \{i \in \{1, \ldots, n\} \mid x_i = y_i = 1\}$, počet odpovídajících shodných jednotkových deterministických vstupů. Cílem výpočtu obvodu je pak určit paritu k. Pro všechna $i \in I$

$$\left\lceil \frac{8\varepsilon \ln 2}{(1-2\varepsilon)^2} + 1 \right\rceil ns + 1 \tag{9.13}$$

a hloubky $d+1$, tj. $L = L(C)$.

Důkaz: Nechť C^p má n deterministických vstupů a n_p náhodných vstupních hradel $x_1^p, \ldots, x_{n_p}^p$, z nichž x_i^p $(1 \leq i \leq n_p)$ je aktivní s pravděpodobností $p_i = \ell^p(x_i^p)$. Pro $\mathbf{r} = (r_1, \ldots, r_{n_p}) \in \{0,1\}^{n_p}$ označme $C(\mathbf{r})$ prahový obvod, který vznikne z C^p náhradou každého náhodného vstupního hradla x_i^p $(1 \leq i \leq n_p)$ hradlem s konstantní funkcí r_i. Zvolme konstantu c tak, že c je nejmenší sudé přirozené číslo, pro které platí:

$$c > \frac{8\varepsilon \ln 2}{(1-2\varepsilon)^2}\,. \tag{9.14}$$

Dále volíme náhodně a nezávisle cn binárních vektorů $\mathbf{r}_j = (r_{j1}, \ldots, r_{jn_p}) \in \{0,1\}^{n_p}$ pro $j = 1, \ldots, cn$ tak, že pravděpodobnost, že $r_{ji} = 1$ $(1 \leq i \leq n_p)$, je p_i. Potom prahový obvod C je konstruován podobně jako v důkazu věty 9.3, tj. skládá se z cn obvodů $C(\mathbf{r}_1), \ldots, C(\mathbf{r}_{cn})$, které mají stejných n deterministických vstupů a jsou připojeny k jednomu výstupnímu prahovému hradlu, které podle věty 6.26 počítá konsenzuální funkci $MAJORITY$. Zřejmě C má požadovanou velikost a hloubku.

Pro každý vstup $\mathbf{x} \in \{0,1\}^n$ obvodu C pravděpodobnost, že výstup $C(\mathbf{r}_j)$ je chybný, je nejvýše ε, protože C^p ε-rozpoznával $L = L(C^p, \varepsilon)$. Uvažujme libovolný pevný vstup $\mathbf{x} \in \{0,1\}^n$ obvodu C. Volbu $\mathbf{r}_1, \ldots, \mathbf{r}_{cn}$ lze interpretovat jako $N = cn$ Bernoulliho pokusů, kde neúspěch znamená, že výstup $C(\mathbf{r}_j)$ je chybný pro vstup $\mathbf{x}$, a tedy jeho pravděpodobnost je nejvýše $\frac{1}{4} < p = \varepsilon < \frac{1}{2}$. Potom podle (ii) z lemmy 9.2 pravděpodobnost, že aspoň polovina z prahových obvodů $C(\mathbf{r}_1), \ldots, C(\mathbf{r}_{cn})$ má chybný výstup pro vstup $\mathbf{x}$, tj. $0 < \beta = 1/(2\varepsilon) - 1 < 1$, je nejvýše

$$B\left(Np(1+\beta), N, p\right) = B\left(\frac{cn}{2}, cn, \varepsilon\right) \leq e^{-\frac{1}{2}(1/(2\varepsilon)-1)^2 cn\varepsilon}\,, \tag{9.15}$$

což lze díky (9.14) dále shora striktně odhadnout

$$B\left(\frac{cn}{2}, cn, \varepsilon\right) < 2^{-n}\,. \tag{9.16}$$

Tedy pro náhodnou volbu $\mathbf{r}_1, \ldots, \mathbf{r}_{cn}$ pravděpodobnost, že výstup prahového obvodu C je chybný pro vstup $\mathbf{x}$, je ostře menší než 2^{-n}. Avšak počet možných deterministických vstupů je 2^n, a proto pravděpodobnost, že výstup C je chybný pro nějaký vstup, je pro tuto volbu jistě ostře menší než 1. Z toho vyplývá, že musí existovat aspoň jedna volba $\mathbf{r}_1, \ldots, \mathbf{r}_{cn}$, při které C rozpoznává L. □

RTC'^0_k, kde $\varepsilon(n) = \frac{1}{2} - n^{-b}$ a b, c jsou konstanty. Podle věty 9.6 existuje obecně neuniformní posloupnost prahových obvodů polynomiální velikosti

$$\left\lceil \frac{8\varepsilon(n)\ln 2}{(1-2\varepsilon(n))^2} + 1 \right\rceil nS(n) + 1 = O\left(n^{2b+c+1}\right) \tag{9.21}$$

a hloubky $k+1$, rozpoznávající jazyk L. □

9.3 Boltzmannovy obvody

Pravděpodobnostní prahové obvody nejsou typickým představitelem neuronových sítí, které se využívají v aplikacích. Nejznámějším pravděpodobnostním modelem neuronových sítí je Boltzmannův stroj (viz podkapitolu 3.4), jehož neurony jsou aktivní s pravděpodobností, která se vypočte ze zvážené sumy vstupů pomocí spojité pravděpodobnostní aktivační funkce. Pro jednoduchost se budeme zabývat jeho acyklickou verzí, tj. *Boltzmannovým obvodem*, který se skládá z tzv. *Boltzmannových hradel.*

Definice 9.11 Boltzmannův obvod $C^B = (V, X, Y, E, \ell^B)$ *je obvod, kde* V *je množina* Boltzmannových hradel *a zobrazení* ℓ^B *přiřadí každému hradlu* $j \in V$ *s* k *vstupy* $x_1, \ldots, x_k$ *celočíselné váhy a práh* $\ell^B(j) = (w_1, \ldots, w_k; h)$ *lineární funkce* $\xi_j = \sum_{i=1}^{k} w_i x_i - h$. *Boltzmannovo hradlo* $j \in V$ *je aktivní, tj.* $y_j = 1$, *s pravděpodobností* $\sigma_j(\xi_j)$, *kde* $\sigma_j : \mathbb{Z} \longrightarrow (0,1)$ *je* pravděpodobnostní aktivační funkce, *např. standardní sigmoida (1.9):*

$$\sigma_j(\xi) = \frac{1}{1 + e^{-\lambda_j \xi}} \tag{9.22}$$

se strmostí $\lambda_j > 0$. *Boltzmannův obvod* C^B *s* $n = |X|$ *vstupy se využívá k* ε*-rozpoznávání jazyka* $L = L(C^B, \varepsilon) \subseteq \{0,1\}^n$ *stejným způsobem jako pravděpodobnostní prahový obvod (viz definici 9.1).*

Ukážeme, že Boltzmannovy obvody nejsou o moc silnější výpočetní prostředek než pravděpodobnostní prahové obvody.

Věta 9.12 (Parberry, Schnitger [215]) *Každý jazyk* $L = L(C^B, \varepsilon) \subseteq \{0,1\}^n$ ε*-rozpoznávaný Boltzmannovým obvodem* C^B *velikosti* s, *hloubky* d *a váhy* $|w|$ *lze* ε*-rozpoznat pravděpodobnostním prahovým obvodem* C^p *velikosti* $(6|w|+4)s$, *hloubky* $2d$ *a váhy* $(4|w|^2 + 10|w| + 4)s$, *tj.* $L = L(C^p, \varepsilon)$.

Důkaz: Každé hradlo j Boltzmannova obvodu C^B nahradíme pravděpodobnostním prahovým podobvodem $C^p(j)$, který realizuje jeho funkci. Podobvod $C^p(j)$ pro Boltzmannovo hradlo j s k vstupy $y_1, \ldots, y_k$ ohodnocenými váhami $w_1^B, \ldots, w_k^B$ a prahem h^B se především skládá s $q \leq 2|w|+1$ dvojic

jsou podle předcházející konstrukce výstupy odpovídajících hradel v první vrstvě rovny příslušnému náhodnému vstupu, tj. $y_{u_i} = y_{v_i} = x_i^p$. Podobně pro všechna $i \notin I$, tj. $x_i = 0$ nebo $y_i = 0$, platí $y_{u_i} = 0$ a $y_{v_i} = 1$. Tedy výstupní prahové hradlo v ve druhé vrstvě má $r = 2|\{i \in I \mid x_i^p = 1\}|$ a $n-k$ jednotkových vstupů. To znamená, že v je aktivní, právě když $r \geq k$, tj. aspoň $\frac{k}{2}$ z k náhodných vstupních hradel $i \in I$ musí být aktivních, u nichž pravděpodobnost aktivity je $\frac{1}{2}$. Tedy můžeme využít (*iii*) z lemmy 9.2, protože pravděpodobnost, že výstup v obvodu C_n^p je 1, je $B(\frac{k}{2}, k, \frac{1}{2})$, a dostáváme, že pravděpodobnostní prahový obvod C_n^p $(\frac{1}{2}, \frac{1}{2} + 1/\sqrt{2\pi k})$-separuje, resp. díky $k \leq n$ $(\frac{1}{2}, \frac{1}{2} + 1/\sqrt{2\pi n})$-separuje $\overline{IP} \cap \{0,1\}^n$.

Zřejmě C_n^p má jednotkové váhy, velikost $O(n)$ a hloubku 2. Tedy podle věty 9.5 existuje pravděpodobnostní prahový obvod s malými váhami, velikosti $O(n)$ a hloubky 2, který $\varepsilon(n)$-rozpoznává $\overline{IP}$ pro

$$\varepsilon(n) = \frac{1 - \frac{1}{2} - p\left(\frac{1}{2} + \frac{1}{\sqrt{2\pi n}} - \frac{1}{2}\right)}{2 - \frac{1}{2} - \frac{1}{\sqrt{2\pi n}} - \frac{1}{2}} = \frac{1}{2} - \frac{1}{8n}, \tag{9.19}$$

kde

$$p = \frac{4n + \sqrt{2\pi n} - 1}{8n}, \tag{9.20}$$

které splňuje (9.7). Nakonec pomocí (i) z lemmy 6.27 získáme posloupnost pravděpodobnostních prahových obvodů velikosti $O(n)$, s malými váhami a hloubky 2, která $(\frac{1}{2} - \frac{1}{8n})$-rozpoznává IP a je zřejmě L-uniformní, protože uvedené konstrukce lze realizovat pomocí Turingova stroje s logaritmickým prostorem. Tedy $IP \in RTC_2^0$. □

Důsledek 9.9 $TC_2^0 \subsetneq RTC_2^0$.

Důkaz: Tvrzení je přímým důsledkem vět 7.56 a 9.8. □

Na druhou stranu díky větě 7.51 je $IP \in TC_3^0$, a tedy podle věty 9.8 jsme prostřednictvím pravděpodobnostních obvodů ušetřili jednu vrstvu. Následující věta pro neuniformní případ ukazuje, že tímto způsobem již nelze hloubku obvodu dále redukovat.

Věta 9.10 (Hajnal, Maass, Pudlák, Szegedy, Turán [91]) *Označme* TC'^0_k, RTC'^0_k *(*$k \geq 1$*) neuniformní verze tříd složitosti* TC_k^0, RTC_k^0. *Potom pro každé* $k \geq 1$ *platí* $RTC'^0_k \subseteq TC'^0_{k+1}$.

Důkaz: Nechť $\mathbf{C}^p = (C_0^p, C_1^p, C_2^p, \ldots)$ je obecně neuniformní posloupnost pravděpodobnostních prahových obvodů polynomiální velikosti $S(n) = O(n^c)$ a konstantní hloubky k, $\varepsilon(n)$-rozpoznávající jazyk $L = L(\mathbf{C}^p, \varepsilon(n)) \in$

Důkaz: Každé hradlo j pravděpodobnostního obvodu C^p nahradíme v obvodu C^B Boltzmannovým hradlem j^B. Nechť nejprve j je náhodné vstupní hradlo, které je aktivní s pravděpodobností $0 < p < 1$. Odpovídající Boltzmannovo hradlo j^B s prahem -1 nebude mít žádné vstupy a pro jeho pravděpodobnostní aktivační funkci platí, že $\sigma_j(1) = p$, tj. např. pro standardní sigmoidu (9.22) stačí volit strmost

$$\lambda_j = \ln \frac{p}{1-p}\,. \tag{9.25}$$

Zřejmě hradlo j^B je aktivní s pravděpodobností p, a tedy realizuje stejnou funkci jako j.

Nyní nechť j představuje prahové hradlo obvodu C^p s reprezentací $(w_1, \ldots, w_k; h)$. Podle lemmy 6.29 můžeme předpokládat, že se jedná o rozhodující reprezentaci, přitom váha obvodu C^B bude $2|w|$. Odpovídající Boltzmannovo hradlo j^B má stejné váhy $w_1, \ldots, w_k$ a práh h a pro jeho pravděpodobnostní aktivační funkci σ_j platí, že $\sigma(1) \geq 1 - \delta/s$ a $\sigma(-1) \leq \delta/s$, tj. např. pro standardní sigmoidu (9.22) stačí volit strmost

$$\lambda_j = \ln \frac{s-\delta}{\delta} > 0\,. \tag{9.26}$$

Zřejmě hradlo j^B počítá stejnou funkci jako j s pravděpodobností chyby nejvýše δ/s.

Boltzmannův obvod C^B se pak chová jinak než původní pravděpodobnostní obvod C^p s pravděpodobností nejvýše δ/s pro každé hradlo, tj. celkově s pravděpodobností menší než δ. Odtud pravděpodobnost chyby při rozpoznávání L je menší než $\varepsilon + \delta$, tj. $L = L(C^B, \varepsilon + \delta)$. □

Věty 9.12, 9.14 ukazují, že z hlediska výpočetní síly jsou Boltzmannovy obvody velmi podobné pravděpodobnostním prahovým obvodům. Také můžeme uvažovat jejich cyklické verze (viz kapitolu 8), tj. Boltzmannovy stroje a pravděpodobnostní cyklické neuronové sítě. Pro jejich konvergentní plně paralelní režim pak můžeme použít techniku z věty 8.7 k vytvoření ekvivalentních obvodů a následně opět pomocí vět 9.12, 9.14 ukázat, že Boltzmannovy stroje jsou z hlediska výpočetní síly velmi podobné pravděpodobnostním cyklickým neuronovým sítím.

9.4 Robustní neuronové sítě

Je známo, že neuronové sítě jsou robustní, tj. při poškození několika neuronů síť nemusí nezbytně ztratit svoji funkčnost. V této podkapitole se pokusíme naznačit možné příčiny tohoto jevu.

prahových hradel $u(\xi'_j)$, $v(\xi'_j)$ (srovnejte s technikou důkazu věty 7.32), které odpovídají všem možným celočíselným hodnotám zvážených sum ξ'_j vstupů (bez prahu):

$$-|w| \leq \xi'_j = \sum_{i=1}^{k} w_i^B y_i \leq |w|\,. \tag{9.23}$$

K oběma hradlům $u(\xi'_j)$, $v(\xi'_j)$ z každé dvojice, které mají práh $h^p(u(\xi'_j)) = -h^p(v(\xi'_j)) = \xi'_j + 1$, jsou připojeny vstupy $y_1, \ldots, y_k$ pomocí vah $w^p(y_i, u(\xi'_j)) = -w^p(y_i, v(\xi'_j)) = w_i^B$ a přidané náhodné vstupní hradlo $x^p(\xi'_j)$ pomocí váhy $w^p(x^p(\xi'_j), u(\xi'_j)) = -w^p(x^p(\xi'_j), v(\xi'_j)) = 1$, které je aktivní ($x^p(\xi'_j) = 1$) s pravděpodobností $\sigma_j(\xi'_j - h^B)$, kde σ_j je pravděpodobnostní aktivační funkce hradla j.

Zřejmě pro skutečnou hodnotu zvážené sumy $\bar{\xi}'_j$ Boltzmannova hradla j je hradlo $u(\xi'_j)$ aktivní, právě když $\bar{\xi}'_j + x^p(\xi'_j) \geq \xi'_j + 1$ a hradlo $v(\xi'_j)$ je aktivní, právě když $\bar{\xi}'_j + x^p(\xi'_j) \leq \xi'_j + 1$. To znamená, že je vždy jedno hradlo z každé z q dvojic aktivní a navíc obě hradla z dvojice $u(\bar{\xi}'_j)$, $v(\bar{\xi}'_j)$ jsou zároveň aktivní, tj. $x^p(\xi'_j) = 1$, s pravděpodobností $\sigma_j(\bar{\xi}'_j - h^B)$. Tedy stačí všechna hradla $u(\xi'_j)$, $v(\xi'_j)$ připojit v $C^p(j)$ přes jednotkové váhy k prahovému hradlu s prahem $q+1$, které realizuje funkci Boltzmannova hradla j a je výstupem podobvodu $C^p(j)$.

Zřejmě vzniklý pravděpodobnostní prahový obvod má velikost nejvýše $s(3(2|w|+1)+1) = (6|w|+4)s$, hloubku $2d$ a váhu $s(2|w|+1)(2|w|+4) = (4|w|^2 + 10|w| + 4)s$. □

Hloubku pravděpodobnostního prahového obvodu z věty 9.12, který simuluje Boltzmannův obvod hloubky d, lze dokonce pomocí techniky z [91] redukovat na $d+1$.

Důsledek 9.13 *Třída jazyků $L \subseteq \{0,1\}^\star$ $\varepsilon(n)$-rozpoznatelných L-uniformní posloupností Boltzmannových obvodů (polynomiální velikosti $S(n) = n^{O(1)}$) konstantní hloubky $D(n) = O(1)$ s malými váhami a chybou*

$$\varepsilon(n) = \frac{1}{2} - \frac{1}{n^{O(1)}} \tag{9.24}$$

se shoduje se složitostní třídou RTC^0.

Důkaz: Tvrzení je přímým důsledkem definice 9.7 a věty 9.12. □

Na druhou stranu pomocí Boltzmannova obvodu lze jednoduše simulovat pravděpodobnostní prahový obvod.

Věta 9.14 *Každý jazyk $L = L(C^p, \varepsilon) \subseteq \{0,1\}^n$ ε-rozpoznávaný pravděpodobnostním prahovým obvodem C^p velikosti s, hloubky d a váhy $|w|$ lze $(\varepsilon+\delta)$-rozpoznat Boltzmannovým obvodem C^B velikosti s, hloubky d a váhy $2|w|$, tj. $L = L(C^B, \varepsilon + \delta)$, pro libovolné $0 < \delta < \frac{1}{2} - \varepsilon$.*

Definice 9.15 *Nechť* $f : \{0,1\}^n \longrightarrow \{0,1\}$ *je booleovská funkce a* C *je obvod s* n *vstupy a jedním výstupem. Řekneme, že* C chybuje při výpočtu f, *jestliže existuje vstup* $\mathbf{x} \in \{0,1\}^n$ *takový, že* $C(\mathbf{x}) \neq f(\mathbf{x})$. *Dále řekneme, že* C *je* (μ,ε)-robustní *pro* f, *kde* $0 \leq \mu + \varepsilon \leq 1$, *jestliže platí, že když každé hradlo v* C *je* poškozeno *nezávisle s pravděpodobností nejvýše* ε, *tj. je* nespolehlivé *s pravděpodobností chyby* ε, *pak pravděpodobnost, že* C *chybuje při výpočtu* f, *je nejvýše* $\mu+\varepsilon$. *Reálnou hodnotu* $\frac{1}{\mu}$ *nazveme* přesností *obvodu* C. *Nakonec řekneme, že obvod* C *je* ε-robustní, *jestliže je* (μ,ε)-*robustní pro nějaké* μ *takové, že* $\mu + \varepsilon < \frac{1}{2}$.

V definici 9.15 zřejmě $\mu \geq 0$, protože pravděpodobnost, že výstupní hradlo je poškozeno, je ε. Intuitivně ε ve výrazu $\mu + \varepsilon$ odpovídá pravděpodobnosti chyby výstupního hradla a μ je pravděpodobnost poškození zbytku obvodu.

Ukážeme, že pro každý spolehlivý prahový obvod existuje větší ekvivalentní prahový obvod, který je robustní, protože kopíruje několikrát výpočet původního obvodu.

Věta 9.16 (Berman, Parberry, Schnitger [35]) *Každou funkci* f, *kterou lze počítat prahovým obvodem* C *velikosti* s *a hloubky* d *s hradly, které mají počet vstupů omezen* q, *je možné počítat pomocí* (μ,ε)-*robustního prahového obvodu* C' *pro* f *velikosti*

$$\frac{4s}{\varepsilon\beta^2}\left(\ln s + \ln\frac{2}{\mu}\right) + 1 \tag{9.27}$$

a hloubky $2d+1$ *pro libovolné*

$$\frac{1}{4(q+1)} \leq \varepsilon < \frac{1}{2(q+1)} \tag{9.28}$$

a $\mu > 0$, *kde*

$$\beta = \frac{1}{2\varepsilon(q+1)} - 1\,. \tag{9.29}$$

Důkaz: Konstruujeme robustní prahový obvod C' tak, že každý spoj v obvodu C nahradíme tzv. *kabelem*, který se skládá z m spojů, a tedy každé hradlo v s $k \leq q$ vstupy nahradíme podobvodem C_v s k vstupními kabely a jedním výstupním kabelem. Řekneme, že spoj v kabelu je *korektní*, jestliže vždy přenáší stejný signál jako původní odpovídající spoj v C. Dále řekneme, že kabel je korektní, jestliže nejvýše θm jeho spojů není korektních, kde $0 \leq \theta \leq 1$ je kladné reálné číslo.

Podobvod C_v se skládá ze dvou vrstev. První vrstva obsahuje m kopií hradla v tak, že j-té hradlo má i-tý vstup od j-tého spoje i-tého vstupního kabele ($1 \leq j \leq m$, $1 \leq i \leq k$). Druhá vrstva se skládá z m hradel, které počítají konsenzuální funkci $MAJORITY$ se vstupy od všech hradel

z první vrstvy. Výstupy hradel ze druhé vrstvy pak tvoří výstupní kabel podobvodu C_v. Nakonec spoje z kabele, který odpovídá výstupu obvodu C, jsou v obvodu C' vstupem hradla $MAJORITY$, které je výstupem obvodu C'. Zřejmě hloubka obvodu C' je $2d+1$.

Nyní předpokládejme, že každé hradlo v C' je poškozeno nezávisle s pravděpodobností ε, pro kterou platí (9.28). Vypočteme pravděpodobnost, že výstupní kabel podobvodu C_v pro libovolné hradlo v s $k \leq q$ vstupy není korektní, za předpokladu, že všechny vstupní kabely C_v jsou korektní. Díky tomuto předpokladu nejvýše $q\theta m$ kopií hradla v v první vrstvě podobvodu C_v chybuje kvůli chybným vstupům. Tedy v nejhorším případě $\frac{1}{2}m - q\theta m = (\frac{1}{2} - q\theta)m$ poškozených kopií hradla v způsobí, že polovina z nich bude chybovat. Proto pravděpodobnost, že více než polovina kopií hradla v v první vrstvě podobvodu C_v chybuje, je menší než $B((\frac{1}{2} - q\theta)m, m, \varepsilon)$ (viz lemmu 9.2). Dále pravděpodobnost, že výstupní kabel podobvodu C_v není korektní, za předpokladu, že méně než polovina hradel v první vrstvě C_v chybuje, je menší než $B(\theta m, m, \varepsilon)$. Z toho vyplývá, že pravděpodobnost, že výstupní kabel podobvodu C_v není korektní, za předpokladu, že všechny vstupní kabely C_v jsou korektní, je menší než součet

$$B\left(\left(\frac{1}{2} - q\theta\right)m, m, \varepsilon\right) + B(\theta m, m, \varepsilon)\,. \tag{9.30}$$

Položme $\theta = \frac{1}{2(q+1)}$. Pravděpodobnost (9.30) lze pak podle (ii) z lemmy 9.2 shora odhadnout:

$$\begin{aligned} B\left(\left(\frac{1}{2} - q\theta\right)m, m, \varepsilon\right) + B(\theta m, m, \varepsilon) &= 2B\left(\frac{m}{2(q+1)}, m, \varepsilon\right) \leq \\ &\leq 2e^{-\frac{1}{2}\beta^2 m\varepsilon}\,, \end{aligned} \tag{9.31}$$

kde $m\varepsilon(1+\beta) = \frac{m}{2(q+1)}$, tj.

$$\beta = \frac{1}{2\varepsilon(q+1)} - 1\,, \tag{9.32}$$

za předpokladu, že $0 \leq \beta \leq 1$, který je podle (9.28) splněn.

Jelikož v obvodu C' je právě s kabelů odpovídajících výstupům s hradel obvodu C, které mohou být nezávisle nekorektní, pravděpodobnost, že kabel odpovídající výstupu obvodu C není korektní, lze podle (9.31) shora odhadnout pomocí

$$2se^{-\frac{1}{2}\beta^2 m\varepsilon}\,, \tag{9.33}$$

což je nejvýše μ, pokud volíme

$$m = \frac{2}{\beta^2\varepsilon}\left(\ln s + \ln\frac{2}{\mu}\right)\,. \tag{9.34}$$

Výstupní hradlo obvodu C', které počítá $MAJORITY$, pak chybuje s pravděpodobností menší než $\mu + \varepsilon$, a tedy prahový obvod C' je (μ, ε)-robustní pro f. Celkový počet hradel v C' je $2ms + 1$. □

Podle věty 9.16 může libovolný prahový obvod s nespolehlivými hradly počítat dostatečně spolehlivě danou funkci za cenu nárůstu velikosti o logaritmický faktor. Navíc přesnost takového obvodu může být zvýšena na libovolnou konstantu při lineárním nárůstu původního obvodu.

Robustnost neuronových sítí lze pomocí věty 9.16 ilustrovat jen pro funkce, které lze počítat pomocí obvodů, které mají hradla s malým počtem vstupů. Pokud např. poškodíme každé desáté hradlo, dostáváme spolehlivý prahový obvod praktické velikosti.

Důsledek 9.17 *Každou funkci, kterou lze počítat klasickým obvodem velikosti s a hloubky d, je možné počítat* 0.1*-robustním prahovým obvodem velikosti* $90s \ln s + 270s + 1$ *a hloubky* $2d + 1$.

Důkaz: Ve větě 9.16 položme $q = 2$, $\varepsilon = 0.1$ a např. $\mu = 0.1$. □

Částečné zobecnění důsledku 9.17 pro hradla s neomezeným počtem vstupů lze najít v [35, 91].

Kapitola 10

Výpočetní síla neuronových sítí

V této kapitole porovnáme obecně cyklické (diskrétní i analogové) neuronové sítě s klasickými modely výpočtů jako jsou konečné automaty a regulární výrazy dané deskriptivní složitosti [2, 137] a Turingovy stroje omezené časové a prostorové složitosti (příp. s orákuly) [26]. Za tímto účelem budeme neuronové sítě využívat jako tzv. *neuronové akceptory* jazyků (nad binární abecedou). Abychom mohli hovořit o složitosti, musí být délka vstupního slova obecně neomezená. Uvažujeme v zásadě dva typy vstupního protokolu. U konečných neuronových sítí máme jeden vstupní neuron, který obvykle postupně během výpočtu čte bity vstupního slova. Pro tento typ vstupního protokolu budeme v následujících dvou podkapitolách uvažovat po řadě diskrétní nebo analogový model sítě. Jinou možností je nekonečná posloupnost neuronových sítí, kde každé délce vstupu odpovídá právě jedna síť s příslušným počtem vstupních neuronů. Proto třetí podkapitola je věnována posloupnostem neuronových sítí. Pro oba typy vstupního protokolu stav výstupního neuronu určuje, zda vstupní slovo patří do daného jazyka. U některých důkazů vět vzhledem k jejich technické náročnosti nastíníme v této kapitole jen základní ideu bez detailů a některé vynecháme s odkazem na původní práce. Také se omezíme jen na deterministické modely neuronových sítí, přitom analogickou analýzu pro pravděpodobnostní neuronové sítě lze nalézt např. v [253].

10.1 Neuromaty

Nejprve definujeme konečný diskrétní neuronový akceptor, kterému budeme díky zřejmé souvislosti s konečnými automaty říkat *neuromat*.

Definice 10.1 Neuromat (neuronový akceptor) $N = (V, inp, out, A, w, h)$ *je cyklická neuronová síť (viz definice 8.1, 8.2) pracující v plně paralelním režimu s jedním vstupním neuronem* $inp \in V$ *a jedním výstupním neuronem* $out \in V$. *Navíc platí, že v příslušné architektuře* (V, E) *nevede do vstupního neuronu* inp *žádná hrana a všechny orientované cesty vedoucí z* inp *do* out *jsou délky aspoň* $k+1$, *kde* $k \geq 1$ *je přirozené číslo nazývané* zpoždění. *Neuromat se využívá k rozpoznávání jazyka* $L \subseteq \{0,1\}^\star$ *nad binární abecedou následujícím způsobem. Nechť* $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$ *je binární* vstupní slovo *(vstup neuromatu), které je formálně zprava doplněno libovolnými bity* x_{n+i} *(*$i \geq 1$*). Stav vstupního neuronu* inp *(včetně počátečního stavu) je v každém časovém kroku výpočtu neuromatu aktualizován podle bitů vstupního slova, tj.* $y_{inp}^{(t)} = x_{t+1}$ *pro* $t \geq 0$. *Označme* $N_k(\mathbf{x}) = y_{out}^{(n+k)}$ *stav výstupního neuronu* out *v čase* $n+k$ *při výpočtu neuromatu* N *pro vstup* $\mathbf{x} \in \{0,1\}^n$. *Potom množina* $L_k(N) = \{\mathbf{x} \in \{0,1\}^\star \mid N_k(\mathbf{x}) = 1\}$ *je jazyk* rozpoznávaný *neuromatem* N *s časovým zpožděním* k, *speciálně* $L(N) = L_1(N)$ *je jazyk rozpoznávaný neuromatem* N.

Ukážeme, že konstantní časové zpoždění při rozpoznávání jazyka pomocí neuromatu lze odstranit za cenu jen lineárního nárůstu jeho velikosti. Proto se v následujícím výkladu můžeme omezit na neuromaty, které odpovídají na vstupní slovo jeden časový krok po jeho načtení.

Věta 10.2 (Šíma [268]) *Pro každý neuromat* $N = (V, inp, out, A, w, h)$ *velikosti* s *se zpožděním* $k \geq 2$ *existuje neuromat* $N' = (V', inp', out', A', w', h')$ *velikosti* $2^k(s-1)+2$ *s jednotkovým zpožděním, který rozpoznává stejný jazyk, tj.* $L_k(N) = L(N')$.

Důkaz: Myšlenka důkazu spočívá v konstrukci neuromatu N', který $k-1$ kroků dopředu simuluje výpočty neuromatu N pro všechna možná $(k-1)$-bitová pokračování vstupu a přeruší ty neplatné výpočty, které nesouhlasí se skutečným vstupem zpožděným $k-1$ kroků. Za tímto účelem se neuromat N' kromě vstupního a výstupního neuronu inp', out' skládá z 2^k podsítí $N_{\mathbf{x}b}$ ($\mathbf{x} \in \{0,1\}^{k-1}$, $b \in \{0,1\}$), z nichž každá simuluje výpočet N pro jedno možné $(k-1)$-bitové pokračování $\mathbf{x} \in \{0,1\}^{k-1}$ vstupního slova. Příslušné neurony v těchto podsítích označíme stejně jako v N a navíc je indexujeme odpovídajícími k bity, tj.

$$V' = \left\{ v_{\tilde{\mathbf{x}}} \mid v \in V - \{inp\}, \tilde{\mathbf{x}} \in \{0,1\}^k \right\} \cup \{inp', out'\}. \qquad (10.1)$$

nulovým stavům žádný vliv na out'. Zbylá polovina těchto neuronů $v_{\tilde{\mathbf{x}}}$ ($w(v, out) \neq 0$) v podsítích $N_{\tilde{\mathbf{x}}}$ má stejné stavy, protože vzdálenost mezi inp a out v N je aspoň $k+1$, a tedy posledních k vstupních bitů nemohlo ovlivnit výstup. Z toho vyplývá, že pro zachování funkce out' stačí vynásobit odpovídající práh $h(out)$ pomocí 2^{k-1}, tj. $h'(out') = 2^{k-1}h(out)$. Uvedeným způsobem je dosaženo správné rozpoznávání jazyka $L_k(N)$ pomocí neuromatu N' o $k-1$ kroků dopředu. Navíc velikost neuromatu N' je $2^k(s-1)+2$. □

Věta 10.3 (Kleene [150]) *Neuromaty a konečné automaty jsou výpočetně ekvivalentní, tj. neuromaty rozpoznávají právě regulární jazyky.*

Důkaz: Neuromat lze chápat jako konečný automat, jehož stavy jsou reprezentovány stavy nevstupních neuronů příslušné sítě a přechodová funkce je popsána pomocí booleovských prahových funkcí těchto neuronů. Počáteční stav konečného automatu je dán množinou iniciálně aktivních neuronů a koncové stavy odpovídají stavům sítě s aktivním výstupním neuronem.

Obráceně každé hraně stavového grafu konečného automatu odpovídá jeden neuron, který kontroluje, zda její ohodnocení souhlasí se stavem vstupního neuronu. Aktivita neuronů odpovídajících hranám, které vstupují do daného stavu, podmiňuje aktivaci neuronů, které odpovídají hranám vystupujícím z tohoto stavu. Jeden iniciálně aktivní neuron na začátku aktivuje neurony, které reprezentují hrany vycházející z počátečního stavu. Výstupní neuron je pak disjunkcí neuronů odpovídajících hranám, které vstupují do koncových stavů. Podobnou podrobněji popsanou konstrukci neuromatu lze najít v důkazu věty 10.5. □

V důkazu věty 10.3 je naznačena konstrukce neuromatu ekvivalentního s konečným automatem, která k m-stavovému deterministickému automatu (ve stavovém grafu automatu z každého vrcholu vychází nejvýše 2 hrany) vytvoří neuromat velikosti $O(m)$. Následující věta zlepšuje tento výsledek tak, že pro malé váhy je již optimální. Na druhou stranu předpokládá obecnější vstupní protokol než v definici 10.1. Vstupní bity jsou neuromatu předkládány prostřednictvím vstupního neuronu postupně — jeden bit během časové periody, která trvá obecně déle než jeden krok. Tuto větu uvádíme bez důkazu, který není konstruktivní. Jiří Wiedermann (osobní komunikace) navrhl pro uvedený obecnější vstupní protokol následující deterministickou konstrukci. Přechodová funkce konečného automatu závisející na stavu a vstupním bitu, které lze kódovat $O(\log m)$ bity, je realizována pomocí neuronové sítě pro výpočet obecné vektorové booleovské funkce $f : \{0,1\}^{O(\log m)} \longrightarrow \{0,1\}^{O(\log m)}$. Neuromat má v tomto případě podle věty 7.41 velikost $O(\sqrt{m \log m})$.

Věta 10.4 (Indyk [139]) *Ke každému m-stavovému (deterministickému) konečnému automatu existuje $5t$-ekvivalentní neuromat velikosti $O(\sqrt{m})$*

Chceme, aby v každém čase $t \geq 0$ výpočtu N' byl stav $y_{v_{\mathbf{x}b}}^{(t)}$ neuronu $v_{\mathbf{x}b}$ ($b \in \{0,1\}$) roven stavu $y_v^{(t+k-1)}$ neuronu $v \in V$ u neuromatu N v čase $t+k-1$ po zpracování dalších $k-1$ bitů $\mathbf{x} \in \{0,1\}^{k-1}$ vstupu. Toho dosáhneme následujícím způsobem.

Na začátku výpočtu N' jsou následující neurony iniciálně aktivní:

$$A' = \left\{ v_{\mathbf{x}b} \in V' \mid y_v^{(k-1)}(\mathbf{x}) = 1,\ \mathbf{x} \in \{0,1\}^{k-1},\ b \in \{0,1\} \right\}, \qquad (10.2)$$

kde $y_v^{(k-1)}(\mathbf{x})$ je stav neuronu v v čase $k-1$ při výpočtu neuromatu N na vstup $\mathbf{x}$. V obecném kroku podsíť $N_{\mathbf{x}b}$ pro $\mathbf{x} \in \{0,1\}^{k-1}$, $b \in \{0,1\}$ počítá svůj nový stav z předchozího stavu podsítě $N_{a\mathbf{x}}$ ($a \in \{0,1\}$). Proto neurony $N_{a\mathbf{x}}$ jsou připojeny k neuronům $N_{\mathbf{x}b}$ pomocí vah, které odpovídají původním váhám v N, tj. $w'(u_{a\mathbf{x}}, v_{\mathbf{x}b}) = w(u,v)$ pro $\mathbf{x} \in \{0,1\}^{k-1}$, $a, b \in \{0,1\}$ a $u, v \in V \setminus \{inp\}$. Podsíť $N_{\mathbf{x}b}$ má konstantní vstup b, který je v případě $b = 1$ zohledněn tak, že prahy $h(v)$ všech neuronů $v_{\mathbf{x}b}$ v této síti jsou zmenšeny o příslušné vstupní váhy $w(inp, v)$ (viz (10.5)).

Vstupní bit $y_{inp'} = c \in \{0,1\}$ neuromatu N' indikuje, že všechny výpočty $N_{a\mathbf{x}}$ pro $a \neq c$ jsou neplatné. Proto vstupní neuron inp' zruší tyto výpočty tak, že vynuluje všechny stavy neuronů v $N_{a\mathbf{x}}$ pomocí následujících vah a prahů:

$$w'(inp', v_{0\mathbf{x}}) = h(v) - 1 - \sum_{u \in V, w(u,v) > 0} w(u,v) \qquad \mathbf{x} \in \{0,1\}^{k-1} \quad (10.3)$$

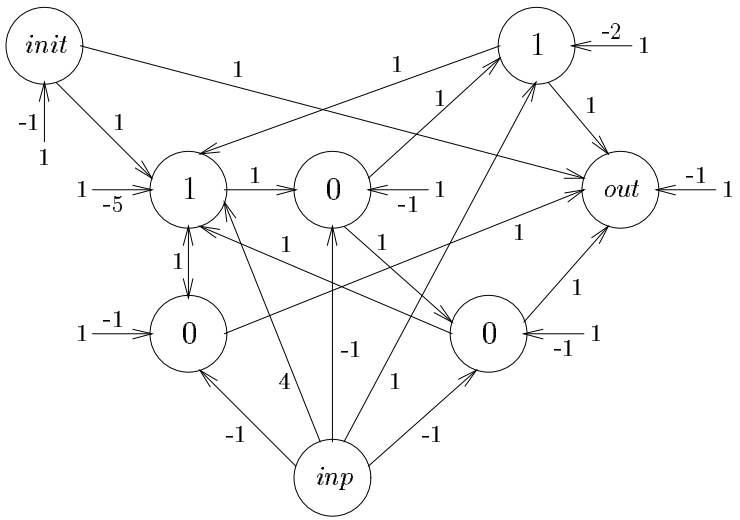
$$w'(inp', v_{1\mathbf{x}}) = -w'(inp', v_{0\mathbf{x}}) \qquad \mathbf{x} \in \{0,1\}^{k-1} \quad (10.4)$$

$$h'(v_{a\mathbf{x}'b}) = h(v) + aw'(inp', v_{a\mathbf{x}'b}) - bw(inp, v) \qquad \mathbf{x}' \in \{0,1\}^{k-2};\ a, b \in \{0,1\} \quad (10.5)$$

pro všechny $v \in V \setminus \{inp\}$. Definice vah (10.3) zajistí vynulování stavů všech neuronů v podsítích $N_{0\mathbf{x}}$ ($\mathbf{x} \in \{0,1\}^{k-1}$), pokud vstupní neuron inp' je aktivní. Podobně váhy (10.4) spolu s členem $aw'(inp', v_{a\mathbf{x}'b})$ u prahů (10.5) vynulují stavy všech neuronů v podsítích $N_{1\mathbf{x}}$ ($\mathbf{x} \in \{0,1\}^{k-1}$), pokud vstupní neuron inp' je pasivní. Tedy všechny neurony v 2^{k-1} podsítích $N_{a\mathbf{x}}$ ($\mathbf{x} \in \{0,1\}^{k-1}$), u nichž a nesouhlasí s předchozím vstupním bitem, mají nulové stavy. To také umožňuje podsíti $N_{\mathbf{x}b}$ realizovat správný výpočet, protože vliv jedné z neplatných podsítí, buď $N_{0\mathbf{x}}$, nebo $N_{1\mathbf{x}}$, které jsou připojeny k $N_{\mathbf{x}b}$, je díky nulovým stavům potlačen.

Neurony $v_{\tilde{\mathbf{x}}} \in V'$ v podsítích $N_{\tilde{\mathbf{x}}}$, jejichž odpovídající neuron $v \in V$ je v N spojen s výstupním neuronem out, jsou také připojeny k výstupnímu neuronu out' sítě N' prostřednictvím příslušných vah, tj. $w'(v_{\tilde{\mathbf{x}}}, out') = w(v, out)$ pro $v \in V$ a $\tilde{\mathbf{x}} \in \{0,1\}^k$. Víme, že polovina z nich nemá díky

disjunkci, a tedy je aktivní, právě když je aktivní aspoň jeden z jeho vstupů. Jeho práh je $h(out) = 1$. Příklad neuromatu, který rozpoznává regulární jazyk $[(1(0+0(1+0)))^\star]$, je na obrázku 10.1, kde jsou také vyznačeny typy neuronů.



Obr. 10.1: Neuromat rozpoznávající regulární jazyk $[(1(0+0(1+0)))^\star]$.

Dolní odhad na velikost neuromatu odpovídajícího regulárnímu výrazu v nejhorším případě ukážeme standardním způsobem. Definujeme regulární výraz $\boldsymbol{\alpha}_m$ $(m \geq 1)$ popisující regulární jazyk $L_m = [\boldsymbol{\alpha}_m]$, který lze upravit na výraz délky $|\boldsymbol{\alpha}_m| = O(m)$:

$$\begin{aligned} \boldsymbol{\alpha}_m &= \left(\sum_{i=0}^{m-3} (1(e+0))^i 10 + (1(e+0))^{m-1}(1+0) \right)^\star = \qquad (10.6) \\ &= (10 + 1(e+0)(10 + \\ &\quad +1(e+0)(\cdots 1(e+0)(10 + 1(e+0)^2(1+0))\cdots)))^\star , \end{aligned}$$

kde symbol e značí prázdné slovo. Dále označme

$$P_m = \left[\sum_{i=0}^{m-1} (1(e+0))^i \right] \qquad (10.7)$$

s malými váhami. Na druhou stranu pro většinu m-stavových konečných automatů každý ekvivalentní neuromat s malými váhami má velikost $\Omega(\sqrt{m})$.

10.1.1 Neuromaty a regulární výrazy

Vzhledem k tomu, že naznačenou konstrukci neuromatu v důkazu věty 10.3 lze použít i pro nedeterministické automaty, budeme se v následujícím výkladu zabývat porovnáním neuromatů s regulárními výrazy, které také popisují regulární jazyky a navíc v sobě skrývají nedeterminismus, tedy jsou pro naše účely přirozenější. Nejprve dokážeme analogii věty 10.4 pro regulární výrazy, tj. ukážeme optimální konstrukci neuromatu z regulárního výrazu.

Věta 10.5 (Šíma [267]) *Pro každý regulární výraz* $\boldsymbol{\alpha}$ *délky* $m = |\boldsymbol{\alpha}|$*, který popisuje regulární jazyk* $[\boldsymbol{\alpha}]$*, lze zkonstruovat neuromat* N *velikosti* $O(m)$ *s malými váhami, který rozpoznává stejný jazyk* $L(N) = [\boldsymbol{\alpha}]$*. Na druhou stranu existují regulární výrazy* $\boldsymbol{\alpha}_m$ *(*$m \geq 1$*) délky* $|\boldsymbol{\alpha}_m| = O(m)$ *takové, že každý neuromat* N_m *(s neomezenými váhami), který rozpoznává regulární jazyk* $L_m = [\boldsymbol{\alpha}_m] = L(N_m)$ *má velikost aspoň* $\Omega(m)$*.*

Důkaz: Princip konstrukce neuromatu N z regulárního výrazu $\boldsymbol{\alpha}$ je podobný jako u konstrukce odpovídajícího nedeterministického automatu. Neuromat N se skládá z neuronů, z nichž každý odpovídá jednomu výskytu symbolu 1 nebo 0 v regulárním výrazu $\boldsymbol{\alpha}$. Podle toho rozlišujeme neurony typu 1 nebo 0. Navíc N má vstupní neuron *inp*, výstupní neuron *out* a inicializační neuron *init*, který je jediný iniciálně aktivní. Architektura sítě odpovídá struktuře regulárnímu výrazu, tj. pro každou orientovanou cestu z *init* do *out* skládající se po řadě z neuronů typu $x_1 \ldots x_n \in \{0,1\}^\star$ slovo $x_1 \ldots x_n \in [\boldsymbol{\alpha}]$ je z regulárního jazyka $[\boldsymbol{\alpha}]$ daného regulárním výrazem $\boldsymbol{\alpha}$. Naopak pro každé takové slovo z $[\boldsymbol{\alpha}]$ existuje příslušná orientovaná cesta z *init* do *out* v topologii N. Je zřejmé, že regulární operace spojení $(\cdot)$ odpovídá prosté cestě, sjednocení $(+)$ znamená větvení a iterace $(^\star)$ je reprezentována cykly. Všechny hrany na uvedených cestách mají jednotkové váhy. Navíc vstupní neuron je připojen ke každém neuronu typu 1 nebo 0.

Výpočet neuromatu probíhá tak, že počáteční aktivita neuronu *init* se šíří po všech cestách v síti, které souhlasí se vstupním slovem. Za tímto účelem neurony daného typu kontrolují, zda vstupní bit souhlasí s uvedeným typem. To znamená, že neuron j typu 1 je aktivní, právě když je vstupní neuron *inp* aktivní a současně je aktivní aspoň jeden z k neuronů typu 1 nebo 0 nebo *init*, které jsou k němu připojeny. Tomu odpovídá váha $w(inp, j) = k$ a práh $h(j) = k + 1$. Podobně neuron j typu 0 je aktivní, právě když je vstupní neuron *inp* pasivní a současně je aktivní aspoň jeden z k neuronů typu 1 nebo 0 nebo *init*, které jsou k němu připojeny. Tomu odpovídá váha $w(inp, j) = -k$ a práh $h(j) = 1$. Nakonec výstupní neuron *out* realizuje

množinu $2^m - 1$ prefixů jazyka L_m, tj. $|P_m| = \Omega(2^m)$. Dá se ukázat [267], že pro každé dva různé prefixy $\mathbf{x}_1, \mathbf{x}_2 \in P_m$ $(\mathbf{x}_1 \neq \mathbf{x}_2)$ jazyka L_m existuje doplnění $\mathbf{x} \in \{0,1\}^\star$ takové, že $\mathbf{x}_1\mathbf{x} \in L_m$ a současně $\mathbf{x}_2\mathbf{x} \notin L_m$. Z toho vyplývá, že neuromat N_m, který rozpoznává $L_m = L(N_m)$, musí mít $\Omega(2^m)$ různých stavů, do kterých se dostane, když jsou na vstupu prefixy z P_m, které mohou být doplněny o $\mathbf{x}$. Tedy neuromat N_m má aspoň $\Omega(m)$ binárních neuronů. □

Podle věty 10.5 by mohlo se zdát, že neuromaty a regulární výrazy jsou v rámci polynomiální deskriptivní složitosti regulárních jazyků ekvivalentní. Avšak následující věta ukazuje, že neuromaty jsou v jistém smyslu silnější, protože k danému neuromatu nelze vždy zkonstruovat odpovídající regulární výraz polynomiální délky.

Věta 10.6 *Pro každé $m \geq 1$ existuje regulární jazyk $L_m = L(N_m)$, který je rozpoznáván neuromatem N_m velikosti $O(m)$, takový, že každý regulární výraz $\boldsymbol{\alpha}_m$, který jej popisuje $L_m = [\boldsymbol{\alpha}_m]$, má délku $|\boldsymbol{\alpha}_m| = \Omega(2^m)$.*

Důkaz: Myšlenka důkazu spočívá v tom, že regulární jazyk L_m $(m \geq 1)$ obsahuje všechna slova délky právě $\Theta(2^m)$. Neuromat N_m, který rozpoznává $L_m = L(N_m)$, využívá k určení délky vstupního slova např. m-bitového čítače z věty 8.18 velikosti $O(m)$ (konstrukce neuronového čítače s nesymetrickými spoji je ještě jednodušší).

Na druhou stranu vzhledem ke konečnosti L_m odpovídající regulární výraz $\boldsymbol{\alpha}_m$ $(L_m = [\boldsymbol{\alpha}_m])$ minimální délky neobsahuje iteraci $(^\star)$. Proto při generování slov jazyka L_m z regulárního výrazu $\boldsymbol{\alpha}_m$ se tento výraz čte jen zleva doprava. To znamená, že $\boldsymbol{\alpha}_m$ má délku aspoň jako délka slov z L_m, tj. $|\boldsymbol{\alpha}_m| = \Omega(2^m)$. □

Deskriptivní sílu neuromatů ve srovnání s regulárními výrazy lze ještě ilustrovat na složitosti problému prázdného jazyka, který je rozpoznáván zadaným neuromatem:

Problém prázdného jazyka zadaného neuromatem NEP (**N**euromaton **E**mptyness **P**roblem):
instance: Neuromat N.
otázka: Je $L(N) \neq \emptyset$ neprázdný?

Věta 10.7 *NEP je $PSPACE$-úplný problém.*

Důkaz: Zřejmě $NEP \in PSPACE$, protože vstupní slovo $\mathbf{x} \in \{0,1\}^\star$ neuromatu N lze díky $PSPACE = NPSPACE$ nedeterministicky hádat postupně po jednotlivých bitech a v polynomiálním prostoru lze simulací výpočtu N pro $\mathbf{x}$ ověřit, zda $\mathbf{x} \in L(N)$. Tvrzení, že NEP je $PSPACE$-těžký problém, dokážeme redukcí libovolného problému v $PSPACE$ na NEP v polynomiálním čase. Nechť tedy $A \in PSPACE$ je problém řešitelný Turingovým

strojem M v polynomiálním prostoru, tj. $A = L(M)$. Dále nechť $\mathbf{x} \in \{0,1\}^\star$ je instance A, tj. vstup pro M. Podobně jako v důkazu věty 8.11 zkonstruujeme v polynomiálním čase odpovídající neuromat $N_{\mathbf{x}}$, který simuluje výpočet M nad $\mathbf{x}$, tak, že $\mathbf{x} \in A$, právě když je neuron *out* na konci této simulace aktivní, tj. $L(N_{\mathbf{x}}) \neq \emptyset$, a tedy $N_{\mathbf{x}} \in NEP$. □

Je známo [141, 142], že problém prázdného jazyka pro konečné automaty i regulární výrazy je řešitelný nedeterministicky v logaritmickém prostoru (je dokonce NL-úplný), zatímco dle věty 10.7 je tento problém pro neuromaty díky $NL \subsetneq PSPACE$ [26] dokazatelně těžší, což také potvrzuje věta 10.6. Rozdíl v deskriptivní síle neuromatů a regulárních výrazů lze také ilustrovat na duálním problému, zda popisovaný jazyk L neobsahuje všechna slova, tj. zda $L \neq \{0,1\}^\star$. Tento problém je pro regulární výrazy $PSPACE$-úplný [2]. To ukazuje, že negace regulárního výrazu může způsobit exponenciální nárůst jeho délky, zatímco podle (i) z lemmy 6.27 lze u neuromatu jednoduše negovat booleovskou prahovou funkci výstupního neuronu *out* při zachování jeho velikosti.

Jako důsledek věty 10.7 je možné ukázat, že problém ekvivalence dvou neuromatů je $PSPACE$-úplný [268]. V tomto případě je doplňkový problém $co\text{-}NEP$, který je také $PSPACE$-úplný, redukován na problém ekvivalence dvou neuromatů tak, že druhý neuromat rozpoznává prázdný jazyk.

10.1.2 Neuronové akceptory binárních řetězců

V následujícím výkladu se zaměříme na speciální podtřídy regulárních výrazů a neuromatů. Nejjednodušším případem regulárních výrazů jsou binární řetězce, tj. odpovídající regulární jazyky obsahují jedno slovo. Pro rozpoznávání binárního řetězce lze zkonstruovat menší neuromat než v obecném případě (viz větu 10.5), který je optimální nejen vzhledem ke své velikosti, ale i z hlediska popisu vah.

Věta 10.8 (Wiedermann [267]) *Pro každý binární řetězec $\mathbf{b} \in \{0,1\}^n$ existuje jeho neuronový akceptor velikosti $O(\sqrt{n})$ a maximální váhy $O(2^{\sqrt{n}})$, který má $O(\sqrt{n})$ hran, a tedy na reprezentaci jeho vah stačí $\Theta(n)$ bitů.*

Důkaz: Uvedeme jen základní myšlenku důkazu, zatímco technické detaily lze najít v [268]. Vstupní slovo $\mathbf{x} \in \{0,1\}^n$ se postupně načítá do vyrovnávací paměti velikosti $p = \sqrt{n}$ neuronů. Při zaplnění této paměti, tj. načtení p bitů vstupu, dochází ke srovnání odpovídajících p bitů řetězce $\mathbf{b}$ a vstupu $\mathbf{x}$. Synchronizace je řízena pomocí hodin s p „minutovými“ neurony, z nichž je aktivní právě jeden neuron odpovídající příslušným p bitům řetězce $\mathbf{b}$, a s p „sekundovými“ neurony, které po každých p krocích (lze např. využít podobný cyklus jako v důkazu věty 8.8) aktualizují minutové neurony tak, že aktivují následující minutový neuron. Ten odpovídá aktuálním p bitům

řetězec $\mathbf{x} \in \{0,1\}^2$ *existuje přirozené číslo* $m_0 > 0$ *takové, že buď* $(\forall m \geq m_0 \ \mathbf{v}_1\mathbf{x}^m\mathbf{v}_2 \in L)$, *a nebo* $(\forall m \geq m_0 \ \mathbf{v}_1\mathbf{x}^m\mathbf{v}_2 \notin L)$.

Věta 10.11 (Šíma [264]) *Regulární jazyk je Hopfieldův, právě když splňuje Hopfieldovu podmínku. Speciálně pro každý regulární jazyk* $L = [\boldsymbol{\alpha}]$, *který splňuje Hopfieldovu podmínku a je zadaný regulárním výrazem* $\boldsymbol{\alpha}$, *lze zkonstruovat Hopfieldův neuromat* N *velikosti* $O(|\boldsymbol{\alpha}|)$, *který* $L = L(N)$ *rozpoznává.*

Důkaz: Nechť tedy nejprve $L = L(N)$ je Hopfieldův jazyk rozpoznávaný symetrickým neuronovým akceptorem N. Neuromatu N předložíme vstupní slovo $\mathbf{v}_1\mathbf{x}^{m_0}$ $(m_0 \geq 1)$, kde $\mathbf{v}_1 \in \{0,1\}^\star$ a $\mathbf{x} \in \{0,1\}^2$ jsou libovolné binární řetězce. Pro dostatečně velké m_0 Hopfieldova síť v čase $t^\star \geq 0$ vstoupí podle věty 8.16 v plně paralelním režimu do cyklu délky menší nebo rovno 2. V důkazu této věty totiž kopie $inp_1 \in V_1$ a $inp_2 \in V_2$ vstupního neuronu inp mají pak díky opakujícímu se vstupnímu 2-bitovému řetězci $\mathbf{x}$ konstantní stavy, tj. $\mathbf{x} = y^{(t)}_{inp_1}y^{(t)}_{inp_2}$ (resp. $\mathbf{x} = y^{(t)}_{inp_2}y^{(t)}_{inp_1}$) pro $t \geq t^\star$. Proto je lze nahradit případnou úpravou prahů $h'(j,1) = h(j) - y_{inp_2}w(inp,j)$ a $h'(j,2) = h(j) - y_{inp_1}w(inp,j)$ pro $j \in V$. Díky nejvýše dvoustavovému cyklu pro každé $\mathbf{v}_2 \in \{0,1\}^\star$ buď $(\forall m \geq m_0 \ \mathbf{v}_1\mathbf{x}^m\mathbf{v}_2 \in L)$, a nebo $(\forall m \geq m_0 \ \mathbf{v}_1\mathbf{x}^m\mathbf{v}_2 \notin L)$. Tedy L splňuje Hopfieldovu podmínku.

Obráceně nechť $L = [\boldsymbol{\alpha}]$ je regulární jazyk zadaný regulárním výrazem $\boldsymbol{\alpha}$ splňující Hopfieldovu podmínku. Podle věty 10.5 zkonstruujeme nejprve obecný neuromat N' velikosti $O(|\boldsymbol{\alpha}|)$, který rozpoznává $L = L(N')$, a ten budeme transformovat na ekvivalentní Hopfieldův neuromat. Pokud regulární výraz $\boldsymbol{\alpha}$ obsahuje jen iterace nejvýše 2-bitových binárních řetězců, pak podle důkazu této věty architektura neuromatu N' obsahuje cykly délky nejvýše 2 s jednotkovými váhami, tj. odpovídá prahovému obvodu, který může mít některé jednotkové zpětné vazby a příp. symetrické jednotkové spoje. V tomto případě lze Hopfieldův neuromat N bez problémů vytvořit z N' pomocí techniky z důkazu věty 8.19.

Hlavní problém tedy spočívá v realizaci obecných iterací pomocí symetrických spojů. Uvažujme podsíť I neuromatu N', která odpovídá pozitivní iteraci $\boldsymbol{\beta}^+$ (iterace $\boldsymbol{\beta}^\star$ lze převést na $e + \boldsymbol{\beta}^+$) nějakého podvýrazu $\boldsymbol{\beta}$ regulárního výrazu $\boldsymbol{\alpha}$. Po aplikaci věty 8.19 každá cesta vedoucí skrz I je ohodnocena klesající posloupností vah, které zamezí zpětnému šíření signálu. Avšak díky iteraci má být signál propagován z libovolného výstupu podsítě I zpět ke každému vstupu I. Na jednu stranu váha takového spoje musí být dostatečně malá, aby se zabránilo zpětnému šíření signálu. Na druhou stranu by tato váha měla být dostatečně velká, aby ovlivnila neuron na vstupu podsítě I. Tyto dva požadavky jsou zřejmě ve sporu.

Dále uvažujme jednoduchý cyklus C v podsíti I, který se skládá z orientované cesty procházející skrz I a jedné zpětné hrany vedoucí z konce této

řetězce $\mathbf{b}$, které se mají porovnat s příslušnými p bity vstupu $\mathbf{x}$. Vlastní porovnání probíhá pomocí dvou komparačních neuronů, k nimž jsou připojeny minutové neurony a neurony vyrovnávací paměti. Váha spoje od minutového k prvnímu komparačnímu neuronu představuje zápornou dekadickou hodnotu odpovídajícího binárního p-bitového podřetězce slova $\mathbf{b}$ a váhy od neuronů vyrovnávací paměti jsou mocniny dvojky od nuly až do řádu $p-1$, které převádí odpovídající binární p-bitový podřetězec vstupu $\mathbf{x}$ do dekadické soustavy. První komparační neuron s nulovým prahem je pak aktivní, právě když dekadické číslo příslušející k části vstupu $\mathbf{x}$ je větší nebo rovno dekadickému číslu reprezentující podřetězce slova $\mathbf{b}$. Podobně pracuje druhý komparační neuron s relací menší nebo rovno. Konjunkce těchto dvou komparačních neuronů zajistí správně porovnání $\mathbf{x}$ s $\mathbf{b}$. Zřejmě uvedený neuronový akceptor má $O(p)$ neuronů a spojů s maximální váhou $O(2^p)$. □

Piotr Indyk (osobní komunikace) navrhl využití neuromatu z věty 10.8 při konstrukci cyklické neuronové sítě s $O(2^{\frac{n}{2}})$ neurony a spoji, která počítá libovolnou booleovskou funkci n proměnných (srovnejte s větou 7.10, jejíž důkaz využívá $O(2^n)$ hran). V tomto případě je 2^n-bitový vektor funkčních hodnot zakódován do $2^{\frac{n}{2}}$ vah po $2^{\frac{n}{2}}$-bitových podřetězcích. Prvních $\frac{n}{2}$ bitů vstupu určuje výběr příslušné váhy (tj. odpovídajícího podřetězce vektoru funkčních hodnot). Dále jsou postupně generovány všechny $2^{\frac{n}{2}}$-bitové řetězce, dokud se nějaký z nich neshoduje s touto váhou, a z něho je extrahován bit v pozici, která se rovná dekadické hodnotě posledních $\frac{n}{2}$ bitů vstupu.

10.1.3 Hopfieldovy jazyky

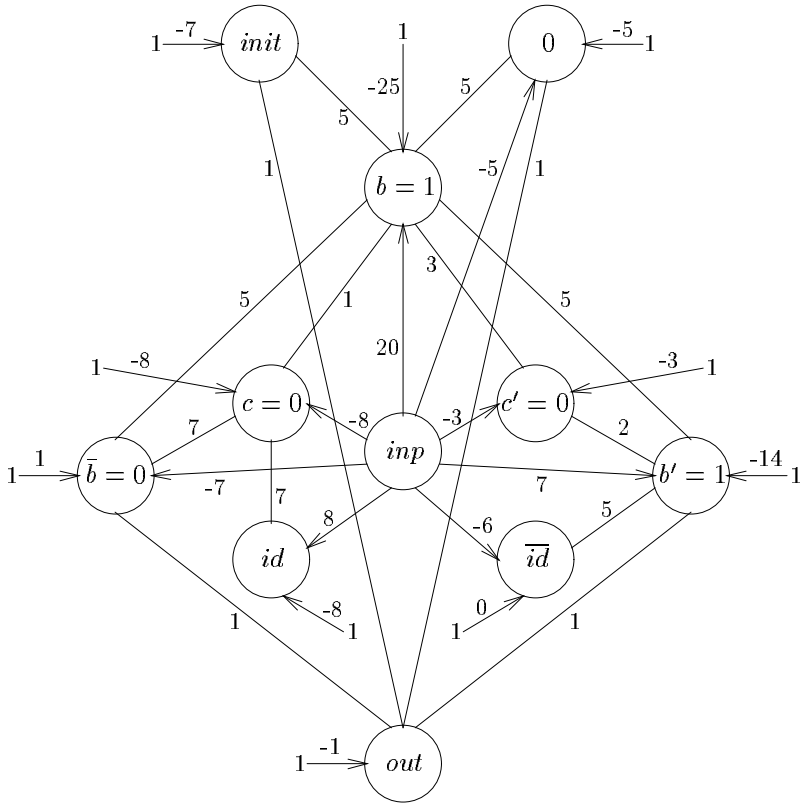
V závěru této podkapitoly se budeme zabývat speciální třídou tzv. *Hopfieldových neuromatů* se symetrickými spoji, která definuje *Hopfieldovy jazyky.*

Definice 10.9 Hopfieldův neuromat *(*symetrický neuronový akceptor*) je neuromat (viz definici 10.1) založený na Hopfieldově síti (viz definici 8.12). Jazyk rozpoznatelný symetrickým neuronovým akceptorem se nazývá* Hopfieldův jazyk.

Ralph Hartley a Harold Szu [94] si všimli, že Hopfieldovy neuromaty vzhledem k odlišným konvergenčním vlastnostem symetrických neuronových sítí (srovnejte věty 8.8 a 8.16) jsou slabším výpočetním prostředkem ve srovnání s konečnými automaty, resp. obecnými neuromaty, a proto nemohou rozpoznávat obecně všechny regulární jazyky. V následujícím výkladu charakterizujeme třídu Hopfieldových jazyků, která je vlastní podtřídou regulárních jazyků, pomocí tzv. *Hopfieldovy podmínky.*

Definice 10.10 *Řekneme, že regulární jazyk* L *splňuje* Hopfieldovu podmínku, *jestliže pro každý prefix a sufix* $\mathbf{v}_1, \mathbf{v}_2 \in \{0,1\}^\star$ *a 2-bitový*

Podobně celý proces transformace je proveden pro všechny iterace v regulárním výrazu $\boldsymbol{\alpha}$. V tomto případě některé iterace mohou být částí jiných iterací a velikost vah ve vnitřní iteraci vyžaduje úpravu vah tak, aby mohla být vtělena do vnější iterace. Je také možné, že neuron *id* musí posílit obě iterace ve stejném místě. Počet jednoduchých cyklů v regulárním výrazu $\boldsymbol{\alpha}$ je $O(|\boldsymbol{\alpha}|)$, proto velikost výsledného Hopfieldova neuromatu N zůstává řádu $O(|\boldsymbol{\alpha}|)$. Navíc $L = L(N)$ je Hopfieldův jazyk.



Obr. 10.2: Hopfieldův neuromat pro regulární jazyk $[(1(0+0(1+0)))^\star]$.

Na obrázku 10.2 je předchozí konstrukce ilustrována na příkladě Hopfieldova neuromatu rozpoznávajícího regulární jazyk $[(1(0+0(1+0)))^\star]$ (srovnejte s obecným neuromatem na obrázku 10.1). Podrobněji komentujeme jednoduchý cyklus neuronů $v_b, v_c, v_{\bar{b}}$. Všimněme si klesající posloupnosti vah

cesty (tj. z výstupu I) na její začátek (tj. ke vstupu I). Nechť typy neuronů v cyklu C odpovídají pozitivní iteraci $\mathbf{a}^+$ binárního řetězce $\mathbf{a} \in \{0,1\}^\star$, jehož délka $|\mathbf{a}| > 2$. Navíc nejprve pro spor předpokládejme, že $\mathbf{a} = \mathbf{x}^k$ pro nějaké $\mathbf{x} \in \{0,1\}^2$ a $k \geq 1$. V Hopfieldově podmínce nechť $\mathbf{v}_2$ je sufix slova z L odpovídající orientované cestě v N' z C do *out*. Podobně nechť $\mathbf{v}_1$ je prefix slova z L odpovídající orientované cestě v N' z *init* do C takový, že pro každé $m \geq 0$ platí $\mathbf{v}_1\mathbf{x}^{mk}\mathbf{v}_2 \in L$ a $\mathbf{v}_1\mathbf{x}^{mk+1}\mathbf{v}_2 \notin L$. Takový prefix $\mathbf{v}_1$ existuje, protože v opačném případě by bylo možné cyklus C realizovat 2-bitovou iterací. Avšak to je ve sporu s Hopfieldovou podmínkou, a proto $\mathbf{a} \neq \mathbf{x}^k$. Tedy řetězce $\mathbf{a}^p$ pro $p \geq 2$ obsahují podřetězec tvaru $bc\bar{b}$, kde $b, c, b \in \{0,1\}$ a $b \neq \bar{b}$. Odtud řetězec $\mathbf{a}$ má tvar buď $\mathbf{a} = \mathbf{a}_1 bc\bar{b}\mathbf{a}_2$ pro $\mathbf{a}_1, \mathbf{a}_2 \in \{0,1\}^\star$, nebo $\mathbf{a} = c\bar{b}\mathbf{a}_2 b$ (např. $\mathbf{a} = 10101$). Kvůli jednoduchosti zápisu se dále omezíme jen na první případ, zatímco druhá varianta je podobná. Tedy dále uvažujme $\mathbf{a} = \mathbf{a}_1 bc\bar{b}\mathbf{a}_2$ s minimální délkou $|\mathbf{a}_2|$.

Nejprve posuneme klesající posloupnost vah v C tak, aby začínala s největší váhou v neuronu v_c, jehož typ právě odpovídá c v $\mathbf{a}$. Za tímto účelem váhy v N' musí být opět pomocí techniky z důkazu věty 8.19 upraveny tak, aby zapojení C v rámci N' zůstalo konzistentní. Např. to znamená, že váha spojů z výstupu I v C do vstupů I je dostatečně veliká, aby realizovala odpovídající iterace. Nyní nám jde o to, jak realizovat propagaci signálu z neuronu v_b (typ v_b odpovídá b v $\mathbf{a}$) do neuronu v_c. Dále předpokládejme, že $b = 1$, zatímco pro $b = 0$ probíhá důkaz podobně. Abychom posílili malou váhu $w(v_b, v_c)$, přidáme k síti nový neuron *id*, který se zpožděním jednoho kroku kopíruje vstupní neuron *inp*, a volíme dostatečně velkou váhu $w(id, v_c)$. Zřejmě, neuron v_b, který kontroluje, zda je vstupní neuron *inp* aktivní ($b = 1$), a nový neuron *id*, který kopíruje vstup, jsou současně aktivní a oba díky součtu vah $w(v_b, v_c) + w(id, v_c)$ umožní propagaci signálu z v_b do v_c. Na druhou stranu ve chvíli, kdy je následující neuron $v_{\bar{b}}$ ($\bar{b} = 0$) aktivní, je neuron *id* pasivní, což zabrání zpětné aktivitě neuronu v_c.

Avšak neuron v_c může být kromě k $v_{\bar{b}}$ připojen i k neuronu $v_{b'}$ (typu b') v podsíti I vně cyklu C, který odpovídá některému symbolu $b' = b = 1$ v regulárním výrazu $\boldsymbol{\alpha}$. Tato situace ve výrazu $\boldsymbol{\beta}$ např. odpovídá symbolu c spojenému se sjednocením nějakých podvýrazů začínajících symbolem $b' = 1$ a $\bar{b} = 0$. V takovém případě by současná aktivita neuronů $v_{b'}$ a *id* způsobila nesprávnou zpětnou aktivitu neuronu v_c. Abychom tomu zabránili, vytvoříme kopii $v_{c'}$ neuronu v_c ($c' = c$), která se chová stejně jako neuron v_c. Tedy stejné neurony, které jsou připojeny k v_c, jsou připojeny k neuronu $v_{c'}$ a hrany, které původně vedly z v_c k $v_{b'}$, jsou pro všechna příslušná b' přepojeny tak, že vychází jen z $v_{c'}$.

Předchozí postup aplikujeme pro všechny jednoduché cykly C v podsíti I odpovídající pozitivní iteraci $\boldsymbol{\beta}^+$. Tyto cykly nejsou nutně disjunktní, ale dekompozice $\mathbf{a} = \mathbf{a}_1 bc\bar{b}\mathbf{a}_2$ s minimální $|\mathbf{a}_2|$ zajistí konzistentní syntézu.

$(7,5,1)$ v tomto cyklu, která začíná a končí v neuronu v_c. Dále neuron *id* umožňuje propagaci signálu z v_b do v_c. Navíc byla vytvořena kopie $v_{c'}$ neuronu v_c, protože neuron v_c byl původně připojen k neuronu $v_{b'}$ ($b' = b$). □

Z věty 10.11 vyplývá, že např. regulární jazyky [(000)*], [(1010)*], které nesplňují Hopfieldovu podmínku, nejsou Hopfieldovými jazyky. Podle věty 8.16 mají Hopfieldovy neuromaty během výpočtu tendenci konvergovat k nejvýše dvěma stavům s vyšší stabilitou (resp. nižší energií). Tyto dva stavy pak nestačí k zapamatování informace (i když konečné) o načtené části vstupu, která je potřebná ke správnému rozpoznání vstupního slova. Intuitivně lze z důkazu věty 10.11 říci, že potenciálně nekonečná vstupní slova Hopfieldových jazyků dodávají Hopfieldovým neuromatům energii (viz (3.29)), tj. porušují jejich stabilitu (viz (8.10)), a zvyšují tak počet dosažitelných stavů s vyšší energií, které kódují konečnou informaci o rozpoznávaném slově. Dalším důsledkem této věty jsou uzávěrové vlastnosti třídy Hopfieldových jazyků.

Důsledek 10.12 *Třída Hopfieldových jazyků je uzavřená na sjednocení, spojení, průnik a doplněk a není uzavřená vůči iteraci.*

Důkaz: Uzavřenost Hopfieldových jazyků vůči sjednocení a spojení je přímým důsledkem věty 10.11. Hopfieldův neuromat pro doplněk Hopfieldova jazyka získáme negací booleovské prahové funkce výstupního neuronu *out* pomocí (*i*) z lemmy 6.27. Z toho vyplývá, že třída Hopfieldových jazyků je také uzavřena na průnik. Nakonec jednoprvkový jazyk {000} je Hopfieldův, zatímco jeho iterace [(000)*] nesplňuje Hopfieldovu podmínku, a proto dle věty 10.11 není Hopfieldovým jazykem. □

Analogii věty 10.7, tj. $PSPACE$-úplnost problému prázdného jazyka (podobně i problému ekvivalence neuromatů), lze ukázat také pro Hopfieldovy neuromaty. Za tímto účelem se simuluje výpočet Turingova stroje s polynomiálním prostorem (který bez újmy na obecnosti vždy končí) pomocí Hopfieldovy sítě využitím technik z důkazů vět 8.11 a 8.21.

10.2 Konečné analogové neuronové sítě

V podkapitole 10.1 jsme formalizovali diskrétní konečný neuronový akceptor, který rozpoznává jen regulární jazyky. V této podkapitole zobecníme tento model pro spojité aktivační funkce. Získáme tak univerzální výpočetní prostředek. Za tímto účelem upravíme definici 8.2 výpočtu cyklické neuronové sítě tak, aby odpovídala aktivní dynamice *analogové neuronové sítě*. Také budeme uvažovat obecně reálné váhy a prahy a podle deskriptivní

složitosti těchto parametrů (racionální, reálné, kolmogorovská složitost) získáme různou výpočetní sílu *analogového neuronového akceptoru*. Na rozdíl od neuromatu (viz definici 10.1) budeme uvažovat nepatrně obecnější vstupní protokol, kde k vstupnímu a výstupnímu datovému neuronu zavedeme dva neurony, které budou signalizovat jejich platnost.

Definice 10.13 Analogová neuronová síť $N = (V, X, Y, \mathbf{y}^{(0)}_{V\setminus X}, w, h)$ *je obdobou diskrétní cyklické neuronové sítě (viz definici 8.1), kde místo množiny iniciálně aktivních neuronů je dán obecně reálný počáteční stav* $\mathbf{y}^{(0)}_{V\setminus X} \in \mathbb{R}^{s-n}$ *($s = |V|$, $n = |X|$) nevstupních neuronů, váhy $w : V \times V \longrightarrow \mathbb{R}$ a prahy $h : V \longrightarrow \mathbb{R}$ jsou obecně reálná čísla. Také výpočet analogové neuronové sítě je podobný jako u diskrétní cyklické neuronové sítě (viz definici 8.2), kde aktualizace binárního stavu neuronu $j \in \alpha_t \subseteq V$ podle (8.4) v čase $t \geq 1$ je upravena pro obecně reálný stav následujícím způsobem (srovnejte s (7.24)):*

$$y_j^{(t)} = \sigma\left(\sum_{i\in V} w_{ji} y_i^{(t-1)} - h_j\right), \tag{10.8}$$

kde $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ je spojitá aktivační funkce. Např. můžeme volit saturovanou lineární funkci (1.8):

$$\sigma(\xi) = \begin{cases} 0 & \xi < 0 \\ \xi & 0 \leq \xi \leq 1 \\ 1 & \xi > 1\,. \end{cases} \tag{10.9}$$

V této podkapitole, pokud neuvedeme explicitně jinak, budeme většinou z důvodů technické náročnosti používat nejjednodušší spojitou sigmoidní funkci (10.9). Navíc saturovaná lineární funkce se také často používá v aplikacích neuronových sítí [29, 175].

Definice 10.14 Analogový neuronový akceptor $N = (V, X, Y, w, h)$ *je obdobou neuromatu (viz definici 10.1), který je založen na analogové neuronové síti pracující v plně paralelním režimu, kde $X = \{inp, v_{inp}\}$, $Y = \{out, v_{out}\}$ a $\mathbf{y}^{(0)}_{V\setminus X} = \mathbf{0}$ představuje nulový stabilní počáteční stav. Rozpoznání jazyka $L \in \{0,1\}^+$ (pro jednoduchost neuvažujeme prázdná slova) pomocí analogového neuronového akceptoru probíhá následujícím způsobem. Nechť $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$ je vstupní slovo. Na začátku výpočtu se prostřednictvím neuronu inp načte vstup $\mathbf{x}$, jehož délka je zadána pomocí vstupního neuronu v_{inp}, a dále jsou oba vstupní neurony pasivní:*

$$y_{inp}^{(t)} = \begin{cases} x_{t+1} & t = 0, \ldots, n-1 \\ 0 & t \geq n \end{cases} \tag{10.10}$$

$$y_{v_{inp}}^{(t)} = \begin{cases} 1 & t = 0, \ldots, n-1 \\ 0 & t \geq n\,. \end{cases} \tag{10.11}$$

vstupního stavu, protože vstupní protokol z definice 10.14 lze pomocí jednoduché podsítě převést v čase $O(n)$ na vstupní stav.

Dále bez újmy na obecnosti předpokládáme, že Turingův stroj místo $k \geq 1$ pásek používá $p = 2k \geq 2$ zásobníků, kde každá dvojice zásobníků reprezentuje obsah jedné pásky rozdělený pozicí čtecí/zapisovací hlavy M na dvě části. Vstupní slovo je tedy na začátku v jednom ze zásobníků. Přechodová funkce M čte vrcholové prvky zásobníků a podle nich a stavu, ve kterém se nachází, s každým zásobníkem případně realizuje jednu ze zásobníkových operací a příp. změní svůj stav. Možné zásobníkové operace jsou odstranění vrcholového prvku a vložení nového prvku (1 nebo 0) na vrchol zásobníku. Je zřejmé, že k-páskový Turingův stroj lze simulovat pomocí Turingova stroje s p zásobníky v reálném čase.

Analogový neuronový akceptor N, který simuluje Turingův stroj M s p zásobníky, má pro každý zásobník tři neurony, které reprezentují obsah zásobníku, vrcholový prvek a informaci o tom, zda je zásobník prázdný. Řídící konečný automat lze implementovat podobně jako ve větě 10.3. Obsah zásobníku $b_1 \ldots b_q \in \{0,1\}^q$ (včetně vstupního slova na začátku) je kódován neuronovým racionálním stavem v intervalu $\langle 0,1) \cap \mathbb{Q}$ pomocí kódovací funkce (10.14):

$$\delta(b_1 \ldots b_q) = \sum_{i=1}^{q} \frac{2b_i + 1}{4^i} \,. \tag{10.16}$$

Tedy vrcholový prvek zásobníku odpovídá ve čtyřkové soustavě první platné číslici stavu příslušného neuronu tak, že bity 1, 0 jsou reprezentovány po řadě čtyřkovými ciframi 3, 1. Navíc předpokládáme $\delta(e) = 0$ pro prázdný zásobník. Tato funkce, na rozdíl např. od binárního kódu, umožňuje jednoduchou neuronovou implementaci zásobníkových operací v konstantním čase a dostatečnou rozlišitelnost různých stavů, protože ne každý neuronový stav z $\langle 0,1)$ reprezentuje obsah zásobníku, ale naopak obor hodnot funkce δ tvoří Cantorovu množinu. To znamená, že např. možné hodnoty (1 nebo 0) na vrcholu zásobníku vymezují obory hodnot funkce δ tak, že mezi nimi je „díra“, tj. $\delta(1\mathbf{b}') \in \langle \frac{3}{4}, 1)$ a $\delta(0\mathbf{b}') \in \langle \frac{1}{4}, \frac{1}{2})$ pro každé $\mathbf{b}' \in \{0,1\}^\star$. Následující prvky zásobníku dále omezují možné hodnoty kódu.

Nakonec ještě nastíníme neuronovou implementaci zásobníkových operací. Nechť $z \in \langle 0,1)$ je stav neuronu kódujícího obsah zásobníku Z. Operace načtení vrcholového prvku

$$top(z) = \sigma(4z - 2) \in \{0,1\} \tag{10.17}$$

lze realizovat pomocí spojité neuronové funkce (10.8), (10.9). Podobně můžeme vypočítat vložení prvku $b \in \{0,1\}$ na vrchol zásobníku Z:

$$push(z,b) = \frac{z}{4} + \frac{2b+1}{4} = \sigma\left(\frac{z}{4} + \frac{2b+1}{4}\right) \in \langle 0,1) \tag{10.18}$$

Podobně v případě končícího výpočtu neuron out v čase $t^\star$ při aktivitě v_{out} signalizuje, zda vstupní slovo $\mathbf{x}$ *patří do jazyka L:*

$$y_{out}^{(t)} = \begin{cases} 1 & \mathbf{x} \in L,\ t = t^\star \\ 0 & jinak \end{cases} \tag{10.12}$$

$$y_{v_{out}}^{(t)} = \begin{cases} 1 & t = t^\star \\ 0 & jinak\,. \end{cases} \tag{10.13}$$

Pokud neuron v_{out} není nikdy aktivní, výpočet akceptoru N neskončí. Tedy množina vstupních slov $L = L(N)$, pro které skončí výpočet s aktivním výstupním neuronem out, je jazyk rozpoznávaný *analogovým neuronovým akceptorem N. Navíc řekneme, že $L = L(N)$ je rozpoznáván v čase $T : \mathbb{N} \longrightarrow \mathbb{N}$, pokud všechny končící výpočty nad vstupem délky n skončí v čase $t^\star \leq T(n)$.*

10.2.1 Racionální váhy

Ukážeme, že konečné analogové neuronové sítě s racionálními váhami jsou z hlediska výpočetní síly i efektivity ekvivalentní Turingovým strojům. Nejprve však ještě navíc zavedeme vstupní protokol analogového neuronového akceptoru, který je alternativou k definici 10.14.

Definice 10.15 *Řekneme, že analogový neuronový akceptor $N = (V, inp, Y, w, h)$ rozpoznává jazyk $L(N)$* pomocí vstupního stavu, *jestliže vstupní slovo $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$ je kódováno racionálním iniciálním stavem vstupního neuronu inp pomocí* kódovací funkce $\delta : \{0,1\}^+ \longrightarrow \langle 0,1)$, *tj. $y_{inp}^{(0)} = \delta(\mathbf{x})$. Budeme používat následující kódovací funkce:*

$$\delta(\mathbf{x}) = \sum_{i=1}^{n} \frac{2x_i + 1}{4^i} \tag{10.14}$$

$$\delta_p(\mathbf{x}) = \sum_{i=1}^{n} \frac{10p^2 - 1 + 4p(x_i - 1)}{(10p^2)^i} \quad p \geq 2\,. \tag{10.15}$$

Věta 10.16 (Siegelmann, Sontag [255]) *Nechť $L = L(M) \subseteq \{0,1\}^\star$ je jazyk rozpoznávaný Turingovým strojem M v čase $T(n)$. Pak existuje analogový neuronový akceptor N konstantní velikosti, s racionálními váhami a prahy (a tedy i stavy), který rozpoznává jazyk $L = L(N)$ v čase $T(n) + O(n)$ a pomocí vstupního stavu s kódovací funkcí δ v čase $O(T(n))$, resp. s δ_p v reálném čase $T(n)$.*

Důkaz: Nastíníme jen základní myšlenky důkazu. Omezíme se jen na konstrukci analogového neuronového akceptoru, který rozpoznává L pomocí

s reálnými váhami a prahy v čase $O(T(n))$ a dále $CIRCUIT(S(n))$, resp. $T-CIRCUIT(S(n))$, třídu jazyků rozpoznatelných (neuniformní) posloupností logických, resp. prahových, obvodů (viz definici 7.19) velikosti $O(S(n))$. Navíc definujeme následující složitostní třídy jazyků:

$$NET-P = \bigcup_{k\geq 0} NET(n^k) \quad (10.20)$$

$$NET-EXP = \bigcup_{k\geq 0} NET(2^{kn}) \quad (10.21)$$

$$CIRCUIT-P = \bigcup_{k\geq 0} CIRCUIT(n^k) \quad (10.22)$$

$$CIRCUIT-EXP = \bigcup_{k\geq 0} CIRCUIT(2^{kn})\,. \quad (10.23)$$

Věta 10.20 (Siegelmann, Sontag [254]) *Nechť $T : \mathbb{N} \longrightarrow \mathbb{N}$, $S : \mathbb{N} \longrightarrow \mathbb{N}$ jsou funkce na přirozených číslech takové, že $T(n), S(n) \geq n$ pro každé přirozené číslo $n \in \mathbb{N}$. Potom pro každé $n \in \mathbb{N}$ platí následující inkluze:*

(i) $NET(T(n)) \subseteq CIRCUIT(T^3(n))$
$CIRCUIT(S(n)) \subseteq NET(nS^2(n))$.

(ii) $NET(T(n)) \subseteq T-CIRCUIT(T^2(n))$
$T-CIRCUIT(S(n)) \subseteq NET(nS^3(n)\log S(n))$.

Důkaz: Nastíníme jen základní myšlenky důkazu, zatímco technické detaily lze najít v [254].

(i) Nejprve nechť N je analogový neuronový akceptor konstantní velikosti s reálnými váhami a prahy, který v čase $T(n)$ rozpoznává jazyk $L = L(N) \subseteq \{0,1\}^+$. Pro každé přirozené číslo $n \in \mathbb{N}$ vytvoříme kopii N_n akceptoru N stejné velikosti, která bude mít jen racionální váhy (resp. prahy) vzniklé zaokrouhlením původních vah (resp. prahů) N na $O(T(n))$ platných bitů, její aktivační saturovaná lineární funkce (10.9) bude navíc také zaokrouhlovat svoji hodnotu na $T(n)$ platných bitů (tj. síť N_n bude mít pouze racionální stavy s přesností $T(n)$) a odpovídající výstup N_n se bude v čase $t \leq T(n)$ shodovat s výstupem N. Analogovou neuronovou síť N_n pak transformujeme na ekvivalentní logický obvod C_n pomocí Savageovy techniky [247] (viz důkaz věty 8.7 pro prahové obvody) tak, že spojitou funkci neuronu sítě N_n realizujeme pomocí speciálního logického podobvodu C'_n. Počet podobvodů C'_n v obvodu C_n bude jen $O(T(n))$, protože N_n má konstantní velikost. Dále podobvod C'_n bude počítat funkci (10.8) s $O(1)$ racionálními vstupy a váhami přesnosti $O(T(n))$ bitů pomocí analogie *(iii)* a odebrání prvku z vrcholu zásobníku Z:

$$pop(z) = 4z - (2top(z)+1) = \sigma(4z-(2top(z)+1)) \in \langle 0,1\rangle\,. \quad (10.19)$$

Při podrobnější analýze se ukazuje, že jeden krok výpočtu M vyžaduje 4 kroky simulace pomocí N, která využívá kódovací funkci δ. Simulaci v reálném čase lze dosáhnout pomocí kódovací funkce δ_p, která určuje specifické velikosti „děr" v Cantorově množině. Poměrně komplikované technické detaily uvedené konstrukce lze najít v [255]. □

Důsledek 10.17 (Siegelmann, Sontag [255]) *Každou parciálně rekurzivní funkci vyčíslitelnou na Turingově stroji v čase $T(n)$ lze počítat pomocí analogové neuronové sítě velikosti 886 neuronů s racionálními váhami v čase $O(T(n))$.*

Důkaz: Analogová neuronová síť zkonstruovaná podle důkazu věty 10.16, která simuluje univerzální Turingův stroj [250] v lineárním čase, se skládá z 886 neuronů [255]. □

Je zřejmé, že na druhou stranu pomocí Turingova stroje lze v polynomiálním čase simulovat analogovou neuronovou síť s racionálními váhami, tedy uvedené modely jsou z hlediska výpočetní síly a efektivity ekvivalentní. Dalším důsledkem této ekvivalence je algoritmická nerozhodnutelnost některých otázek jako např. problém zastavení analogové neuronové sítě (srovnejte se složitostí HNN pro diskrétní neuronové sítě ve větě 8.11). Zformulujeme bez důkazu příslušný důsledek, který využívá simulaci (univerzálního) Turingova stroje s časovou složitostí $T(n)$ pomocí analogové neuronové sítě velmi malé velikosti s racionálními váhami v čase $O(n^2T(n))$ [139].

Důsledek 10.18 (Indyk [139]) *Problém zastavení analogové neuronové sítě velikosti 25 neuronů s racionálními váhami pro daný vstup (s podobným vstupním protokolem jako v definici 10.14) je algoritmicky nerozhodnutelný problém.*

10.2.2 Reálné váhy

V tomto odstavci budeme uvažovat analogové neuronové akceptory s reálnými váhami a prahy a ukážeme, že jsou výpočetně silnější než Turingovy stroje. Pro přesnější charakterizaci jejich efektivity je porovnáme s neuniformními posloupnostmi obvodů (viz odstavec 7.1.4) a dokážeme analogii věty 8.7 pro analogové neuronové sítě. Nejprve definujeme příslušné složitostní třídy problémů.

Definice 10.19 *Nechť $T : \mathbb{N} \longrightarrow \mathbb{N}$, $S : \mathbb{N} \longrightarrow \mathbb{N}$ jsou funkce na přirozených číslech. Označme $NET(T(n))$ třídu jazyků rozpoznatelných analogovými neuronovými akceptory (viz definici 10.14) konstantní velikosti,*

z věty 7.46 (sčítáme $O(T(n))$ $O(T(n))$-bitových čísel) a s využitím analogie věty 7.17 pro alternující obvody (tj. v důkazu věty 7.17 obvod C_2 je stejný jako C_2^A). Tedy C'_n má $O(T^2(n))$ hradel a velikost C_n je $O(T^3(n))$. Posloupnost $\mathbf{C} = (C_0, C_1, C_2, \ldots)$ logických obvodů velikosti $O(T^3(n))$ rozpoznává jazyk $L = L(\mathbf{C})$.

Na druhou stranu nechť $\mathbf{C} = (C_0, C_1, C_2, \ldots)$ je neuniformní posloupnost logických obvodů velikosti $S(n)$ rozpoznávající jazyk $L = L(\mathbf{C}) \subseteq \{0,1\}^+$. Zkonstruujeme analogový neuronový akceptor N konstantní velikosti a s reálnými váhami, který rozpoznává stejný jazyk $L = L(N)$. Nejprve zakódujeme posloupnost $\mathbf{C}$ pomocí reálného čísla $code(\mathbf{C}) \in (0,1)$ tak, že využijeme sudé cifry devítkové soustavy, aby kódy všech možných těchto posloupností obvodů tvořily podobně jako v důkazu věty 10.16 Cantorovu množinu, a tedy byly od sebe dostatečně vzdálené. Jeden obvod C_n velikosti $O(S(n))$ hradel lze díky možnému kvadratickému počtu jejich spojů kódovat pomocí $O(S^2(n))$ devítkových cifer. Tyto kódy $code(C_n)$ zřetězené v pořadí podle n tvoří kód $code(\mathbf{C})$. Neuronový akceptor N zahrnuje analogovou podsíť 16 neuronů, která obsahuje reálnou váhu $code(\mathbf{C})$ a na základě délky n vstupního slova (tj. aktivity vstupního neuronu v_{inp}) vygeneruje kód $code(C_n) \in (0,1) \cap \mathbb{Q}$ příslušného logického obvodu C_n jako racionální stav nějakého svého neuronu. Čas potřebný k odseparování kódu $code(C_n)$ z kódu $code(\mathbf{C})$ je úměrný součtu délek předcházejících kódů $code(C_k)$ pro $k < n$, tj. $O(nS^2(n))$. Další podsíť pak vyhodnotí výstup C_n pro vstupní slovo zakódované pomocí jiné podsítě ze vstupního neuronu inp do racionálního stavu nějakého neuronu. Tuto podsíť s méně než 1000 neurony lze např. podle věty 10.16 zkonstruovat z třípáskového Turingova stroje, který vyhodnotí příslušný obvod v čase $O(|code(C_n)| + n) = O(S^2(n))$. Výsledný analogový neuronový akceptor N tedy pracuje v čase $O(nS^2(n))$.

(ii) Důkaz probíhá podobně jako v části (i), avšak místo logických obvodů pracujeme s prahovými obvody. Při konstrukci prahové analogie podobvodu C'_n použijeme větu 7.33. Obráceně při kódování prahového obvodu C_n reprezentace váhy podle důsledku 6.39 představuje $O(S(n)\log S(n))$ bitů, tj. $|code(C_n)| = O(S^3(n)\log S(n))$. □

Kvůli charakterizaci výpočetní síly analogových neuronových sítí s reálnými váhami připomeneme některé pojmy z teorie neuniformní složitosti [26].

Definice 10.21 *Nechť $\mathcal{L} \subseteq \mathcal{P}(\{0,1\}^\star)$ je třída jazyků a $\mathcal{F} \subseteq \{f : \mathbb{N} \longrightarrow \{0,1\}^\star\}$ je třída* poradních *(advice) funkcí. Označme např.*

$$poly = \{f : \mathbb{N} \longrightarrow \{0,1\}^\star \mid (\exists polynom\ p)(\forall n)\ |f(n)| \leq p(n)\}$$
$$log = \{f : \mathbb{N} \longrightarrow \{0,1\}^\star \mid (\exists c > 0)(\forall n)\ |f(n)| \leq c \cdot \log n\}$$

třídy polynomiálních a logaritmických poradních funkcí. Také budeme někdy hodnoty poradních funkcí formálně chápat jako přirozená čísla, která vzniknou interpretací binárních řetězců z $\{0,1\}^\star$ jako čísel ve dvojkové soustavě. Definujeme třídu jazyků

$$\mathcal{L}/\mathcal{F} = \{\{\mathbf{x} \in \{0,1\}^\star \mid (\mathbf{x}, f(|\mathbf{x}|)) \in L\} \mid L \in \mathcal{L},\ f \in \mathcal{F}\}\ . \qquad (10.24)$$

Podobně se definuje třída jazyků $Pref - \mathcal{L}/\mathcal{F}$, kde navíc pro poradní funkce platí, že pro každé $n \leq m$ je $f(n)$ prefixem $f(m)$. Dále řekneme, že množina $Q \subseteq \{0,1\}^\star$ je řídká*, jestliže existuje polynom $p(n)$ takový, že počet slov v Q délky nejvýše n je nejvýše $p(n)$, tj. pro každé přirozené číslo n je $|\{\mathbf{x} \in Q \mid |\mathbf{x}| \leq n\}| \leq p(n)$.*

Poradní funkce mají stejnou úlohu jako orákula u Turingových strojů, avšak jejich hodnota závisí jen na délce vstupu. Je také například známo, že $Pref - P/poly = P/poly$ a $Pref - P/log = P/log$ [28, 108]. Následující věta formuluje známou souvislost neuniformních posloupností obvodů a turingovských výpočtů s poradními funkcemi.

Věta 10.22 (Pippenger [224]) *Označme $P(Q)$ třídu jazyků rozpoznatelných Turingovým strojem v polynomiálním čase s orákulem Q. Pak*

$$P/poly = \bigcup_{Q \text{ řídká}} P(Q) = CIRCUIT - P\,. \qquad (10.25)$$

Nyní již zformulujeme důsledek, který ukazuje, že analogové neuronové akceptory s reálnými váhami rozpoznávají v polynomiálním čase právě jazyky z třídy $P/poly$, tj. problémy, které řeší Turingův stroj v polynomiálním čase s polynomiální poradní funkcí, resp. s řídkým orákulem. Pokud připustíme exponenciální čas, pak analogové neuronové sítě s reálnými váhami rozpoznávají již všechny jazyky.

Důsledek 10.23 (Siegelmann, Sontag [254])

(i) $NET - P = CIRCUIT - P = P/poly = \bigcup_{Q \text{ řídká}} P(Q)$.

(ii) $NET - EXP = CIRCUIT - EXP = \mathcal{P}(\{0,1\}^\star)$.

Důkaz:

(i) Tvrzení je přímým důsledkem vět 10.20 a 10.22.

(ii) Tvrzení je přímým důsledkem vět 10.20 a 7.7. □

V důkazu věty 10.20 využíváme při konstrukci analogové neuronové sítě pouze jednu reálnou váhu a ostatní váhy jsou racionální. Na druhou stranu vyžadujeme nekonečnou přesnost váhy, což je prakticky nesplnitelný předpoklad, díky kterému podle důsledku 10.23 analogové neuronové sítě představují silnější výpočetní prostředek než jsou Turingovy stroje, které nerozpoznávají všechny jazyky. Nicméně např. ve fyzice dynamické chování systému pružin a závaží závisí na pevnostních a třecích konstantách, jejichž zaokrouhlení na libovolný počet míst může za dostatečně dlouhý čas ovlivnit dynamiku uvedeného systému. Nelze vyloučit, že kvantové jevy v biologických nervových systémech vykazují podobné vlastnosti.

Na závěr tohoto odstavce ještě uvedeme bez důkazu větu, která dosvědčuje, že také analogové neuronové sítě s diferencovatelnou aktivační funkcí hyperbolickým tangens (1.10), s níž je manipulace technicky náročnější než se saturovanou lineární funkcí (10.8), simulují Turingovy stroje.

Věta 10.24 (Kilian, Siegelmann [148]) *Existuje analogová neuronová síť $N = (V, \emptyset, out, \mathbf{y}_V^{(0)}, w, h)$ velikosti s, s reálnými váhami a aktivační funkcí hyperbolickým tangens:*

$$\sigma(\xi) = \frac{2}{1+e^{-\xi}} - 1 = \frac{1-e^{-\xi}}{1+e^{-\xi}}\,, \tag{10.26}$$

která nemá vstupy a má jeden výstup $out \in V$, a dále existuje rekurzivní kódovací funkce $\delta(M, \mathbf{x}) \in \mathbb{R}^s$ pro libovolný Turingův stroj M a jeho vstup $\mathbf{x}$ taková, že M se zastaví na vstup $\mathbf{x}$, právě když stav výstupního neuronu $y_{out}^{(t^\star)} > \frac{3}{4}$ v nějakém čase $t^\star$ výpočtu sítě N s počátečním stavem $\mathbf{y}_V^{(0)} = \delta(M, \mathbf{x})$. Naopak výpočet M na vstup $\mathbf{x}$ neskončí, právě když stav výstupního neuronu $y_{out}^{(t)} < \frac{1}{4}$ v každém čase t výpočtu sítě N s počátečním stavem $\mathbf{y}_V^{(0)} = \delta(M, \mathbf{x})$.

10.2.3 Kolmogorovská složitost vah

V tomto odstavci se omezíme na polynomiální výpočty analogových neuronových akceptorů. Podle věty 10.16 neuronové sítě s racionálními váhami pak rozpoznávají právě jazyky z třídy P a podle důsledku 10.23 sítě s reálnými váhami řeší problémy z třídy $P/poly$. Ukážeme, že mezi P a $P/poly$ je hierarchie tříd složitosti, kterou lze přirozeně definovat tak, že klademe jistá omezení na množství informace zakódované v reálných váhách analogových neuronových akceptorů pracujících v polynomiálním čase. Množství informace ve váhových parametrech sítě budeme měřit pomocí *kolmogorovské složitosti* [172]. Intuitivně kolmogorovská složitost nějakého objektu odpovídá deskriptivní složitosti programu, který tento objekt vygeneruje, je-li

vstupem např. nějakého univerzálního Turingova stroje. Budeme používat následující definici kolmogorovské složitosti (nekonečných) binárních řetězců.

Definice 10.25 *Nechť U je nějaký pevný* univerzální Turingův stroj, *který pro kód $code(M) \in \{0,1\}^\star$ libovolného Turingova stroje M a jeho vstup $\mathbf{x}$ simuluje výpočet M nad $\mathbf{x}$ s výstupem $U(code(M), \mathbf{x})$. Nechť dále $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ jsou funkce na přirozených číslech. Řekneme, že (nekonečný) binární řetězec $\boldsymbol{\alpha} \in \{0,1\}^\infty \cup \{0,1\}^\star$ má* kolmogorovskou složitost *$\boldsymbol{\alpha} \in K[f(n), g(n)]$, jestliže existuje nekonečný binární řetězec $\boldsymbol{\beta} \in \{0,1\}^\infty$ (kód programu) takový, že pro nekonečně mnoho $n \in \mathbb{N}$ a pro všechna $m \geq f(n)$ univerzální Turingův stroj U v čase $g(n)$ vypočte prvních n bitů $\boldsymbol{\alpha}$ pomocí prvních m bitů $\boldsymbol{\beta}$, tj. $U(\beta_1 \ldots \beta_m, n) = \alpha_1 \ldots \alpha_n$. Podobně se definuje kolmogorovská složitost $K[f(n)]$, avšak bez omezení na čas výpočtu U. Dále definujeme odpovídající složitostní třídy (nekonečných) binárních řetězců*

$$K[\mathcal{F}, \mathcal{G}] = \bigcup_{f \in \mathcal{F}, g \in \mathcal{G}} K[f, g] \tag{10.27}$$

$$K[\mathcal{F}] = \bigcup_{f \in \mathcal{F}} K[f] \tag{10.28}$$

pro třídy funkcí $\mathcal{F}, \mathcal{G} \subseteq \{f : \mathbb{N} \longrightarrow \mathbb{N}\}$.

Chápeme-li v souladu s definicí 10.21 *poly* a *log* jako třídy funkcí na přirozených číslech, pak např. $K[log, poly]$ je třída (nekonečných) binárních řetězců, jejichž prefixy lze v polynomiálním čase vygenerovat z logaritmicky dlouhých prefixů jiných nekonečných binárních řetězců. Nebo $K[n + O(1), poly] = \{0,1\}^\infty \cup \{0,1\}^\star$ jsou již všechny binární řetězce, protože vstup pro univerzální Turingův stroj U může obsahovat kopírovací program konstantní délky $O(1)$. Dále např. $K[n - O(1)]$ představuje třídu tzv. *kolmogorovských náhodných posloupností*, které již nelze kompresovat více než o konstantní počet bitů, nebo $K[O(1)]$ je třída rekurzivních binárních řetězců. Definici 10.25 využijeme při definici kolmogorovské složitosti vah analogového neuronového akceptoru

Definice 10.26 *Nechť $\delta : \{0,1\}^\star \cup \{0,1\}^\infty \longrightarrow \langle 0,1\rangle$ je kódovací funkce (srovnejte s (10.14)), která (příp. nekonečnému) binárnímu řetězci $\boldsymbol{\alpha} \in \{0,1\}^\star \cup \{0,1\}^\infty$ přiřadí reálné číslo z intervalu $\langle 0,1\rangle$ např. pomocí následujícího předpisu:*

$$\delta(\boldsymbol{\alpha}) = \sum_{i=1}^{|\boldsymbol{\alpha}|} \frac{\alpha_i}{2^i}\,. \tag{10.29}$$

Zřejmě zobrazení δ není prosté, nicméně definujeme inverzní zobrazení $\delta_0^{-1} : \langle 0,1\rangle \longrightarrow \{0,1\}^\star \cup \{0,1\}^\infty$, tj. binární reprezentaci reálného čísla $x \in \langle 0,1\rangle$

charakterizuje třídu $NET-P(K[log, poly]) = Pref-P/log = P/log$ jazyků rozpoznatelných v polynomiálním čase analogovými neuronovými akceptory s váhami logaritmické kolmogorovské složitosti (s polynomiálním omezením na čas jejich generování) pomocí známé složitostní třídy P/log.

Věta 10.29 (Balcázar, Gavaldà, Siegelmann [27]) *Nechť $\mathcal{F}, \mathcal{G} \subseteq \{f : \mathbb{N} \longrightarrow \mathbb{N}\}$ jsou třídy funkcí uzavřené na $O(\cdot)$ a $\mathcal{F} < \mathcal{G}$. Pak*

$$NET-P(K[\mathcal{F}, poly]) \subsetneqq NET-P(K[\mathcal{G}, poly]) . \tag{10.31}$$

Věta 10.29 dosvědčuje existenci hierarchie složitostních tříd mezi P a $P/poly$ definovanou polynomiálními výpočty analogových neuronových akceptorů s váhami dané kolmogorovské složitosti. Například můžeme uvažovat posloupnost tříd funkcí $\mathcal{F}_k = O(f_k)$, kde $f_1 = \log$ a $f_{k+1} = \log f_k$ pro $k \geq 1$, pro kterou lze snadno ukázat, že $\mathcal{F}_{k+1} < \mathcal{F}_k$. Dostaneme tak hierarchii složitostních tříd $NET-P(K[\mathcal{F}_{k+1}, poly]) \subsetneqq NET-P(K[\mathcal{F}_k, poly])$ $(k \geq 1)$, v níž jsou všechny inkluze ostré.

10.3 Posloupnosti neuronových sítí

V předchozích dvou podkapitolách jsme uvažovali diskrétní a analogové konečné neuronové sítě. V této podkapitole budeme v analogii s posloupnostmi obvodů (viz odstavec 7.1.4) uvažovat obecně neuniformní nekonečné posloupnosti diskrétních cyklických neuronových sítí, kde každé délce vstupu odpovídá právě jedna síť s příslušným počtem vstupních neuronů. Posloupnosti cyklických neuronových sítí budeme využívat k rozpoznávání jazyků. Také definujeme příslušné složitostní třídy pro polynomiální velikost cyklických sítí, resp. Hopfieldových sítí (popř. s malými váhami).

Definice 10.30 *Definujeme nekonečnou* posloupnost cyklických neuronových sítí $\mathbf{N} = (N_0, N_1, N_2, \ldots)$, *v níž neuronová síť $N_n = (V_n, X_n, Y_n, A_n, w_n, h_n)$ $(n \geq 0)$ má $|X_n| = n$ vstupů.* Velikost *posloupnosti* $\mathbf{N}$ *je taková funkce $S : \mathbb{N} \longrightarrow \mathbb{N}$ na přirozených číslech, že velikost sítě N_n v této posloupnosti je pro každé $n \geq 0$ menší nebo rovna $S(n)$. Řekneme, že posloupnost cyklických neuronových sítí s jedním výstupem, tj. $|Y_n| = 1$ pro $n \geq 0$,* rozpoznává jazyk $L = L(\mathbf{N}) \subseteq \{0,1\}^\star$, *jestliže pro každý vstup $\mathbf{x} \in \{0,1\}^n$ síť N_n konverguje v plně paralelním režimu s výstupem 1, právě když $\mathbf{x} \in L$. Označme $PNETS$ třídu jazyků rozpoznatelných posloupnostmi cyklických neuronových sítí polynomiální velikosti $S(n) = O(n^c)$, kde c je konstanta, dále označme $PHNETS$, resp. $PHNETSW$, třídu jazyků rozpoznatelných posloupnostmi Hopfieldových sítí polynomiální velikosti, resp. s malými váhami.*

tak, že vybereme např. libovolný prvek $\delta_0^{-1}(x) \in \delta^{-1}(x)$ ze vzoru $\delta^{-1}(x)$, protože prvky $\delta^{-1}(x)$ mají (až na malou aditivní konstantu) stejnou kolmogorovskou složitost. Řekneme, že reálná váha $w \in \mathbb{R}$ má kolmogorovskou složitost $w \in K[f(n), g(n)]$, jestliže binární reprezentace její desetinné části $w' \in \langle 0, 1)$ má kolmogorovskou složitost $\delta_0^{-1}(w') \in K[f(n), g(n)]$. Dále pro třídu funkcí $\mathcal{F} \subseteq \{f : \mathbb{N} \longrightarrow \mathbb{N}\}$ definujeme třídu $NET-P(K[\mathcal{F}, poly])$ jazyků rozpoznatelných analogovými neuronovými akceptory s váhami kolmogorovské složitosti $K[\mathcal{F}, poly]$ v polynomiálním čase.

Před tím, než zformulujeme bez důkazu příslušné věty ohledně polynomiálních výpočtů analogových neuronových akceptorů s reálnými váhami dané kolmogorovské složitosti, definujeme ještě požadované vlastnosti tříd funkcí na přirozených číslech a jejich hierarchie.

Definice 10.27 *Nechť $\mathcal{F}, \mathcal{G} \subseteq \{f : \mathbb{N} \longrightarrow \mathbb{N}\}$ jsou třídy funkcí na přirozených číslech. Řekneme, že $\mathcal{F}$ je třída* rozumných *poradních funkcí, jestliže splňuje následující tři podmínky:*

1. *$\mathcal{F} \subseteq poly$.*
2. *$\mathcal{F}$ je* uzavřená na $O(\cdot)$, *tj. pro libovolné funkce $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ když $g = O(f)$ a $f \in \mathcal{F}$, pak $g \in \mathcal{F}$.*
3. *Pro každý polynom p a libovolnou funkci $f \in \mathcal{F}$ existuje $g \in \mathcal{F}$, jejíž hodnotu $g(n)$ lze počítat Turingovým strojem v polynomiálním čase vzhledem k n a platí, že složená funkce $f \circ p \leq g$.*

Dále definujeme hierarchii tříd funkcí tak, že $\mathcal{F} < \mathcal{G}$, jestliže existuje funkce $g \in \mathcal{G}$ vyčíslitelná v polynomiálním čase a $g = o(n)$ taková, že pro každý polynom p a libovolnou funkci $f \in \mathcal{F}$ platí $f \circ p(n) = o(g(n))$. Připomeňme, že $f = o(g)$, právě když pro každé $\varepsilon > 0$ existuje n_0 tak, že pro všechna $n \geq n_0$ platí $f(n) < \varepsilon g(n)$.

Věta 10.28 (Balcázar, Gavaldà, Siegelmann [27]) *Nechť $\mathcal{F} \subseteq \{f : \mathbb{N} \longrightarrow \mathbb{N}\}$ je třída rozumných poradních funkcí, pak*

$$NET-P(K[\mathcal{F}, poly]) = Pref-P/\mathcal{F} . \tag{10.30}$$

Dá se např. snadno ukázat, že *poly* (podobně *log*) tvoří třídu rozumných poradních funkcí. Proto důsledek 10.23 je speciálním případem věty 10.28, tj. $NET-P(K[poly, poly]) = Pref-P/poly = P/poly = NET-P$, protože kolmogorovská třída složitosti $K[poly, poly]$ již zahrnuje všechny reálné váhy. Podobně věta 10.16 v případě polynomiálních výpočtů je také speciálním případem věty 10.28, tj. $NET-P(K[O(1), poly]) = P$, protože racionální váhy mají konstantní kolmogorovskou složitost. Věta 10.28 ale také např.

Pro $P/poly \subseteq PHNETSW$ konstruujeme Hopfieldovu síť $N_{|\mathbf{x}|}$, která simuluje Turingův stroj M pracující v polynomiálním čase $T(n) = O(n^c)$ (c je konstanta) pro vstup $(\mathbf{x}, f(|\mathbf{x}|))$ polynomiální délky. Podobně jako v důkazu věty 10.31 nejprve pomocí techniky z důkazu věty 8.11 vytvoříme odpovídající obecnou cyklickou neuronovou $N'_{|\mathbf{x}|}$ síť polynomiální velikosti s, jejíž váha je omezená nějakou konstantou W závisející na M a která se zastaví v polynomiálním čase $O(T(n))$. Ze sítě $N'_{|\mathbf{x}|}$ vytvoříme pomocí věty 8.21 $O(t)$-ekvivalentní Hopfieldovu síť $N_{|\mathbf{x}|}$ velikosti $O(s^2)$. Z důkazu této věty maximální váha podsítě N_T je konstantní $O(W) = O(1)$, a proto maximální váha $N_{|\mathbf{x}|}$ je dána maximální váhou použitého čítače. V tomto případě díky polynomiálnímu výpočtu $N'_{|\mathbf{x}|}$ můžeme použít čítač s $O(\log T(n))$ bity a podle důkazu věty 8.18 maximální váha $O(\log T(n))$-bitového čítače je polynomiální $2^{O(\log T(n))} = n^{O(c)}$. Tedy $N_{|\mathbf{x}|}$ má malé váhy. □

V následujícím výkladu dokážeme věty, které charakterizují výpočetní sílu posloupností cyklických neuronových sítí tak, že porovnávají příslušné složitostní třídy z definice 10.30 se známými třídami z teorie neuniformní složitosti (viz definici 10.21).

Věta 10.31 (Lepley, Miller [169]) $PNETS = PSPACE/poly$.

Důkaz: Nejdříve nechť $L \in PNETS$, tj. máme posloupnost $\mathbf{N} = (N_0, N_1, N_2, \ldots)$ cyklických neuronových sítí polynomiální velikosti, která rozpoznává $L = L(\mathbf{N})$. Zřejmě existuje Turingův stroj M, který pro vstup $(\mathbf{x}, code(N_{|\mathbf{x}|}))$ simuluje výpočet $N_{|\mathbf{x}|}$ na vstup $\mathbf{x}$ v lineárním prostoru. Označme jazyk $A = L(M) \in PSPACE$. Dále definujeme poradní funkci $f : \mathbb{N} \longrightarrow \{0,1\}^\star$, jejíž hodnota je $f(n) = code(N_n)$ pro $n \in \mathbb{N}$. Polynomiální velikost $\mathbf{N}$ implikuje $f \in poly$. Tedy $L = \{\mathbf{x} \in \{0,1\}^\star \mid (\mathbf{x}, f(|\mathbf{x}|)) \in A\} \in PSPACE/poly$. Z toho vyplývá, že $PNETS \subseteq PSPACE/poly$.

Nechť tedy naopak $L = \{\mathbf{x} \in \{0,1\}^\star \mid (\mathbf{x}, f(|\mathbf{x}|)) \in A\} \in PSPACE/poly$ pro nějaký jazyk $A = L(M) \in PSPACE$ rozpoznávaný Turingovým strojem M v polynomiálním prostoru a poradní funkci $f \in poly$. Pomocí techniky z důkazu věty 8.11 zkonstruujeme cyklickou neuronovou síť $N_{|\mathbf{x}|}$ polynomiální velikosti, která simuluje výpočet Turingova stroje M s polynomiálním prostorem pro vstup $(\mathbf{x}, f(|\mathbf{x}|))$, tak, že hodnotu $f(|\mathbf{x}|)$ polynomiální délky např. zakódujeme do množiny iniciálních neuronů sítě $N_{|\mathbf{x}|}$. Získáme tak obecně neuniformní posloupnost $\mathbf{N} = (N_0, N_1, N_2, \ldots)$ cyklických sítí polynomiální velikosti, která rozpoznává $L = L(\mathbf{N}) \in PNETS$. Tedy $PSPACE/poly \subseteq PNETS$. □

Věta 10.32 (Orponen [209])

(i) $PHNETS = PSPACE/poly$.

(ii) $PHNETSW = P/poly$.

Důkaz:

(i) Důkaz tvrzení je obdobou důkazu věty 10.31. Při simulaci Turingova stroje M s polynomiálním prostorem pro vstup $(\mathbf{x}, f(|\mathbf{x}|))$ pomocí Hopfieldovy sítě $N_{|\mathbf{x}|}$ navíc bez újmy na obecnosti předpokládáme, že M se zastaví pro každý vstup, a využijeme větu 8.21.

(ii) Důkaz tvrzení je podobný důkazu (i). Pro $PHNETSW \subseteq P/poly$ simulaci plně paralelního výpočtu Hopfieldovy sítě $N_{|\mathbf{x}|}$ polynomiální velikosti, s malými váhami na vstup $\mathbf{x}$ lze pomocí Turingova stroje realizovat v polynomiálním čase, protože výpočet $N_{|\mathbf{x}|}$ vstoupí podle věty 8.16 v polynomiálním čase do cyklu délky nejvýše 2.

tj. nezáleží na nich. Řekneme, že výstup $\mathbf{y} = C_\ell^A(\mathbf{x}) \in \{0,1\}^m$ *obvodu pro vstup* $\mathbf{x} \in \{0,1\}^n$ souhlasí *(*je konzistentní*) s požadovaným výstupem* $\mathbf{d} \in \{0,1,\star\}^m$*, a značíme* $\mathbf{y} \models \mathbf{d}$*, jestliže pro všechna* $j = 1,\ldots,m$ *požadovaný binární bit* $d_j \in \{0,1\}$ *implikuje* $y_j = d_j$*. Podobně funkce* C_ℓ^A *obvodu* C *je konzistentní s tréninkovou množinou* $\mathcal{T}$*, značíme* $C_\ell^A \models \mathcal{T}$*, jestliže pro všechna* $(\mathbf{x},\mathbf{d}) \in \mathcal{T}$ *platí* $C_\ell^A(\mathbf{x}) \models \mathbf{d}$*.*

Tréninkový problém LP (**L**oading **P**roblem [143]):
instance: Architektura obvodu A; tréninková množina $\mathcal{T}$ pro A.
výstup (vyhledávací verze): Konfigurace ℓ taková, že $C_\ell^A \models \mathcal{T}$.
otázka (rozhodovací verze): Existuje konfigurace ℓ taková, že $C_\ell^A \models \mathcal{T}$?

Je zřejmé, že vyhledávací verze LP je nejméně tak těžká jako jeho rozhodovací verze, protože algoritmus, který hledá příslušnou konzistentní konfiguraci také rozhoduje o její existenci. Naším cílem je ukázat, že LP je obecně algoritmicky těžký problém, a pro tento účel je tedy postačující omezit se na rozhodovací verzi LP. Nejprve zpřesníme definici rozhodovací verze LP pro danou třídu hradlových funkcí tak, že formálně definujeme jazyk $Perf_{\mathcal{F}}$ (**Perf**ormability).

Definice 11.2 *Nechť* $\mathcal{F}$ *je třída hradlových funkcí. Dále označme* $A = (V,X,Y,E)$ *architekturu,* ℓ *konfiguraci a* $\mathcal{T}$ *tréninkovou množinu. Definujeme následující jazyk (problém):*

$$Perf_{\mathcal{F}} = \left\{(A,\mathcal{T}) \;\middle|\; (\exists \ell : V \longrightarrow \mathcal{F})\, C_\ell^A \models \mathcal{T}\right\}. \qquad (11.2)$$

Vhodnou volbou třídy $\mathcal{F}$ prahových funkcí dostaneme tréninkový problém pro různé modely acyklických neuronových sítí, např. pro logické nebo (analogové) prahové obvody (viz kapitolu 7).

Zamysleme se nyní nad relevancí LP, resp. $Perf_{\mathcal{F}}$ pro učení acyklických neuronových sítí. Naše úvahy vychází z recenze [32] knihy [143] a z článku [263]. Při učení (acyklických) neuronových sítí v praktických aplikacích máme většinou k dispozici tréninkovou množinu, kterou se snažíme naučit síť zvolené pevné architektury s danou třídou $\mathcal{F}$ prahových funkcí. Hlavní námitkou proti adekvátnosti LP je, že pevná architektura je součástí vstupu, zatímco v praxi obvykle hledáme architekturu, pro kterou by učení sítě bylo efektivní, např. při aplikaci učícího algoritmu backpropagation (viz odstavec 2.2.6). Avšak prvotním cílem výzkumu neuronových sítí je univerzální efektivní učící pravidlo pro libovolnou architekturu sítě. Navíc LP můžeme zobecnit pro danou třídu $\mathcal{A}$ architektur, např.

$$Perf_{\mathcal{F}}^{\mathcal{A}} = \left\{\mathcal{T} \;\middle|\; (\exists A \in \mathcal{A})(\exists \ell : V \longrightarrow \mathcal{F})\, C_\ell^A \models \mathcal{T}\right\}, \qquad (11.3)$$

který pro $|\mathcal{A}| = 1$ splývá s původním problémem $Perf_{\mathcal{F}}$, proto je aspoň tak těžký. Na druhou stranu LP bez omezení na přípustné architektury nemá

Kapitola 11

Složitost učení neuronových sítí

V předchozích kapitolách této části jsme se zabývali výpočetní sílou neuronových sítí, resp. efektivitou jejich aktivní dynamiky. Ukázali jsme, že různé modely neuronových sítí jsou srovnatelné s klasickými výpočetními modely. V této a následující kapitole se zaměříme na složitostní analýzu učení neuronových sítí, tj. na adaptivní dynamiku, v níž spočívá hlavní přínos neuronových sítí (viz podkapitolu 1.4).

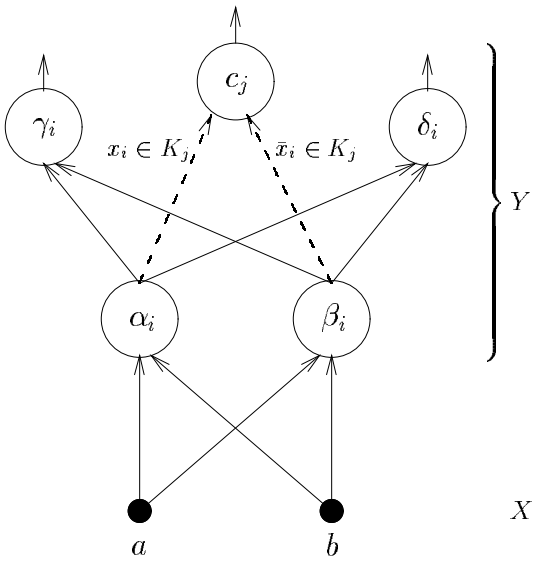
11.1 Tréninkový problém

Abychom mohli studovat fenomén učení neuronových sítí z hlediska teorie složitosti, zformulujeme nejprve tzv. *tréninkový problém* (training problem, resp. loading problem) pro acyklické neuronové sítě, resp. pro obvody (viz podkapitolu 7) a budeme diskutovat jeho relevanci v aplikacích neuronových sítí při jejich učení z příkladů.

Definice 11.1 *Nechť* $C = (V,X,Y,E,\ell)$ *je obvod (viz definici 7.1) s architekturou* $A = (V,X,Y,E)$ *(*$n = |X|$*,* $m = |Y|$*) a konfigurací* ℓ*. Budeme značit* $C_\ell^A : \{0,1\}^n \longrightarrow \{0,1\}^m$ *funkci* C *(viz definici 7.2) určenou architekturou* A *a konfigurací* ℓ*. Dále množina*

$$\mathcal{T} = \{(\mathbf{x}_k,\mathbf{d}_k) \mid \mathbf{x}_k \in \{0,1\}^n,\, \mathbf{d}_k \in \{0,1,\star\}^m,\, k = 1,\ldots,p\} \qquad (11.1)$$

p tréninkových vzorů *(*příkladů*), tj. dvojic vstupů* $\mathbf{x}_k$ *a odpovídajících požadovaných výstupů* $\mathbf{d}_k$ *(*$k = 1,\ldots,p$*) obvodu* C*, se nazývá* tréninková množina *pro architekturu* A*, kde hodnoty* $\star$ *u požadovaných výstupů jsou irelevantní,*



Obr. 11.1: Podobvod odpovídající proměnné x_i.

Definujeme architekturu $A = (V, X, Y, E)$ obvodu, který má jen dva vstupy $X = \{a, b\}$ a pro každou booleovské proměnnou $x_i \in \chi$, resp. pro každou klauzuli K_j, $(j = 1, \ldots, p)$ instance $3SAT$, obvod obsahuje čtyři, resp. jedno, odpovídající výstupní hradlo

$$Y = \{\alpha_i, \beta_i, \gamma_i, \delta_i \mid x_i \in \chi\} \cup \{c_j \mid j = 1, \ldots, p\}, \tag{11.4}$$

tj. $V = X \cup Y$. Čtveřice hradel odpovídajících booleovské proměnné $x_i \in \chi$ tvoří dvouvrstvý podobvod po dvou výstupních hradlech v první (α_i, β_i) a druhé (γ_i, δ_i) vrstvě. Hradlo c_j ve druhé vrstvě příslušející ke klauzuli K_j je spojeno s jedním hradlem v první vrstvě odpovídajícím x_i, přesněji s α_i, pokud $x_i \in K_j$, resp. s β_i, jestliže $\bar{x}_i \in K_j$. Příslušný podobvod je naznačen na obrázku 11.1. Následuje formální definice množiny orientovaných hran architektury A:

$$\begin{aligned} E \;=\; & \{(a, \alpha_i), (a, \beta_i), (b, \alpha_i), (b, \beta_i), \\ & (\alpha_i, \gamma_i), (\alpha_i, \delta_i), (\beta_i, \gamma_i), (\beta_i, \delta_i) \mid x_i \in \chi\} \cup \\ & \cup \{(\alpha_i, c_j) \mid x_i \in K_j\} \cup \{(\beta_i, c_j) \mid \bar{x}_i \in K_j\}. \end{aligned} \tag{11.5}$$

smysl. Pro každou tréninkovou množinu $\mathcal{T}$ lze totiž efektivně sestrojit architekturu A (např. každý neuron v první skryté vrstvě sítě odpovídá jednomu tréninkovému vzoru) a k ní konfiguraci ℓ takovou, že $C_\ell^A \models \mathcal{T}$. Avšak taková neuronová síť spíše odpovídá tabulce v paměti klasické von neumannovské architektury počítače a nelze u ní hovořit o generalizaci (viz odstavec 1.4.1). To zeslabuje zmíněnou námitku.

Další námitkou proti relevanci LP může být fakt, že LP je formulován pro klasický počítač (např. Turingův stroj), zatímco vlastní učení je součástí adaptivního režimu neuronové sítě. To znamená, že např. algoritmus pro LP má neomezený globální přístup k architektuře A a k tréninkovým vzorům z $\mathcal{T}$. V tomto případě je ale distribuovaný adaptivní režim sítě při řešení LP omezenější, a tedy LP je z tohoto hlediska pro klasický počítač lehčí. Na druhou stranu neuronová síť může využít paralelismu, avšak dosažené zrychlení sekvenčního algoritmu řešícího LP na klasickém počítači je úměrné nejvýše velikosti architektury $O(|A|)$, tj. je lineární vzhledem k délce vstupu LP. Z uvedeného vyplývá, že z hlediska složitosti je LP relevantním modelem učení acyklických neuronových sítí v tom smyslu, že učení sítí je nejméně tak těžké jako LP.

Uvažujme architekturu s jedním perceptronem, který počítá booleovskou prahovou funkci. Odpovídající tréninkovou množinu, kterou lze zadat jen polynomiálním počtem (vzhledem k počtu vstupů) všech tréninkových vzorů s jednotkovým výstupem, zatímco u ostatních vzorů se implicitně předpokládá nulový výstup, je možno chápat jako booleovskou funkci v disjunktivní normální formě polynomiální délky, kde jednotlivé monomy odpovídají tréninkovým vzorům. Podle věty 6.41 je však o takové funkci těžké algoritmicky rozhodnout, zda je booleovskou prahovou funkcí, tj. zda uvedený perceptron lze naučit uvedenou tréninkovou množinu. To naznačuje, že LP je algoritmicky těžký problém, tj. je velmi nepravděpodobné, že za uvedených předpokladů existuje obecně efektivní učící algoritmus pro acyklické neuronové sítě. Následující věta tuto hypotézu potvrzuje.

Věta 11.3 (Judd [143]) *Pro $\{AND, OR\} \subseteq \mathcal{F}$ je $Perf_\mathcal{F}$ NP-těžký problém.*

Důkaz: Důkaz tvrzení provedeme polynomiální redukcí NP-úplného problému $3SAT$ [47] na $Perf_\mathcal{F}$. Problém $3SAT$ je variantou problému SAT (viz důkaz věty 8.9), v níž instance mají v klauzulích nejvýše tři literály. Nechť tedy $\chi = \{x_1, \ldots, x_n\}$ je množina n booleovských proměnných a $K_1, \ldots, K_p$ jsou klauzule s nejvýše třemi literály ($|K_j| \leq 3$ pro $j = 1, \ldots, p$) booleovské formule f v konjunktivní normální formě, tj. instance problému $3SAT$. V polynomiálním čase zkonstruujeme odpovídající instanci $(A, \mathcal{T})$ problému $Perf_\mathcal{F}$ takovou, že f je splnitelná, právě když $(A, \mathcal{T}) \in Perf_\mathcal{F}$.

Dále tréninková množina $\mathcal{T} = \{I_2, I_3, I_1\}$ obsahuje tři tréninkové vzory:

$$I_1 = (01, (\star\star 01)^n 1^p) \tag{11.6}$$
$$I_2 = (00, (0000)^n 0^p) \tag{11.7}$$
$$I_3 = (11, (1111)^n 1^p)\,. \tag{11.8}$$

Nechť tedy nejprve f je splnitelná a ohodnocení $\omega : \chi \longrightarrow \{0,1\}$ proměnných dosvědčuje splnitelnost f. Definujeme konfiguraci $\ell : V \longrightarrow \mathcal{F}$ ($\{AND, OR\} \subseteq \mathcal{F}$):

$$\ell(\alpha_i) = \begin{cases} OR & \omega(x_i) = 1 \\ AND & \omega(x_i) = 0 \end{cases} \quad i = 1, \ldots, n \tag{11.9}$$
$$\ell(\beta_i) = \begin{cases} AND & \omega(x_i) = 1 \\ OR & \omega(x_i) = 0 \end{cases} \quad i = 1, \ldots, n \tag{11.10}$$
$$\ell(\gamma_i) = AND \quad \ell(\delta_i) = OR \quad i = 1, \ldots, n \tag{11.11}$$
$$\ell(c_j) = OR \quad j = 1, \ldots, p\,, \tag{11.12}$$

u které lze jednoduše ověřit konzistenci $C^A_\ell \models \mathcal{T}$.

Nechť naopak $\ell : V \longrightarrow \mathcal{F}$ je konfigurace taková, že $C^A_\ell \models \mathcal{T}$. Označme pro každé hradlo $j \in V$ odpovídající funkci $f_j = \ell(j) \in \mathcal{F}$. Z konzistence tréninkových vzorů I_1 a I_2 dostáváme

$$f_{\delta_i}(f_{\alpha_i}(0,1), f_{\beta_i}(0,1)) = 1 \neq 0 = f_{\delta_i}(f_{\alpha_i}(0,0), f_{\beta_i}(0,0))\,, \tag{11.13}$$

což spolu s konzistencí I_2 implikuje, že

$$f_{\alpha_i}(0,1) \neq f_{\alpha_i}(0,0) = 0 \text{ nebo } f_{\beta_i}(0,1) \neq f_{\beta_i}(0,0) = 0\,. \tag{11.14}$$

Podobně z konzistence tréninkových vzorů I_1 a I_3 dostáváme

$$f_{\gamma_i}(f_{\alpha_i}(0,1), f_{\beta_i}(0,1)) = 0 \neq 1 = f_{\gamma_i}(f_{\alpha_i}(1,1), f_{\beta_i}(1,1))\,, \tag{11.15}$$

což spolu s konzistencí I_3 implikuje, že

$$f_{\alpha_i}(0,1) \neq f_{\alpha_i}(1,1) = 1 \text{ nebo } f_{\beta_i}(0,1) \neq f_{\beta_i}(1,1) = 1\,. \tag{11.16}$$

Ze vztahů (11.14) a (11.16) vyplývá, že

$$f_{\alpha_i}(0,1) \neq f_{\beta_i}(0,1) \quad i = 1, \ldots, n\,. \tag{11.17}$$

Nyní uvažujme klauzuli K_j ($1 \leq j \leq p$), která má např. tvar $K_j = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$. Z konzistence tréninkových vzorů I_1 a I_2 dostaneme:

$$f_{c_j}(f_{\alpha_1}(0,1), f_{\beta_2}(0,1), f_{\beta_3}(0,1)) = 1 \quad \neq \tag{11.18}$$
$$\neq \quad 0 = f_{c_j}(f_{\alpha_1}(0,0), f_{\beta_2}(0,0), f_{\beta_3}(0,0))\,,$$

což spolu s konzistencí I_2 implikuje, že

$$f_{\alpha_1}(0,1) \neq 0 \text{ nebo } f_{\beta_2}(0,1) \neq 0 \text{ nebo } f_{\beta_3}(0,1) \neq 0\,. \tag{11.19}$$

Formule (11.19) lze podle (11.17) přepsat:

$$f_{\alpha_1}(0,1) = 1 \text{ nebo } f_{\alpha_2}(0,1) = 0 \text{ nebo } f_{\alpha_3}(0,1) = 0\,. \tag{11.20}$$

Získaná formule (11.20) pak odpovídá sémantice klauzule $K_j = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$, když x_i chápeme jako „$f_{\alpha_i}(0,1) = 1$“. Obecně tedy ohodnocení $\omega : \chi \longrightarrow \{0,1\}$ proměnných, definované $\omega(x_i) = f_{\alpha_i}(0,1)$ pro $i = 1, \ldots, n$, splňuje f. □

Důkaz věty 11.3 lze jednoduše upravit [143] tak, že příslušná redukce nevyžaduje irelevantní výstupy, které nemusí odpovídat situaci při učení neuronových sítí v praktických aplikacích. Dále alternativní důkaz této věty [143] pro nepatrně silnější třídu hradlových funkcí (kromě AND a OR obsahuje i jejich negace a negace jejich vstupů, tj. podle lemmy 6.27 je obsažena ve třídě booleovských prahových funkcí) uvažuje i skryté neurony, jejichž hodnoty nejsou nijak předepsány tréninkovou množinou. Z uvedeného vyplývá, že odstraněním některých technických předpokladů (irelevantní výstupy, absence skrytých neuronů), které zpřehledňují redukci v důkazu věty 11.3, se složitost LP nemění.

Jako důsledek věty 11.3 dostáváme NP-úplnost $Perf_{\mathcal{F}}$ pro různé třídy $\mathcal{F}$ hradlových funkcí.

Důsledek 11.4 (Judd [143]) *$Perf_{\mathcal{F}}$ je NP-úplný problém např. pro následující třídy $\mathcal{F}$ hradlových funkcí:*

(i) *$\mathcal{F} = \{AND, OR\}$ (logické obvody).*

(ii) *$\mathcal{F}$ je třída booleovských prahových funkcí (prahové obvody).*

(iii) *$\mathcal{F}$ je třída booleovských funkcí.*

Důkaz: Uvedené třídy $\mathcal{F}$ hradlových funkcí obsahují funkce AND a OR, a tedy NP-těžkost problému $Perf_{\mathcal{F}}$ je přímým důsledkem věty 11.3.

(i) Zřejmě $Perf_{\mathcal{F}} \in NP$, protože nedeterministický algoritmus v polynomiálním čase uhádne pro každé hradlo j jeho funkci $\ell(j) \in \{AND, OR\}$ a ověří konzistenci funkce obvodu s tréninkovou množinou.

(ii) Obdobně jako v části (i), avšak v tomto případě se hádají reprezentace booleovských prahových funkcí, které lze podle věty 6.30 reprezentovat pomocí polynomiálního počtu bitů.

(ii) *Omezené tréninkové množiny:*

- *Tréninková množina $\mathcal{T}$ je* monotónní, *tj. existuje monotónní funkce $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ taková, že $f \models \mathcal{T}$ (f je monotónní, jestliže všechny $x \leq x'$, tj. pro $x_i = 1$ je také $x'_i = 1$, implikují $f(x) \leq f(x')$ nebo jestliže všechny $x \leq x'$ implikují $f(x) \geq f(x')$).*
- *Tréninková množina $\mathcal{T}$ je generována pomocí funkce $C^A_{\ell_\mathcal{G}}$ obvodu se stejnou architekturou A a konfigurací $\ell_\mathcal{G} : V \longrightarrow \mathcal{G}$ z třídy $\mathcal{G}$ hradlových funkcí, pro kterou platí $\{AND, OR\} \subseteq \mathcal{G} \subseteq \mathcal{F}$, tj. $C^A_{\ell_\mathcal{G}} \models \mathcal{T}$. (Přesněji uvažujeme modifikovaný výpočetní problém $Perf^\mathcal{G}_\mathcal{F}$, který pro vstup $(A, \mathcal{T})$ má výstup 1, jestliže existuje $\ell : V \longrightarrow \mathcal{F}$ tak, že $C^A_\ell \models \mathcal{T}$, a má výstup 0, jestliže neexistuje $\ell_\mathcal{G} : V \longrightarrow \mathcal{F}$ tak, že $C^A_{\ell_\mathcal{G}} \models \mathcal{T}$. Problém $Perf^\mathcal{G}_\mathcal{F}$ je NP-těžký (resp. NP-úplný) např. i pro $\mathcal{G} = \{AND, OR\}$ a $\mathcal{F}$ třídu všech booleovských funkcí.)*

(iii) *Zeslabení konzistence:*

- *Funkce obvodu $C^A_\ell(\mathbf{x}) \models \mathbf{d}$ je konzistentní pro aspoň 67% celých tréninkových vzorů $(\mathbf{x}, \mathbf{d}) \in \mathcal{T}$ z tréninkové množiny $\mathcal{T}$.*

Důkaz: Instance $(A, \mathcal{T})$ problému $Perf_\mathcal{F}$ konstruovaná při polynomiální redukci z instance $3SAT$ v důkazu věty 11.3 splňuje současně všechny podmínky uvedené v tvrzení. Např. zeslabení konzistence (*iii*) je dosaženo tak, že 67% tréninkových vzorů ze tří představuje již všechny tři vzory. Tedy NP-úplnost LP platí pro konjunkci uvedených omezení. □

11.2 Mělké a hluboké architektury

Dalším důležitým speciálním případem tréninkového problému (viz podkapitolu 11.1) je omezení se na tzv. *mělké architektury.*

Definice 11.7 *Nechť $A = (V, X, Y, E)$ je architektura obvodu a $\mathcal{F}$ je třída jeho hradlových funkcí. Množinu $sc(j) \subseteq V$ hradel, které ovlivňují výstup $j \in Y$ (včetně j), tj. vede z nich orientovaná cesta do j, nazveme* kužel *(support cone) výstupu j. Podobně označme množinu $sc(Y') = \bigcup_{j \in Y'} sc(j)$ hradel kuželů vybraných výstupů z $Y' \subseteq Y$. Dále definujeme tzv. SCI-*graf *(support cone interaction graph) $G = (Y, I)$, což je neorientovaný graf interakcí kuželů v A, jehož vrcholy jsou výstupy z Y a hrany mezi výstupy indikují neprázdnost průniku odpovídajících kuželů, tj. $I = \{\{i, j\} \mid sc(i) \cap sc(j) \neq \emptyset, i, j \in Y\}$. Třídu $sccs(j) = \{\ell_j : sc(j) \longrightarrow \mathcal{F}\}$ všech částečných konfigurací kužele $j \in Y$ nazveme* prostor konfigurací kužele j *(support cone configuration space). Řekneme, že $\mathcal{A}$ je třída* mělkých architektur, *jestliže pro každou architekturu*

(iii) Obdobně jako v části (*i*), avšak reprezentace obecných booleovských funkcí mohou být exponenciální (viz větu 7.9). Na druhou stranu pro každé hradlo stačí uhádnout jen tu část reprezentace (na zbytku se předpokládají např. nulové hodnoty), která se uplatní při výpočtu obvodu pro vstupy tréninkových vzorů, a tedy ji lze reprezentovat jen polynomiálním počtem bitů (vzhledem k velikosti tréninkové množiny). □

Větu 11.3 lze také zobecnit pro analogové prahové obvody (viz definici 7.42) s omezenou monotónní aktivační funkcí [143]. Jako důsledek, který uvedeme bez důkazu, obdržíme složitost LP pro analogové prahové obvody se standardní sigmoidou, kterou využívá učící algoritmus backpropagation.

Věta 11.5 (Judd [143]) *$Perf_\mathcal{F}$ je NP-úplný problém pro třídu $\mathcal{F}$ funkcí (7.24) složených z lineární a aktivační funkce, pokud uvažujeme standardní sigmoidu (7.25) a separaci $\Omega(1)$ binárního výstupu (7.26).*

Herbert Wiklicky dokonce ukázal [291], že LP pro neuronové sítě vyššího řádu (viz odstavec 1.3.2), tj. prahové obvody s hradly, které místo lineární funkce vstupů počítají polynom, je algoritmicky nerozhodnutelný problém.

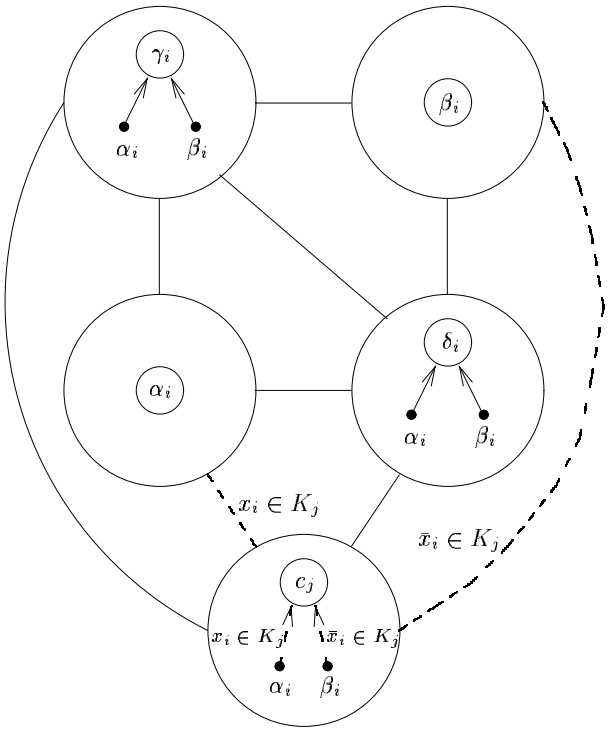
Z důsledku 11.4 a věty 11.5 vyplývá, že složitost problému $Perf_\mathcal{F}$ v důkazu věty 11.3 v podstatě nezávisí na třídě $\mathcal{F}$ hradlových funkcí, a tedy se dále, kde není třeba, budeme odkazovat na LP bez specifikace $\mathcal{F}$. Přesněji řečeno odchylky od NP-úplnosti $Perf_\mathcal{F}$ jsou pro triviální $\mathcal{F}$ ($|\mathcal{F}| = 1$), kdy problém LP není NP-těžký, a pro složité $\mathcal{F}$ (bez polynomiálního popisu), kdy LP není v NP. Tedy můžeme uzavřít tvrzením, že LP je v obecném případě algoritmicky těžký problém s diskutovanými důsledky pro učení acyklických neuronových sítí.

Možnou příčinou nepříznivé složitosti LP by mohla být jeho přílišná obecnost. Proto se v následujícím výkladu budeme zabývat speciálními případy LP, které by mohly zefektivnit jeho řešení. Avšak třída instancí LP musí být nekonečná, jinak by jeho složitost byla konstantní $O(1)$. Dalším jednoduchým důsledkem důkazu věty 11.3 je NP-úplnost LP i pro jeho velmi omezené instance.

Důsledek 11.6 (Judd [143]) *Problém $Perf_\mathcal{F}$ je NP-úplný i pro konjunkci následujících omezení instancí $(A = (V, X, Y, E), \mathcal{T})$:*

(i) *Omezené architektury:*

- *Hloubka architektury A je nejvýše 2 (1 skrytá vrstva).*
- *Počet vstupů každého hradla v architektuře A je nejvýše 3.*
- *Počet vstupů architektury A je $|X| = 2$.*



Obr. 11.2: SCI-graf podobvodu z obrázku 11.1.

Pro tento účel popíšeme speciální *křížovou konstrukci*. Uvažujme zkřížení hran $(q, q'), (r, r') \in E$ vedoucích v architektuře A z první do druhé vrstvy (tj. např. $(\alpha_i, c_j), (\beta_{i'}, c_{j'}) \in E$). Přidáme průnikový vrchol u a místo hrany (q, q') uvažujeme hrany (q, u), (q', u) a podobně hranu (r, r') nahradíme hranami $(r, u), (r', u)$, tj. hradla q', r' se přesunou do první skryté vrstvy. Navíc každá z dvojic hradel q, q' a r, r' má svoji kopii vstupů z $X = \{a, b\}$ architektury A, tj. a_q, b_q a a_r, b_r. Při šíření signálu těmito hranami v případě vstupu $a = 0$, $b = 1$ chceme, aby výstup z hradla q odpovídal vstupu do hradla q' a podobně aby výstup z hradla r odpovídal vstupu do hradla r'. Navrhneme tréninkové vzory pro tyto hradla, které uvedenou vlastnost zajistí až na negaci, tj. nová realizace původních hran $(q, q'), (r, r')$ neguje procházející signál. Opětovnou negaci lze docílit pomocí techniky z důkazu věty 11.3

$A \in \mathcal{A}$ je maximální velikost $\max_{j \in Y} |sccs(j)| \leq c$ *prostoru konfigurací kužele jejího výstupu omezena konstantou c.*

Zřejmě obvody s mělkými architekturami mají omezenou hloubku, počet vstupů hradel a velikost $|\mathcal{F}|$ třídy hradlových funkcí. Přesto následující věta ukazuje, že LP je těžký i pro mělké architektury dokonce se silnými omezeními na strukturu SCI-grafu. Tato omezení totiž v důkazu věty 11.3 nezabrání vlivu volby částečné konfigurace v konstruovaném obvodu na volbu konfigurace v celém tomto obvodu, a tedy LP v tomto případě nelze rozdělit na podúlohy, jejichž řešení by bylo efektivní.

Věta 11.8 (Judd [143]) *LP je NP-úplný problém pro*

(i) *třídu mělkých architektur.*

(ii) *třídu mělkých architektur (hloubky 2) s planárním (rovinným) SCI-grafem za předpokladu, že třída $\mathcal{F}$ hradlových funkcí obsahuje aspoň AND, OR, jejich negace a negace jejich proměnných.*

(iii) *třídu mělkých architektur s mřížkovým SCI-grafem $G = (Y, I)$, kde $I = \{\{v_{ji}, v_{j+1,i}\}, \{v_{ji}, v_{j,i+1}\} \mid 1 \leq j, i < \sqrt{|Y|}, v_{ji}, v_{j+1,i}, v_{j,i+1} \in Y\}$.*

Důkaz:

(i) Stačí si uvědomit, že polynomiální redukce v důkazu věty 11.3 využívá obvody s kužely velikosti nejvýše 4 ($sc(c_j)$) a také počet vstupů hradel není větší než 3 (srovnejte s (i) ve větě 11.6). Tedy velikost prostoru konfigurací kužele je pro každý výstup shora omezena konstantou a konstruovaný obvod má mělkou architekturu.

(ii) Na obrázku 11.2 je zobrazen SCI-graf architektury podobvodu z obrázku 11.1, který při redukci $3SAT$ na LP v důkazu věty 11.3 odpovídá proměnné x_i booleovské formule f. Ve vrcholech SCI-grafu jsou navíc kromě (zakroužkovaných) označení těchto vrcholů znázorněny i odpovídající kužele s původní strukturou. Je zajímavé, že původní orientovaný graf architektury $A = (V, X, Y, E)$ (obrázek 11.1) bez vstupů a jeho neorientovaný SCI-graf $G = (Y, I)$ (obrázek 11.2) bez spojů s c_j jsou až na orientaci hran izomorfní, a tedy je můžeme podle potřeby zaměňovat. Chceme dokázat, že existuje rovinné nakreslení (tj. bez křížení hran) SCI-grafu G architektury A celého obvodu konstruovaného v důkazu věty 11.3. Nakreslení na obrázku 11.2 SCI-grafu architektury uvedeného podobvodu je rovinné, avšak spojení architektur těchto podobvodů odpovídajících booleovským proměnným x_i s hradly $c_j \in Y$ příslušejícími klauzulím K_j může obecně způsobit zkřížení hran.

	$\cdots$	a_q	b_q	a_r	b_r	$\cdots$	,	$\cdots$	q	q'	r	r'	u	$\cdots$
P_1	$\cdots$	0	0	0	0	$\cdots$	,	$\cdots$	0	0	0	0	0	$\cdots$
P_2	$\cdots$	0	1	0	0	$\cdots$	,	$\cdots$	$\star$	$\star$	0	0	1	$\cdots$
P_3	$\cdots$	1	1	1	1	$\cdots$	,	$\cdots$	1	1	1	1	1	$\cdots$
P_4	$\cdots$	0	1	1	1	$\cdots$	,	$\cdots$	$\star$	$\star$	1	1	0	$\cdots$
P_5	$\cdots$	0	0	0	1	$\cdots$	,	$\cdots$	0	0	$\star$	$\star$	1	$\cdots$
P_6	$\cdots$	1	1	0	1	$\cdots$	,	$\cdots$	1	1	$\star$	$\star$	0	$\cdots$

Tabulka 11.1: Relevantní část tréninkové množiny pro křížovou konstrukci.

(viz (11.17)). Relevantní část tréninkové množiny vyžaduje 6 tréninkových vzorů a je naznačena v tabulce 11.1.

Nyní porovnáním P_1 a P_2 (podobně jako (11.13) a (11.14) v důkazu věty 11.3) dostáváme, že $f_q(0,1) \neq 0$ nebo $f_{q'}(0,1) \neq 0$, a porovnáním P_3 a P_4 obdržíme, že $f_q(0,1) \neq 1$ nebo $f_{q'}(0,1) \neq 1$, tedy $f_q(0,1) \neq f_{q'}(0,1)$. Podobně porovnáním P_1 a P_5 dostáváme, že $f_r(0,1) \neq 0$ nebo $f_{r'}(0,1) \neq 0$, a porovnáním P_3 a P_6 obdržíme, že $f_r(0,1) \neq 1$ nebo $f_{r'}(0,1) \neq 1$, tedy $f_r(0,1) \neq f_{r'}(0,1)$. Zřejmě $f_r(0,1), f_q(0,1)$ jsou negativní kopie $f_{q'}(0,1), f_{r'}(0,1)$.

V architektuře A je jen polynomiální počet křížových bodů, tedy pomocí uvedené křížové konstrukce dostaneme odpovídající architekturu s planárním SCI-grafem v polynomiálním čase. Navíc je tato architektura podobně jako v části (*i*) mělká. Avšak konfigurace, která by byla konzistentní s modifikovanou tréninkovou množinou (viz tabulku 11.1), vyžaduje třídu $\mathcal{F}$ hradlových funkcí obsahující např. booleovské funkce, které nejsou prahové. Uvedenou křížovou konstrukci lze zobecnit pro třídu $\mathcal{F}$ hradlových funkcí, která obsahuje aspoň AND, OR, jejich negace a negace jejich proměnných. Poměrně komplikované detaily tohoto zobecnění lze najít v [143].

(iii) Podobně jako v části (*ii*) jde o to, jak šířit informaci v mřížce. Pro diagonální směr z v_{ji} do $v_{j+1,i+1}$ se uplatní technika z důkazu věty 11.3 (viz (11.17)) a pro přímý směr o 2 hrany, tj. např. z v_{ji} do $v_{j+2,i}$, využijeme křížovou konstrukci z (*ii*). □

Ve snaze dále zesílit omezení na SCI-grafy architektur tak, aby LP byl řešitelný v polynomiálním čase, definujeme tzv. *stromovou dekompozici* (tree-decomposition) grafu a *šířku ramene* (treewidth) grafu. Tento pojem byl objeven nezávisle na sobě několika autory (např. Stephen Judd jej objevil v kontextu LP a pojmenoval *armwidth*). Ukazuje se, že často obecně

NP-úplné grafové problémy lze řešit v polynomiálním čase pro podtřídu grafů s omezenou šířkou ramene [41].

Definice 11.9 *Řekneme, že* (Υ, D) *je* stromová dekompozice *neorientovaného grafu* $G = (V, E)$, *jestliže* $\Upsilon = \{V_j \subseteq V \mid j \in J\}$ *jsou podmnožiny vrcholů* G *a* D *je strom s množinou vrcholů* J *takový, že*

(i) $V = \bigcup_{j \in J} V_j$.

(ii) *Pro každou hranu* $\{u, v\} \in E$ *existuje* $j \in J$ *tak, že* $u, v \in V_j$.

(iii) *Pro všechny* $i, j, k \in J$, *pokud* j *leží na cestě z* i *do* k *ve stromě* D, *pak* $V_i \cap V_k \subseteq V_j$.

Šířka stromové dekompozice (Υ, D) *je* $\max_{j \in J} |V_j| - 1$ *a* šířka ramene *grafu* G *je minimální šířka přes všechny jeho stromové dekompozice. Navíc řekneme, že stromová dekompozice je* kompresovaná, *pokud pro každé* $i, j \in J$ *platí* $V_i \not\subseteq V_j$.

Lemma 11.10

(i) *Pro danou stromovou dekompozici* (Υ, D) *grafu* $G = (V, E)$ *lze sestrojit kompresovanou stromovou dekompozici stejné šířky v lineárním čase.*

(ii) *V každé kompresované stromové dekompozici* (Υ, D) *grafu* $G = (V, E)$ *s* $r = |V|$ *vrcholy má strom* D *nejvýše* r *vrcholů.*

Důkaz:

(i) Nechť $\Upsilon = \{V_j \subseteq V \mid j \in J\}$ a strom $D = (J, F)$ má množinu vrcholů J a hran F. Jestliže nějaké $V_k \subseteq V_i$, pak podle (*iii*) z definice 11.9 také $V_k = V_k \cap V_i \subseteq V_j$ pro $\{k, j\} \in F$ a j leží na cestě z k do i ve stromě D. V tomto případě odstraníme V_k z Υ a vrchol k z D tak, že spojíme všechny j' sousedy k kromě j ($\{j', k\} \in F$ a $j \neq j' \in J$) s vrcholem j. Tímto neprodloužíme žádnou cestu v D (zachováme platnost (*iii*) v definice 11.9), ani nezměníme šířku stromové dekompozice. Opakujeme tento postup, pokud třeba, pro každou hranu z F a v lineárním čase sestrojíme kompresovanou stromovou dekompozici.

(ii) Důkaz probíhá indukcí dle r. Případ $r = 1$ je triviální. Předpokládejme, že tvrzení platí pro $r - 1$. Rozšíříme stromovou dekompozici o nový vrchol i, který připojíme k listu $j \in J$ ve stromě D. Jistě existuje vrchol $v_i \in V_i \setminus V_j$ v grafu G, protože uvažujeme kompresovanou stromovou dekompozici, v níž $V_i \not\subseteq V_j$. Všechny cesty do i ve stromě D vedou přes j, a tedy neexistuje $k \in J$ takové, že $v_i \in V_k$. Kdyby totiž $v_i \in V_k$, pak by $v_i \in V_k \cap V_i \subseteq V_j$, což je spor. To znamená, že počet vrcholů v G roste aspoň tak rychle jako vrcholy v D. □

3. { $|J(\delta)| > 1$ }
   Pro každý bezprostřední podstrom δ_j stromu δ (tj. $\{r(\delta), r(\delta_j)\} \in F(\delta)$) vypočti $\Lambda_j :=$`Solve(`δ_j`)`.

4. Vypočti $\Lambda := \{\ell' \in \Lambda' \mid (\forall j)\,(\exists \ell_j \in \Lambda_j)\, \ell' \cong \ell_j\}$.

5. Vrať Λ.

Tvrdíme, že existuje $\ell' \in$`Solve(`δ`)`, právě když existuje $\ell \in scpc[\varrho(\delta)]$ takové, že $\ell' = \ell | sc(Y_{r(\delta)})$.

Důkaz probíhá indukcí dle výšky δ. V případě nulové výšky, kdy δ obsahuje jen kořen, tj. $Y_{r(\delta)} = \varrho(\delta)$, je $\Lambda = \Lambda' = scpc[\varrho(\delta)]$ podle kroků 1, 2 procedury `Solve`, a tedy tvrzení platí. V indukčním kroku nechť tvrzení platí pro všechny bezprostřední podstromy δ_j stromu δ ($\{r(\delta), r(\delta_j)\} \in F(\delta)$).

Nechť nejprve $\ell' \in$`Solve(`δ`)`. Podle kroků 1, 4 procedury `Solve` to znamená, že $\ell' \in scpc[Y_{r(\delta)}]$, tj. $\ell' : sc(Y_{r(\delta)}) \longrightarrow \mathcal{F}$, a podle kroků 3, 4 navíc pro každé j ($\{r(\delta), r(\delta_j)\} \in F(\delta)$) existuje $\ell'_j \in$`Solve(`δ_j`)`, tj. $\ell'_j : sc(Y_{r(\delta_j)}) \longrightarrow \mathcal{F}$, takové, že

$$\ell' \cong \ell'_j\,. \tag{11.23}$$

Podle indukčního předpokladu pro δ_j tedy existuje $\ell_j \in scpc[\varrho(\delta_j)]$, tj. $\ell_j : sc(\varrho(\delta_j)) \longrightarrow \mathcal{F}$, takové, že

$$\ell'_j = \ell_j \left| sc\left(Y_{r(\delta_j)}\right)\right.\,. \tag{11.24}$$

Ukážeme, že

$$sc\left(\varrho(\delta_j)\right) \cap sc\left(Y_{r(\delta)}\right) \subseteq sc\left(Y_{r(\delta_j)}\right)\,, \tag{11.25}$$

Nechť $v \in sc(\varrho(\delta_j)) \cap sc(Y_{r(\delta)})$ je hradlo obvodu, tj. $v \in sc(Y_i) \cap sc(Y_{r(\delta)})$ pro nějaký vrchol $i \in J(\delta_j)$ podstromu δ_j. Tedy existují dvě výstupní hradla $y_1 \in Y_i$ a $y_2 \in Y_{r(\delta)}$ taková, že $v \in sc(y_1) \cap sc(y_2)$. Podle definice 11.7 SCI-grafu je pak $\{y_1, y_2\} \in I$ hrana grafu G. Dále podle (*ii*) z definice 11.9 stromové dekompozice potom existuje vrchol $k \in J$ stromu D takový, že $y_1, y_2 \in Y_k$. Protože D je strom, musí v něm existovat cesta z k do i. Nejprve uvažujme případ, že na této cestě leží vrchol $r(\delta_j)$. Potom podle (*iii*) z definice 11.9 stromové dekompozice platí, že $Y_k \cap Y_i \subseteq Y_{r(\delta_j)}$, ale $y_1 \in Y_k \cap Y_i$, a proto $y_1 \in Y_{r(\delta_j)}$. Z toho vyplývá, že $sc(y_1) \subseteq sc(Y_{r(\delta_j)})$, a tedy $v \in sc(Y_{r(\delta_j)})$. Nechť naopak vrchol $r(\delta_j)$ neleží na cestě z k do i ve stromě D. To ale znamená, že vrchol k se nachází v podstromě δ_j, a tedy vrchol $r(\delta_j)$ leží ve stromě D na cestě z k do $r(\delta)$. Opět podle (*iii*) z definice 11.9 stromové dekompozice dostáváme, že $Y_k \cap Y_{r(\delta)} \subseteq Y_{r(\delta_j)}$, ale $y_2 \in Y_k \cap Y_{r(\delta)}$, a proto $y_2 \in Y_{r(\delta_j)}$. Z toho dále vyplývá, že $sc(y_2) \subseteq sc(Y_{r(\delta_j)})$, a tedy $v \in sc(Y_{r(\delta_j)})$. Tím jsme dokázali vztah (11.25), tj. průnik definičních oborů ℓ_j a ℓ' je obsažen v definičním oboru ℓ'_j.

Ukážeme polynomiální algoritmus, který řeší LP pro mělké architektury s omezenou šířkou ramene SCI-grafu. Pro tento účel zavedeme některé pojmy a značení.

Definice 11.11 *Nechť $A = (V, X, Y, E)$ je architektura obvodu a $V_1, V_2 \subseteq V$ jsou podmnožiny jejích hradel. Řekneme, že částečné konfigurace $\ell_{V_1} : V_1 \longrightarrow \mathcal{F}$ a $\ell_{V_2} : V_2 \longrightarrow \mathcal{F}$ jsou* kompatibilní *a značíme $\ell_{V_1} \cong \ell_{V_2}$, jestliže pro každé hradlo $j \in V_1 \cap V_2$ platí $\ell_{V_1}(j) = \ell_{V_2}(j)$. Dále $\ell' = \ell_{V_1} \cup \ell_{V_2}$ je* sjednocení *kompatibilních částečných konfigurací ℓ_{V_1}, ℓ_{V_2}, jestliže $\ell' : V_1 \cup V_2 \longrightarrow \mathcal{F}$ a $\ell' \cong \ell_{V_1}$, $\ell' \cong \ell_{V_2}$. Naopak pro* restrikci $\ell' = \ell_{V_1}|V_2$ *částečné konfigurace ℓ_{V_1} na množinu hradel $V_2 \subseteq V_1$ platí, že $\ell' : V_2 \longrightarrow \mathcal{F}$ a $\ell' \cong \ell_{V_1}$. Nechť dále $\mathcal{T}$ je tréninková množina pro A. Řekneme, že částečná konfigurace $\ell_{Y'} : sc(Y') \longrightarrow \mathcal{F}$ pro $Y' \subseteq Y$ je* konzistentní *s tréninkovou množinou $\mathcal{T}$ a značíme $\ell_{Y'} \models \mathcal{T}$, jestliže pro každou úplnou konfiguraci $\ell : V \longrightarrow \mathcal{F}$ kompatibilní s $\ell_{Y'} \cong \ell$, každý tréninkový vzor $(\mathbf{x}, \mathbf{d}) \in \mathcal{T}$ a každé výstupní hradlo $j \in Y'$ platí $(C^A_\ell(\mathbf{x}))_j \models d_j$.*

Věta 11.12 (Judd [143]) *LP pro mělké architektury $A = (V, X, Y, E)$ s omezenou šířkou q ramene SCI-grafu $G = (Y, I)$ lze řešit v lineárním čase, pokud je na vstupu dána stromová dekompozice (Υ, D) grafu G šířky q.*

Důkaz: Bez újmy na obecnosti nechť (Υ, D) je kompresovaná stromová dekompozice, protože obecnou stromovou dekompozici lze podle (*i*) z lemmy 11.10 kompresovat v lineárním čase. Označme $\Upsilon = \{Y_j \subseteq Y \mid j \in J\}$, kde J je množina vrcholů stromu D, a dále podmnožinu

$$\varrho(\delta) = \bigcup_{j \in J(\delta)} Y_j \tag{11.21}$$

vrcholů z Y pro podstrom δ stromu D, kde $J(\delta)$ je množinou vrcholů δ. Podobně $F(\delta)$ je množina hran δ a $r(\delta)$ značí kořen podstromu δ. Dále pro podmnožinu $Z \subseteq Y$ vrcholů SCI-grafu označme třídu

$$scpc[Z] = \{\ell_Z : sc(Z) \longrightarrow \mathcal{F} \mid \ell_Z \models \mathcal{T}\} \tag{11.22}$$

částečných konfigurací konzistentních s danou tréninkovou množinou $\mathcal{T}$ pro A. Zřejmě pro každé $\ell \in scpc[\varrho(D)] = scpc[Y]$ platí $C^A_\ell \models \mathcal{T}$.

Nejprve definujeme následující rekurzivní proceduru `Solve(`δ`)`, která pro podstrom δ stromu D vrátí množinu $\Lambda \subseteq scpc[Y_{r(\delta)}]$.

`Solve(`δ`)`:

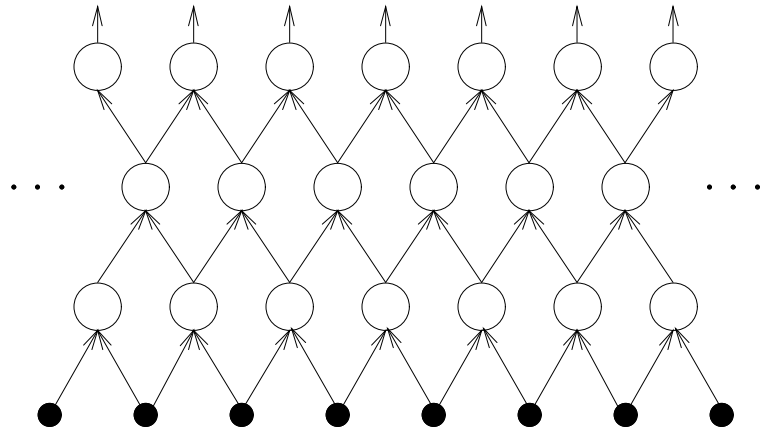
1. Vypočti $\Lambda' := scpc[Y_{r(\delta)}]$.

2. Pro $|J(\delta)| = 1$ polož $\Lambda := \Lambda'$ a pokračuj krokem 5.

Důsledek 11.13 (Judd [143]) *Nechť jsou splněny předpoklady věty 11.12 a navíc nechť šířka $q \leq Q(n)$ ramene SCI-grafu je obecně omezena funkcí $Q(n)$, která závisí na velikosti instance LP. Složitost LP pak závisí na funkci $Q(n)$ následujícím způsobem:*

$Q(n)$	*složitost LP*
$O(1)$	*lineární*
$O(\log n)$	*polynomiální*
$\Omega(n^\varepsilon)$, $0 < \varepsilon \leq 1$	*NP-úplný.*

Důkaz: Pro $Q(n) = O(1)$ dostáváme přímo tvrzení věty 11.12 a případ $Q(n) = \log n$ je přímým důsledkem důkazu této věty. Důkaz, že LP je NP-úplný pro mělkou architekturu s polynomiální šířkou $Q(n)$ ramene stromové dekompozice SCI-grafu, je podobný jako důkaz věty 11.8. Při polynomiální redukci $3SAT$ na LP navíc přidáme k architektuře odpovídající instance LP izolovaná hradla tak, aby počet hradel byl $s' = Q^{-1}(s) \geq s$ $(0 < \varepsilon \leq 1)$, kde s je původní počet hradel a $Q^{-1}(n)$ je inverzní funkce k polynomu $Q(n)$, což zajistí polynomiální čas redukce. Šířka q stromové dekompozice SCI-grafu se tak nezmění a podle (ii) z lemmy 11.10 je menší než počet výstupů obvodu, a tedy $q \leq s = Q(s')$. □



Obr. 11.3: Sloupcová architektura hloubky $d = 3$.

Polynomiální algoritmus pro LP z věty 11.12 předpokládal, že stromová dekompozice s minimální šířkou příslušného SCI-grafu je zadána na vstupu. Nalezení šířky ramene je totiž pro obecný graf NP-úplný problém [23]. Avšak

Uvažujme nyní $v \in sc(\varrho(\delta_j)) \cap sc(Y_{r(\delta)})$ hradlo obvodu. Podle (11.25) je $v \in sc(Y_{r(\delta_j)})$, a tedy $v \in sc(Y_{r(\delta)}) \cap sc(Y_{r(\delta_j)})$. Z (11.23) pak vyplývá, že $\ell'(v) = \ell'_j(v)$, a proto podle (11.24) je $\ell'(v) = \ell_j(v)$. To znamená, že $\ell' \cong \ell_j$ a můžeme definovat sjednocení $\ell_j^+ = \ell_j \cup \ell' \in scpc[\varrho(\delta_j) \cup Y_{r(\delta)}]$, tj. $\ell_j^+ : sc(\varrho(\delta_j) \cup Y_{r(\delta)}) \longrightarrow \mathcal{F}$. Ukážeme, že ℓ_j^+ jsou vzájemně kompatibilní. Nechť tedy $v \in sc(\varrho(\delta_j) \cup Y_{r(\delta)}) \cap sc(\varrho(\delta_{j'}) \cup Y_{r(\delta)}) = (sc(\varrho(\delta_j)) \cap sc(\varrho(\delta_{j'}))) \cup sc(Y_{r(\delta)})$ je hradlo obvodu. Dále ukážeme, že $v \in sc(Y_{r(\delta)})$. Předpokládejme proto, že $v \in sc(\varrho(\delta_j)) \cap sc(\varrho(\delta_{j'}))$. To znamená, že existují $i \in J(\delta_j)$ a $i' \in J(\delta_{j'})$ takové, že $v \in sc(Y_i) \cap sc(Y_{i'})$, a dále $y \in Y_i$ a $y' \in Y_{i'}$, pro které $v \in sc(y) \cap sc(y')$. Podle definice 11.7 SCI-grafu je pak $\{y, y'\} \in I$ hrana grafu G a podle (ii) z definice 11.9 stromové dekompozice potom existuje vrchol $k \in J$ stromu D takový, že $y, y' \in Y_k$. Kořen $r(\delta)$ leží na cestě buď z k do i, nebo z k do i', protože vrcholy $i, i' \in J(\delta)$ leží v jeho různých podstromech $\delta_j, \delta_{j'}$. Potom podle (iii) z definice 11.9 stromové dekompozice platí buď $y \in Y_k \cap Y_i \subseteq Y_{r(\delta)}$, nebo $y' \in Y_k \cap Y_{i'} \subseteq Y_{r(\delta)}$, tj. buď $v \in sc(y) \subseteq sc(Y_{r(\delta)})$, nebo $v \in sc(y') \subseteq sc(Y_{r(\delta)})$, a tedy $v \in sc(Y_{r(\delta)})$. To znamená, že $\ell_j^+(v) = \ell'(v) = \ell_{j'}^+(v)$, a tedy $\ell_j^+ \cong \ell_{j'}^+$. Nyní můžeme definovat

$$\ell = \bigcup_j \ell_j^+ = \ell' \cup \bigcup_j \ell_j \in scpc[\varrho(\delta)]\,, \tag{11.26}$$

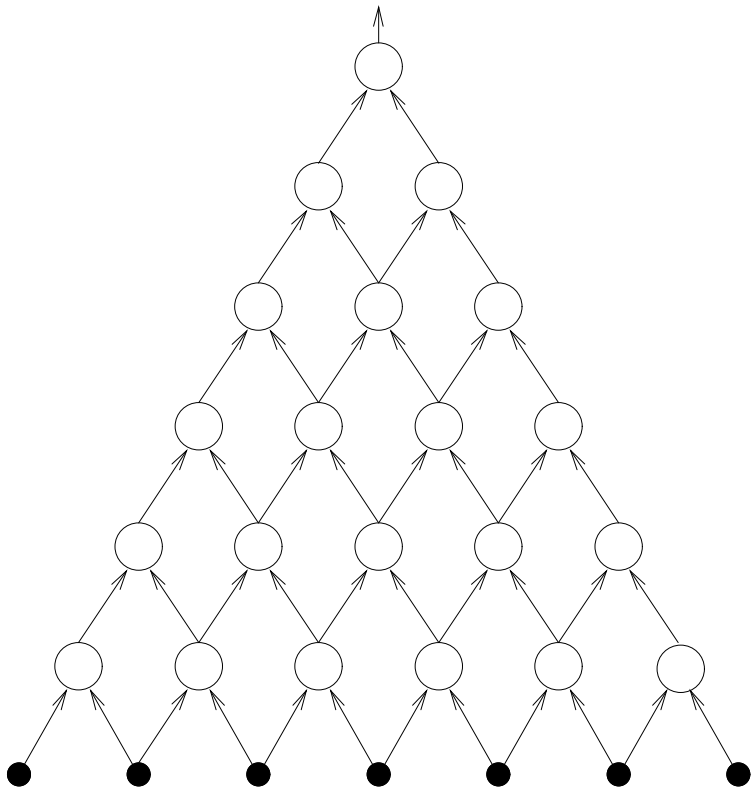
pro kterou zřejmě platí $\ell' = \ell | sc(Y_{r(\delta)})$.

Jestliže naopak existuje $\ell \in scpc[\varrho(\delta)]$, pak procedura `Solve`, která prochází všechny relevantní částečné konfigurace, musí najít $\ell' = \ell | sc(Y_{r(\delta)})$. Tím jsme ukázali uvedené tvrzení o proceduře `Solve`.

Nyní použijeme proceduru `Solve` k řešení zadaného LP tak, že zvolíme libovolně kořen $r(D)$ stromu D a zavoláme `Solve(`D`)`, která vrátí $\Lambda =$ `Solve(`D`)`. Víme, že $\ell' \in \Lambda$, právě když existuje $\ell \in scpc[\varrho(D)]$ takové, že $\ell' = \ell | sc(Y_{r(D)})$, tj. právě když $C_\ell^A \models \mathcal{T}$. Tedy LP má řešení, právě když $\Lambda \neq \emptyset$.

Na závěr odhadneme časovou složitost naznačeného algoritmu. Počet volání procedury odpovídá počtu vrcholů stromu D kompresované stromové dekompozice (Υ, D), který je podle (ii) z lemmy 11.10 shora omezen počtem $|Y|$ vrcholů SCI-grafu G, tedy je lineární vzhledem k velikosti instance LP. Čas na výpočet procedury `Solve` (kromě jejího rekurzivního volání) je $O(|scpc[Y_{r(\delta)}]|) = O(c^q)$, kde $\max_{j \in Y} |sccs(j)| \leq c = O(1)$ je konstantní podle definice 11.7 mělké architektury a také $q = O(1)$ je omezená šířka ramene SCI-grafu G. Tedy výpočet `Solve` proběhne v konstantním čase a celkově LP lze za uvedených předpokladů řešit v lineárním čase. □

Větu 11.12 lze zobecnit pro šířky ramene SCI-grafu shora omezené funkcí, která závisí na velikosti instance LP. Podle (ii) z lemmy 11.10 se můžeme omezit na nejvýše lineární funkce.



Obr. 11.5: Trojúhelníková architektura hloubky $d = 6$.

dující věty o složitosti uvedeného LP klade určitá omezení na třídu hradlových funkcí (např. vylučuje některé booleovské prahové funkce), avšak při dvou vstupech hradel tyto předpoklady pravděpodobně nemají vliv na složitost LP.

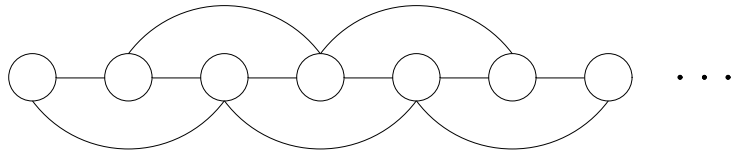
Věta 11.14 (Šíma [263]) *Nechť $\mathcal{F}$ je třída netriviálních symetrických monotónních booleovských hradlových funkcí, tj.*

(i) $\{OR, \overline{AND}\} \subseteq \mathcal{F}$ *nebo* $\{AND, \overline{OR}\} \subseteq \mathcal{F}$

(ii) $\mathcal{F} \subseteq \{0, 1, OR, AND, \overline{OR}, \overline{AND}\}$.

Pak pro třídu trojúhelníkových architektur $Perf_{\mathcal{F}}$ je NP-úplný problém.

v praxi se obvykle používají regulární architektury, u nichž lze jednoduše určit stromové dekompozice jejich SCI-grafů s minimální šířkou. Například tzv. *sloupcová* (columnar line) architektura hloubky 3 je na obrázku 11.3. Obecně tato architektura má omezenou hloubku d a každé hradlo má dva vstupy ze dvou hradel (resp. vstupů), které jsou pod ním v předchozí vrstvě. Příslušný SCI-graf sloupcové architektury z obrázku 11.3 je načrtnut na obrázku 11.4, kde každý vrchol grafu odpovídá výstupnímu hradlu a hrany představují neprázdné průniky odpovídajících kuželů. Stromová dekompozice takového SCI-grafu s minimální šířkou je v tomto případě cesta, kde každý vrchol stromu sdružuje sousední 3 výstupní hradla. Tedy šířka ramene SCI-grafu z obrázku 11.4 je 2 a v obecném případě je $d-1$. Sloupcová architektura splňuje předpoklady věty 11.12, a tedy LP lze pro ni řešit v lineárním čase. Také podle vyjádření neurofyziologů sloupcová architektura údajně připomíná korové struktury v některých částech lidského mozku, kde spoje jsou lokalizované okolo neuronu.



Obr. 11.4: SCI-graf sloupcové architektury z obrázku 11.3.

Na druhou stranu učící algoritmus z věty 11.12 je prakticky nepoužitelný, protože multiplikativní konstanta jeho lineární složitosti je příliš velká. Z důkazu této věty vyplývá, že příslušná konstanta závisí podstatně na maximální velikosti prostoru konfigurací kužele. V uvedeném algoritmu se využívá toho, že u mělkých architektur je tato velikost konstantní, a proto lze konzistentní částečné konfigurace kužele vyhledat z třídy všech takových konfigurací v konstantním čase tak, že prověříme všechny možnosti. Efektivní implementace tohoto algoritmu by vyžadovala efektivní proceduru pro nalezení konzistentní částečné konfigurace kužele. Složitost takové procedury závisí na hloubce obvodu. Proto se uvažují tzv. *hluboké* architektury, které mají neomezenou hloubku. Prototypem takové architektury je tzv. *trojúhelníková* (triangular) architektura, jejíž struktura odpovídá kuželu sloupcové architektury bez omezení na hloubku. Příklad trojúhelníkové architektury hloubky $d = 6$ je na obrázku 11.5. Složitost LP pro třídu trojúhelníkových architektur, které jsou pravděpodobně nejjednodušší regulární hluboké architektury, vystihuje problém učení hlubokých architektur. Důkaz násle-

Důkaz: Zřejmě $Perf_{\mathcal{F}} \in NP$, protože nedeterministický algoritmus v polynomiálním čase uhádne konfiguraci, tj. pro každé hradlo určí jeho funkci z $\mathcal{F} \subseteq \{0,1,OR,AND,\overline{OR},\overline{AND}\}$, tj. z nejvýše šesti možných, a ověří, zda funkce sítě je konzistentní s tréninkovou množinou.

Důkaz NP-těžkosti $Perf_{\mathcal{F}}$ provedeme polynomiální redukcí NP-úplného problému SAT (viz důkaz věty 8.9). Nechť tedy $\chi = \{x_1,\ldots,x_n\}$ je množina n booleovských proměnných a $K_1,\ldots,K_p$ jsou klauzule booleovské formule f v konjunktivní normální formě, tj. instance problému SAT. V polynomiálním čase zkonstruujeme odpovídající instanci $(A,\mathcal{T})$ problému $Perf_{\mathcal{F}}$ takovou, že f je splnitelná, právě když $(A,\mathcal{T}) \in Perf_{\mathcal{F}}$.

Bez újmy na obecnosti budeme předpokládat, že pozitivní literály (proměnné) se v klauzulích formule f střídají pravidelně s negativními literály (negace proměnných) při daném uspořádání proměnných χ. V opačném případě, když v nějaké klauzuli K_j $(1 \leq j \leq p)$ po sobě následují např. dva pozitivní literály $\cdots x_{i_1} \vee x_{i_3} \cdots$ pro $1 \leq i_1 < i_3 \leq n$ (podobně negativní literály), přidáme novou proměnnou x_{i_2} v uspořádání χ mezi x_{i_1} a x_{i_3}, tj. $1 \leq i_1 < i_2 < i_3 \leq n$. Dále opravíme klauzuli K_j pomocí příslušného negativního literálu $\cdots x_{i_1} \vee \bar{x}_{i_2} \vee x_{i_3} \cdots$ a navíc přidáme klauzuli $K_{j'} = \{x_{i_2}\}$ obsahující jen pozitivní literál x_{i_2}. Klauzule $K_{j'}$ vynutí jednotkovou hodnotu proměnné x_{i_2} při splňujícím ohodnocení a zabrání tak vlivu přidaného negativního literálu $\bar{x}_{i_2}$ na klauzuli K_j. Opakováním uvedeného postupu získáme v polynomiálním čase formuli f v požadovaném tvaru.

Trojúhelníková architektura $A = (V,X,Y,E)$ (viz obrázek 11.5) odpovídající instance LP je dána svojí hloubkou $d = 2n-1$, tj. má $2n$ vstupů a jeden výstup. Příslušná tréninková množina $\mathcal{T}$ obsahuje jeden tréninkový vzor s požadovaným nulovým výstupem a p tréninkových vzorů s požadovaným jednotkovým výstupem, které odpovídají klauzulím K_j pro $j = 1,\ldots,p$:

$$\begin{aligned} \mathcal{T} &= \{((10)^n,0)\} \cup \\ &\quad \cup \left\{(\beta_{j1}\ldots\beta_{jn},1) \mid \beta_{ji} \in \{0,1\}^2,\, i = 1,\ldots,n;\, j = 1,\ldots,p\right\}, \end{aligned} \tag{11.27}$$

kde

$$\beta_{ij} = \begin{cases} 11 & x_i \in K_j \\ 00 & \bar{x}_i \in K_j \\ 01 & x_i, \bar{x}_i \notin K_j;\ x_{\pi_j(i)} \in K_j \text{, resp. } \bar{x}_{\varrho_j(i)} \in K_j \\ 10 & x_i, \bar{x}_i \notin K_j;\ \bar{x}_{\pi_j(i)} \in K_j \text{, resp. } x_{\varrho_j(i)} \in K_j \end{cases} \tag{11.28}$$

pro $j = 1,\ldots,p$, $i = 1,\ldots,n$ a

$$\pi_j(i) = \max\{i' < i \mid x_{i'} \in K_j \text{ nebo } \bar{x}_{i'} \in K_j\} \tag{11.29}$$

$$\varrho_j(i) = \min\{i' > i \mid x_{i'} \in K_j \text{ nebo } \bar{x}_{i'} \in K_j\}. \tag{11.30}$$

Předchozí formální zápis ilustrujeme na příkladě. Uvažujeme booleovskou formuli f s $n = 4$ proměnnými, tj. $\chi = \{x_1,x_2,x_3,x_4\}$, v konjunktivní normální formě s $p = 2$ klauzulemi

$$K_1 = \{x_1,\bar{x}_3\},\quad K_2 = \{\bar{x}_2,x_3,\bar{x}_4\}. \tag{11.31}$$

Odpovídající instance LP bude mít trojúhelníkovou architekturu hloubky $d = 7$ a tréninkovou množinu

$$\mathcal{T} = \{(10101010,0),(11010010,1),(01001100,1)\}. \tag{11.32}$$

Dále se omezíme na případ, kdy např. $\{OR,\overline{AND}\} \subseteq \mathcal{F}$, protože důkaz pro $\{AND,\overline{OR}\} \subseteq \mathcal{F}$ je analogický. Stačí totiž znegovat požadované výstupy tréninkových vzorů z $\mathcal{T}$ v instanci LP a v následujícím popisu nahradit OR, $\overline{AND}$ po řadě jejich negacemi $\overline{OR}$, AND.

Částečnou konfiguraci $\ell_{V_1} : V_1 \longrightarrow \mathcal{F}$ první vrstvy $V_1 \subseteq V$ obvodu budeme značit pomocí hradlových funkcí

$$f_1,g_1,f_2,g_2,\ldots,f_{n-1},g_{n-1},f_n \in \mathcal{F} \tag{11.33}$$

a odpovídající výstupy hradel z V_1 pro daný vstup $\mathbf{x} = x_1,\ldots,x_{2n} \in \{0,1\}^{2n}$ obvodu pomocí vektoru

$$\begin{aligned} \mathbf{y}_{V_1}(\mathbf{x}) &= (f_1(x_1,x_2),g_1(x_2,x_3),\ldots \\ &\quad \ldots,g_{n-1}(x_{2n-2},x_{2n-1}),f_n(x_{2n-1},x_{2n})) \in \{0,1\}^{2n-1}. \end{aligned} \tag{11.34}$$

Nechť tedy nejprve $\ell : V \longrightarrow \mathcal{F}$ je konfigurace taková, že $C_\ell^A \models \mathcal{T}$, a tomu odpovídá částečná konfigurace $\ell_{V_1} \cong \ell$, kterou značíme (11.33). Podle (11.27) platí

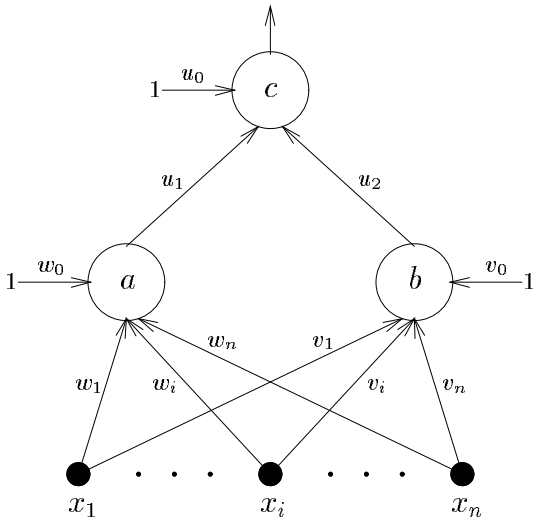
$$\mathbf{y}_{V_1}((10)^n) \neq \mathbf{y}_{V_1}(\mathbf{x}) \quad (\mathbf{x},1) \in \mathcal{T} \tag{11.35}$$

pro všechny tréninkové vzory s jednotkovým výstupem, protože pro vstup $(10)^n$ je požadován nulový výstup. Z konstrukce (11.28) je také zřejmé, že hradlové funkce g_i pro $i = 1,\ldots,n-1$ nemají na vztah (11.35) žádný vliv, protože pro jejich možné vstupy 1,0 nebo 0,1 tréninkových vzorů z $\mathcal{T}$ je $g_i(1,0) = g_i(0,1)$ $(i = 1,\ldots,n-1)$ kvůli symetrii $g_i \in \mathcal{F} \subseteq \{0,1,OR,AND,\overline{OR},\overline{AND}\}$. Tedy podmínka (11.35) musí být splněna pomocí hradlových funkcí $f_1,\ldots,f_n$. Uvažujme příklad (11.32), kde vztah (11.35) lze přepsat následujícím způsobem:

$$(f_1(1,0) \neq f_1(1,1) \text{ nebo } f_3(1,0) \neq f_3(0,0)) \text{ a} \tag{11.36}$$

$$(f_2(1,0) \neq f_2(0,0) \text{ nebo } f_3(1,0) \neq f_3(1,1) \text{ nebo } f_4(1,0) \neq f_4(0,0)).$$

(viz obrázek 11.6), je NP-úplný. V tomto důkazu je naopak velikost architektury (3) a počet výstupů (1) konstantní, zatímco počet tréninkových vzorů a vstupů roste. Tento výsledek platí pro třídu booleovských hradlových funkcí, které jsou prahové, tj. pro vícevrstvé perceptrony. Uvažovaná architektura představuje vlastně nejjednodušší vícevrstvou síť s pravidelnou topologií. Nejprve zavedeme následující značení.



Obr. 11.6: Architektura prahové 3-sítě.

Definice 11.15 *Řekneme, že obvod $C = (V, X, Y, E, \ell)$ je* 3-síť, *pokud pro architekturu $A = (V, X, Y, E)$ platí, že $V = \{a, b, c\}$, $X = \{x_1, \ldots, x_n\}$, $Y = \{c\}$ a*

$$E = \{(x_i, a), (x_i, b) \,|\, i = 1, \ldots, n\} \cup \{(a, c), (b, c)\}\,. \tag{11.40}$$

Nejprve uvažujeme prahovou 3-síť, *kde C je prahový obvod. Označme reprezentace příslušných booleovských prahových funkcí:*

$$\ell(a) = (w_1, \ldots, w_n; -w_0) \tag{11.41}$$
$$\ell(b) = (v_1, \ldots, v_n; -v_0) \tag{11.42}$$
$$\ell(c) = (u_1, u_2; -u_0)\,, \tag{11.43}$$

Navíc víme, že nekonstantní funkce $f_i \in \mathcal{F} \subseteq \{0, 1, OR, AND, \overline{OR}, \overline{AND}\}$ $(i = 1, \ldots, n)$ jsou monotónní, tj. $f_i(0,0) \neq f_i(1,1)$. Tedy po označení např. $\mathcal{F}_A = \{AND, \overline{AND}\}$ ze vztahu (11.36) dostáváme:

$$(f_1 \in \mathcal{F}_A \text{ nebo } f_3 \notin \mathcal{F}_A) \text{ a} \tag{11.37}$$
$$(f_2 \notin \mathcal{F}_A \text{ nebo } f_3 \in \mathcal{F}_A \text{ nebo } f_4 \notin \mathcal{F}_A)\,,$$

což odpovídá sémantice původní formule (11.31) příslušné instance SAT, pokud x_i chápeme jako „$f_i \in \mathcal{F}_A$“. Obecně definujeme ohodnocení $\omega : \chi \longrightarrow \{0,1\}$ proměnných f tak, že

$$\omega(x_i) = \begin{cases} 1 & f_i \in \mathcal{F}_A \\ 0 & \text{jinak} \end{cases} \qquad i = 1, \ldots, n\,, \tag{11.38}$$

které splňuje f, a tedy f je splnitelná.

Nechť naopak f je splnitelná a ohodnocení $\omega : \chi \longrightarrow \{0,1\}$ proměnných dosvědčuje splnitelnost f. Definujeme konfiguraci $\ell : V \longrightarrow \mathcal{F}$ ($\{OR, \overline{AND}\} \subseteq \mathcal{F}$), tj. nejprve ℓ_{V_1}:

$$f_i \equiv \begin{cases} \overline{AND} & \omega(x_i) = 1 \\ OR & \omega(x_i) = 0 \end{cases} \qquad i = 1, \ldots, n\,, \tag{11.39}$$
$$g_i \equiv \overline{AND} \quad i = 1, \ldots, n-1\,.$$

Dále ve druhé vrstvě hradla počítají funkci $\overline{AND}$ a ve zbylých vrstvách OR. Podle (11.39) platí $y_{V_1}((10)^n) = 1^{2n-1}$ a z ekvivalence (11.37) a (11.35) dostaneme, že $y_{V_1}(\mathbf{x}) \neq 1^{2n-1}$ pro $(\mathbf{x}, 1) \in \mathcal{T}$. Z uvedeného vyplývá, že výpočet obvodu s definovanou konfigurací ℓ v dalších vrstvách vede k požadovaným výstupům, tj. $C_\ell^A \models \mathcal{T}$, a tedy $(A, \mathcal{T}) \in Perf_{\mathcal{F}}$. □

11.3 Neuronové sítě se 3 neurony

V této podkapitole se budeme zabývat jiným typem důkazu NP-úplnosti LP než v podkapitole 11.1. V důkazu věty 11.3 je instance těžkého problému $3SAT$ zakódována do nepravidelné architektury obvodu, a proto v tomto případě NP-úplnost LP je pravděpodobně dána komplikovanou pevnou architekturou, která je na vstupu učícího algoritmu. V praxi se však obvykle používají vícevrstvé sítě, které mají pravidelnou topologii. Uvedený výsledek také např. nezávisí na volbě třídy hradlových funkcí (viz důsledek 11.4 a poznámku k větě 11.5). Navíc v uvedeném Juddově důkazu roste velikost obvodu a počet výstupních hradel, zatímco počet tréninkových vzorů (3) a vstupů (2) je konstantní.

Budeme prezentovat alternativní důkaz Avrima Bluma a Ronalda Rivesta [38], že LP pro tzv. *3-sítě*, tj. pro dvouvrstvé sítě jen se třemi neurony

$(A, \mathcal{T}) \in LP$ má řešení. Architektura A prahové 3-sítě C je dána počtem vstupů $n = |Q|$ a tréninková množina $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$ se skládá z $|\mathcal{T}| = n+p+1$ vzorů, kde

$$\begin{aligned} \mathcal{T}_0 &= \{(0^{i-1}10^{n-i}, 0) \,|\, i = 1, \ldots, n\} \\ \mathcal{T}_1 &= \{(0^n, 1)\} \cup \\ &\quad \cup \{(x_{j1} \ldots x_{jn}, 1) \,|\, x_{ji} = 1 \leftrightarrow q_i \in r_j,\ i = 1, \ldots, n;\ j = 1, \ldots, p\}\,. \end{aligned} \tag{11.48}$$

Uvedený formální zápis ilustrujeme na příkladě. Uvažujeme množinu $Q = \{q_1, q_2, q_3\}$ s $n = 3$ prvky a systém $R = \{r_1, r_2\}$ s $p = 2$ jejími podmnožinami:

$$r_1 = \{q_1, q_2\},\ \ r_2 = \{q_2, q_3\}\,. \tag{11.49}$$

Odpovídající instance LP bude mít architekturu 3-sítě s $n = 3$ vstupy a tréninkovou množinu $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$ s $|\mathcal{T}| = 6$ vzory, kde

$$\mathcal{T}_0 = \{(100, 0), (010, 0), (001, 0)\} \tag{11.50}$$

$$\mathcal{T}_1 = \{(000, 1), (110, 1), (011, 1)\}\,. \tag{11.51}$$

Nechť tedy Q_1, Q_2 je řešení SSP. Definujeme konfiguraci ℓ prahové 3-sítě C, tj. reprezentace booleovských prahových funkcí hradel a, b:

$$w_0 = 1 \qquad w_i = \begin{cases} -2 & q_i \in Q_1 \\ 2n & q_i \notin Q_1 \end{cases} \qquad i = 1, \ldots, n \tag{11.52}$$

$$v_0 = 1 \qquad v_i = \begin{cases} -2 & q_i \in Q_2 \\ 2n & q_i \notin Q_2 \end{cases} \qquad i = 1, \ldots, n\,. \tag{11.53}$$

Ověříme, že $C_\ell^A \models \mathcal{T}$. Pro $q_i \in Q_1$ podle (11.52) platí, že $\xi_a(0^{i-1}10^{n-i}) = w_0 + w_i = -1 < 0$ pro tréninkový vzor $(0^{i-1}10^{n-i}, 0) \in \mathcal{T}_0$, tj. $y_a(0^{i-1}10^{n-i}) = 0$, a proto $y_c(0^{i-1}10^{n-i}) = 0$ se shoduje s požadovaným nulovým výstupem. Podobně pro $q_i \in Q_2$ podle (11.53) platí, že $\xi_b(0^{i-1}10^{n-i}) = v_0 + v_i = -1 < 0$ pro tréninkový vzor $(0^{i-1}10^{n-i}, 0) \in \mathcal{T}_0$, tj. $y_b(0^{i-1}10^{n-i}) = 0$, a proto $y_c(0^{i-1}10^{n-i}) = 0$ se shoduje s požadovaným nulovým výstupem. To znamená, že $C_\ell^A \models \mathcal{T}_0$. Zřejmě pro tréninkový vzor $(0^n, 1) \in \mathcal{T}_1$ platí podle (11.52) a (11.53), že $\xi_a(0^n) = \xi_b(0^n) = 1$, a tedy i $y_a(0^n) = y_b(0^n) = y_c(0^n) = 1$. Pro tréninkový vzor $(x_{j1} \ldots x_{jn}, 1) \in \mathcal{T}_1$ odpovídající podmnožině $r_j \in R$ platí podle (11.52), že $\xi_a(x_{j1} \ldots x_{jn}) = w_0 + \sum_{q_i \in r_j} w_i \geq 0$, tj. $y_a(x_{j1} \ldots x_{jn}) = 1$, protože díky $r_j \not\subseteq Q_1$ existuje aspoň jedno $q_i \in r_j \setminus Q_1$. Podobně platí $\xi_b(x_{j1} \ldots x_{jn}) = v_0 + \sum_{q_i \in r_j} v_i \geq 0$ podle (11.53), tj. $y_b(x_{j1} \ldots x_{jn}) = 1$, protože díky $r_j \not\subseteq Q_2$ existuje aspoň jedno $q_i \in r_j \setminus Q_2$, tedy $y_c(x_{j1} \ldots x_{jn}) = 1$. Z toho vyplývá, že $C_\ell^A \models \mathcal{T}_1$, a tedy $C_\ell^A \models \mathcal{T}$.

zvážené sumy $\xi_j(\mathbf{x})$ (včetně opačného prahu) a výstupy $y_j(\mathbf{x})$ hradel $j \in V$ pro vstup $\mathbf{x} \in \{0,1\}^n$:

$$\xi_a(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i \qquad y_a(\mathbf{x}) = \sigma(\xi_a(\mathbf{x})) \tag{11.44}$$

$$\xi_b(\mathbf{x}) = v_0 + \sum_{i=1}^{n} v_i x_i \qquad y_b(\mathbf{x}) = \sigma(\xi_b(\mathbf{x})) \tag{11.45}$$

$$\xi_c(\mathbf{x}) = u_0 + u_1 y_a(\mathbf{x}) + u_2 y_b(\mathbf{x}) \qquad y_c(\mathbf{x}) = \sigma(\xi_c(\mathbf{x}))\,, \tag{11.46}$$

kde σ je aktivační funkce ostrá nelinearita (1.7):

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0 \\ 0 & \xi < 0\,. \end{cases} \tag{11.47}$$

Architektura prahové 3-sítě je znázorněna na obrázku 11.6.

Intuitivně nejprve naznačíme, proč by LP pro prahové 3-sítě měl být těžký. Předpokládejme pro jednoduchost, že $\ell(c) = AND$. Pro tréninkový vzor $(\mathbf{x}, 1) \in \mathcal{T}$ s požadovaným jednotkovým výstupem musí být obě hradla a, b aktivní, tj. $y_a(\mathbf{x}) = y_b(\mathbf{x}) = 1$. V opačném případě, kdy požadovaný výstup tréninkového vzoru je nulový, obecně existují tréninkové vzory $(\mathbf{x}, 0) \in \mathcal{T}$, pro které je $y_a(\mathbf{x}) \neq y_b(\mathbf{x})$, jinak by stačilo jen jedno prahové hradlo pro řešení LP. Avšak v takovém případě má učící algoritmus exponenciální počet možných voleb $y_a(\mathbf{x}) \in \{0,1\}$ $(y_a \neq y_b)$ pro tréninkové vzory $(\mathbf{x}, 0) \in \mathcal{T}$. Nyní již zformulujeme příslušnou větu.

Věta 11.16 (Blum, Rivest [38]) *LP pro prahové 3-sítě je NP-úplný problém.*

Důkaz: Důkaz, že LP je ve třídě NP, probíhá stejně jako v části (ii) důsledku 11.4. Nejprve ukážeme NP-těžkost LP pro $\ell(c) = AND$, tj. např. $u_1 = 1$, $u_2 = 1$ a $u_0 = -2$ (viz větu 6.26), a později důkaz zobecníme pro obecnou booleovskou prahovou funkci $\ell(c)$. Postupujeme tak, že následující NP-úplný *problém rozkladu množiny* [178] redukujeme v polynomiálním čase na LP.

Problém rozkladu množiny SSP (**S**et-**S**plitting **P**roblem):
instance: Konečná množina Q a soubor jejích podmnožin $R = \{r_j \subseteq Q \,|\, j = 1, \ldots, p\}$.
otázka: Existuje disjunktní rozklad množiny $Q = Q_1 \cup Q_2$ $(Q_1 \cap Q_2 = \emptyset)$ takový, že $r_j \not\subseteq Q_1$ a $r_j \not\subseteq Q_2$ pro všechna $j = 1, \ldots, p$?

Nechť tedy (Q, R) je instance SSP a označme $Q = \{q_1, \ldots, q_n\}$, $R = \{r_j \subseteq Q \,|\, j = 1, \ldots, p\}$. V polynomiálním čase zkonstruujeme odpovídající instanci $(A, \mathcal{T})$ problému LP takovou, že $(Q, R) \in SSP$ má řešení, právě když

Nechť naopak ℓ je řešení LP, tj. $C_\ell^A \models \mathcal{T}$. Definujeme disjunktní rozklad množiny $Q = Q_1 \cup Q_2$ následujícím způsobem:

$$Q_1 = \{q_i \in Q \mid w_0 + w_i < 0\} \tag{11.54}$$
$$Q_2 = \{q_i \in Q \mid v_0 + v_i < 0\} \setminus Q_1\,. \tag{11.55}$$

Zřejmě $w_0 \geq 0$ a $v_0 \geq 0$, protože $(0^n, 1) \in \mathcal{T}_1$. Dále uvažujme $r_j \in R$. Kdyby $r_j \subseteq Q_1$, pak by podle (11.54) platilo $\xi_a(x_{j1} \ldots x_{jn}) = w_0 + \sum_{q_i \in r_j} w_i < 0$ pro odpovídající tréninkový vzor $(x_{j1} \ldots x_{jn}, 1) \in \mathcal{T}_1$, tj. $y_a(x_{j1} \ldots x_{jn}) = 0$, a tedy $y_c(x_{j1} \ldots x_{jn}) = 0$, což je ve sporu s požadovaným jednotkovým výstupem tohoto vzoru, a proto $r_j \not\subseteq Q_1$. Podobně $r_j \not\subseteq Q_2$. To znamená, že SSP má řešení $Q = Q_1 \cup Q_2$.

Nyní zobecníme předchozí důkaz NP-těžkosti LP pro obecnou booleovskou prahovou funkci $\ell(c)$. Stačí modifikovat konstrukci (11.48) instance LP tak, že nová architektura A' 3-sítě bude mít $n+3$ vstupů, dále původní tréninkové vzory z $\mathcal{T}$ budou mít 3 nové vstupy nulové a navíc přidáme tréninkové vzory tak, aby nová tréninková množina $\mathcal{T}' = \mathcal{T}_0' \cup \mathcal{T}_1'$ vynutila $\ell(c) = AND$:

$$\mathcal{T}_0' = \mathcal{T}_0 \cup \{(0^n 100, 0), (0^n 010, 0), (0^n 001, 0), (0^n 111, 0)\} \tag{11.56}$$
$$\mathcal{T}_1' = \mathcal{T}_1 \cup \{(0^n 101, 1), (0^n 011, 1)\}\,.$$

Konzistentní konfiguraci (11.52) a (11.53), kde navíc $\ell(c) = AND$, pak dodefinujeme pro 3 nové vstupy následujícím způsobem:

$$w_{n+1} = w_{n+2} = 2, \qquad w_{n+3} = -2 \tag{11.57}$$
$$v_{n+1} = v_{n+2} = -2, \qquad v_{n+3} = 2\,. \tag{11.58}$$

Naopak nechť konfigurace $\ell \models \mathcal{T}'$ je konzistentní s tréninkovou množinou $\mathcal{T}'$. Ukážeme, že nové vzory (11.56) vynucují $\ell(c) = AND$. Nejprve si uvědomíme, že jedno prahové hradlo tyto nové vzory nerozliší. Kdyby tomu tak bylo např. u hradla a, pak by podle (11.56) platilo

$$y_a(0^{n+3}) = y_a(0^n 101) \neq y_a(0^n 100) = y_a(0^n 001) \tag{11.59}$$

a stejná znaménka sčítanců na levé straně rovnice (11.60) by se lišila od stejných znamének sčítanců na pravé straně:

$$\xi_a(0^{n+3}) + \xi_a(0^n 101) = 2w_0 + w_{n+1} + w_{n+3} = \xi_a(0^n 100) + \xi_a(0^n 001)\,, \tag{11.60}$$

což je spor.

Dále ukážeme, že nemůže zároveň nastat:

$$y_a(0^n 100) = y_a(0^n 010) = y_a(0^n 001) = y_a(0^n 111) \tag{11.61}$$
$$y_b(0^n 100) = y_b(0^n 010) = y_b(0^n 001) = y_b(0^n 111)\,.$$

Naopak předpokládejme, že (11.61) platí. Nutně

$$y_a(0^{n+3}) \neq y_a(0^n 100) \text{ nebo } y_b(0^{n+3}) \neq y_b(0^n 100)\,, \tag{11.62}$$

protože $(0^{n+3}, 1) \in \mathcal{T}_1'$ a $(0^n 100, 0) \in \mathcal{T}_0'$. Zřejmě obě nerovnice ve formuli (11.62) neplatí současně, protože by pak podle podobné úvahy jako v (11.60), tj.

$$\xi_j(0^n 100) + \xi_j(0^n 001) = \xi_j(0^{n+3}) + \xi_j(0^n 101) \tag{11.63}$$

pro $j = a, b$, kde $\xi_j(0^n 100)$ a $\xi_j(0^{n+3})$ mají různá znaménka a $\xi_j(0^n 100)$ a $\xi_j(0^n 001)$ mají podle (11.61) stejná znaménka, platilo, že

$$y_a(0^n 101) = y_a(0^n 100) \text{ a } y_b(0^n 101) = y_b(0^n 100)\,, \tag{11.64}$$

což je ve sporu s konzistencí ℓ, protože $(0^n 101, 1) \in \mathcal{T}_1'$. Tedy nechť např.

$$y_a(0^{n+3}) = y_a(0^n 100) \text{ a } y_b(0^{n+3}) \neq y_b(0^n 100) \tag{11.65}$$

(opačný případ analogicky), čemuž odpovídá následující přepis (11.64):

$$y_a(0^n 101) \neq y_a(0^n 100) \text{ a } y_b(0^n 101) = y_b(0^n 100) \tag{11.66}$$

a podobně

$$y_a(0^n 011) \neq y_a(0^n 100) \text{ a } y_b(0^n 011) = y_b(0^n 100)\,. \tag{11.67}$$

Ze vztahů (11.61), (11.66) a (11.67) dostáváme

$$y_a(0^n 101) = y_a(0^n 011) \neq y_a(0^n 001) = y_a(0^n 111) \tag{11.68}$$

a opět aplikací známé úvahy stejná znaménka sčítanců na levé straně následující rovnice by se lišila od stejných znamének sčítanců na pravé straně:

$$\begin{aligned} \xi_a(0^n 101) + \xi_a(0^n 011) &= 2w_0 + w_{n+1} + w_{n+2} + 2w_{n+3} = \\ &= \xi_a(0^n 001) + \xi_a(0^n 111)\,, \end{aligned} \tag{11.69}$$

což je spor. Tedy (11.61) nemůže nastat.

Víme, že jedno prahové hradlo nové vzory nerozliší. Dále neplatí (11.61), tj. nové vzory s nulovým požadovaným výstupem nemají všechny y_a, resp. y_b stejné. Tedy jediná možná booleovská prahová funkce výstupního hradla c, která vyhovuje konzistenci ℓ, je $\ell(c) = AND$, popř. $\overline{OR}$, kterou však lze podle (*ii*) z lemmy 6.27 převést na AND. □

Větu 11.16 lze dokázat pro různé speciální i obecnější případy, jak ukazuje následující důsledek s obdobným důkazem, který neuvádíme.

11.4 Kaskádové architektury

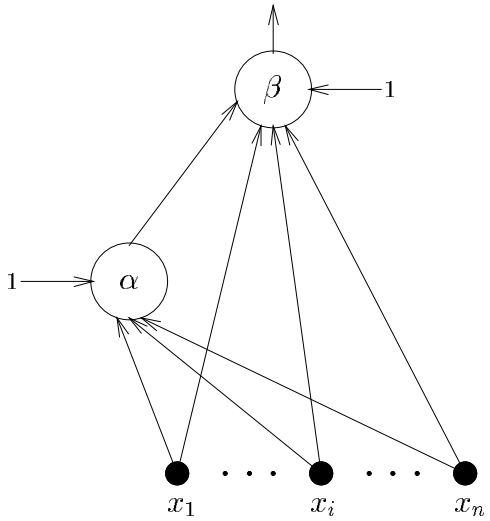
Dalším důsledkem věty 11.16 je NP-úplnost tréninkového problému pro tzv. *kaskádovou 2-síť*, která odpovídá nejjednodušší kaskádové neuronové síti (viz podkapitolu 2.4) jen se dvěma perceptrony.

Definice 11.20 *Řekneme, že prahový obvod $C = (V, X, Y, E, \ell)$ je* kaskádová 2-síť, *pokud pro architekturu $A = (V, X, Y, E)$ platí, že $V = \{\alpha, \beta\}$, $X = \{x_1, \ldots, x_n\}$, $Y = \{\beta\}$ a*

$$E = \{(x_i, \alpha), (x_i, \beta) \,|\, i = 1, \ldots, n\} \cup \{(\alpha, \beta)\}\,. \tag{11.72}$$

Podobně jako v definici 11.15 zváženou sumu (včetně opačného prahu) značíme $\xi_j(\mathbf{x})$ a výstup $y_j(\mathbf{x})$ pro hradlo $j \in V$ a vstup $\mathbf{x} \in \{0,1\}^n$.

Architektura kaskádové 2-sítě je znázorněna na obrázku 11.7.



Obr. 11.7: Architektura kaskádové 2-sítě.

Věta 11.21 (Lin, Vitter [173]) *LP pro kaskádové 2-sítě je NP-úplný problém.*

Důkaz: Důkaz, že LP je ve třídě NP, probíhá stejně jako v části (*ii*) důsledku 11.4. Důkaz NP-těžkosti LP pro kaskádové 2-sítě provedeme polynomiální redukcí LP pro prahové 3-sítě s $\ell(c) = AND$, který je podle důkazu

Důsledek 11.17 (Blum, Rivest [38])

(i) *LP pro prahové 3-sítě je NP-úplný problém, i když platí jedna z následujících podmínek:*

- *$w_i = v_i \in \{-1, 1\}$ pro $i = 1, \ldots, n$.*
- *$\ell(c) = XOR$ a vstup $\mathbf{x} \in \{-1, 0, 1\}^n$.*

(ii) *LP pro obvod s polynomiálním počtem prahových hradel v první vrstvě vzhledem k počtu vstupů a s jedním výstupním AND hradlem ve druhé vrstvě (prahová 3-síť s $\ell(c) = AND$ je jeho speciální případ) je NP-úplný problém.*

Nyní upozorníme na zajímavou skutečnost, že složitost LP závisí na zakódování vstupních hodnot. Nejprve zavedeme následující značení.

Definice 11.18 *Označme $\mathbf{T}_N$ třídu tréninkových množin, které jsou konzistentní s prahovým hradlem s n vstupy, a $\mathbf{T}_{3NX}$ třídu tréninkových množin konzistentních s prahovou 3-sítí s n vstupy a $\ell(c) = XOR$. Podobně $\mathbf{T}_{N^2}$ je třída tréninkových množin, které jsou konzistentní s prahovým hradlem s $2n + n(n-1)/2$ vstupy, které odpovídají n původním vstupům, n jejich mocninám a $n(n-1)/2$ součinům všech různých dvojic.*

Lemma 11.19 (Blum, Rivest [38]) $\mathbf{T}_N \subseteq \mathbf{T}_{3NX} \subseteq \mathbf{T}_{N^2}$.

Důkaz: Zřejmě $\mathbf{T}_N \subseteq \mathbf{T}_{3NX}$. Také důkaz $\mathbf{T}_{3NX} \subseteq \mathbf{T}_{N^2}$ je snadný. Pro tréninkový vzor $(\mathbf{x}, 1) \in \mathcal{T} \in \mathbf{T}_{3NX}$ platí

$$\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)\left(v_0 + \sum_{i=1}^{n} v_i x_i\right) < 0\,, \tag{11.70}$$

což lze přepsat

$$\begin{aligned} -w_0 v_0 \quad &+ \quad \sum_{i=1}^{n} -(w_0 v_i + v_0 w_i) x_i + \sum_{i=1}^{n} -v_i w_i x_i^2 + \\ &+ \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} -(w_i v_j + v_i w_j) x_i x_j > 0\,, \end{aligned} \tag{11.71}$$

a tedy $(\mathbf{x}, 1) \in \mathcal{T} \in \mathbf{T}_{N^2}$. □

Zřejmě LP pro $\mathbf{T}_N, \mathbf{T}_{N^2}$ je řešitelný pomocí lineárního programování v polynomiálním čase [145], zatímco LP pro třídu $\mathbf{T}_{3NX}$, která se podle lemmy 11.19 nachází v uspořádání podle inkluze mezi nimi, je podle (*i*) z důsledku 11.17 NP-úplný problém. Tato zajímavá skutečnost je pravděpodobně dána tím, že tréninkové vzory z $\mathbf{T}_{N^2}$ jsou předzpracovány, a proto je pro ně LP lehčí.

věty 11.16 NP-úplný. Nechť tedy $(A,\mathcal{T})$ je instance LP pro prahovou 3-síť s n vstupy a $\ell(c)=AND$, přitom využijeme značení z definice 11.15. V polynomiálním čase zkonstruujeme instanci $(A',\mathcal{T}')$ tréninkového problému pro kaskádové 2-sítě (viz definici 11.20) s $n+2$ vstupy takovou, že $(A,\mathcal{T})$ má řešení, právě když $(A',\mathcal{T}')$ má řešení. Tréninková množina $\mathcal{T}'$ pro A' má následující tvar:

$$\mathcal{T}'=\{(\mathbf{x}00,d),(\mathbf{x}10,0),(\mathbf{x}01,0),(\mathbf{x}11,1)\,|\,(\mathbf{x},d)\in\mathcal{T}\}\,. \tag{11.73}$$

Nechť nejprve ℓ je konfigurace prahové 3-sítě (viz definici 11.15) taková, že $C_\ell^A\models\mathcal{T}$. Definujeme konfiguraci ℓ' pro odpovídající kaskádovou 2-síť, tj. reprezentace booleovských prahových funkcí hradel α a β:

$$\begin{aligned}\ell(\alpha) &= (w_1,\ldots,w_n,-W-|w_0|,-W-|w_0|;-w_0) &(11.74)\\ \ell(\beta) &= (v_1,\ldots,v_n,2V,2V,3V+v_0;3V)\,, &(11.75)\end{aligned}$$

kde volíme $W>\sum_{i=1}^n|w_i|$, $V>\sum_{i=1}^n|v_i|$ a $w(\alpha,\beta)=3V+v_0$ je váha spoje (α,β). Nyní je snadné ověřit, že $C_{\ell'}^{A'}\models\mathcal{T}'$. Zřejmě $y_\alpha(\mathbf{x}00)=y_a(\mathbf{x})$ a

$$\xi_\beta(\mathbf{x}00)=-3V+(\xi_b(\mathbf{x})-v_0)+(3V+v_0)y_\alpha(\mathbf{x}00)\,, \tag{11.76}$$

tj. pro $y_a(\mathbf{x})=1$ je $\xi_\beta(\mathbf{x}00)=\xi_b(\mathbf{x})$, a tedy $y_\beta(\mathbf{x}00)=y_b(\mathbf{x})$. To pro $y_a(\mathbf{x})=1$ díky $\ell(c)=AND$ znamená, že $y_c(\mathbf{x})=y_\beta(\mathbf{x}00)$. V případě, že $y_a(\mathbf{x})=0$, tj. $y_c(\mathbf{x})=0$, je podle (11.76) $\xi_\beta(\mathbf{x}00)<0$, tj. $y_\beta(\mathbf{x}00)=0$. Dále $\xi_\alpha(\mathbf{x}10)=\xi_\alpha(\mathbf{x}01)=\xi_a(\mathbf{x})-W-|w_0|<0$, tj. $y_\alpha(\mathbf{x}10)=y_\alpha(\mathbf{x}01)=0$, a $\xi_\beta(\mathbf{x}10)=\xi_\beta(\mathbf{x}01)=-3V+(\xi_b(\mathbf{x})-v_0)+2V<0$, tj. $y_\beta(\mathbf{x}10)=y_\beta(\mathbf{x}01)=0$ se shoduje s požadovaným nulovým výstupem. Nakonec $\xi_\alpha(\mathbf{x}11)=\xi_a(\mathbf{x})-2W-2|w_0|<0$, tj. $y_\alpha(\mathbf{x}11)=0$, a $\xi_\beta(\mathbf{x}11)=-3V+(\xi_b(\mathbf{x})-v_0)+4V\geq 0$, tj. $y_\beta(\mathbf{x}11)=1$ se shoduje s požadovaným jednotkovým výstupem.

Nechť naopak ℓ' je konfigurace pro kaskádovou 2-síť taková, že $C_{\ell'}^{A'}\models\mathcal{T}'$. Označme

$$\begin{aligned}\ell'(\alpha) &= (w_1,\ldots,w_n,w_{n+1},w_{n+2};-w_0) &(11.77)\\ \ell'(\beta) &= (v_1,\ldots,v_n,v_{n+1},v_{n+2},v;-v_0)\,,\end{aligned}$$

kde $w(\alpha,\beta)=v$ je váha spoje (α,β). Dále definujeme následující dvě množiny vstupů pro A':

$$\begin{aligned}Q_0 &= \left\{\mathbf{x}x_{n+1}x_{n+2}\in\{0,1\}^{n+2}\;\middle|\;v_0+\sum_{i=1}^{n+2}v_ix_i\geq 0,\,(\mathbf{x},d)\in\mathcal{T}\right\}\\ Q_1 &= \left\{\mathbf{x}x_{n+1}x_{n+2}\in\{0,1\}^{n+2}\;\middle|\;v_0+\sum_{i=1}^{n+2}v_ix_i+v\geq 0,\,(\mathbf{x},d)\in\mathcal{T}\right\}.\end{aligned}$$

Nejprve předpokládejme, že $v\geq 0$. Odtud dostáváme následující implikace:

$$\begin{aligned}\mathbf{x}x_{n+1}x_{n+2}\in Q_0 &\rightarrow (\mathbf{x}x_{n+1}x_{n+2},1)\in\mathcal{T}' &(11.78)\\ \mathbf{x}x_{n+1}x_{n+2}\notin Q_1 &\rightarrow (\mathbf{x}x_{n+1}x_{n+2},0)\in\mathcal{T}' &(11.79)\\ (\mathbf{x}x_{n+1}x_{n+2},1)\in\mathcal{T}' &\rightarrow \mathbf{x}x_{n+1}x_{n+2}\in Q_0\vee &(11.80)\\ &\quad\vee(y_\alpha(\mathbf{x}x_{n+1}x_{n+2})=1\wedge\mathbf{x}x_{n+1}x_{n+2}\in Q_1)\,.\end{aligned}$$

Pokud pro všechna $(\mathbf{x},1)\in\mathcal{T}$ je $\mathbf{x}00\in Q_0$, pak definujeme následující konfiguraci ℓ pro prahovou 3-síť s $\ell(c)=AND$:

$$\ell(a)=\ell(b)=(v_1,\ldots,v_n;-v_0)\,. \tag{11.81}$$

Zřejmě v tomto případě pro $(\mathbf{x},1)\in\mathcal{T}$ platí $y_a(\mathbf{x})=y_b(\mathbf{x})=1$, tj. $y_c(\mathbf{x})=1$, a $(\mathbf{x},0)\in\mathcal{T}$ implikuje $(\mathbf{x}00,0)\in\mathcal{T}'$, pro které platí $\xi_\beta(\mathbf{x}00)=v_0+\sum_{i=1}^n v_ix_i+vy_\alpha(\mathbf{x}00)<0$, z čehož díky $v\geq 0$ vyplývá, že $\xi_a(\mathbf{x})=\xi_b(\mathbf{x})=v_0+\sum_{i=1}^n v_ix_i<0$, tj. $y_a(\mathbf{x})=y_b(\mathbf{x})=y_c(\mathbf{x})=0$ pro $(\mathbf{x},0)\in\mathcal{T}$. Odtud $C_\ell^A\models\mathcal{T}$.

Pokud naopak existuje aspoň jeden tréninkový vzor $(\mathbf{x}',1)\in\mathcal{T}$ takový, že $\mathbf{x}'00\notin Q_0$, pak podle (11.80) platí současně $y_\alpha(\mathbf{x}'00)=1$ a $\mathbf{x}'00\in Q_1$. Ukážeme sporem, že v takovém případě to platí již pro všechny $(\mathbf{x},1)\in\mathcal{T}$, tj.

$$(y_\alpha(\mathbf{x}00)=1)\wedge(\mathbf{x}00\in Q_1)\quad(\mathbf{x},1)\in\mathcal{T}\,. \tag{11.82}$$

Nechť tedy (11.82) neplatí pro nějaký vzor $(\tilde{\mathbf{x}},1)\in\mathcal{T}$. Podle (11.80) pak $\tilde{\mathbf{x}}00\in Q_0$, což spolu s $(\tilde{\mathbf{x}}01,0),(\tilde{\mathbf{x}}01,0)\in\mathcal{T}'$ implikuje $v_{n+1},v_{n+2}<0$, a tedy $\tilde{\mathbf{x}}11\notin Q_0$. Avšak $(\tilde{\mathbf{x}}11,1)\in\mathcal{T}'$, a proto podle (11.80) platí $y_\alpha(\tilde{\mathbf{x}}11)=1$ a $\tilde{\mathbf{x}}11\in Q_1$, což spolu s $v_{n+1},v_{n+2}<0$ implikuje $\tilde{\mathbf{x}}01,\tilde{\mathbf{x}}10\in Q_1$. Z $\tilde{\mathbf{x}}10\in Q_1$ plyne $v_0+\sum_{i=1}^n v_i\tilde{x}_i+v_{n+1}+v\geq 0$, ale díky $(\tilde{\mathbf{x}}10,0)\in\mathcal{T}'$ dostáváme $\xi_\beta(\tilde{\mathbf{x}}10)=v_0+\sum_{i=1}^n v_i\tilde{x}_i+v_{n+1}+vy_\alpha(\tilde{\mathbf{x}}10)<0$, a tedy nutně $y_\alpha(\tilde{\mathbf{x}}10)=0$, tj. $\xi_\alpha(\tilde{\mathbf{x}}10)=w_0+\sum_{i=1}^n w_i\tilde{x}_i+w_{n+1}<0$. Podobně z $\tilde{\mathbf{x}}01\in Q_1$ vyplývá $y_\alpha(\tilde{\mathbf{x}}01)=0$, tj. $\xi_\alpha(\tilde{\mathbf{x}}01)=w_0+\sum_{i=1}^n w_i\tilde{x}_i+w_{n+2}<0$. Na druhou stranu víme, že $y_\alpha(\tilde{\mathbf{x}}11)=1$, tj. $\xi_\alpha(\tilde{\mathbf{x}}11)=w_0+\sum_{i=1}^n w_i\tilde{x}_i+w_{n+1}+w_{n+2}\geq 0$, z čehož dostáváme $w_{n+1},w_{n+2}>0$.

Dále z $\mathbf{x}'00\notin Q_0$ díky $v_{n+1},v_{n+2}<0$ plyne také $\mathbf{x}'11\notin Q_0$. Avšak $(\mathbf{x}'11,1)\in\mathcal{T}'$, a tedy podle (11.80) $y_\alpha(\mathbf{x}'11)=1$ a $\mathbf{x}'11\in Q_1$, tj. $v_0+\sum_{i=1}^n v_ix_i'+v_{n+1}+v_{n+2}+v\geq 0$. Protože navíc $v_{n+1},v_{n+2}<0$, dostáváme $\mathbf{x}'01,\mathbf{x}'10\in Q_1$. Víme, že $y_\alpha(\mathbf{x}'00)=1$ a $w_{n+1},w_{n+2}>0$, a tedy také $y_\alpha(\mathbf{x}'01)=y_\alpha(\mathbf{x}'10)=1$, což spolu s $\mathbf{x}'01,\mathbf{x}'10\in Q_1$ implikuje $y_\beta(\mathbf{x}'01)=y_\beta(\mathbf{x}'10)=1$, což je spor s $C_{\ell'}^{A'}\models\mathcal{T}'$. Tím jsme ukázali (11.82), a můžeme definovat konfiguraci ℓ prahové 3-sítě s $\ell(c)=AND$:

$$\begin{aligned}\ell(a) &= (w_1,\ldots,w_n;-w_0) &(11.83)\\ \ell(b) &= (v_1,\ldots,v_n;-v_0-v)\,,\end{aligned}$$

vrstvě, jedním výstupním hradlem v druhé vrstvě a s třídou hradlových lineárních prahových funkcí (viz definici 6.3), LP je řešitelný v polynomiálním čase [57, 191]. Avšak příslušný algoritmus závisí exponenciálně na počtu vstupů a velikosti obvodu, a tedy je pro větší úlohy prakticky nepoužitelný.

V této podkapitole dále zobecníme důkaz věty 11.16 a ukážeme NP-těžkost LP pro analogovou prahovou 3-síť s aktivační funkcí standardní sigmoidou a s reálnými váhami, která odpovídá modelu neuronové sítě používané v učícím algoritmu backpropagation v praktických aplikacích. To představuje dolní odhad složitosti LP pro tento model neuronové sítě. Na druhou stranu je velmi pravděpodobné, že příslušný LP je rekurzivní [185] (horní odhad). Tvrzení následující věty také ještě vyžaduje technický předpoklad, který omezuje výstupní hradlovou funkci $\ell(c)$ a jehož odstranění je pravděpodobně technickou otázkou. Nicméně tento předpoklad je např. splněn, pokud výstupní prahové hradlo c má nulový práh, což je častý předpoklad. Uvedený výsledek potvrzuje praktické zkušenosti s neefektivitou učícího algoritmu backpropagation [270, 271] a naznačuje, že jej pravděpodobně nelze vylepšit, protože překážky spočívají v samotné složitosti tréninkového problému. Nejprve zformulujeme zmiňovaný technický předpoklad.

Definice 11.22 *Řekneme, že analogová prahová 3-síť (viz definici 7.42 a 11.15) splňuje* výstupní separační podmínku *pro reálné koeficienty* $\ell(c) = (u_1, u_2; -u_0)$ *lineární funkce výstupního hradla* c*, jestliže v případě, že čísla* u_0, $u_0 + u_1$, $u_0 + u_2$, $u_0 + u_1 + u_2$ *nemají všechna stejná znaménka, platí*

$$u_0(u_0 + u_1)(u_0 + u_2)(u_0 + u_1 + u_2) \leq 0\,. \tag{11.88}$$

Výstupní separační podmínka má jednoduchý geometrický význam. Říká, že přímka s rovnicí $u_0 + u_1 y_a + u_2 y_b = 0$ neodděluje ostře žádné dva body z množiny $\{[0,0],[1,0],[0,1],[1,1]\}$ od zbylých dvou. Zřejmě pro $u_0 = 0$ (přímka prochází počátkem $[0,0]$, který již proto nemůže být ostře oddělen) je tato podmínka splněna.

Věta 11.23 (Šíma [266]) *LP pro analogové prahové 3-sítě s aktivační funkcí standardní sigmoidou (1.9) a s nulovou separací (7.26) binárního výstupu (tj. $\ell(c)$ je booleovská prahová funkce), které splňují výstupní separační podmínku (speciálně $u_0 = 0$) z definice 11.22, je NP-těžký problém.*

Důkaz: Důkaz NP-těžkosti LP pro uvedené analogové prahové 3-sítě probíhá podobně jako v důkazu věty 11.16 polynomiální redukcí NP-úplného problému SSP na LP. Nechť tedy (Q, R) je instance SSP a označme $Q = \{q_1, \ldots, q_n\}$, $R = \{r_j \subseteq Q \mid j = 1, \ldots, p\}$. V polynomiálním čase zkonstruujeme odpovídající instanci $(A, \mathcal{T})$ problému LP takovou, že $(Q, R) \in SSP$ má řešení, právě když $(A, \mathcal{T}) \in LP$ má řešení. Architektura A analogové

pro kterou $y_a(\mathbf{x}) = y_b(\mathbf{x}) = y_c(\mathbf{x}) = 1$ pro každé $(\mathbf{x}, 1) \in \mathcal{T}$. Na druhou stranu $(\mathbf{x}, 0) \in \mathcal{T}$ implikuje $(\mathbf{x}00, 0) \in \mathcal{T}'$, pro které platí $\xi_\beta(\mathbf{x}00) = v_0 + \sum_{i=1}^{n} v_i x_i + v y_\alpha(\mathbf{x}00) < 0$. Pokud $y_\alpha(\mathbf{x}00) = 1$, pak $\xi_b(\mathbf{x}) = v_0 + v + \sum_{i=1}^{n} v_i x_i < 0$, tj. $y_b(\mathbf{x}) = 0$, a tedy $y_c(\mathbf{x}) = 0$. Pro $y_\alpha(\mathbf{x}00) = y_a(\mathbf{x}) = 0$ také $y_c(\mathbf{x}) = 0$. Odtud $C_\ell^A \not\models \mathcal{T}$.

V případě, že $v < 0$, postupujeme podobně. Modifikujeme definici triviální konfigurace (11.81):

$$\ell(a) = \ell(b) = (v_1, \ldots, v_n; -v_0 - v)\,. \tag{11.84}$$

Místo podmínky (11.82) dokážeme

$$(y_\alpha(\mathbf{x}00) = 0) \wedge (\mathbf{x}00 \in Q_0) \quad (\mathbf{x}, 1) \in \mathcal{T}\,, \tag{11.85}$$

což umožní upravit definici konfigurace (11.83) následovně:

$$\begin{aligned} \ell(a) &= (-w_1, \ldots, -w_n; w_0 + \eta) \\ \ell(b) &= (v_1, \ldots, v_n; -v_0)\,, \end{aligned} \tag{11.86}$$

kde volíme

$$0 < \eta \leq \min_{(\mathbf{x},1)\in\mathcal{T}} \left(-w_0 - \sum_{i=1}^{n} w_i x_i \right)\,. \tag{11.87}$$

□

11.5 Backpropagation není efektivní

Věta 11.16, resp. důsledek 11.17 ukazuje, že pravděpodobně neexistuje efektivní algoritmus pro učení vícevrstvých neuronových sítí s pravidelnou architekturou, které se často používají v praktických aplikacích. Tyto výsledky platí pro diskrétní neuronové sítě s perceptrony, které počítají booleovské prahové funkce, na jejichž vlastnosti se příslušné důkazy odkazují. Avšak v praxi se nejčastěji používá učící algoritmus backpropagation (viz podkapitolu 2.2), který vyžaduje diferencovatelnou (tj. spojitou) aktivační funkci. Proto někteří inženýři považovali uvedené výsledky za irelevantní (NP-úplnost ve větě 11.5 vychází spíše z nepravidelnosti architektury než z použité třídy hradlových funkcí) a snaha byla zobecnit větu 11.16 pro standardní sigmoidu (1.9). Např. Klaus-U. Höffgen dokázal, že LP pro analogovou prahovou 3-síť (viz definice 7.42 a 11.15) s jednotkovými váhami je NP-úplný problém [116]. Nebo autoři článku [57] ukázali NP-úplnost LP pro analogové prahové 3-sítě, resp. analogové kaskádové 2-sítě, se saturovanou lineární aktivační funkcí (1.8) a reálnými váhami.

Na druhou stranu, pokud připustíme obecně celočíselné (resp. racionální) vstupy a uvažujeme obvod s konstantním počtem vstupů a hradel v první

že $\xi_a(0^{i-1}10^{2n+1-i}) = w_0 + w_i = -2n+1$ a $\xi_b(0^{i-1}10^{2n+1-i}) = v_0 + v_i = -2n+1$ pro tréninkový vzor $(0^{i-1}10^{2n+1-i},0) \in \mathcal{T}_0$ ($q_i \in Q_2$), tj. $\xi_c(0^{i-1}10^{2n+1-i}) = -2\sigma(-2n+1)+\sigma(-2n+1) < 0$, a proto podle (7.26) $y_c(0^{i-1}10^{2n+1-i}) = 0$ se shoduje s požadovaným nulovým výstupem. To znamená, že $C_\ell^A \models \mathcal{T}_0$. Zřejmě pro tréninkový vzor $(0^{2n+1},1) \in \mathcal{T}_1$ podle (11.91) platí, že $\xi_a(0^{2n+1}) = -1$ a $\xi_b(0^{2n+1}) = 1$, a tedy $\xi_c(0^{2n+1}) = -2\sigma(-1)+\sigma(1) \geq 0$, tj. $y_c(0^{2n+1}) = 1$ podle (7.26). Pro tréninkový vzor $(x_{j1}\ldots x_{jn}0^{n+1},1) \in \mathcal{T}_1$ odpovídající podmnožině $r_j \in R$ podle (11.91) platí, že $\xi_a(x_{j1}\ldots x_{jn}0^{n+1}) = w_0+\sum_{q_i\in r_j} w_i \leq -1+2(n-1)-2n+2 = -1$, protože díky $r_j \not\subseteq Q_1$ existuje aspoň jedno $q_i \in r_j \setminus Q_1$. Podobně podle (11.91) platí, že $\xi_b(x_{j1}\ldots x_{jn}0^{n+1}) = v_0 + \sum_{q_i\in r_j} v_i \geq 1-2n(n-1)+2n^2-2n = 1$, protože díky $r_j \not\subseteq Q_2$ existuje aspoň jedno $q_i \in r_j \setminus Q_2$. To znamená, že $\xi_c(x_{j1}\ldots x_{jn}0^{n+1}) \geq -2\sigma(-1)+\sigma(1) \geq 0$, což podle (7.26) implikuje $y_c(x_{j1}\ldots x_{jn}0^{n+1}) = 1$. Z toho vyplývá, že $C_\ell^A \models \mathcal{T}_1$.

Podobně z (11.91) vyplývá, že $\xi_a(0^{n+i-1}10^{n-i}1) = w_0+w_{n+i}+w_{2n+1} = -1$ a $\xi_b(0^{n+i-1}10^{n-i}1) = v_0+v_{n+i}+v_{2n+1} = 1$ pro tréninkový vzor $(0^{n+i-1}10^{n-i}1,1) \in \mathcal{T}_1'$ ($1 \leq i \leq n$), a tedy $\xi_c(0^{n+i-1}10^{n-i}1) = -2\sigma(-1)+\sigma(1) \geq 0$, což podle (7.26) implikuje $y_c(0^{n+i-1}10^{n-i}1) = 1$. To znamená, že $C_\ell^A \models \mathcal{T}_1'$. Dále pro $(0^{2n}1,0) \in \mathcal{T}_0'$ platí podle (11.91), že $\xi_a(0^{2n}1) = w_0+w_{2n+1} = 0$ a $\xi_b(0^{2n}1) = v_0+v_{2n+1} = 2n+1$, a tedy $\xi_c(0^{2n}1) = -2\sigma(0)+\sigma(2n+1) < 0$, z čehož podle (7.26) vyplývá, že $y_c(0^{2n}1) = 0$. Konečně pro tréninkový vzor $(0^n x_{j1}\ldots x_{jn}1,0) \in \mathcal{T}_0'$ odpovídající podmnožině $r_j \in R$ platí podle definice vah (11.91), že $\xi_a(0^n x_{j1}\ldots x_{jn}1) = w_0+\sum_{i\in r_j} w_{n+i}+w_{2n+1} = -|r_j|$ a $\xi_b(0^n x_{j1}\ldots x_{jn}1) = v_0+\sum_{i\in r_j} v_{n+i}+v_{2n+1} = 1-2n|r_j|+2n$. Ze vztahu $2 \leq |r_j| \leq n$ plyne $-1+2n|r_j|-2n-|r_j| > 0$, což implikuje $e^{|r_j|}(1-2e^{-1+2n|r_j|-2n-|r_j|}) < 0$, z čehož dále dostáváme $-2(1+e^{-(1-2n|r_j|+2n)})+(1+e^{|r_j|}) < 0$. Nakonec vydělíme výrazem $(1+e^{-(1-2n|r_j|+2n)})(1+e^{|r_j|}) > 0$ a můžeme uzavřít $\xi_c(0^n x_{j1}\ldots x_{jn}1) = -2\sigma(-|r_j|)+\sigma(1-2n|r_j|+2n) < 0$, tj. podle (7.26) $y_c(0^n x_{j1}\ldots x_{jn}1) = 0$ pro $(0^n x_{j1}\ldots x_{jn}1,0) \in \mathcal{T}_0'$. Proto $C_\ell^A \models \mathcal{T}_0'$, a tedy $C_\ell^A \models \mathcal{T}$.

Nechť naopak $C_\ell^A \models \mathcal{T}$ pro konfiguraci ℓ splňující výstupní separační podmínku. Protože $\mathcal{T}_0 \cup \mathcal{T}_0' \neq \emptyset$, $\mathcal{T}_1 \cup \mathcal{T}_1' \neq \emptyset$ a $0 < y_a, y_b < 1$, reálná čísla u_0, u_0+u_1, u_0+u_2, $u_0+u_1+u_2$ odpovídající po řadě hraničním bodům $\{[0,0],[1,0],[0,1],[1,1]\}$ nemají všechna stejná znaménka. Tedy podle (11.88) platí

$$u_0(u_0+u_1)(u_0+u_2)(u_0+u_1+u_2) \leq 0\,. \tag{11.92}$$

Omezíme se na případ, kdy mezi reálnými čísly u_0, u_0+u_1, u_0+u_2, $u_0+u_1+u_2$ je právě jedno kladné a zbylá tři čísla jsou nekladná. V tomto případě dále využijeme jen tréninkové vzory z $\mathcal{T}_0 \cup \mathcal{T}_1$ a jim odpovídající váhy (resp. prahy) $w_0,\ldots,w_n$, $v_0,\ldots,v_n$ a u_0,u_1,u_2. Opačný případ, kdy

prahové 3-sítě C je dána počtem vstupů $2n+1$ a tréninková množina $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_0' \cup \mathcal{T}_1'$ se skládá z $|\mathcal{T}| = 2(n+p+1)$ vzorů, kde

$$\begin{aligned}
\mathcal{T}_0 &= \{(0^{i-1}10^{2n+1-i},0) \mid i = 1,\ldots,n\} \\
\mathcal{T}_1 &= \{(0^{2n+1},1)\} \cup \\
&\quad \{(x_{j1}\ldots x_{jn}0^{n+1},1) \mid x_{ji} = 1 \leftrightarrow q_i \in r_j,\ i = 1,\ldots,n;\ j = 1,\ldots,p\} \\
\mathcal{T}_0' &= \{(0^{2n}1,0)\} \cup \\
&\quad \{(0^n x_{j1}\ldots x_{jn}1,0) \mid x_{ji} = 1 \leftrightarrow q_i \in r_j,\ i = 1,\ldots,n;\ j = 1,\ldots,p\} \\
\mathcal{T}_1' &= \{(0^{n+i-1}10^{n-i}1,1) \mid i = 1,\ldots,n\}\,.
\end{aligned} \tag{11.89}$$

Uvedený formální zápis ilustrujeme na příkladě z důkazu věty 11.16, kde instanci (11.49) problému SSP odpovídá instance $(A,\mathcal{T})$ problému LP, v níž architektura A 3-sítě má 7 vstupů a tréninková množina obsahuje následujících 12 vzorů:

$$\begin{aligned}
\mathcal{T}_0 &= \{(1000000,0),(0100000,0),(0010000,0)\} \\
\mathcal{T}_1 &= \{(0000000,1),(1100000,1),(0110000,1)\} \\
\mathcal{T}_0' &= \{(0000001,0),(0001101,0),(0000111,0)\} \\
\mathcal{T}_1' &= \{(0001001,1),(0000101,1),(0000011,1)\}\,.
\end{aligned} \tag{11.90}$$

Nechť tedy Q_1, Q_2 je řešení SSP. Definujeme konfiguraci ℓ analogové prahové 3-sítě C, která splňuje výstupní separační podmínku ($u_0 = 0$):

$$\begin{array}{llll}
w_0 = -1 & w_i = \begin{cases} 2 & q_i \in Q_1 \\ -2n+2 & q_i \notin Q_1 \end{cases} & & i = 1,\ldots,n \\
w_{2n+1} = 1 & w_{n+i} = -1 & & i = 1,\ldots,n \\
v_0 = 1 & v_i = \begin{cases} -2n & q_i \in Q_2 \\ 2n^2-2n & q_i \notin Q_2 \end{cases} & & i = 1,\ldots,n \\
v_{2n+1} = 2n & v_{n+i} = -2n & & i = 1,\ldots,n \\
u_0 = 0 & u_1 = -2 & u_2 = 1\,. &
\end{array} \tag{11.91}$$

Ověříme, že $C_\ell^A \models \mathcal{T}$. Pro $q_i \in Q_1$ podle (11.91) platí, že $\xi_a(0^{i-1}10^{2n+1-i}) = w_0+w_i = 1$ a $\xi_b(0^{i-1}10^{2n+1-i}) = v_0+v_i = 2n^2-2n+1$ pro tréninkový vzor $(0^{i-1}10^{2n+1-i},0) \in \mathcal{T}_0$ ($q_i \in Q_1$), tj. $\xi_c(0^{i-1}10^{2n+1-i}) = -2\sigma(1)+\sigma(2n^2-2n+1) < 0$, kde $\sigma = 1/(1+e^{-\xi})$ je standardní sigmoida (1.9), a proto podle (7.26) $y_c(0^{i-1}10^{2n+1-i}) = 0$ se shoduje s požadovaným nulovým výstupem. Podobně pro $q_i \in Q_2$ podle (11.91) platí,

újmy na obecnosti označme prvky $r_j = \{q_1, \ldots, q_r\} \subseteq Q$, kde $r = |r_j| \leq n$. Z $C_\ell^A \models \mathcal{T}_0$ a $C_\ell^A \models \mathcal{T}_1$ podle (11.89) a (7.26) dostáváme:

$$\begin{aligned} \xi_c(0^{i-1}10^{2n+1-i}) = u_0 + u_1\sigma(w_0 + w_i) + u_2\sigma(v_0 + v_i) &< 0 \\ i = 1, \ldots, r \leq n & \\ \xi_c(0^{2n+1}) = u_0 + u_1\sigma(w_0) + u_2\sigma(v_0) &\geq 0 \\ \xi_c(x_{j1}\ldots x_{jn}0^{n+1}) = u_0 + u_1\sigma\left(w_0 + \sum_{i=1}^{r} w_i\right) + u_2\sigma\left(v_0 + \sum_{i=1}^{r} v_i\right) &\geq 0\,, \end{aligned}$$

což lze pomocí explicitního tvaru (1.9) standardní sigmoidy σ upravit:

$$e^{v_0+v_i}\,(U_2 + U_3e^{w_i}) < U_0 + U_1e^{w_i} \quad i = 1, \ldots, r \tag{11.102}$$

$$e^{v_0}\,(U_2 + U_3) \geq U_0 + U_1 \tag{11.103}$$

$$e^{v_0+V_m}\,\left(U_2 + U_3e^{W_m}\right) \geq U_0 + U_1e^{W_m}\,, \tag{11.104}$$

kde (11.94) lze přepsat:

$$\begin{aligned} U_0 &= -u_0 > 0 \\ U_1 &= -(u_0 + u_1)e^{w_0} \geq 0 \\ U_2 &= u_0 + u_2 \leq 0 \\ U_3 &= (u_0 + u_1 + u_2)e^{w_0} > 0 \end{aligned} \tag{11.105}$$

a $W_k = \sum_{i=1}^{k} w_i$, $V_k = \sum_{i=1}^{k} v_i$ pro $k = 1, \ldots, r$.

Podle (11.105) víme, že $U_0 + U_1, U_0 + U_1e^{W_m} > 0$, a tedy z nerovností (11.103) a (11.104) vyplývá, že $U_2 + U_3 > 0$ a $U_2 + U_3e^{W_m} > 0$. Navíc ukážeme, že $U_2 + U_3e^{w_i} > 0$ a $U_2 + U_3e^{W_i} > 0$ pro $i = 1, \ldots, r$. V případě, že $r_j \subseteq Q_1$, pro všechny $i = 1, \ldots, r$ platí $w_i \geq 0$, a tedy $U_2 + U_3e^{W_i} \geq U_2 + U_3e^{w_i} \geq U_2 + U_3 > 0$ pro $i = 1, \ldots, r$. V případě, že $r_j \subseteq Q_2$, pro všechny $i = 1, \ldots, r$ platí $w_i < 0$, a tedy $U_2 + U_3e^{w_i} \geq U_2 + U_3e^{W_i} \geq U_2 + U_3e^{W_m} > 0$ pro $i = 1, \ldots, r$. To nám umožní vyjádřit (11.102) a (11.103) následujícím způsobem:

$$e^{v_0+v_i} < \frac{U_0 + U_1e^{w_i}}{U_2 + U_3e^{w_i}} \quad i = 1, \ldots, r \tag{11.106}$$

$$e^{-v_0} \leq \frac{U_2 + U_3}{U_0 + U_1}\,. \tag{11.107}$$

Dále ukážeme indukcí dle k, že nerovnosti (11.106) a (11.107) implikují

$$e^{v_0+V_k} < \frac{U_0 + U_1e^{W_k}}{U_2 + U_3e^{W_k}} \quad k = 1, \ldots, r\,. \tag{11.108}$$

je jedno z čísel u_0, $u_0 + u_1$, $u_0 + u_2$, $u_0 + u_1 + u_2$ záporné a zbylá tři jsou nezáporná, lze totiž převést na předchozí případ pomocí tréninkových vzorů z $\mathcal{T}_0' \cup \mathcal{T}_1'$. Za tímto účelem jsou definovány nové váhy $w_0', \ldots, w_n'$, $v_0', \ldots, v_n'$ a u_0', u_1', u_2' konfigurace ℓ' analogové prahové 3-sítě C následujícím způsobem:

$$\begin{array}{llll} w_0' = w_0 + w_{2n+1} & w_i' = w_{n+i} & i = 1, \ldots, n & \\ v_0' = v_0 + v_{2n+1} & v_i' = v_{n+i} & i = 1, \ldots, n & \\ u_0' = -u_0 & u_1' = -u_1 & u_2' = -u_2\,. & \end{array} \tag{11.93}$$

Zřejmě podle (11.93) je mezi čísly u_0', $u_0' + u_1'$, $u_0' + u_2'$, $u_0' + u_1' + u_2'$ právě jedno kladné a zbylá jsou nekladná, jak požadujeme. Navíc podle (11.93) platí $C_{\ell'}^A \models \mathcal{T}_0 \cup \mathcal{T}_1$, protože $C_{\ell'}^A \models \mathcal{T}_0' \cup \mathcal{T}_1'$.

Dále budeme bez újmy na obecnosti předpokládat, že

$$u_0 < 0 \;\; u_0 + u_1 \leq 0 \;\; u_0 + u_2 \leq 0 \;\; u_0 + u_1 + u_2 > 0\,, \tag{11.94}$$

kde ostrost první nerovnosti vyplývá z ostatních. Ostatní tři kombinace znamének lze opět převést na (11.94) při zachování funkce C_ℓ^A analogové prahové 3-sítě C. Tuto transformaci naznačíme jen pro jeden případ, zatímco ostatní jsou podobné. Nechť např. $u_0 \leq 0$, $u_0 + u_1 < 0$, $u_0 + u_2 > 0$, $u_0 + u_1 + u_2 \leq 0$. V tomto případě definujeme nové váhy $w_i' = -w_i$, $v_i' = v_i$ pro $i = 0, \ldots, n$ a $u_0' = u_0 + u_1$, $u_1' = -u_1$, $u_2' = u_2$. Jelikož pro standardní sigmoidu σ platí $\sigma(-\xi) = 1 - \sigma(\xi)$, výstupy y_a', y_b' hradel a, b v první vrstvě pro tyto nové váhy jsou $y_a' = 1 - y_a$ a $y_b' = y_b$. Ověříme, že analogová prahová 3-síť s novými váhami počítá původní funkci C_ℓ^A, tj.

$$\xi_c' = u_0' + u_1'y_a' + u_2'y_b' = u_0 + u_1 - u_1(1 - y_a) + u_2y_b = u_0 + u_1y_a + u_2y_b\,. \tag{11.95}$$

Nové váhy navíc splňují (11.94):

$$u_0' = u_0 + u_1 < 0 \tag{11.96}$$

$$u_0' + u_1' = u_0 + u_1 - u_1 = u_0 \leq 0 \tag{11.97}$$

$$u_0' + u_2' = u_0 + u_1 + u_2 \leq 0 \tag{11.98}$$

$$u_0' + u_1' + u_2' = u_0 + u_1 - u_1 + u_2 = u_0 + u_2 > 0\,. \tag{11.99}$$

Nyní již můžeme definovat disjunktní rozklad množiny $Q = Q_1 \cup Q_2$ následujícím způsobem:

$$Q_1 = \{q_i \in Q \mid w_i \geq 0\} \tag{11.100}$$

$$Q_2 = Q \setminus Q_1\,. \tag{11.101}$$

Ukážeme sporem, že Q_1, Q_2 je řešením instance (Q, R) problému SSP. Nechť tedy naopak existuje $r_j \in R$ takové, že buď $r_j \subseteq Q_1$, nebo $r_j \subseteq Q_2$. Bez

vstupu LP. Možným východiskem z této situace je umožnit učícímu algoritmu volit architekturu sítě, avšak kvůli generalizační schopnosti požadovat její minimalitu (viz odstavec 2.2.6), kterou lze formalizovat různými způsoby. V tomto případě se hovoří o tzv. *konstruktivním učení* [54]. Nicméně pro různá kritéria minimality architektury je možné ukázat, že konstruktivní tréninkový problém je NP-úplný. Na druhou stranu je v literatuře popsáno několik polynomiálních konstruktivních učících algoritmů neuronových sítí, které naleznou architekturu konzistentní s tréninkovými vzory, avšak díky omezeným generalizačním schopnostem je lze jen v omezené míře prakticky použít [68, 192, 232, 240].

Také původní neurofyziologické motivace nám potvrzují, že musí existovat nějaká možnost efektivního učení, protože živé organismy jsou přece jen schopny učit se v reálném čase, i když např. u člověka to představuje několik vývojových let jedince. Avšak v případě živých organismů je situace odlišná tím, že jedinec navíc dědí genetickou informaci, která podstatným způsobem přispívá k jeho schopnosti učit se. Některé funkce jsou již dokonce předprogramovány, např. části mozku člověka, které řídí základní funkce organismu. To u modelů neuronových sítí může odpovídat počáteční konfiguraci sítě, která nemusí odpovídat její smysluplné funkci, ale představuje výchozí stav adaptace, ze kterého lze síť učit efektivně. Proto např. počáteční nastavení vah učícího algoritmu backpropagation může rozhodujícím způsobem ovlivnit dobu a kvalitu učení (viz odstavec 2.2.2). Otázka, kde se v přírodě vzala tato počáteční informace, tj. zda např. v průběhu evoluce člověka nebo při stvoření světa, a zda ji budeme schopny někdy popsat, je pak již filozofický problém.

11.6 Učení cyklických neuronových sítí

V této podkapitole připojíme jen stručnou poznámku o složitosti učení cyklických neuronových sítí. Nejprve zformulujeme obecný tréninkový problém pro Hopfieldovy sítě (viz podkapitoly 8.3 a 8.4), jehož složitost je doposud otevřený problém [66].

Tréninkový problém pro Hopfieldovy sítě HNS (**H**opfield **N**et **S**ynthesis):
instance: Celé číslo r; tréninková množina $\mathcal{T} = \{\mathbf{x}_k \in \{0,1\}^n \mid k = 1, \ldots, p\}$ taková, že mezi každými dvěma různými vzory $\mathbf{x}_k, \mathbf{x}_j \in \mathcal{T}$ je Hammingova vzdálenost $H(\mathbf{x}_k, \mathbf{x}_j) > 2r$ (viz definici 8.23) pro $1 \leq k < j \leq p$.
otázka: Existuje Hopfieldova síť velikosti n taková, že každý tréninkový vzor $\mathbf{x}_k \in \mathcal{T}$ je jejím stabilním stavem s poloměrem atrakce aspoň r?

Nejprve pro $k = 1$ nerovnost (11.108) odpovídá přesně (11.106) pro $i = 1$. Dále předpokládejme, že (11.108) platí pro nějaké $k \in \{1, \ldots, r-1\}$. Potom pomocí (11.106), (11.107) a faktu, že všechny členy jsou kladné, obdržíme následující nerovnost:

$$\begin{aligned} e^{v_0+V_{k+1}} &= e^{v_0+V_k} e^{v_0+v_{k+1}} e^{-v_0} < \\ &< \frac{U_0 + U_1 e^{W_k}}{U_2 + U_3 e^{W_k}} \cdot \frac{U_0 + U_1 e^{w_{k+1}}}{U_2 + U_3 e^{w_{k+1}}} \cdot \frac{U_2 + U_3}{U_0 + U_1} . \end{aligned} \quad (11.109)$$

Chceme dokázat (11.108) pro k nahrazené $k+1$. Za tímto účelem stačí podle (11.109) ukázat:

$$\frac{U_0 + U_1 e^{W_k}}{U_2 + U_3 e^{W_k}} \cdot \frac{U_0 + U_1 e^{w_{k+1}}}{U_2 + U_3 e^{w_{k+1}}} \cdot \frac{U_2 + U_3}{U_0 + U_1} \leq \frac{U_0 + U_1 e^{W_{k+1}}}{U_2 + U_3 e^{W_{k+1}}} , \quad (11.110)$$

což lze přepsat jako

$$(U_1U_2 - U_0U_3)\left(e^{W_k} - 1\right)\left(e^{w_{k+1}} - 1\right)\left(U_1U_3 e^{W_{k+1}} - U_0U_2\right) \leq 0 . \quad (11.111)$$

Prostřední dva činitelé $e^{W_k} - 1$ a $e^{w_{k+1}} - 1$ v nerovnosti (11.111) jsou buď oba nezáporné, když $r_j \subseteq Q_1$, nebo oba nekladné, pokud $r_j \subseteq Q_2$. Navíc podle (11.105) je první činitel v (11.111) záporný a poslední je nezáporný. Tedy nerovnost (11.111) platí, čímž jsme ukázali (11.108) speciálně pro $k = m$, a tedy negaci (11.104), což je spor. Proto $r_j \not\subseteq Q_1$ $r_j \not\subseteq Q_2$ pro $r_j \in R$ a Q_1, Q_2 je řešení instance (Q, R) problému SSP. □

Důsledek 11.24 *LP pro analogový prahový obvod s aktivační funkcí standardní sigmoidou (1.9), který má daný konstantní počet hradel v první vrstvě a jedno výstupní prahové hradlo ve druhé vrstvě s nulovým prahem, je NP-těžký problém.*

Důkaz: Stačí stejně jako v důkazu věty 11.23 polynomiálně redukovat problém SSP na LP pro analogový prahový obvod se standardní sigmoidou, který má 2 hradla v první vrstvě a jedno výstupní prahové hradlo ve druhé vrstvě s nulovým prahem. □

Uvedené negativní výsledky o složitosti LP potvrzují praktickou zkušenost s neefektivitou používaných učících heuristik neuronových sítí a naznačují vnitřní překážky problému učení. Neznamenají, že se nám v daném konkrétním případě nepodaří neuronovou síť, např. s nevelkou architekturou, naučit malý počet tréninkový vzorů v reálném čase, ale ukazují, že existují zadání tréninkového problému, která žádný učící algoritmus pravděpodobně neřeší efektivně. Jak jsme již několikrát upozornili, jedním z důležitých předpokladů těchto výsledků je pevná architektura, která je součástí

Zřejmě ve formulaci HNS nejde jen o to, aby tréninkové vzory byly stabilními stavy Hopfieldovy sítě, ale také aby odpovídající oblasti atrakce byly dostatečně velké. Tento požadavek je předpokladem využití Hopfieldovy sítě jako asociativní paměti (viz odstavec 3.2.2). Pokud se omezíme jen na to, aby tréninkové vzory odpovídaly stabilním stavům, problém HNS je řešitelný v polynomiálním čase.

Věta 11.25 *Problém HNS pro $r = 0$ je řešitelný v polynomiálním čase.*

Důkaz: Nechť $(r; \mathcal{T})$ je instance problému HNS, kde

$$\mathcal{T} = \{\mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \{0,1\}^n \mid k = 1, \ldots, p\} \tag{11.112}$$

je tréninková množina. Sestavíme soustavu lineárních nerovnic, která má řešení, právě když tréninkové vzory z $\mathcal{T}$ jsou stabilními stavy Hopfieldovy sítě $N = (V, X, Y, A, w, h)$ velikosti n. Neznámé této soustavy reprezentují váhy w a prahy h sítě N:

$$\begin{aligned} &\sum_{i=1}^{n} x_{ki} w_{ji} \begin{cases} \geq h_j & x_{kj} = 1 \\ < h_j & x_{kj} = 0 \end{cases} \qquad j = 1, \ldots, n;\ k = 1, \ldots, p \\ &w_{ij} = w_{ji} \quad 1 \leq i, j \leq n\,. \end{aligned} \tag{11.113}$$

Soustavu (11.113) lze vyřešit pomocí lineárního programování v polynomiálním čase [145]. □

Složitost HNS pro $r \geq 1$ není známa. Pro konstantní r je HNS v NP, pokud se omezíme na malé váhy. Věta 8.24 napovídá, že HNS bude v obecném případě pravděpodobně těžký problém.

Pro obecné analogové neuronové sítě (viz podkapitolu 10.2) je situace ještě horší [57]. Z důsledku 10.18 vyplývá, že LP není v případě racionálních vah ani rekurzivní.

že $K \subseteq \Xi_n$ je koncept *dimenze n a $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ je třída těchto konceptů. Uvažujme také obecnou* třídu konceptů $\mathcal{K} = \bigcup_{n\geq 0} \mathcal{K}_n \subseteq \mathcal{P}(\Xi)$ *ve vstupním prostoru Ξ. Řekneme, že třída konceptů $\mathcal{K} \subseteq \mathcal{P}(\Xi)$ je* triviální, *jestliže buď $|\mathcal{K}| = 1$, nebo $\mathcal{K} = \{K_1, K_2\}$ tak, že $\Xi = K_1 \cup K_2$. Omezíme se na* reprezentaci *konceptu $K \subseteq \Xi_n$ pomocí prahového obvodu C_n pevné architektury s n vstupy a jedním výstupem, jehož funkce $C_n : \Xi_n \longrightarrow \{0,1\}$ představuje charakteristickou funkci K, tj. pro každé $\mathbf{x} \in \Xi_n$ platí, že $C_n(\mathbf{x}) = 1$, právě když $\mathbf{x} \in K$, nebo také $K = L(C)$ podle definice 7.52.* Velikost *$s(K)$ konceptu K je pak počet bitů potřebných pro reprezentaci nejmenšího obvodu, který K reprezentuje. Podobně podle definice 7.19 třída posloupností $\mathbf{C}$ obvodů s jedním výstupem určuje třídu konceptů $\mathcal{K} = L(\mathbf{C})$.*

Nyní zformalizujeme základní model učení, který přesně vymezuje, kdy lze efektivně najít reprezentaci tzv. *hypotézy* (konceptu), která je s velkou pravděpodobností dobrou aproximací neznámého *cílového konceptu* z dané třídy konceptů, pomocí tréninkových vzorů, které odpovídají tomuto konceptu a jsou generovány náhodně s neznámým pravděpodobnostním rozdělením. Obecně se uvažuje větší třída hypotéz než je třída cílových konceptů, avšak pro naše účely je postačující omezit se na případ tzv. *vlastního učení* (proper learning), kdy obě třídy splývají a v našem případě jsou reprezentovány acyklickou neuronovou sítí. To znamená, že neznámý cílový koncept i hledaná hypotéza, která jej aproximuje, jsou jen z takové třídy konceptů, které lze reprezentovat acyklickou neuronovou sítí.

Definice 12.2 (Valiant [275]) *Řekneme, že třída $\mathcal{K} = \bigcup_{n\geq 0} \mathcal{K}_n$ konceptů reprezentovaných posloupnostmi prahových obvodů je PAC-*naučitelná, *jestliže existuje* učící algoritmus *U takový, že pro každé $n \geq 0$, pro každý* cílový koncept *$K \in \mathcal{K}_n$, pro každé pravděpodobnostní rozdělení $D : \Xi_n \longrightarrow \langle 0,1\rangle$ vstupního prostoru Ξ_n dimenze n, pro každou* přesnost *$0 < \varepsilon < 1$ a každou* konfidenci *$0 < \delta < 1$ algoritmus U, který má na vstupu n, ε, δ a tréninkovou posloupnost vzorů $(\mathbf{x}_k, d_k) \in \Xi_n \times \{0,1\}$ ($k \geq 1$), kde vstup $\mathbf{x}_k \in \Xi_n$ je generován náhodně a nezávisle podle pravděpodobnostního rozdělení D a $d_k = 1$ (*pozitivní příklad*), právě když $\mathbf{x}_k \in K$ (v případě $d_k = 0$, tj. $\mathbf{x}_k \notin K$, hovoříme o* negativním příkladě*), skončí v čase $P(n, s(K), \frac{1}{\varepsilon}, \frac{1}{\delta})$, kde P je polynom, s* hypotézou *$H \in \mathcal{K}_n$ na výstupu takovou, že s pravděpodobností aspoň $1 - \delta$ (vzhledem k možným tréninkovým posloupnostem)* chyba

$$E(H, K) = \sum_{\mathbf{x} \in (K \Delta H)} D(\mathbf{x}) \leq \varepsilon\,, \tag{12.1}$$

kde $K \Delta H = (K \setminus H) \cup (H \setminus K)$ je symetrická diference.

Zamysleme se nad tím, jaký je rozdíl mezi tréninkovým problémem LP (viz kapitolu 11) a PAC-modelem. U LP hledáme konfiguraci neuronové

Kapitola 12

Generalizace neuronových sítí

Obecně heuristiky v umělé inteligenci často postrádají složitostní analýzu. Podobně tomu bylo i v oblasti strojového učení, a proto se Leslie G. Valiant počátkem 80.let s úspěchem pokusil o formalizaci učení a generalizace z hlediska teorie složitosti a vytvořil tak známý *PAC-model* (**P**robably **A**pproximately **C**orrect learning). Jeho pionýrská práce [275] dala později vznik celému informatickému odvětví, tzv. *výpočetní teorii učení* (computational learning theory) [20, 22]. V rámci této disciplíny bylo dosaženo mnoho výsledků, které jsou relevantní pro učení a generalizaci neuronových sítí. Modely neuronových sítí tak slouží jako oblast aplikace a testování obecné teorie učení, ale také jako zdroj motivací pro další výzkum. V této kapitole naznačíme jen základní myšlenky obecné výpočetní teorie učení a uvedeme nejdůležitější výsledky týkající se učení neuronových sítí.

12.1 PAC-model

Budeme se zabývat učením tzv. *konceptů* z tréninkových vzorů. Např. v oblasti rozpoznávání obrazů písmeno „A“ lze považovat za koncept a vzory jsou konkrétní výskyty tohoto písmene (viz motivační příklad z odstavce 1.3.1). Tedy koncept lze chápat jako podmnožinu vstupního prostoru. Pro reprezentaci konceptů budeme v našem případě používat acyklickou neuronovou síť s jedním výstupem, jehož hodnota určuje, zda daný vstupní příklad je instancí příslušného konceptu.

Definice 12.1 *Nechť Ξ_n je vstupní prostor dimenze n, tj. např. $\Xi_n = \{0,1\}^n$ (resp. $\Xi_n = \mathbb{R}^n$) a označme obecný* vstupní prostor *$\Xi = \bigcup_{n\geq 0} \Xi_n$. Řekneme,*

s dotazy na shodu (příp. s dotazy na příslušnost) [19]. Každý dotaz na shodu polynomiálního přesného učícího algoritmu lze totiž v PAC-modelu nahradit dostatečně velkým počtem náhodně generovaných tréninkových vzorů odpovídajícího cílového konceptu, které se použijí při testu konzistence hypotézy buď k odhalení protipříkladu, nebo k potvrzení hypotézy, která je již pravděpodobně dobrou aproximací cílového konceptu.

12.2 Počet tréninkových vzorů

Před tím, než se v rámci PAC-modelu budeme zabývat efektivní naučitelností konceptů reprezentovaných pomocí acyklických neuronových sítí, věnujeme tuto podkapitolu jednodušší otázce, totiž kolik je při tom potřeba a kolik stačí tréninkových vzorů. Tento problém je užitečný pro praktické učení neuronových sítí, protože jeho řešením dostaneme odhad velikosti tréninkové množiny pro úspěšnou generalizaci. Počet náhodných příkladů, které požaduje učící algoritmus v nejhorším případě, se nazývá *vzorková složitost* (sample complexity), která je jedním ze základních předmětů výzkumu ve výpočetní teorii učení. Zřejmě vzorkovou složitost lze studovat nezávisle na reprezentaci konceptu a výpočetní složitosti učícího algoritmu. Za tímto účelem definujeme tzv. *(ε,δ)-učící algoritmus.*

Definice 12.4 *Řekneme, že U je* (ε,δ)-učící algoritmus *pro třídu $\mathcal{K} \subseteq \mathcal{P}(\Xi)$ konceptů ve vstupním prostoru Ξ, kde $0 < \varepsilon < 1$ je přesnost a $0 < \delta < 1$ je konfidence, jestliže pro každý cílový koncept $K \in \mathcal{K}$ a pro každé pravděpodobnostní rozdělení $D : \Xi \longrightarrow \langle 0,1\rangle$ vstupního prostoru Ξ, algoritmus U, který má na vstupu p tréninkových vzorů generovaných náhodně a nezávisle podle pravděpodobnostního rozdělení D, nalezne hypotézu $H \in \mathcal{K}$ takovou, že s pravděpodobností aspoň $1-\delta$ chyba*

$$E(H,K) = \sum_{\mathbf{x}\in(K\,\Delta H)} D(\mathbf{x}) \leq \varepsilon\,. \tag{12.2}$$

Dále hypotéza $H \subseteq \Xi$ je konzistentní *se vzorkem p tréninkových vzorů, jestliže obsahuje všechny pozitivní příklady a neobsahuje žádný negativní příklad z tohoto vzorku. Učící algoritmus je* konzistentní, *jestliže nalezne hypotézu konzistentní se zadanou tréninkovou množinou.*

Ukážeme triviální horní odhad velikosti tréninkové množiny (ε,δ)-učícího algoritmu pro konečnou třídu konceptů.

Věta 12.5 (Blumer, Ehrenfeucht, Haussler, Warmuth [39], Vapnik [276]) *Nechť $\mathcal{K}$ je konečná třída konceptů. Potom každý učící algoritmus, který nalezne hypotézu $H \in \mathcal{K}$ konzistentní s $\frac{1}{\varepsilon}\ln\frac{|\mathcal{K}|}{\delta}$ tréninkovými vzory je (ε,δ)-učící algoritmus pro $\mathcal{K}$.*

sítě, která je konzistentní s danými tréninkovými vzory, přitom funkce sítě pro vstupy, které nejsou pokryty tréninkovou množinou, není nijak určena. Tedy LP nepostihuje generalizační vlastnosti sítě. Na druhou stranu PAC-model předpokládá existenci cílového konceptu, který je fixován učitelem. Tréninkové vzory odpovídající tomuto konceptu jsou generovány náhodně s nějakým pravděpodobnostním rozdělením, tj. při každém běhu učícího algoritmu je k dispozici obecně jiná tréninková množina. Pomocí ní by pak učící algoritmus měl pokaždé najít v polynomiálním čase reprezentaci hypotézy, tj. konfiguraci neuronové sítě, která aproximuje uvedený cílový koncept pravděpodobně s malou chybou vzhledem k danému rozdělení pravděpodobností tréninkových vzorů. To znamená, že cílový koncept určuje jednoznačně požadovanou funkci sítě, která je parciálně zadána tréninkovou množinou, zatímco generalizační chyba učení se měří vzhledem ke všem instancím tohoto konceptu s přihlédnutím k jejich výskytům. Čas běhu algoritmu závisí polynomiálně nejen na velikosti reprezentace sítě, ale také na přesnosti a konfidenci výsledné hypotézy. Naučenou síť je pak možno využívat v prostředí se stejnou pravděpodobností výskytu vzorů, jako byla učena. Z uvedeného vyplývá, že PAC-model je vhodným rámcem pro studium generalizačních vlastností neuronových sítí. V podkapitole 12.3 ukážeme, že PAC-model v jistém smyslu zahrnuje LP.

U PAC-modelu učení v definici 12.2 se také někdy předpokládá odlišný vstupní protokol, kdy učící algoritmus sám klade učiteli dotazy. Uvedeme následující dva typy dotazů, i když ve výpočetní teorii učení se uvažují i další varianty [19]. Budeme se zabývat *dotazy na shodu* hypotézy s cílovým konceptem a při tzv. *aktivním učení* se používají *dotazy na příslušnost* libovolného vzoru k cílovému konceptu [18].

Definice 12.3 *Řekneme, že třída konceptů reprezentovaných posloupnostmi prahových obvodů je* PAC-naučitelná s dotazy na shodu a příslušnost, *jestliže existuje učící algoritmus U z definice 12.2, který místo tréninkových vzorů na vstupu klade učiteli* dotazy na shodu, *tj. zda předložená reprezentace hypotézy H již reprezentuje cílový koncept K s tím, že učitel buď tuto skutečnost potvrdí, nebo v opačném případě generuje libovolný protipříklad $\mathbf{x} \in (K\Delta H)$, a dále* dotazy na příslušnost, *zda $\mathbf{x} \in \Xi_n$ volené algoritmem U je instancí konceptu K, tj. zda $\mathbf{x} \in K$. Učitel v obou případech odpovídá v konstantním čase.*

Další varianta PAC-modelu uvažuje dané pevné rozdělení pravděpodobností tréninkových vzorů, např. rovnoměrné rozdělení [37]. Nebo se např. studuje tzv. *přesné učení* (on-line learning, exact identification) [19], kdy je požadováno, aby učící algoritmus nalezl přesnou reprezentaci cílového konceptu pomocí dotazů na shodu, příp. na příslušnost. Zřejmě přibližné učení v PAC-modelu (příp. s dotazy na příslušnost) není těžší než přesné učení

Důkaz: Pravděpodobnost, že libovolná hypotéza $H \in \mathcal{K}$ s chybou $E(H,K) > \varepsilon$ není konzistentní s jedním náhodným tréninkovým vzorem je zřejmě podle (12.2) větší než ε, a tedy opačně pravděpodobnost její konzistence s jedním vzorem je nejvýše $1-\varepsilon$ a s p náhodně a nezávisle vybranými tréninkovými vzory je nejvýše $(1-\varepsilon)^p \leq e^{-\varepsilon p}$. Protože existuje nejvýše $|\mathcal{K}|$ hypotéz, pravděpodobnost, že nějaká hypotéza v $\mathcal{K}$ má chybu více než ε a je konzistentní s p s náhodnými tréninkovými vzory je nejvýše $|\mathcal{K}|e^{-\varepsilon p}$. Tedy pro $p = \frac{1}{\varepsilon}\ln\frac{|\mathcal{K}|}{\delta}$ je tato pravděpodobnost nejvýše δ. To znamená, že s pravděpodobností aspoň $1-\delta$ je $E(H,K) \leq \varepsilon$ pro $H \in \mathcal{K}$ konzistentní s $\frac{1}{\varepsilon}\ln\frac{|\mathcal{K}|}{\delta}$ tréninkovými vzory. □

Věta 12.5 předpokládá konečnou velikost třídy konceptů. V případě nekonečných tříd konceptů se místo toho obecně uvažuje tzv. *Vapnik-Chervonenkisova dimenze* [277], která se stala jedním z ústředních pojmů výpočetní teorie učení.

Definice 12.6 *Řekneme, že podmnožina $Q \subseteq \Xi$ je* atomizována *(shattered) třídou konceptů $\mathcal{K} \subseteq \mathcal{P}(\Xi)$ ve vstupním prostoru Ξ, jestliže pro každou podmnožinu $R \subseteq Q$ existuje koncept $K \in \mathcal{K}$ takový, že $K \cap Q = R$.* Vapnik-Chervonenkisova dimenze *(VC-dimenze) $VC(\mathcal{K})$ třídy konceptů $\mathcal{K}$ je největší velikost podmnožiny vstupního prostoru, která je atomizována třídou $\mathcal{K}$.*

Nyní zformulujeme bez důkazu zobecněnou větu 12.5 pro konečnou VC-dimenzi.

Věta 12.7 (Blumer, Ehrenfeucht, Haussler, Warmuth [40]) *Nechť $VC(\mathcal{K})$ je konečná VC-dimenze třídy konceptů $\mathcal{K}$. Potom každý učící algoritmus konzistentní s tréninkovou množinou velikosti*

$$O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{VC(\mathcal{K})}{\varepsilon}\log\frac{1}{\varepsilon}\right) \tag{12.3}$$

je (ε,δ)-učící algoritmus pro $\mathcal{K}$.

Ukážeme také dolní odhad pro velikost tréninkové množiny (ε,δ)-učícího algoritmu, který se jen nepatrně liší od horního odhadu ve větě 12.7.

Věta 12.8 (Ehrenfeucht, Haussler, Kearns, Valiant [59]) *Nechť $\mathcal{K}$ je netriviální třída konceptů a U je (ε,δ)-učící algoritmus pro $\mathcal{K}$. Potom U vyžaduje aspoň*

$$\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{VC(\mathcal{K})}{\varepsilon}\right) \tag{12.4}$$

tréninkových vzorů.

Důkaz: Nechť U je (ε,δ)-učící algoritmus pro $\mathcal{K}$ se vzorkovou složitostí p. Dolní odhad p se obvykle dokazuje tak, že se nejprve zvolí „těžké“ rozdělení pravděpodobností na Ξ a množina konceptů $\mathcal{K}' \subseteq \mathcal{K}$ a pak se ukáže, že pro dostatečně velké p algoritmus U odpoví hypotézou H, která má s velkou pravděpodobností velkou chybu $E(H,K)$ vzhledem k nějakému konceptu $K \in \mathcal{K}'$.

Speciálně, abychom nejprve ukázali, že p je aspoň $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\delta})$, využijeme toho, že $\mathcal{K}$ je netriviální (viz definici 12.1), a tedy existují vzory $\mathbf{x}_1, \mathbf{x}_2 \in \Xi$ a množina konceptů $\mathcal{K}' = \{K_1, K_2\} \subseteq \mathcal{K}$ taková, že $\mathbf{x}_1 \in K_1$, právě když $\mathbf{x}_1 \in K_2$, a zároveň $\mathbf{x}_2 \in K_1$, právě když $\mathbf{x}_2 \notin K_2$. Uvažujme pravděpodobnostní rozdělení $D : \Xi \longrightarrow \langle 0,1\rangle$, pro které $D(\mathbf{x}_1) = 1-\varepsilon$ a $D(\mathbf{x}_2) = \varepsilon$. Tedy tréninková posloupnost délky p pro U obsahuje jen $\mathbf{x}_1$ s pravděpodobností $(1-\varepsilon)^p$. V tomto případě se výsledná hypotéza H liší v $\mathbf{x}_2$ buď vzhledem k cílovému konceptu K_1, nebo K_2, tj. buď $E(H,K_1) \geq \varepsilon$, nebo $E(H,K_2) \geq \varepsilon$. To znamená, že nutně $(1-\varepsilon)^p < \delta$, protože U je (ε,δ)-učící algoritmus. Z toho vyplývá, že $p\ln(1-\varepsilon) < \ln\delta$, což lze díky $\ln(1-\varepsilon) < 0$ upravit

$$p > \frac{1}{\ln(1-\varepsilon)}\ln\delta = \frac{1}{-\ln(1-\varepsilon)}\ln\frac{1}{\delta}\,, \tag{12.5}$$

kde

$$-\ln(1-\varepsilon) = \ln\left(1+\frac{\varepsilon}{1-\varepsilon}\right) \leq \frac{\varepsilon}{1-\varepsilon}\,, \tag{12.6}$$

a tedy

$$p > \frac{1-\varepsilon}{\varepsilon}\ln\frac{1}{\delta}\,. \tag{12.7}$$

Zbývá ukázat, že $p = \Omega(VC(\mathcal{K})/\varepsilon)$. Nechť $Q = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{d-1}\} \subseteq \Xi$ dosvědčující $VC(\mathcal{K}) = |Q| = d$ je atomizována třídou konceptů $\mathcal{K} \subseteq \mathcal{P}(\Xi)$ ve vstupním prostoru Ξ a položme $Q' = Q\setminus\{\mathbf{x}_0\}$. Uvažujme pravděpodobnostní rozdělení $D : \Xi \longrightarrow \langle 0,1\rangle$, pro které $D(\mathbf{x}_0) = 1-8\varepsilon$ a $D(\mathbf{x}) = 8\varepsilon/(d-1)$ pro $\mathbf{x} \in Q'$. Tedy s pravděpodobností $1-8\varepsilon+(d-1)\cdot 8\varepsilon/(d-1) = 1$ tréninkové vzory pro U jsou z Q. Dále volme množinu

$$\mathcal{K}' = \{K \in \mathcal{K} \mid K \cap Q' \in \mathcal{P}(Q'), \mathbf{x}_0 \in K\}\,, \tag{12.8}$$

která díky tomu, že Q je atomizována třídou konceptů $\mathcal{K}$, obsahuje $|\mathcal{P}(Q')| = 2^{d-1}$ konceptů, a předpokládejme, že $p = \frac{d-1}{32\varepsilon}$. Ukážeme, že pro nějaké $K \in \mathcal{K}'$ chyba $E(H,K)$ výsledné hypotézy H algoritmu U je větší nebo rovna ε s pravděpodobností aspoň δ pro dostatečně malé ε,δ, tj. pravděpodobnost

$$\mathbf{P}\{E(H,K) \geq \varepsilon\} \geq \delta\,. \tag{12.9}$$

Navíc bez újmy na obecnosti předpokládejme, že vždy $\mathbf{x}_0 \in H$, protože příslušnou úpravou algoritmu U se chyba $E(H,K)$ pro $K \in \mathcal{K}'$ nezvýší.

což nastane s pravděpodobností 8ε, a celkový počet tréninkových vzorů je $\frac{d-1}{32\varepsilon}$. Tedy pravděpodobnost, že počet tréninkových vzorů pro U, které jsou z Q', je nejméně $(d-1)/2$, lze podle (9.2) shora odhadnout pomocí

$$B\left(2\frac{d-1}{32\varepsilon}8\varepsilon, \frac{d-1}{32\varepsilon}, 8\varepsilon\right) \leq e^{-\frac{d-1}{8}}\,. \tag{12.17}$$

To znamená, že

$$\mathbf{P}\{G\} = 1 - B\left(\frac{d-1}{2}, \frac{d-1}{32\varepsilon}, 8\varepsilon\right) \geq 1 - e^{-\frac{d-1}{8}}\,. \tag{12.18}$$

Dosazením (12.16) a (12.18) do (12.10) dostáváme pro $d \geq 2$ a $\varepsilon \leq \frac{1}{8}$ dolní odhad pravděpodobnosti

$$\mathbf{P}\{E(H,K) \geq \varepsilon\} \geq \frac{1}{7}\left(1 - e^{-\frac{d-1}{8}}\right) \geq \frac{1}{100}\,. \tag{12.19}$$

Proto, jestliže $\delta \leq \frac{1}{100}$ a $p \leq \frac{d-1}{32\varepsilon}$, U nemůže být (ε,δ)-učící algoritmus, a nutně tedy $p = \Omega(VC(\mathcal{K})/\varepsilon)$, což spolu s (12.7) implikuje tvrzení (12.4). □

Dolní a horní odhad na velikost tréninkové množiny ve větách 12.7 a 12.8 byl formulován pro obecnou třídu konceptů dané VC-dimenze. Abychom mohli tyto výsledky použít při učení neuronových sítí, v následujících větách určíme hodnotu této dimenze pro perceptron a acyklické neuronové sítě. V případě perceptronu budeme potřebovat Radonovu větu.

Věta 12.9 (Radon [90]) *Nechť $n \geq 1$ je přirozené číslo a Q je množina $|Q| \geq n+2$ bodů v euklidovském prostoru E_n dimenze n. Potom existuje neprázdná podmnožina $\emptyset \neq R \subseteq Q$ množiny Q taková, že*

$$conv(R) \cap conv(Q \setminus R) \neq \emptyset\,, \tag{12.20}$$

kde $conv(R)$ značí konvexní obal množiny R.

Věta 12.10 (Wenocur, Dudley [285]) *Nechť $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ je třída konceptů ve vstupním prostoru Ξ_n, kde $\{0,1\}^n \subseteq \Xi_n \subseteq \mathbb{R}^n$, které jsou reprezentovány lineárními prahovými funkcemi s n proměnnými (viz definici 6.3). Pak $VC(\mathcal{K}_n) = n+1$.*

Důkaz: Uvažujme libovolnou množinu $Q \subseteq \Xi_n$ velikosti $|Q| > n+1$. Podle Radonovy věty 12.9 existuje $\emptyset \neq R \subseteq Q$, pro kterou platí (12.20), tj. neexistuje lineární prahová funkce $f : \Xi_n \longrightarrow \{0,1\}$, která by byla charakteristickou funkcí pro R ($f(\mathbf{x}) = 1$, právě když $\mathbf{x} \in R$). To znamená, že neexistuje koncept $K \in \mathcal{K}_n$ takový, že $K \cap Q = R$, a tedy Q není atomizována třídou $\mathcal{K}_n$ konceptů ve vstupním prostoru Ξ_n. Z toho vyplývá, že $VC(\mathcal{K}_n) \leq n+1$.

Označme G jev, že počet tréninkových vzorů pro algoritmus U, které jsou z Q', je nejvýše $(d-1)/2$. Potom pravděpodobnost (12.9) lze odhadnout pomocí podmíněné pravděpodobnosti následujícím způsobem:

$$\mathbf{P}\{E(H,K) \geq \varepsilon\} \geq \mathbf{P}\{E(H,K) \geq \varepsilon \,|\, G\} \cdot \mathbf{P}\{G\}\,. \tag{12.10}$$

Nejprve tedy předpokládejme, že nastane jev G, tj. nejvýše $(d-1)/2$ tréninkových vzorů je z Q'. V tom případě výsledná hypotéza H algoritmu U nezávisí na zbylých aspoň $d-1-(d-1)/2 = (d-1)/2$ vzorech z Q'. Označme množinu těchto vzorů $Q_H \subseteq Q'$ ($|Q_H| \geq (d-1)/2$) a zavedeme predikát $e_H(K,\mathbf{x})$, pro který $e_H(K,\mathbf{x}) = 1$, jestliže $\mathbf{x} \in K \Delta H$, a $e_H(K,\mathbf{x}) = 0$ jinak. Odhadneme zdola průměrnou chybu $E(H,K)$ přes všechny koncepty $K \in \mathcal{K}'$:

$$\begin{aligned}
\frac{1}{|\mathcal{K}'|}\sum_{K\in\mathcal{K}'} E(H,K) &\geq \frac{1}{2^{d-1}}\sum_{K\in\mathcal{K}'}\sum_{\mathbf{x}\in Q_H} e_H(K,\mathbf{x})D(\mathbf{x}) = \\
= \frac{1}{2^{d-1}}\cdot\frac{8\varepsilon}{d-1}\sum_{\mathbf{x}\in Q_H}\sum_{K\in\mathcal{K}'} e_H(K,\mathbf{x}) &= \frac{1}{2^{d-1}}\cdot\frac{8\varepsilon}{d-1}\sum_{\mathbf{x}\in Q_H}\frac{2^{d-1}}{2} \geq \\
&\geq \frac{8\varepsilon}{d-1}\cdot\frac{d-1}{2}\cdot\frac{1}{2} = 2\varepsilon\,.
\end{aligned} \tag{12.11}$$

Odtud vyplývá, že střední hodnota

$$\mathbf{E}\left[\frac{1}{|\mathcal{K}'|}\sum_{K\in\mathcal{K}'} E(H,K)\,\middle|\, G\right] = \frac{1}{2^{d-1}}\sum_{K\in\mathcal{K}'}\mathbf{E}[E(H,K)\,|\,G] \geq 2\varepsilon\,, \tag{12.12}$$

a tedy existuje nějaký koncept $K \in \mathcal{K}'$ takový, že

$$\mathbf{E}[E(H,K)\,|\,G] \geq 2\varepsilon\,. \tag{12.13}$$

Na druhou stranu z předpokladu, že U vždy správně klasifikuje $\mathbf{x}_0$ vyplývá, že $E(H,K) \leq 8\varepsilon$. Z toho dostaneme horní odhad střední hodnoty:

$$\mathbf{E}[E(H,K)\,|\,G] \leq 8\varepsilon p_H + \varepsilon(1-p_H)\,, \tag{12.14}$$

kde $p_H = \mathbf{P}\{E(H,K) \geq \varepsilon \,|\, G\}$, což spolu s (12.13) implikuje

$$2\varepsilon \leq 8\varepsilon p_H + \varepsilon(1-p_H)\,, \tag{12.15}$$

a tedy

$$p_H = \mathbf{P}\{E(H,K) \geq \varepsilon \,|\, G\} \geq \frac{1}{7}\,. \tag{12.16}$$

Pro dolní odhad $\mathbf{P}\{G\}$ v (12.10) využijeme (*ii*) z lemmy 9.2 pro $\beta = 1$. V našem případě neúspěšný pokus znamená, že tréninkový vzor $\mathbf{x}$ je z Q',

charakteristických funkcí podmnožin $R \subseteq Q$. Ze vztahu (12.23) vyplývá, že $2^p \leq p^{2w}$, tj.

$$p \leq 2w \log p\,, \tag{12.24}$$

z čehož po úpravě dostáváme $4w \geq 2p/\log p$. Zřejmě $2p/\log p \geq \log p$, a tedy $4w \geq \log p$, což lze přepsat jako $\log 4w \geq \log\log p$. Horní odhad pro $\log\log p$ využijeme, když zlogaritmujeme (12.24), a dostáváme $\log p \leq \log 2w + \log\log p \leq \log 2w + \log 4w = O(\log w)$, což zpětně dosadíme za $\log p$ do (12.24) a obdržíme $p = O(w \log w)$. To znamená, že $VC(\mathcal{K}_n) = O(w \log w)$. □

Ukážeme překvapivý fakt, že horní odhad ve větě 12.12 nelze obecně zlepšit, i když by se mohlo zdát, že VC-dimenze u acyklické neuronové sítě nemůže být větší než součet VC-dimenzí u jejích neuronů, tj. podle věty 12.10 by byla $O(w)$. Následující věta tuto domněnku vyvrací a lze ji interpretovat jako matematickou evidenci teze, že síť neuronů není jen součtem svých komponent.

Věta 12.13 (Maass [182]) *Nechť $\mathcal{K} = \bigcup_{n\geq 0}\mathcal{K}_n$, kde $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ a $\{0,1\}^n \subseteq \Xi_n \subseteq \mathbb{R}^n$, je třída konceptů reprezentovaná posloupnostmi prahových obvodů $\mathbf{C} = (C_0, C_1, C_2, \ldots)$ ($L(\mathbf{C}) \in \mathcal{K}$) velikosti $S(n) = O(n)$ a hloubky 3, kde prahový obvod C_n ($L(C_n) \in \mathcal{K}_n$, $n \geq 0$) má n vstupů, první vrstva se skládá z $\Omega(n)$ hradel, ve druhé vrstvě je $4\log n$ hradel, ve třetí vrstvě je jedno výstupní hradlo a hradla (včetně vstupů) z jedné vrstvy jsou spojena se všemi hradly následující vrstvy. Tedy počet vah a prahů v obvodu C_n je $w = \Theta(n^2)$. Potom $VC(\mathcal{K}_n) = \Omega(n^2 \log n) = \Omega(w \log w)$.*

Důkaz: Nechť přirozené číslo $m = 2^e \in \mathbb{N}$ je mocninou 2 a položme $n = 2m + \log m$, tj. $2m = n - \log m \geq n - m$, a tedy $n/3 \leq m \leq n$, což znamená, že $m = \Theta(n)$. Označme binární vektory $\mathbf{e}_i = (0^{i-1}10^{m-i}) \in \{0,1\}^m$ pro $i = 1, \ldots, m$ a $\mathbf{e}'_i = (0^{i-1}10^{\log m - i}) \in \{0,1\}^{\log m}$ pro $i = 1, \ldots, \log m$. Dále uvažujme množinu

$$Q = \left\{(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_{i_3}) \mid 1 \leq i_1, i_2 \leq m,\ 1 \leq i_3 \leq \log m\right\} \subseteq \{0,1\}^n \subseteq \Xi_n \tag{12.25}$$

velikosti $|Q| = m^2 \log m = \Omega(n^2 \log n)$. Ukážeme, že Q je atomizována třídou konceptů $\mathcal{K}_n$, tj. $VC(\mathcal{K}_n) = \Omega(n^2 \log n) = \Omega(w \log w)$. Nechť tedy $R \subseteq Q$. Sestrojíme obvod C_n s $n = 2m + \log m$ vstupy, který počítá charakteristickou funkci $F : \{0,1\}^n \longrightarrow \{0,1\}$ množiny R, tj. pro každé $\mathbf{x} \in Q$ platí, že $F(\mathbf{x}) = C_n(\mathbf{x}) = 1$, právě když $\mathbf{x} \in R$.

Za tímto účelem definujeme zobrazení $g : \{1, \ldots, m\}^2 \longrightarrow \{0,1\}^{\log m}$ následujícím způsobem:

$$g(i_1, i_2) = \left(F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_1), \ldots, F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_{\log m})\right) \tag{12.26}$$

Na druhou stranu nechť $Q = \{0^n\} \cup \{0^{i-1}10^{n-i} \mid i = 1, \ldots, n\} \subseteq \Xi_n$ je množina velikosti $|Q| = n+1$ a $R \subseteq Q$. Definujeme reprezentaci $(w_1, \ldots, w_n; h)$ lineární prahové funkce $f : \Xi_n \longrightarrow \{0,1\}$, která je charakteristickou funkcí konceptu $K \in \mathcal{K}_n$:

$$w_i = \begin{cases} 1 & 0^{i-1}10^{n-i} \in R \\ -1 & 0^{i-1}10^{n-i} \notin R \end{cases} \qquad i = 1, \ldots, n\,. \tag{12.21}$$

$$h = \begin{cases} 0 & 0^n \in R \\ 1 & 0^n \notin R \end{cases}$$

Z definice (12.21) je zřejmé, že pro každé $\mathbf{x} \in Q$ platí $f(\mathbf{x}) = 1$, právě když $\mathbf{x} \in R$. Tedy $K \cap Q = R$ a Q je atomizována třídou $\mathcal{K}_n$ konceptů ve vstupním prostoru Ξ_n. Z toho vyplývá, že $VC(\mathcal{K}_n) \geq n + 1$, a tedy z předchozího $VC(\mathcal{K}_n) = n + 1$. □

Pro horní odhad VC-dimenze se často používá následující Sauerova lemma. Pomocí ní shora odhadneme tuto dimenzi pro třídu konceptů reprezentovaných acyklickými neuronovými sítěmi.

Lemma 12.11 (Sauer [246]) *Nechť Ξ je konečný vstupní prostor velikosti $|\Xi| = p$, $\mathcal{K} \subseteq \mathcal{P}(\Xi)$ je třída konceptů v Ξ taková, že $VC(\mathcal{K}) \leq d$. Potom platí*

$$|\mathcal{K}| \leq \sum_{i=0}^{d} \binom{p}{i} \leq p^d + 1\,. \tag{12.22}$$

Věta 12.12 (Cover [50], Baum, Haussler [34]) *Nechť $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ je třída konceptů ve vstupním prostoru $\Xi_n \subseteq \mathbb{R}^n$ reprezentovaných prahovými obvody (obecně s lineárními prahovými hradlovými funkcemi) s n vstupy, jedním výstupem a w váhami a prahy (tj. $w = |E| + s$ pro množinu hran E a velikost obvodu s). Pak $VC(\mathcal{K}_n) = O(w \log w)$.*

Důkaz: Nechť množina $Q \subseteq \mathbb{R}^n$ velikosti $|Q| = p \geq 2$ je atomizována třídou $\mathcal{K}_n$ konceptů. Označme V množinu hradel prahového obvodu C, který reprezentuje koncept $K \in \mathcal{K}_n$, a $d_G(j)$ značí počet vstupů hradla $j \in V$. Prahové hradlo $j \in V$ obvodu C reprezentuje podle věty 12.10 třídu konceptů VC-dimenze $d_G(j) + 1$. Tedy podle Sauerovy lemmy 12.11 toto hradlo může počítat nejvýše $p^{d_G(j)+1} + 1$ různých lineárních prahových funkcí na konečném definičním oboru $\Xi \subseteq \mathbb{R}^{d_G(j)}$ velikosti $|\Xi| = p$. Proto prahové obvody C reprezentující koncepty z $\mathcal{K}_n$ počítají pro vstupy z Q nejvýše

$$\prod_{j \in V} \left(p^{d_G(j)+1} + 1\right) \leq p^{2w} \tag{12.23}$$

různých funkcí $C : Q \longrightarrow \{0,1\}$. Na druhou stranu, protože Q je atomizována $\mathcal{K}_n$, prahové obvody C reprezentující koncepty z $\mathcal{K}_n$ musí počítat všech 2^p

Dále ve druhé vrstvě bude prahové hradlo q_b s prahem $h(q_b) = \frac{m}{2} + 1$, spojené se všemi hradly u_j, v_j $(j = 0, \ldots, m-1)$ z první vrstvy pomocí vah $w(u_j, q_b) = w(v_j, q_b) = (bin(j))_b$, a tedy jeho zváženou sumu (včetně prahu) lze vyjádřit:

$$\begin{aligned}\xi_{q_b}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) &= \sum_{j=0}^{m-1} w(u_j, q_b) y_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) + \\ &\quad + \sum_{j=0}^{m-1} w(v_j, q_b) y_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) - h(q_b) = \\ &= \sum_{j=0}^{m-1} \left(y_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) + y_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2})\right) (bin(j))_b - \frac{m}{2} - 1\,. \quad (12.33)\end{aligned}$$

V sumě (12.33) je právě $\frac{m}{2}$ sčítanců, pro které je $(bin(j))_b = 1$ (ostatní jsou nulové), a podle (12.31) jsou tyto sčítance aspoň 1 a podle (12.32) existuje mezi nimi člen větší než 2, právě když $g(i_1, i_2) = bin(j)$. Tedy podle (12.33) je $\xi_{q_b}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) \geq 0$, právě když existuje $j \in \{0, \ldots, m-1\}$ takové, že $g(i_1, i_2) = bin(j)$ a současně $(bin(j))_b = 1$. To znamená, že q_b je aktivní, právě když $F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_b) = (g(i_1, i_2))_b = 1$, a tedy prahové hradlo q_b počítá $F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_b)$ pro vstup $\mathbf{e}_{i_1}\mathbf{e}_{i_2}$ a pevné $b \in \{1, \ldots, \log m\}$.

Výpočet $F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_{i_3})$ pro libovolné $\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_{i_3} \in \{0,1\}^n$ na vstupu lze pak ve třech vrstvách realizovat pomocí $\log m$ hradel q_b ve druhé vrstvě pro všechny možné hodnoty $b \in \{1, \ldots, \log m\}$ a jednoduchého podobvodu, který kontroluje, zda q_{i_3} je aktivní.

Na závěr důkazu jen stručně nastíníme, jak odstranit zjednodušující předpoklad, že zobrazení g_{i_2} jsou bijekce pro všechna $i_2 \in \{1, \ldots, m\}$, zatímco detaily lze najít v [182]. Myšlenka je založena na výsledku [181, 204], že pro každé zobrazení $g : \{1, \ldots, m\}^2 \longrightarrow \{0,1\}^{\log m}$ existují čtyři zobrazení $g_k : \{1, \ldots, m\}^2 \longrightarrow \{0,1\}^{\log m}$ pro $k = 1, \ldots, 4$ taková, že jejich projekce $g_{ki_2} : \{1, \ldots, m\} \longrightarrow \{0,1\}^{\log m}$, tj. $g_{ki_2}(i_1) = g_k(i_1, i_2)$ $(k = 1, \ldots, 4)$, jsou bijektivní pro každé $i_2 \in \{1, \ldots, m\}$ a platí

$$g(i_1, i_2) = \begin{cases} g_1(i_1, i_2) \oplus g_2(i_1, i_2) & i_1 \leq \frac{m}{2} \\ g_3(i_1, i_2) \oplus g_4(i_1, i_2) & i_1 > \frac{m}{2}\,. \end{cases} \quad (12.34)$$

Podle předchozího zkonstruujeme 4 oddělené podobvody pro výpočet g_1, g_2, g_3, g_4, které podle (12.34) zkombinujeme tak, aby výsledný obvod C_n počítal $F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_{i_3})$. Zřejmě C_n má v první vrstvě $\Omega(n)$ hradel a druhá vrstva se skládá ze $4 \log n$ hradel. □

Hloubku obvodů ve větě 12.13 lze pro reálné vstupy, tj. $\Xi_n = \mathbb{R}^n$, snížit na 2 [245].

a jeho projekce $g_{i_2} : \{1, \ldots, m\} \longrightarrow \{0,1\}^{\log m}$ pro $i_2 = 1, \ldots, m$ vztahem $g_{i_2}(i_1) = g(i_1, i_2)$. Pro jednoduchost předpokládejme, že zobrazení g_{i_2} jsou bijekce pro všechna $i_2 \in \{1, \ldots, m\}$, a později naznačíme, jak tento zjednodušující předpoklad odstranit. Tedy existují inverzní zobrazení $g_{i_2}^{-1} : \{0,1\}^{\log m} \longrightarrow \{1, \ldots, m\}$.

Nejprve zkonstruujeme podobvod, který pro vstup $\mathbf{e}_{i_1}\mathbf{e}_{i_2} \in \{0,1\}^{2m}$ počítá hodnotu $F(\mathbf{e}_{i_1}\mathbf{e}_{i_2}\mathbf{e}'_b)$ pro pevné $b \in \{1, \ldots, \log m\}$. První vrstva se skládá z m párů prahových hradel u_j, v_j pro $j = 0, \ldots, m-1$ (srovnejte s technikou důkazu věty 7.32), které mají nulový práh $h(u_j) = h(v_j) = 0$ $(j = 0, \ldots, m-1)$ a jsou spojeny se vstupními neurony $x_{11}, \ldots, x_{1m}, x_{21}, \ldots, x_{2m}$ pomocí spojů s následujícími váhami:

$$\begin{array}{llll} w(x_{1i}, u_j) = i & w(x_{2i}, u_j) = -g_i^{-1}(bin(j)) & j = 0, \ldots, m-1 \\ w(x_{1i}, v_j) = -i & w(x_{2i}, v_j) = g_i^{-1}(bin(j)) & i = 1, \ldots, n\,, \end{array} \quad (12.27)$$

kde $bin(j) \in \{0,1\}^{\log m}$ je binární reprezentace čísla $j \in \{0, \ldots, m-1\}$.

Vypočteme zvážené sumy (včetně prahu) $\xi_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2})$, $\xi_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2})$ prahových hradel u_j, v_j pro vstup $\mathbf{e}_{i_1}\mathbf{e}_{i_2} \in \{0,1\}^{2m}$ tak, že využijeme definici vah (12.27) a faktu, že i-tá složka $(\mathbf{e}_{i_1})_i = 0$ pro $i \neq i_1$ a $(\mathbf{e}_{i_2})_i = 0$ pro $i \neq i_2$:

$$\begin{aligned}\xi_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) &= \sum_{i=1}^{m} w(x_{1i}, u_j)(\mathbf{e}_{i_1})_i + \sum_{i=1}^{m} w(x_{2i}, u_j)(\mathbf{e}_{i_2})_i - h(u_j) = \\ &= w(x_{1i_1}, u_j) + w(x_{2i_2}, u_j) = i_1 - g_{i_2}^{-1}(bin(j)) \quad (12.28) \\ \xi_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) &= \sum_{i=1}^{m} w(x_{1i}, v_j)(\mathbf{e}_{i_1})_i + \sum_{i=1}^{m} w(x_{2i}, v_j)(\mathbf{e}_{i_2})_i - h(v_j) = \\ &= w(x_{1i_1}, v_j) + w(x_{2i_2}, v_j) = -i_1 + g_{i_2}^{-1}(bin(j)) \quad (12.29)\end{aligned}$$

pro $j = 0, \ldots, m-1$. Porovnáním (12.28) a (12.29) dostáváme

$$\xi_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = -\xi_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = i_1 - g_{i_2}^{-1}(bin(j)) \quad j = 0, \ldots, m-1\,. \quad (12.30)$$

Tedy podle (12.30) je aspoň jedno hradlo z každého páru u_j, v_j $(j = 0, \ldots, m-1)$ aktivní, tj.

$$\text{buď } y_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = 1\,, \text{ nebo } y_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = 1\,. \quad (12.31)$$

Navíc obě hradla u_j, v_j $(0 \leq j \leq m-1)$ jsou aktivní současně, tj.

$$y_{u_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = y_{v_j}(\mathbf{e}_{i_1}\mathbf{e}_{i_2}) = 1\,, \quad (12.32)$$

právě když $i_1 = g_{i_2}^{-1}(bin(j))$, tj. $g_{i_2}(i_1) = bin(j)$, což lze přepsat $g(i_1, i_2) = bin(j)$.

Důkaz:

(i) Tvrzení je přímým důsledkem vět 12.8, 12.7 a 12.10.

(ii) Tvrzení je přímým důsledkem vět 12.8, 12.13 a 12.7, 12.12.

(iii) Tvrzení je přímým důsledkem vět 12.8, 12.7 a poznámky k důkazu věty 12.13. □

Důsledek 12.14 také udává dolní odhad časové složitosti učení acyklických neuronových sítí v PAC-modelu, protože učící algoritmus musí aspoň načíst tréninkové vzory potřebné pro generalizaci.

Také existují dolní odhady na velikost tréninkové množiny při učení v PAC-modelu s dotazy na shodu [274], resp. s dotazy na příslušnost [60] (viz definici 12.3), které nesníží podstatně vzorkovou složitost (ε, δ)-učícího algoritmu.

12.3 PAC-model a tréninkový problém

V podkapitole 12.2 jsme se zabývali vzorkovou složitostí učení v PAC-modelu, tj. velikostí tréninkové množiny, která pro acyklické neuronové sítě zajistí (např. podle důsledku 12.14) správnou generalizaci. Zřejmě polynomiální počet tréninkových vzorů je nutnou podmínkou k efektivní naučitelnosti v PAC-modelu, avšak není postačující. K úplné charakterizaci PAC-naučitelnosti je navíc potřeba polynomiální pravděpodobnostní algoritmus, který řeší vyhledávací verzi tréninkového problému v polynomiálním čase pro architekturu, která reprezentuje učené koncepty. To znamená, že efektivní generalizace neuronových sítí v jistém smyslu vyžaduje efektivní řešení tréninkového problému. Před tím, než ukážeme příslušný vztah mezi PAC-modelem a LP, připomeneme definici pravděpodobnostní třídy složitosti RP.

Definice 12.15 *Pravděpodobnostní algoritmus pro vyhledávací problém nalezne s pravděpodobností aspoň $\frac{1}{2}$ jeho správné řešení, pokud existuje, a v opačném případě vždy ohlásí, že neexistuje. Speciálně pro rozhodovací verze problémů např. pravděpodobnostní Turingův stroj M_p* rozhoduje *jazyk $L \subseteq \{0,1\}^\star$, značíme $L = L(M_p)$, jestliže pro každé vstupní slovo $\mathbf{x} \in \{0,1\}^\star$, pokud $\mathbf{x} \in L$, pak M_p s pravděpodobností aspoň $\frac{1}{2}$ slovo $\mathbf{x}$ přijímá (výstup je 1), a pokud $\mathbf{x} \notin L$, pak M_p vždy $\mathbf{x}$ zamítá (výstup je 0). Označme RP třídu jazyků (problémů) rozhodnutelných pravděpodobnostním Turingovým strojem v polynomiálním čase.*

Věta 12.13 také platí pro analogové prahové obvody s reálnými váhami a se saturovanou lineární aktivační funkcí (1.8), resp. standardní sigmoidou (1.9), jak lze nahlédnout z jejího důkazu. Nedávný výsledek [159] zlepšil tento dolní odhad na $\Omega(w^2)$ pro analogové prahové obvody s velkou třídou aktivačních funkcí včetně (1.8) a (1.9). Jsou také známy horní odhady VC-dimenze, tj. analogie věty 12.12 pro analogové prahové obvody s reálnými vstupy. V případě saturované lineární aktivační funkce je VC-dimenze $O(w^2)$ [79] a pro standardní sigmoidu je nejvýše $O(w^4)$ [147], což se od uvedeného dolního odhadu liší kvadraticky.

Z uvedených odhadů VC-dimenze prahových obvodů lze určit velikost tréninkové množiny při učení acyklických neuronových sítí pro jejich správnou generalizaci, tj. vzorkovou složitost příslušného (ε, δ)-učícího algoritmu.

Důsledek 12.14 *Počet p tréninkových vzorů (ε, δ)-učícího algoritmu pro třídu konceptů reprezentovaných pomocí*

(i) *lineárních prahových funkcí s n proměnnými je aspoň*

$$p = \Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{n+1}{\varepsilon}\right) \tag{12.35}$$

a stačí

$$p = O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{n+1}{\varepsilon}\log\frac{1}{\varepsilon}\right) . \tag{12.36}$$

(ii) *prahových obvodů s w váhami a prahy je aspoň*

$$p = \Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{w\log w}{\varepsilon}\right) \tag{12.37}$$

a stačí

$$p = O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{w\log w}{\varepsilon}\log\frac{1}{\varepsilon}\right) . \tag{12.38}$$

(iii) *analogových prahových obvodů s aktivační funkcí standardní sigmoidou (1.9), s w váhami a prahy je aspoň*

$$p = \Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{w^2}{\varepsilon}\right) \tag{12.39}$$

a stačí

$$p = O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{w^4}{\varepsilon}\log\frac{1}{\varepsilon}\right) . \tag{12.40}$$

Z věty 12.16 vyplývá, že perceptron s reálnými vstupy je PAC-naučitelný.

Důsledek 12.17 *Třída konceptů $\mathcal{K}$ reprezentovaná pomocí lineárních prahových funkcí je PAC-naučitelná, pokud uvažujeme logaritmickou složitost operací nad reálnými čísly.*

Důkaz: Podle věty 12.10 je VC-dimenze třídy $\mathcal{K}$ lineární. Dále vyhledávací verzi tréninkového problému pro architekturu s jedním hradlem, které počítá lineární prahovou funkci, lze řešit pomocí lineárního programování v polynomiálním čase [145]. Tvrzení je pak přímým důsledkem věty 12.16. □

Věta 12.16 má jednodušší formulaci pro binární vstupní prostor, která např. také implikuje PAC-naučitelnost perceptronu s binárními vstupy.

Věta 12.18 (Natarajan [203]) *Nechť $\mathcal{K} = \bigcup_{n\geq 0}\mathcal{K}_n$, kde $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ a $\Xi_n = \{0,1\}^n$, je třída polynomiálně velkých konceptů reprezentovaná posloupnostmi prahových obvodů. Potom $\mathcal{K}$ je PAC-naučitelná, právě když $\log|\mathcal{K}_n|$ je polynomiální vzhledem k n a existuje polynomiální pravděpodobnostní algoritmus, který řeší vyhledávací verzi tréninkového problému (viz podkapitolu 11.1) pro architekturu, která reprezentuje $\mathcal{K}_n$.*

Důkaz: Podle věty 12.16 stačí dokázat, že $VC(\mathcal{K}_n)$ je polynomiální vzhledem k n, právě když $\log|\mathcal{K}_n|$ je polynomiální vzhledem k n. Z definice 12.6 vyplývá, že

$$2^{VC(\mathcal{K}_n)} \leq |\mathcal{K}_n|\,, \tag{12.42}$$

protože pro každou podmnožinu $R \subseteq Q$ atomizované množiny Q musí existovat aspoň jeden koncept $K \in \mathcal{K}_n$ takový, že $K \cap Q = R$. Ze vztahu (12.42) vyplývá, že $VC(\mathcal{K}_n) = O(\log|\mathcal{K}_n|)$. Na druhou stranu pro $\Xi_n = \{0,1\}^n$ je $|\Xi_n| = 2^n$. Tedy podle Sauerovy lemmy 12.11 dostáváme:

$$|\mathcal{K}_n| \leq (2^n)^{VC(\mathcal{K}_n)} + 1\,, \tag{12.43}$$

což implikuje $\log|\mathcal{K}_n| = O(nVC(\mathcal{K}_n))$. □

Důsledek 12.19 *Třída konceptů $\mathcal{K}$ reprezentovaná pomocí booleovských prahových funkcí je PAC-naučitelná.*

Důkaz: Důkaz probíhá podobně jako důkaz důsledku 12.17 s tím, že můžeme využít větu 12.18, protože počet booleovských prahových funkcí n proměnných je podle důsledku 6.33 $2^{\Theta(n^2)}$, tj. $\log|\mathcal{K}_n| = \Theta(n^2)$. □

Uvedenou charakterizaci efektivního učení v PAC-modelu lze také využít pro důkaz toho, že daná třída konceptů není PAC-naučitelná za předpokladu, že $RP \neq NP$ (zřejmě $RP \subseteq NP$), který implikuje známou hypotézu $P \neq NP$. Zřejmě se při takovém důkazu můžeme omezit jen na

Věta 12.16 (Blumer, Ehrenfeucht, Haussler, Warmuth [40]) *Nechť $\mathcal{K} = \bigcup_{n\geq 0}\mathcal{K}_n$, kde $\mathcal{K}_n \subseteq \mathcal{P}(\Xi_n)$ a $\Xi_n = \mathbb{R}^n$, je třída polynomiálně velkých konceptů reprezentovaná posloupnostmi prahových obvodů. Potom $\mathcal{K}$ je PAC-naučitelná, právě když $VC(\mathcal{K}_n)$ je polynomiální vzhledem k n a existuje polynomiální pravděpodobnostní algoritmus, který řeší vyhledávací verzi tréninkového problému (viz podkapitolu 11.1) pro architekturu, která reprezentuje $\mathcal{K}_n$.*

Důkaz: Nechť tedy nejprve třída $\mathcal{K}$ konceptů je PAC-naučitelná pomocí učícího algoritmu U, tj. U je polynomiální časové složitosti. Podle věty 12.8 U vyžaduje aspoň $\Omega(VC(\mathcal{K}))$ tréninkových vzorů, které musí načíst, tedy jeho časová složitost je aspoň $\Omega(VC(\mathcal{K}))$, a proto nutně $VC(\mathcal{K}_n)$ je polynomiální vzhledem k n.

Dále s využitím algoritmu U ukážeme existenci polynomiálního pravděpodobnostního algoritmu U_p pro řešení vyhledávací verze LP. Podle definice 12.2 algoritmus U pracuje jen pro cílové koncepty z dané třídy konceptů $\mathcal{K}$, avšak lze jej jednoduše modifikovat tak, že pro koncepty, které nelze reprezentovat pomocí dané architektury, U po vyčerpání omezeného polynomiálního času odpoví např. libovolnou hypotézou. Nechť tedy

$$\mathcal{T} = \{(\mathbf{x}_k, d_k) \mid \mathbf{x}_k \in \Xi_n,\, d_k \in \{0,1\},\, k = 1,\ldots,p\} \tag{12.41}$$

je tréninková množina. Pro aplikaci U definujeme pravděpodobnostní rozdělení $D : \Xi_n \longrightarrow \langle 0,1\rangle$ vstupního prostoru, které je pro tréninkové vzory z $\mathcal{T}$ rovnoměrné, tj. $D(\mathbf{x}_k) = \frac{1}{p}$ pro $\mathbf{x}_k \in \mathcal{T}$ a $D(\mathbf{x}) = 0$ jinak. Algoritmus U_p generuje tréninkové vzory podle pravděpodobnostního rozdělení D a zavolá algoritmus U s přesností $\varepsilon = \frac{1}{p+1}$ a konfidencí $\delta = \frac{1}{2}$, který díky zvolené přesnosti s pravděpodobností aspoň $\frac{1}{2}$ odpoví hypotézou $K \in \mathcal{K}$, resp. její reprezentací pomocí prahového obvodu, která je konzistentní s $\mathcal{T}$, pokud existuje. Algoritmus U_p konzistenci K s $\mathcal{T}$ v polynomiálním čase ověří a pokud souhlasí, předá ji na výstupu. V opačném případě ohlásí, že řešení LP neexistuje. Zřejmě U_p je polynomiální pravděpodobnostní algoritmus, který řeší vyhledávací verzi tréninkového problému.

Na druhou stranu předpokládejme, že $VC(\mathcal{K}_n)$ je polynomiální vzhledem k n a existuje polynomiální pravděpodobnostní algoritmus U_p, který řeší vyhledávací verzi LP. Učící algoritmus U pro danou přesnost ε a konfidenci δ načte tréninkovou množinu polynomiální velikosti (12.3) a pro ní zavolá pravděpodobnostní algoritmus U_p, který pro příslušnou architekturu nalezne reprezentaci konzistentní hypotézy. Zřejmě podle věty 12.7 je U v PAC modelu korektní učící algoritmus pro $\mathcal{K}$. □

binární vstupní prostor $\Xi_n = \{0,1\}^n$, protože negativní výsledky o PAC-naučitelnosti pro binární případ lze jednoduše převést na reálný vstupní prostor vhodnou volbou „těžkého“ pravděpodobnostního rozdělení, které je nenulové jen pro binární vstupy.

Věta 12.20 (Pitt, Valiant [225]) *Nechť $\mathcal{K}$ je třída polynomiálně velkých konceptů reprezentovaná posloupnostmi prahových obvodů. Jestliže $RP \neq NP$ a LP pro tuto reprezentaci je NP-těžký, pak $\mathcal{K}$ není PAC-naučitelná.*

Důkaz: Protože LP pro $\mathcal{K}$ je NP-těžký problém, $LP \in RP$ by implikovalo, že $RP = NP$, a tedy platí $LP \notin RP$. Podle věty 12.16 pak $\mathcal{K}$ nemůže být PAC-naučitelná. □

Využitím výsledků o NP-úplnosti LP z kapitoly 11 a pomocí věty 12.20 dostáváme, že mnohé architektury acyklických neuronových sítí nemohou (za předpokladu $RP \neq NP$) při učení z příkladů efektivně generalizovat koncepty, které lze pomocí nich reprezentovat.

Důsledek 12.21 *Jestliže $RP \neq NP$, pak třída $\mathcal{K} = \bigcup_{n\geq 0} \mathcal{K}_n$ konceptů není PAC-naučitelná, pokud $\mathcal{K}_n$ je reprezentována např. pomocí*

(i) *prahové 3-sítě.*

(ii) *prahového obvodu hloubky 2, s polynomiální počtem prahových hradel v první vrstvě a s jedním výstupním AND hradlem ve druhé vrstvě.*

(iii) *kaskádové 2-sítě.*

(iv) *analogové prahové 3-sítě (resp. analogové kaskádové 2-sítě) se saturovanou lineární aktivační funkcí (1.8).*

(v) *analogové prahové 3-sítě se standardní sigmoidální aktivační funkcí (1.9) s polynomiální reprezentací vah (včetně prahů) a s nulovým prahem u výstupního hradla.*

(vi) *analogového prahového obvodu hloubky 2 s polynomiální reprezentací (včetně prahů), se zadaným konstantním počtem hradel v první vrstvě a s jedním výstupním hradlem s nulovým prahem ve druhé vrstvě.*

Důkaz: Pro aplikaci věty 12.20 je nejprve třeba ověřit, že koncepty z třídy $\mathcal{K}_n$ mají ve všech případech (i)–(vi) polynomiální velikost. Pro diskrétní prahové obvody tento fakt vyplývá z věty 6.30, pro případ (iv) lze příslušný argument najít v [57] a v případě (v), (vi) je tento požadavek vyjádřen v předpokladech. Tvrzení pak plyne z věty 12.20 a

(i) z věty 11.16.

(ii) z části (ii) důsledku 11.17.

(iii) z věty 11.21.

(iv) z NP-úplnosti LP pro analogové prahové obvody se saturovanou lineární aktivační funkcí [57] (viz podkapitolu 11.5).

(v) z věty 11.23.

(vi) z důsledku 11.24. □

Bez důkazu zformulujeme analogii věty 12.20 a příklad její aplikace pro PAC-model s dotazy na shodu a příslušnost (podobná věta platí také pro přesné učení s dotazy na shodu [19] s předpokladem $P \neq NP$). V tomto případě se místo LP uvažuje tzv. *problém reprezentace.*

Věta 12.22 (Aizenstein, Hellerstein, Pitt [3], Hegedüs [99]) *Nechť $\mathcal{K}$ je třída polynomiálně velkých konceptů reprezentovaná posloupnostmi prahových obvodů. Jestliže $NP \neq co-NP$ a* problém reprezentace, *tj. rozhodnout, zda daná booleovská formule v disjunktivní normální formě reprezentuje koncept z $\mathcal{K}$, je NP-těžký, pak $\mathcal{K}$ není PAC-naučitelná s dotazy na shodu a příslušnost (viz definici 12.3).*

Důsledek 12.23 (Hegedüs [101]) *Nechť $\mathcal{K}$ je třída konceptů reprezentovaných pomocí prahového obvodu hloubky 2, se 3 prahovými hradly v první vrstvě a s jedním výstupním OR hradlem ve druhé vrstvě. Problém reprezentace pro $\mathcal{K}$ je NP-těžký. Proto, pokud $NP \neq co-NP$, třída $\mathcal{K}$ není PAC-naučitelná s dotazy na shodu a příslušnost.*

12.4 Perceptronový učící algoritmus

Podle důsledku 12.17 se můžeme vyhnout těžkosti učení (viz důsledek 12.21), pokud se omezíme jen na nejjednodušší případ perceptronu, který počítá lineární prahovou funkci. Kromě polynomiálního algoritmu pro lineární programování využitém v důkazu důsledku 12.17 existuje např. efektivní algoritmus pro učení perceptronu v modelu přesného učení s dotazy na shodu [184]. Na druhou stranu v této podkapitole budeme analyzovat klasický perceptronový učící algoritmus (viz podkapitolu 2.1), o kterém ukážeme, že není efektivní.

Nejprve dokážeme klasický výsledek o konvergenci perceptronového učícího algoritmu. Pro jednoduchost se omezíme na jeden perceptron s nulovým prahem (viz lemmu 6.7) a omezenou doménou (viz podkapitolu 6.2). Dále

prahová funkce f podle (6.16) separabilní. Tedy uvedený skalární součin $\mathbf{w}^{(t)}\mathbf{w}$ můžeme pomocí ξ_{min} zdola odhadnout:

$$\mathbf{w}^{(t)}\mathbf{w} = \sum_{k=1}^{t} \mathbf{x}'^{(k)}\mathbf{w} \geq t\xi_{min} > 0\,, \tag{12.50}$$

což lze upravit:

$$\left|\mathbf{w}^{(t)}\mathbf{w}\right|^2 \geq t^2\xi_{min}^2\,. \tag{12.51}$$

Pomocí Cauchy-Schwarzovy nerovnosti $|\mathbf{w}^{(t)}\mathbf{w}|^2 \leq ||\mathbf{w}^{(t)}||^2 \cdot ||\mathbf{w}||^2$ ze vztahu (12.51) dostáváme:

$$\left\|\mathbf{w}^{(t)}\right\|^2 \geq \frac{\xi_{min}^2}{||\mathbf{w}||^2}t^2\,. \tag{12.52}$$

Na druhou stranu zřejmě podle (12.45) platí

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} + \mathbf{x}'^{(k)}\,, \tag{12.53}$$

z čehož dostáváme

$$\left\|\mathbf{w}^{(k)}\right\|^2 = \left\|\mathbf{w}^{(k-1)}\right\|^2 + 2\mathbf{w}^{(k-1)}\mathbf{x}'^{(k)} + \left\|\mathbf{x}'^{(k)}\right\|^2\,. \tag{12.54}$$

Předpokládáme, že $f^{(k-1)}(\mathbf{x}^{(k)}) \neq d^{(k)}$, a tedy $\mathbf{w}^{(k-1)}\mathbf{x}'^{(k)} \leq 0$. Díky tomu lze (12.54) přepsat:

$$\left\|\mathbf{w}^{(k)}\right\|^2 - \left\|\mathbf{w}^{(k-1)}\right\|^2 \leq \left\|\mathbf{x}'^{(k)}\right\|^2\,. \tag{12.55}$$

Nyní sčítáme (12.55) pro $k = 1,\ldots,t$ a shora odhadneme $||\mathbf{w}^{(t)}||^2$ pomocí x_{max}:

$$\left\|\mathbf{w}^{(t)}\right\|^2 \leq \sum_{k=1}^{t}\left\|\mathbf{x}'^{(k)}\right\|^2 \leq tx_{max}\,. \tag{12.56}$$

Porovnáním (12.52) a (12.56) dostáváme

$$\frac{\xi_{min}^2}{||\mathbf{w}||^2}t^2 \leq tx_{max}\,, \tag{12.57}$$

z čehož vyplývá:

$$t \leq \frac{||\mathbf{w}||^2\, x_{max}}{\xi_{min}^2}\,, \tag{12.58}$$

a tedy nutně existuje $t^\star$ splňující (12.46) a $f^{(t^\star)}(\mathbf{x}_k) = d_k$ pro $k = 1,\ldots,p$. □

v adaptivní dynamice (2.5) budeme uvažovat počáteční nulovou konfiguraci a rychlost konvergence $\varepsilon = 1$.

Věta 12.24 (Rosenblatt [239]) *Nechť $\Xi_n \subseteq \mathbb{R}^n$ je omezený vstupní prostor. Označme $(w_1^{(t)},\ldots,w_n^{(t)};0)$ reprezentaci lineární prahové funkce $f^{(t)}$: $\Xi_n \longrightarrow \{0,1\}$ s n vstupy v čase $t \geq 0$ učení, kde na začátku volíme nulové váhy, tj. $w_i^{(0)} = 0$ pro $i = 1,\ldots,n$. Nechť*

$$\mathcal{T} = \{(\mathbf{x}_k, d_k) \mid \mathbf{x}_k \in \Xi_n,\, d_k \in \{0,1\},\, k = 1,\ldots,p\} \tag{12.44}$$

je odpovídající tréninková množina a nechť existuje reprezentace $(w_1,\ldots,w_n;0)$ lineární prahové funkce $f : \Xi_n \longrightarrow \{0,1\}$ konzistentní s $\mathcal{T}$, tj. $f(\mathbf{x}_k) = d_k$ pro $k = 1,\ldots,p$. Dále předpokládejme, že v čase $t \geq 1$ je vybrán tréninkový vzor $(\mathbf{x}^{(t)}, d^{(t)}) \in \mathcal{T}$, pro který $f^{(t-1)}(\mathbf{x}^{(t)}) \neq d^{(t)}$, a podle něj je vytvořena reprezentace $f^{(t)}$ následujícím způsobem:

$$w_i^{(t)} = \begin{cases} w_i^{(t-1)} + x_i^{(t)} & f^{(t-1)}(\mathbf{x}^{(t)}) = 0 \;\; a \;\; d^{(t)} = 1 \\ w_i^{(t-1)} - x_i^{(t)} & f^{(t-1)}(\mathbf{x}^{(t)}) = 1 \;\; a \;\; d^{(t)} = 0 \end{cases} \tag{12.45}$$

pro i=1,...,n, což je jiná formulace perceptronového učícího pravidla (2.5). Potom existuje čas $t^\star$, kdy $f^{(t^\star)}(\mathbf{x}_k) = d_k$ pro $k = 1,\ldots,p$, a platí:

$$t^\star \leq \frac{||\mathbf{w}||^2\, x_{max}}{\xi_{min}^2}\,, \tag{12.46}$$

kde $\mathbf{w} = (w_1,\ldots,w_n)$, tj. $||\mathbf{w}||^2 = \sum_{i=1}^n w_i^2$, a

$$x_{max} = \max\{||\mathbf{x}||^2 \mid \mathbf{x} \in \Xi_n\} \tag{12.47}$$

$$\xi_{min} = \min\left\{\sum_{i=1}^{n} w_i x_i > 0 \;\middle|\; \mathbf{x} \in \Xi_n\right\}\,. \tag{12.48}$$

Důkaz: Podle (12.45) můžeme vyjádřit váhový vektor $\mathbf{w}^{(t)}$ reprezentace lineární prahové funkce $f^{(t)}$ v čase $t \geq 1$ jako součet

$$\mathbf{w}^{(t)} = \sum_{k=1}^{t}\mathbf{x}'^{(k)}, \quad \text{kde } \mathbf{x}'^{(k)} = \begin{cases} \mathbf{x}^{(k)} & d^{(k)} = 1 \\ -\mathbf{x}^{(k)} & d^{(k)} = 0\,. \end{cases} \tag{12.49}$$

Rovnici (12.49) vynásobíme zprava pomocí váhového vektoru $\mathbf{w}$ reprezentace f konzistentní s $\mathcal{T}$. Díky této konzistenci je $\mathbf{x}'^{(k)}\mathbf{w} \geq 0$ pro $k = 1,\ldots,t$. Navíc můžeme bez újmy na obecnosti předpokládat, že $\mathbf{x}'^{(k)}\mathbf{w} \neq 0$ ($k = 1,\ldots,t$), protože pro konečný počet p vstupů tréninkových vzorů je lineární

Čas perceptronového učícího algoritmu ve větě 12.24 je měřen pomocí počtu adaptací váhového vektoru. Uvažuje se zde model přesného učení s dotazy na shodu tak, že každá negativní odpověď na dotaz poskytne jako protipříklad tréninkový vzor, ve kterém perceptron chybuje. Pro booleovskou doménu tak získáme horní odhad časové složitosti perceptronového učícího algoritmu v tomto modelu.

Důsledek 12.25 *Nechť jsou splněny předpoklady věty 12.24 a navíc* $\Xi_n = \{0,1\}^n$. *Potom perceptronový učící algoritmus (12.45) konverguje k reprezentaci booleovské prahové funkce konzistentní s tréninkovou množinou* $\mathcal{T}$ *nejpozději po* $t^\star = O(n^{n+3})$ *adaptacích vah.*

Důkaz: Zřejmě pro $\Xi_n = \{0,1\}^n$ je $x_{max} = n$ a podle věty 6.30 existuje celočíselný váhový vektor $\mathbf{w}$, a tedy $\xi_{min} \geq 1$. Navíc podle této věty známe horní odhad váhy booleovské prahové funkce, pomocí kterého dostáváme $||\mathbf{w}||^2 \leq n \max_i |w_i|^2 \leq n(n+1)^{n+1} = O(n^{n+2})$. Dohromady pak podle (12.46) obdržíme $t^\star = O(n^{n+3})$. □

Pro booleovskou prahovou funkci s malými váhami lze analogickou úvahou jako v důkazu důsledku 12.25 získat polynomiální časovou složitost perceptronového učícího algoritmu. Také v PAC-modelu se speciálním pravděpodobnostním rozdělením vzorů [30] stačí perceptronovému algoritmu čas $O(n^2/\varepsilon^3)$, kde ε je přesnost. Avšak pro obecné váhy tento algoritmus zřejmě vyžaduje čas úměrný aspoň váze $|w|$ booleovské prahové funkce f, protože podle (12.45) přírůstek absolutní hodnoty libovolné váhy perceptronu při jedné adaptaci je nejvýše jednotkový. P. M. Lewis [171] dokonce ukázal, že v obecném případě je potřeba aspoň $|w|^2/(n+1)$ adaptací, což spolu s dolním odhadem váhy $|w|$ ve větě 6.38 potvrzuje exponenciální čas perceptronového učícího algoritmu. Tento výsledek lze zesílit pro PAC-model s obecným pravděpodobnostním rozdělením vzorů. Příslušné tvrzení uvádíme bez důkazu.

Věta 12.26 (Schmitt [252]) *Nechť jsou splněny předpoklady důsledku 12.25. Potom třída konceptů reprezentovaných pomocí booleovských prahových funkcí není v* PAC*-modelu naučitelná pomocí perceptronového učícího algoritmu (12.45).*

V literatuře [31, 33, 37] bylo v poslední době publikováno několik relativně komplikovaných učících algoritmů pro dvouvrstvý perceptron s konstantním počtem (např. 2, resp. méně než 4) skrytých neuronů (resp. vstupů), které jsou v jistém smyslu teoreticky efektivní v různých variantách PAC-modelu (např. s dotazy na shodu, resp. se speciálním pravděpodobnostním rozdělením vzorů), avšak buď jsou prakticky exponenciální, nebo v reálných úlohách vykazují malou generalizační schopnost [168], a proto je lze v praxi pravděpodobně zatím jen v omezené míře použít.

Část III

Aproximace funkcí pomocí neuronových sítí

Tato část se dělí na dvě kapitoly, z nichž první se zabývá otázkou, jaké funkce umíme přibližně vyjádřit různými modely dopředných sítí. Druhá kapitola pojednává o vlastnosti sítí, kterou nazýváme funkční ekvivalence a o jejím možném využití k urychlení učícího procesu.

V první kapitole zkoumáme sílu jednotlivých modelů neuronových sítí na základě toho, jaké funkce dokáže daný model aproximovat, tj. vyjádřit s libovolnou přesností. Bude nás zajímat vlastnost *univerzální aproximace*, což je schopnost sítě aproximovat každou spojitou nebo integrovatelnou funkci. Pokud nějaký model tuto vlastnost má, znamená to, že je v jistém smyslu univerzálním výpočetním prostředkem — je teoreticky schopen reprezentovat všechny „rozumné“ funkce.

Postupně se v této kapitole zaměříme na perceptronové sítě s jednou a dvěma skrytými vrstvami, na RBF sítě, sítě se semi-lokálními jednotkami a na kaskádové sítě. Kromě vlastních výsledků nás bude zajímat i použitý důkazový aparát, který je velmi rozmanitý a sahá od nezvyklých prostředků kalkulu rozšířených v minulosti po hluboké věty funkcionální analýzy. Většina důkazů není uvedena do všech podrobností, klademe důraz na vyložení hlavních myšlenek a do technických detailů nebo částí vyžadujících pokročilý matematický aparát se pouštět nebudeme.

Ve druhé kapitole si vysvětlíme, co to je *funkční ekvivalence*, budeme zkoumat, jak vypadají funkčně ekvivalentní perceptronové a RBF sítě a podáme charakterizaci funkční ekvivalence pro běžně používané přechodové funkce.

V další části druhé kapitoly se seznámíme s genetickými algoritmy, které jsou samy o sobě zajímavou alternativou k jiným učícím či optimalizačním postupům. Navíc, právě genetické učení nám umožní — na rozdíl od gradientních metod — využít charakterizaci funkční ekvivalence k redukci prohledávacího prostoru a tím i ke zrychlení celého procesu učení. Ukážeme si tzv. kanonický genetický algoritmus, který slouží k učení RBF sítí.

V dalším budeme pracovat s prostory funkcí, které jsou vždy definovány pomocí nějaké třídy funkcí a odpovídající normy těchto funkcí. Dva pro nás nejdůležitější prostory budou: (srovnej [235]):

- Prostor všech spojitých funkcí nad kompaktním prostorem X (označovaný $C(X)$) se supremovou normou:

$$||f|| = \sup_{x \in X} |f(x)| .$$

- Prostor $\mathcal{L}_p$ funkcí, pro něž je $|f|^p$ integrovatelná na množině X, s tzv. $\mathcal{L}_p$ normou:

$$||f|| = \left[\int_{x \in X} |f(x)|^p dx \right]^{\frac{1}{p}} .$$

Pohybujeme-li se v prostoru $\mathbb{R}^n$, často za kompaktní $X \subset \mathbb{R}^n$ bereme jednotkovou krychli $\mathcal{I}^n$.

Norma indukuje metriku známým vztahem $\rho(x,y) = ||x-y||$. Na prostoru se supremovou normou měříme tedy vzdálenost dvou funkcí jako supremum rozdílu jejich funkčních hodnot. U $\mathcal{L}_p$ normy počítáme integrál (mocniny) jejich rozdílu. Vzdálenost měřená $\mathcal{L}_p$ metrikou bere v úvahu globální chování obou funkcí, zatímco u supremové metriky záleží i na případných lokálních výkyvech. Dokážeme si tedy představit funkce, které jsou „blízko“ měříme-li jejich vzdálenost $\mathcal{L}_p$ metrikou, ale „daleko“ dle supremové metriky. Opačně ovšem, jsou-li dvě funkce blízko měřeno supremovou metrikou, jsou blízko i dle $\mathcal{L}_p$.

Definice 13.1 *Nechť U je třída funkcí, T její podmnožina a ρ metrika na U. Třídu T nazveme* aproximátorem *vzhledem k (U,ρ), je-li T hustá v U (vzhledem k topologii indukované metrikou ρ). Je-li třída funkcí T aproximátorem vzhledem ke třídě spojitých reálných funkcí (nebo $\mathcal{L}_p$ funkcí), nazveme ji* univerzálním aproximátorem.

Mocným nástrojem k dokazování aproximačních schopností je *Stone-Weierstrassova věta*. Připomeňme, že podalgebra nějaké algebry je její podmnožina uzavřená na součin.

Věta 13.2 (Stone-Weierstrasse) *Nechť S je Hausdorffův prostor a $A \subseteq C(S)$ uzavřená podalgebra. Pokud A obsahuje všechny konstantní funkce a funkce rozdělující body (tj. pro každé $x,y \in S\ \exists f \in A; f(x) \neq f(y)$) a je-li nadto symetrická (tj. z $f \in A$ plyne $\bar{f} \in A$), pak $A = C(S)$.*

Kapitola 13

Univerzální aproximace

V této kapitole budeme vyšetřovat aproximační vlastnosti jednotlivých modelů dopředných sítí. V úvodu si definujeme potřebné pojmy společné pro celou kapitolu a pak se budeme věnovat jednotlivým modelům sítí. U perceptronových sítí se dvěma skrytými vrstvami využijeme Kolmogorovovu větu o reprezentaci spojitých funkcí více proměnných pomocí spojitých funkcí jedné proměnné. Dále si ukážeme poměrně obecné aproximační věty o perceptronových sítích s jednou skrytou vrstvou a o RBF sítích. Podíváme se, proč sítě se semi-lokálními jednotkami, na rozdíl od ostatních modelů, vlastnost univerzální aproximace nemají. Nakonec si dokážeme aproximační výsledek pro podtřídu kaskádových sítí, které budeme studovat v oboru komplexních čísel pomocí klasického matematického aparátu řetězcových zlomků.

13.1 Základní pojmy

Vlastnost univerzální aproximace je definována pomocí základních pojmů matematické topologie, které si nyní velmi stručně připomeneme.

Podmnožina $V \subset X$ topologického prostoru $(X,\mathcal{T})$ je *otevřená*, když ke každému $x \in V$ existuje okolí obsažené ve V. Množina $F \subset X$ je *uzavřená*, je-li $X \setminus F$ otevřená. Pro $A \subset X$ definujeme *uzávěr* $\overline{A}$ jako nejmenší uzavřenou množinu obsahující A. Dále říkáme, že $A \subset X$ je *hustá*, je-li $\overline{A} = X$, tedy uzávěr této množiny je již celý prostor.

Topologický prostor X je *Hausdorffův*, když pro každou dvojici různých bodů $x_1, x_2 \in X$ existují otevřené množiny $U_1, U_2 \subset X$ takové, že $x_1 \in U_1$, $x_2 \in U_2$ a $U_1 \cap U_2 = \emptyset$. Jelikož často pracujeme s metrickými prostory, připomeňme, že každý prostor s topologií indukovanou metrikou je Hausdorffův.

kde jedna funkce ϕ nahradila všechny ϕ_q a ψ_{pq} jsou nahrazeny výrazem $\lambda^{pq}\psi_q$, přičemž λ je reálná konstanta a ψ_q jsou monotónně rostoucí funkce. Navíc, ψ_q nezávisí na funkci f, kterou vyjadřujeme, ale jsou pro danou dimenzi stejné.

Využití Kolmogorovovy věty v neuronových sítích navrhl v roce 1987 Hecht-Nielsen [104], který si všiml, že vzorec (13.2) připomíná zápis funkce realizované sítí se třemi vrstvami. Vstupní vrstvu takové sítě tvoří n vstupních jednotek, ve skryté vrstvě je $2n+1$ jednotek a jedna jednotka je výstupní. Mezi vstupní a skrytou vrstvou jsou přechodové funkce počítající hodnoty $\lambda^{pq}\psi_q$, přechodovou funkcí mezi skrytou a výstupní vrstvou je ϕ. Funkce ϕ je také jediná, která závisí na aproximované funkci, ostatní parametry jsou pro danou dimenzi pevné.

Tento návrh se stal předmětem kritiky Girosiho a Poggia, kteří ve svém článku [75] nazvaném výmluvně *„Kolmogorov theorem is irrelevant"* předkládají následující výhrady k této aplikaci Kolmogorovovy věty.

- Funkce ψ_q nejsou hladké, mají dokonce fraktální grafy, nehodí se tedy jako přechodové funkce jednotek v neuronové síti.
- Funkce ϕ se konstruuje v závislosti na hodnotách funkce f a není tedy reprezentovatelná v parametrické formě, čili se také nedá použít jako standardní přechodová funkce neuronu.

Kůrková v [163] ukázala, že Kolmogorovovu větu lze využít i pro běžné perceptronové sítě. Podstata tohoto přístupu je, že nebudeme trvat na přesném vyjádření funkce, ale spokojíme se (jako obvykle) s její libovolně přesnou aproximací. V tom případě dostaneme síť se dvěma skrytými vrstvami perceptronů, kde jednotky v první, resp. druhé skryté vrstvě budou aproximovat funkce ψ, resp. ϕ.

Tato myšlenka se skrývá za dvěma tvrzeními z následujícího odstavce. První věta je existenční a ukáže nám, že tento postup je možný. Výsledek prezentovaný ve druhé větě lze odvodit postupem podobným důkazu klasické Kolmogorovovy věty. Podrobná analýza pak poskytne i horní odhad počtu jednotek ve skrytých vrstvách sítě.

13.2.2 Kolmogorovova věta a aproximace pomocí NS

V tomto odstavci se tedy budeme zabývat sítěmi s perceptronovými jednotkami ve skrytých vrstvách a jednou lineární výstupní jednotkou. Vstupy sítě budeme uvažovat z jednotkové krychle $\mathcal{I}^n$, výstupem je libovolné reálné číslo. Uvažujme perceptrony se sigmoidální aktivační funkcí $\sigma : \mathbb{R} \to \mathcal{I}$ pro kterou platí:

$$\lim_{z \to -\infty} \sigma(z) = 0$$

13.2 Kolmogorovova věta

V následujícím oddílu této kapitoly si ukážeme, že perceptronové sítě se dvěma skrytými vrstvami mají univerzální aproximační schopnost. V další podkapitole pak vyslovíme silnější tvrzení o perceptronových sítích, avšak důkazový aparát, který prezentujeme v této části, je velmi zajímavý, neboť používá hlubokou a možná i překvapivou Kolmogorovovu větu o reprezentaci vícerozměrných spojitých funkcí. Nejprve si osvětlíme pozadí vzniku celého problému a jeho řešení a poté si ukážeme, jak lze tuto větu aplikovat v oblasti neuronových sítí.

13.2.1 Hilbertův problém a Kolmogorovo řešení

V roce 1900, přednesl Hilbert na druhém mezinárodním kongresu matematiků v Paříži slavnou přednášku, v níž formuloval 23 problémů, které podle něj budou nejpodstatnější pro vývoj matematiky ve dvacátém století. Jako problém s pořadovým číslem třináct vyslovil Hilbert domněnku, že kořeny polynomické rovnice

$$x^7 + ax^3 + bx^2 + cx + 1 = 0$$

jako funkce tří reálných koeficientů a, b, c není možné vyjádřit jen pomocí součtu a složení funkcí nejvýše dvou proměnných. Za touto konkrétně formulovanou hypotézou se skrýval obecnější problém řešení polynomických rovnic vyšších stupňů.

Problém dlouho vzdoroval snažení matematiků o jeho důkaz či vyvrácení, až Arnold v roce 1957 [24] Hilbertovu hypotézu vyvrátil. Ve stejném roce pak Kolmogorov přišel s mnohem silnějším výsledkem [160], který říkal, že libovolnou spojitou funkci n proměnných lze vyjádřit jen pomocí součtu a složení spojitých funkcí jedné proměnné.

Věta 13.3 *Buďte $n \geq 2$ přirozené číslo a $f(x_1, \ldots, x_n)$ spojitá funkce $f : \mathcal{I}^n \to \mathbb{R}$. Pak existují spojité funkce jedné proměnné ϕ_q a ψ_{pq} $(p = 1, \ldots, n;$ $q = 1, \ldots, 2n+1)$ tak, že platí:*

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi_q \left(\sum_{p=1}^{n} \psi_{pq}(x_p) \right) . \tag{13.1}$$

V současné době existuje ještě obecnější modifikace této věty, na níž mají zásluhu Sprecher a Lorentz [261, 177]. V této variantě se funkční předpis (13.1) změnil na:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi \left(\sum_{p=1}^{n} \lambda^{pq} \psi_q(x_p) \right) , \tag{13.2}$$

a

$$\lim_{z\to\infty}\sigma(z)=1\,.$$

Definice 13.4 Schodištěm typu σ *nazveme množinu všech funkcí* $f:\mathbb{R}\to\mathbb{R}$ *tvaru:*

$$f(x)=\sum_{i=1}^{k}a_i\sigma(b_ix+c_i)\,, \tag{13.3}$$

kde $a_i,b_i,c_i\in\mathbb{R}$. *Schodiště typu* σ *budeme značit symbolem* $\mathcal{S}(\sigma)$.

V následujících tvrzeních budeme ještě potřebovat definice těchto pojmů:

Definice 13.5 *Funkce* $\omega_f:(0,\infty)\to\mathbb{R}$ *se nazývá* modul spojitosti *funkce* $f:\mathcal{I}^n\to\mathbb{R}$, *platí-li*

$$\omega_f(\delta)=\sup\{|f(x_1,\dots,x_n)-f(y_1,\dots,y_n)|;\ (x_1,\dots,x_n),(y_1,\dots,y_n)\in\mathcal{I}^n:$$
$$|x_i-y_i|<\delta\ \ \forall i=1,\dots,n\}\,.$$

Vyslovme nyní naši první aproximační větu, která vychází z Kolmogorovovy věty a říká, že funkce ϕ a ψ mohou být schodiště.

Věta 13.6 *Buďte* $n\in\mathbb{N}$, $n\geq 2$, $\sigma:\mathbb{R}\to\mathcal{I}$ *sigmoidální funkce,* $f\in C(\mathcal{I}^n)$ *a* ε *kladné reálné číslo. Pak existuje* $k\in\mathbb{N}$ *a funkce* $\phi_i,\psi_{pi}\in\mathcal{S}(\sigma)$ *takové, že*

$$\left|f(x_1,\dots,x_n)-\sum_{i=1}^{k}\phi_i\left(\sum_{p=1}^{n}\psi_{pi}x_p\right)\right|<\varepsilon\ \ \forall(x_1,\dots,x_n)\in\mathcal{I}^n\,.$$

K důkazu věty využijeme následující lemma, které uvedeme bez důkazu:

Lemma 13.7 *Nechť* $\sigma:\mathbb{R}\to\mathcal{I}$ *je sigmoida a* $\langle a,b\rangle\subset\mathbb{R}$ *uzavřený interval. Pak množina všech funkcí* $f:\langle a,b\rangle\to\mathbb{R}$ *tvaru*

$$f(x)=\sum_{i=1}^{k}w_i\sigma(v_ix+u_i)\,,$$

kde $w_i,v_i,u_i\in\mathbb{R}$, *je hustá v* $C(\langle a,b\rangle)$.

Důkaz věty 13.6:

Důkaz je jednoduchý. Uvažujme reprezentaci funkce f dle věty 13.3 a za pomoci lemmatu 13.7 ukážeme, že původní funkce ϕ a ψ můžeme nahradit schodišti g a h. Čili, dle 13.3 máme:

$$f(x_1,\dots,x_n)=\sum_{q=1}^{2n+1}\phi_q\left(\sum_{p=1}^{n}\psi_{pq}(x_p)\right)\,.$$

Vezměme nyní $\langle a,b\rangle\subset\mathbb{R}$ takový, aby pro všechna $p=1,\dots,n$ a $q=1,\dots,2n+1$ bylo $\psi_{pq}(\mathcal{I}^n)\subseteq\langle a,b\rangle$.

Podle lemmatu 13.7 pro každé $q=1,\dots,2n+1$ existují funkce $g_q\in\mathcal{S}(\sigma)$ takové, že

$$|g_q(x)-\phi_q(x)|<\frac{\varepsilon}{2n(2n+1)}\,,$$

pro všechna $x,y\in\langle a,b\rangle$.

Jelikož g_q jsou stejnoměrně spojité, existuje $\delta\in\mathbb{R}$ taková, že

$$|g_q(x)-g_q(y)|<\frac{\varepsilon}{2n(2n+1)}\,,$$

pro všechna $x,y\in\langle a,b\rangle$, pro něž je navíc $|x-y|<\delta$.

Pro každé $p=1,\dots,n$ a $q=1,\dots,2n+1$ existují funkce $h_{pq}\in\mathcal{S}(\sigma)$ takové, že platí:

$$|h_{pq}(x)-\psi_{pq}(x)|<\delta\,,\ \ \forall x\in\mathcal{I}\,.$$

Celkem tedy pro každé $(x_1,\dots,x_n)\in\mathcal{I}^n$ platí:

$$\left|\sum_{q=1}^{2n+1}g_q\left(\sum_{p=1}^{n}h_{pq}(x_p)\right)-f(x_1,\dots,x_n)\right|<\varepsilon\,.$$

Tím je věta dokázána. □

Nyní uvedeme hlavní větu této podkapitoly, která hovoří o aproximaci spojitých funkcí perceptronovou sítí se dvěma skrytými vrstvami. Kromě existenčního tvrzení je výsledkem i horní odhad potřebného počtu neuronů.

Věta 13.8 *Buďiž* $n\in\mathbb{N}$, $n\leq 2$, $\sigma:\mathbb{R}\to\mathcal{I}$ *sigmoidální funkce,* $f\in C(\mathcal{I}^n)$ *a* ε *kladné reálné číslo. Pak pro každé* $m\in\mathbb{N}$ *takové, že* $m\geq 2n+1$ *a* $\frac{n}{m-n}+v<\frac{\varepsilon}{||f||}$ *a* $\omega_f(1/m)<v\frac{m-n}{2m-3n}$ *pro nějaké kladné* $v\in\mathbb{R}$, *může být* f *aproximována s přesností* ε *perceptronovou sítí se dvěma skrytými vrstvami, přičemž v první skryté vrstvě má síť* $nm(m+1)$ *jednotek a ve druhé* $m^2(m+1)^n$ *jednotek s aktivační funkcí* σ. *Navíc, všechny váhy a prahy sítě, s výjimkou vah spojujících druhou skrytou vrstvu a výstupní jednotku, jsou pevné pro všechny funkce* g *pro něž platí:* $||g||\leq||f||$ *a zároveň* $\omega_g\leq\omega_f$.

Než stručně ukážeme hlavní myšlenky důkazu této věty, uvedeme lemma, které se v něm využívá. Lemma 13.9 říká, že každá konečná množina schodů může být aproximována funkcí náležející do množiny $\mathcal{S}(\sigma)$. Množinou schodů myslíme 2k reálných čísel $x_1<y_1<x_2<y_2<\ldots<x_k<y_k$ a libovolné zobrazení $g:\{1,\dots,k\}\to\mathbb{R}$.

13.3 Perceptronové sítě s jednou skrytou vrstvou

13.3.1 Motivace a definice

Asi nejvíce výsledků o univerzální aproximaci neuronových sítí se týká perceptronových sítí s jednou skrytou vrstvou. Příčinou je jistě fakt, že tyto sítě jsou v praxi nejčastěji používaným modelem a navíc jedna skrytá vrstva je v jistém smyslu minimální architektura zachovávající rysy vícevrstvých perceptronů. Motivací pro výzkum aproximačních vlastností byly hlavně dobré praktické zkušenosti s tím, jak tento model dokáže aproximovat nejrůznější funkce. Jak píše Hornik, Stinchcombe a White ve svém článku [128]: „Jsou úspěchy, které potkáváme v praxi, odrazem nějaké hluboké a fundamentální aproximační schopnosti, nebo jde jen o šťastné náhody způsobené selekcí kladných výsledků a příznivou volbou problémů?“

Na přelomu osmdesátých a devadesátých let se tak objevila řada prací, ve kterých se dokazovala univerzální aproximace této architektury s různými omezeními kladenými na aktivační funkci. Gallant a White [71] například dokázali, že síť s aktivační funkcí $\cos(x)$ je univerzálním aproximátorem, protože má totožné aproximační vlastnosti jako reprezentace funkce Fourierovou řadou. Hornik, Stinchcombe a White [128] tento výsledek rozšířili na třídu tzv. *squashing* funkcí, která obsahuje např. logistickou sigmoidu. Mezi další práce s tímto typem výsledků patří například [170, 127] a další.

V dalším odstavci si uvedeme dva důležité výsledky publikované Hornikem v roce 1991 [127] a podrobněji se zastavíme u dosud nejobecnějšího výsledku autorů Leshna, Lina, Pinkuse a Shockena z roku 1993 [170]. Předtím opět uvedeme několik potřebných definic.

Definice 13.10 *Nechť funkce f je definována skoro všude vzhledem k lebesgueovské míře μ na měřitelné množině Ω v $\mathbb{R}^n$. Pak f nazveme* omezenou skoro všude, *pokud existuje $c \in \mathbb{R}$ takové, že míra množiny $\{x \in \mathbb{R}^n;\ |f(x)| > c\}$ je rovna nule. Prostor takových funkcí označíme $\mathcal{L}^\infty(\Omega)$. Norma na $\mathcal{L}^\infty(\Omega)$ se definuje jako:*

$$||f||_{\mathcal{L}^\infty(\Omega)} = ess\sup_{x\in\Omega} |f(x)|\,,$$

kde ess sup *je největší dolní odhad všech možných hodnot c.*

Definice 13.11 *Nechť funkce f je definována skoro všude vzhledem k lebesgueovské míře μ na otevřené množině Ω v $\mathbb{R}^n$. Pak f nazveme* lokálně omezenou skoro všude, *když pro každou kompaktní množinu $K \subset \Omega$ je $f \in \mathcal{L}^\infty(K)$. Prostor těchto funkcí označíme $\mathcal{L}^\infty_{\mathrm{loc}}(\Omega)$.*

Lemma 13.9 *Buďte $\sigma : \mathbb{R} \to \mathcal{I}$ sigmoida, ε kladné reálné číslo, $k \in \mathbb{N}$, $x_1 < y_1 < x_2 < y_2 < \ldots < x_k < y_k$ reálná čísla a $g : \{1, \ldots, k\} \to \mathbb{R}$ zobrazení. Pak existuje funkce $\phi \in \mathcal{S}(\sigma)$ taková, že*

$$|\phi(x) - g(j)| < \varepsilon;\ \forall x \in \langle x_j, y_j\rangle;\ \forall j = 1, \ldots, k \tag{13.4}$$

a

$$||\phi|| \le \max\{|g(j)|; j = 1, \ldots, k\} + \varepsilon\,.$$

Důkaz lemmatu: Každou ze sigmoid můžeme pomocí parametrů b_i a c_i posunout a „zeštíhlet“ tak, že je pro hodnoty argumentů $z < y_{i-1}$ dostatečně malá (blízká nule) a naopak, pro $z > x_i$ blízko 1. (Dostatečně malá v našem případě znamená menší než $\frac{\varepsilon}{4Mk}$, kde $M = \max\{|g(j)|;\ j = 1, \ldots, k\}$.) Nakonec nastavíme a_i na $g(i) - g(i-1)$. Lze lehce ověřit, že podmínka (13.4) je tím splněna. □

Důkaz věty 13.8:

Jelikož kompletní důkaz této věty je velmi technicky náročný (ač nepoužívá aparát z pokročilých partií analýzy), uvedeme jen základní myšlenky. Zájemce o podrobnosti odkazujeme na článek [163].

Princip důkazu spočívá v tom, že nejprve rozdělíme krychli $\mathcal{I}^n$ na malé krychle pomocí schodišťových funkcí $\psi_p \in \mathcal{S}(\sigma)$; $p = 1, \ldots, n$. Definujeme zobrazení $\Psi : \mathcal{I}^n \to \mathbb{R}$ takto: $\Psi(x_1, \ldots, x_n) = \sum_{p=1}^n \psi_p(x_p)$. Zobrazení Ψ tedy definuje strukturu malých krychlí, u nichž hrany korespondují se schody u ψ_p a mezery mezi nimi odpovídají šikmým částem funkcí ψ_p. Funkce Ψ je navíc zvolena tak, aby se malé krychle zobrazily na uzavřené a oddělené intervaly na reálné přímce. Nyní těmto intervalům přiřadíme hodnoty, kterých aproximovaná funkce f nabývá v odpovídajících malých krychlích. Tím dostaneme konečnou množinu schodů, kterou podle lemmatu 13.9 aproximujeme funkcí $\phi \in \mathcal{S}(\sigma)$.

Funkce $\phi \circ \Psi$ nám tak aproximuje f na podmnožině $\mathcal{I}^n$, která je tvořena sjednocením malých krychlí. Čím jemnější jsou schody funkcí ψ_p, tím přesnější aproximace dosáhneme. Navíc, naše funkce je vyjádřitelná parametricky, neboť se skládá jen ze schodišť.

Posledním problémem je, že funkce f není aproximována na mezerách mezi schody ψ_p. To lze napravit tím, že celý popsaný postup uděláme vícekrát, pokaždé s vhodně posunutými schody definujícími rozdělení krychle na menší krychličky. Ukazuje se, že dostatečný počet opakování je $2n+1$ (proto je v první sumě vzorce (13.1) $2n + 1$ členů). Důvodem je kombinatorický princip zásuvek: potřebujeme zaručit, aby každý bod byl ve více rozděleních, v nichž se nachází v nějaké krychličce, než v těch, kde je v mezeře mezi nimi. Jelikož bod v krychli má n souřadnic, z nichž každá může padnout do mezery jiného rozdělení, potřebujeme alespoň $n+(n+1) = 2n+1$ rozdělení.

Pečlivou formalizací těchto idejí lze nakonec dojít k odhadům počtu jednotek prezentovaných ve tvrzení. □

1. Ukažme nejprve snadný směr implikace: *Je-li σ polynom, pak $\mathcal{P}_n(\sigma)$ není husté v $C(\mathbb{R}^n)$* .

   Když je σ polynom stupně k, pak i $\sigma(\mathbf{w}\cdot\mathbf{x}+b)$ je polynom stupně k pro všechna $\mathbf{w}, b$ a $\mathcal{P}_n(\sigma)$ je právě množina všech polynomů stupně nejvýše k. Tedy $\mathcal{P}_n(\sigma)$ není hustá v $C(\mathbb{R}^n)$.

2. Nyní předpokládejme, že σ není polynom a ukažme nejprve následující tvrzení: *Je-li $\mathcal{P}_1(\sigma)$ hustá v $C(\mathbb{R})$, pak je $\mathcal{P}_n(\sigma)$ hustá v $C(\mathbb{R}^n)$.*

   Uvažujeme množinu V funkcí tvaru:

$$V = \{f : \mathbb{R}^n \to \mathbb{R};\ \sum_{i=1}^{k} f_i(\mathbf{a}_i\cdot\mathbf{x}); \mathbf{a}_i \in \mathbb{R}^n, f_i \in C(\mathbb{R}), k \in \mathbb{N}\}\,.$$

   Tato množina je hustá v $C(\mathbb{R}^n)$, což plyne např. z věty 13.43, která je též v [279]. Vezměme nyní libovolnou $g(\mathbf{x}) \in C(\mathbb{R}^n)$, $K \subset \mathbb{R}^n$ kompaktní a $\varepsilon > 0$. Pak existují $\mathbf{a}_i \in \mathbb{R}^n$, $f_i \in C(\mathbb{R})$ takové, že

$$\left| g(\mathbf{x} - \sum_{i=1}^{k} f_i(\mathbf{a}_i\cdot\mathbf{x}) \right| < \frac{\varepsilon}{2}\,,$$

   pro každé $x \in K$.

   Dále existují čísla $\alpha_i, \beta_i \in \mathbb{R}$ $(i = 1,\ldots,k)$ taková, že $\{\mathbf{a}_i\cdot\mathbf{x};\ \mathbf{x} \in K\} \subseteq \langle\alpha_i,\beta_i\rangle$. Jelikož $\mathcal{P}_1(\sigma)$ je hustá v $\langle\alpha_i,\beta_i\rangle$, existují reálná čísla c_{ij}, w_{ij}, b_{ij} $(i = 1,\ldots,k;\ j = 1,\ldots,m_i)$ taková, že

$$\left| f_i(y) - \sum_{j=1}^{m_i} c_{ij}\sigma(w_{ij}y_i + b_{ij}) \right| < \frac{\varepsilon}{2k}\,,$$

   pro všechna $y \in \langle\alpha_i,\beta_i\rangle$.

   Takže

$$\left| g(\mathbf{x} - \sum_{i=1}^{k}\sum_{j=1}^{m_i} c_{ij}\sigma(\mathbf{a}_i\cdot\mathbf{x}) \right| < \varepsilon\,,$$

   pro každé $x \in K$. Čili $\mathcal{P}_n(\sigma)$ je hustá v $C(\mathbb{R}^n)$.

3. Dále ukážeme, že má-li σ všechny derivace, je $\mathcal{P}_1(\sigma)$ hustá v $C(\mathbb{R})$.

   Ukážeme, že pomocí derivování $\sigma(wx+b)$ umíme získat všechny mocniny x v uzávěru $\mathcal{P}_1(\sigma)$, a ten tedy obsahuje všechny polynomy. Z Weierstrassovy věty 13.2 pak ihned plyne, že $\mathcal{P}_1(\sigma)$ je hustá v $C(\mathbb{R})$.

Nyní definujeme přípustnou množinu aktivačních funkcí, které použijeme při vyslovení tvrzení 13.16.

Definice 13.12 *Označme M množinu funkcí z $\mathcal{L}^\infty_{\mathrm{loc}}(\Omega)$, které mají následující vlastnost. Uzávěr množiny bodů nespojitosti libovolné funkce z M má Lebesgueuovu míru nula.*

Definice 13.13 *Označme $\mathcal{P}_n(\sigma)$ množinu funkcí realizovatelných sítí s n vstupy, jednou skrytou vrstvou perceptronů s aktivační funkcí σ a lineární výstupní jednotkou:*

$$\mathcal{P}_n(\sigma) = \{f : \mathbb{R}^n \to \mathbb{R};\ f(\mathbf{x}) = \sum_{i=1}^{k} v_i\sigma(\mathbf{w}_i\cdot\mathbf{x}+b_i);\ \mathbf{w}_i \in \mathbb{R}^n, v_i, b_i \in \mathbb{R}, k \in \mathbb{N}\}\,.$$

13.3.2 Hlavní výsledky

Uveďme nejprve dva výsledky publikované Hornikem, které hovoří o univerzální aproximaci jak v $\mathcal{L}_p$, tak v $C(X)$ pro kompaktní X. Všimněme si, jaká jsou omezení kladená na přechodovou funkci. Musí jít o nekonstantní (poměrně zřejmé), omezenou a spojitou (aproximujeme-li spojité funkce, viz věta 13.15) funkci.

Věta 13.14 *Je-li aktivační funkce omezená a nekonstantní, pak perceptronová síť s jednou skrytou vrstvou může aproximovat každou funkci z $\mathcal{L}_p$ s libovolnou přesností, za předpokladu, že je k dispozici dostatečně mnoho jednotek ve skryté vrstvě.*

Věta 13.15 *Je-li aktivační funkce spojitá, omezená a nekonstantní na kompaktní množině $X \subset \mathbb{R}^n$, pak může perceptronová síť s jednou skrytou vrstvou aproximovat každou funkci z $C(X)$ libovolně přesně, je-li k dispozici dostatečně mnoho jednotek ve skryté vrstvě.*

Otázka, zda spojitost aktivační funkce je nutná, byla nějakou dobu otevřená a vyřešila ji až práce [170], jejíž hlavní výsledek nyní uvedeme. Podmínky kladené na σ v následujícím tvrzení jsou překvapivě slabé — k univerzální aproximaci spojitých funkcí stačí, aby aktivační funkce nebyla polynomem.

Věta 13.16 *Buď $\sigma \in M$. Množina funkcí $\mathcal{P}_n(\sigma)$ je hustá v $C(\mathbb{R}^n)$ právě tehdy, když je σ nepolynomiální.*

Důkaz: Důkaz věty rozdělíme do čtyř částí, z nichž u tří, které nejsou příliš technicky náročné, se zastavíme podrobněji.

13.4 Aproximace RBF sítěmi

13.4.1 Úvod

Podívejme se nyní na aproximační vlastnosti RBF sítí. Zavedeme si dvě množiny funkcí realizovaných RBF sítěmi. První z nich odpovídá sítím, jejichž RBF jednotky mají stejné hodnoty parametrů určující jejich šířku. Druhá třída funkcí odpovídá obecným RBF sítím.

Definice 13.18 *Označme $\mathcal{R}_n^0(K)$ množinu funkcí reprezentovaných sítí s n vstupy, libovolným počtem RBF jednotek, které mají stejnou šířku σ a aktivační funkci K, a jednou lineární výstupní jednotkou. $\mathcal{R}_n^0(K)$ je tedy množina funkcí $f : \mathbb{R}^n \to \mathbb{R}$*

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i K\left(\frac{\mathbf{x} - \mathbf{c}_i}{\sigma}\right) ,$$

kde $w_i, \sigma \in \mathbb{R}$, $\mathbf{c}_i \in \mathbb{R}^n$, $k \in \mathbb{N}$ a $K : \mathbb{R}^n \to \mathbb{R}$.

Definice 13.19 *Označme $\mathcal{R}_n(K)$ množinu funkcí reprezentovaných sítí s n vstupy, libovolným počtem RBF jednotek s aktivační funkcí K a jednou lineární výstupní jednotkou. $\mathcal{R}_n(K)$ je tedy množina funkcí $f : \mathbb{R}^n \to \mathbb{R}$*

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i K\left(\frac{\mathbf{x} - \mathbf{c}_i}{\sigma_i}\right) ,$$

kde $w_i, \sigma_i \in \mathbb{R}$, $\mathbf{c}_i \in \mathbb{R}^n$, $k \in \mathbb{N}$ a $K : \mathbb{R}^n \to \mathbb{R}$.

Všimněme si, že na rozdíl od kapitoly 5 je funkce K definována obecněji — nemusí se jednat o složení nějaké funkce (třeba Gaussovy) s normou. Obvykle se uvažují radiálně symetrické funkce K, my však radiální symetrii použijeme jen při formulaci jednoho výsledku, takže, pokud to explicitně neuvedeme, neklademe na K žádné další požadavky.

Definice 13.20 *Funkci K nazveme* radiálně symetrickou vzhledem k normě $||\cdot||$, *platí-li, že $||x|| = ||y||$ implikuje $K(x) = K(y)$.*

13.4.2 Výsledky

V tomto odstavci uvedeme hlavní výsledky týkající se aproximačních vlastností RBF sítí. Asi nás nepřekvapí, že RBF sítě jsou univerzálními aproximátory vzhledem k $C(\mathbb{R}^n)$ i $\mathcal{L}_p(\mathbb{R}^n)$. Uvidíme, že podmínky kladené na aktivační funkci se v jednotlivých případech mírně liší. Většina zde uvedených vět pochází od autorů Parka a Sandberga [216, 217]. Důkazy využívají vesměs pokročilejší partie matematické analýzy a nebudeme je uvádět.

Má-li σ všechny derivace, pak pro každé $w, b, h \in \mathbb{R}$, $h \neq 0$ platí:

$$\frac{\sigma((w+h)+b) - \sigma(wx+b)}{h} \in \mathcal{P}_1(\sigma) .$$

Tedy $\frac{\partial}{\partial w}\sigma(wx+b) \in \overline{\mathcal{P}_1(\sigma)}$. Stejně dokážeme, že $\frac{\partial^k}{\partial w^k}\sigma(wx+b) \in \overline{\mathcal{P}_1(\sigma)}$ pro každé $k \in \mathbb{N}$ a pro všechna $w, b \in \mathbb{R}$. Ovšem

$$\frac{\partial^k}{\partial w^k}\sigma(wx+b) = x^k \sigma^{(k)}(wx+b) ,$$

(kde $\sigma^{(k)}$ značí k-tou derivaci σ).

Jelikož σ není polynom, existuje $d_k \in \mathbb{R}$ takové, že $\sigma^{(k)}(d_k) \neq 0$. Takže

$$x^k \sigma^{(k)}(d_k) = \frac{\partial^k}{\partial w^k}\sigma(wx+b) \text{ pro } w = 0, b = d_k .$$

Tedy $x^k \sigma^{(k)}(d_k) \in \overline{\mathcal{P}_1(\sigma)}, \overline{\mathcal{P}_1(\sigma)}$ obsahuje všechny polynomy a můžeme použít Weierstrassovu větu 13.2.

4. V poslední části důkazu, kterou již nebudeme probírat, se použitím konvolucí a předchozího kroku dokáže, že $\mathcal{P}_1(\sigma)$ je hustá v $C(\mathbb{R})$ a tedy můžeme použít tvrzení kroku 2 pro dokončení důkazu i druhé implikace tvrzení □

13.3.3 Poznámky

Následující tvrzení je důsledkem předchozí věty vyslovené pro prostor $\mathcal{L}_p$.

Důsledek 13.17 *Buď $\sigma \in M$. Množina funkcí $\mathcal{P}_n(\sigma)$ je hustá v $\mathcal{L}_p$ pro $1 \leq p < \infty$ právě tehdy, když je σ nepolynomiální.*

Hlavní věta předchozího odstavce je svým tvrzením velmi silná. Ukazuje, že zajímá-li nás schopnost univerzální aproximace, postačující jsou perceptronové sítě jen s jednou skrytou vrstvou a jakoukoliv nepolynomiální aktivační funkcí. Podobný výsledek s jiným postupem důkazu pochází i od Mhaskara a Micchelliho [193].

Ilustrujme ještě jednu zajímavou věc, a to je důležitost prahu v předchozích úvahách. Představme si aktivační funkci $\sigma(x) = \sin(x)$, která bohatě splňuje všechny předpoklady věty 13.16 (mimochodem, i věty 13.15). Jak vypadá množina funkcí tvaru $\{sin(w \cdot x);\ w \in \mathbb{R}\}$? Předně, obsahuje jen liché funkce, takže nelze aproximovat nějakou sudou funkci (např. $\cos(x)$) na intervalu $\langle -1, 1\rangle$. Čili tato množina funkcí není hustá na $C(\langle -1, 1\rangle)$. To lze snadno napravit právě přidáním posunu, neboť víme, že $\sin(x + \frac{\pi}{2}) = \cos(x)$. U jiných funkcí naopak nehraje posunutí roli, jako je tomu např. u e^x.

Výjimkou bude důkaz věty 13.23, který je zajímavý svou technikou. Idea použít pro aproximační důkazy v neuronových sítích dvou elegantních a silných nástrojů — Hahn-Banachovy a Rieszovy věty — byla poprvé použita v Cybenkově článku [52] z roku 1989.

První z následujících vět hovoří o hustotě množiny $\mathcal{R}_n^0(K)$ v prostoru $\mathcal{L}_1(\mathbb{R}^n)$. Má tvar ekvivalence, udává tedy nutnou a postačující podmínku pro aproximaci pomocí RBF sítí s uniformními šířkami.

Věta 13.21 *Nechť $K : \mathbb{R}^n \to \mathbb{R}$ je integrovatelná, pak $\mathcal{R}_n^0(K)$ je hustá v $\mathcal{L}_1(\mathbb{R}^n)$ právě tehdy, když*

$$\int_{\mathbb{R}^n} K(x)dx \neq 0\,.$$

Stejné podmínky platí i pro aproximaci funkcí v $\mathcal{L}_1(\mathbb{R}^n)$ pomocí obecné RBF sítě:

Věta 13.22 *Nechť $K : \mathbb{R}^n \to \mathbb{R}$ je integrovatelná, pak $\mathcal{R}_n(K)$ je hustá v $\mathcal{L}_1(\mathbb{R}^n)$ právě tehdy, když*

$$\int_{\mathbb{R}^n} K(x)dx \neq 0\,. \tag{13.5}$$

Poznámka: Je zajímavé, že pro aproximaci funkcí z $\mathcal{L}_2(\mathbb{R}^n)$ pomocí funkcí z množiny $\mathcal{R}_n(K)$ se podařilo nahradit podmínku (13.5) slabší podmínkou požadující nenulovost Fourierovy transformace funkce K skoro všude na určitých podmnožinách $\mathbb{R}^n$.

Obecné tvrzení o $\mathcal{L}_p(\mathbb{R}^n)$ aproximaci funkcemi z $\mathcal{R}_n(K)$ má obdobný tvar, ale je již vysloveno pouze jako implikace.

Věta 13.23 *Nechť $p \geq 1$, $K : \mathbb{R}^n \to \mathbb{R}$ je integrovatelná funkce taková, že platí:*

$$\int_{\mathbb{R}^n} K(x)dx \neq 0$$

a zároveň

$$\int_{\mathbb{R}^n} |K(x)dx|^p < \infty\,.$$

Pak je $\mathcal{R}_n(K)$ hustá v $\mathcal{L}_p(\mathbb{R}^n)$.

Důkaz: Důkaz se provádí sporem. Předpokládejme, že $\mathcal{R}_n(K)$ není hustá v $\mathcal{L}_p(\mathbb{R}^n)$. Potom podle Hahn-Banachovy věty [242] existuje omezený funkcionál $\Lambda : \mathcal{L}_p(\mathbb{R}^n) \to \mathbb{R}$ takový, že

$$\Lambda\left(\overline{\mathcal{R}_n(K)}\right) = 0\,, \tag{13.6}$$

ale

$$\Lambda\left(\overline{\mathcal{L}_p(\mathbb{R}^n)}\right) \neq 0\,. \tag{13.7}$$

Nyní použijeme Rieszovu větu o reprezentaci [242], podle které můžeme Λ vyjádřit jako:

$$\Lambda(f) = \int_{\mathbb{R}^n} f(x)g_\Lambda(x)dx\,,$$

kde $g_\Lambda(x)$ je funkce v $\mathcal{L}_q(\mathbb{R}^n)$ a pro q platí $\frac{1}{p} + \frac{1}{q} = 1$. Konkrétně, z (13.6) plyne, že

$$\int_{\mathbb{R}^n} \frac{1}{\sigma^n} K\left(\frac{\mathbf{x}-\mathbf{c}}{\sigma}\right) g_\Lambda(x)dx = 0\,, \tag{13.8}$$

pro všechna $c \in \mathbb{R}^n$ a $\sigma > 0$.

Pomocí technického lemmatu o konvolucích se dá dokázat, že ze vztahu (13.8) plyne, že g_Λ je nula skoro všude. To by ale znamenalo, že Λ je nulový funkcionál, což je spor s (13.7). □

Následující dvě věty charakterizují podmínky nutné k univerzální aproximaci spojitých funkcí pomocí $\mathcal{R}_n(K)$. Obě jsou důsledky silnějšího tvrzení z [217].

Věta 13.24 *Nechť $K : \mathbb{R}^n \to \mathbb{R}$ je integrovatelná, spojitá a splňuje*

$$\int_{\mathbb{R}^n} K(x)dx \neq 0\,. \tag{13.9}$$

Pak $\mathcal{R}_n(K)$ je hustá v $C(X)$ pro kompaktní $X \subset \mathbb{R}^n$.

Ve druhé větě je podmínka (13.9) nahrazena radiální symetrií vzhledem k euklidovské normě.

Věta 13.25 *Nechť $K : \mathbb{R}^n \to \mathbb{R}$ je integrovatelná, spojitá a radiálně symetrická vzhledem k euklidovské normě. Pak $\mathcal{R}_n(K)$ je hustá v $C(X)$ pro kompaktní $X \subset \mathbb{R}^n$.*

13.5 Sítě se semi-lokálními jednotkami

13.5.1 Úvod

V této podkapitole se budeme věnovat aproximačním vlastnostem sítí se semi-lokálními jednotkami, které jsme popsali v podkapitole 5.2. Řekli jsme si, že Hartman a Keeler se ve svém návrhu snažili najít jakýsi kompromis mezi perceptrony a lokálními RBF jednotkami a navrhli jednotku, která počítá součet gaussovských funkcí v jednotlivých dimenzích. Experimentální

výsledky ukázali, že semi-lokální gaussovské sítě se učí rychleji než perceptrony a — na rozdíl od RBF sítí — nemají potíže s irelevantními vstupy.

Otázku univerzální aproximace semi-lokálních gaussovských sítí vyřešila Kůrková v roce 1993 [164], a to negativně. V následujícím odstavci si vyložíme její výsledek, který ukazuje, že tyto sítě nejsou univerzálními aproximátory. Univerzální aproximace dosáhneme, když povolíme buď obecnější jednotky namísto semi-lokálních, nebo složitější architekturu. Tato tvrzení uvedeme a dokážeme v dalším odstavci.

Uveďme nyní několik potřebných definic. Kvůli jednoduššímu zápisu budeme funkce počítané neuronovou sítí vyjadřovat pomocí operace složení zobrazení $\circ$.

Gaussovská semi-lokální jednotka počítá funkci $f : \mathbb{R}^n \to \mathbb{R}$ tvaru:

$$f(x_1, \ldots, x_n) = \sum_{r=1}^{n} w_r e^{-\left(\frac{x_r - c_r}{\sigma_r}\right)^2} , \tag{13.10}$$

kde $w_r, c_r, \sigma_r \in \mathbb{R}$, $\sigma_r \neq 0$. Označme si nyní Gaussovu funkci jako $\gamma(z) = exp(-z^2)$, dále si zaveďme projekci do r-té souřadnice $\pi_r : \mathbb{R}^n \to \mathbb{R}$ $(r = 1, \ldots, n)$ definovanou $\pi_r(x_1, \ldots, x_n) = x_r$. Nakonec definujme afinní zobrazení $\alpha_r(z) = \frac{z}{\sigma_r} - \frac{c_r}{\sigma_r} = a_r z + b_r$, kde $a_r = \frac{1}{\sigma_r} \neq 0$ a $b_r = \frac{c_r}{\sigma_r}$. Nyní můžeme (13.10) přepsat takto:

$$f = \sum_{r=1}^{n} w_r \gamma \circ \alpha_r \circ \pi_r . \tag{13.11}$$

Definujme nyní tři třídy funkcí. První z nich jsou funkce realizovatelné semi-lokální gaussovskou sítí, druhá obsahuje funkce realizované sítí s obecnějším typem jednotek (význam pochopíme později v odstavci 13.5.3). Třetí třída funkcí jsou funkce počítané sítí, jež sestává ze vstupní vrstvy, vrstvy semi-lokálních jednotek, lineární vrstvy a opět vrstvy semi-lokálních jednotek. I o ní se v odstavci 13.5.3 dozvíme více. Připomeňme si ještě, že opět budeme uvažovat sítě s jednou výstupní jednotkou a n vstupy z jednotkové krychle $\mathcal{I}^n$.

Definice 13.26 *Označme $\mathcal{G}_n$ množinu* funkcí $f : \mathcal{I}^n \to \mathbb{R}$ realizovatelných semi-lokální gaussovskou sítí *s n vstupy, libovolným počtem semi-lokálních jednotek a jednou lineární výstupní jednotkou. Funkce $f \in \mathcal{G}_n$ je tedy tvaru:*

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{h} v_i \left(\sum_{r=1}^{n} w_{ir} \gamma \circ \alpha_{ir} \circ \pi_r \right) , \tag{13.12}$$

kde $v_i, w_{ir} \in \mathbb{R}$, $h \in \mathbb{N}$.

Definice 13.27 *Označme $\mathcal{G}_n^1$ množinu funkcí $f : \mathcal{I}^n \to \mathbb{R}$ tvaru:*

$$f = \sum_{j=1}^{k} u_j \gamma \circ \beta_j , \tag{13.13}$$

kde $k \in \mathbb{N}$, $u_j \in \mathbb{R}$, $\beta_j : \mathcal{I}^n \to \mathbb{R}$ jsou obecné afinní transformace, tj. $\beta_j(x_1, \ldots, x_n) = \sum_{i=1}^{n} d_{ji} x_i + e_j$, kde $d_{ji}, e_j \in \mathbb{R}$.

Definice 13.28 *Označme $\mathcal{G}_n^2$ množinu funkcí $g : \mathcal{I}^n \to \mathbb{R}$ tvaru:*

$$g = \sum_{j=1}^{q} w_j \gamma \circ \alpha_j \circ \pi_j \circ f_j , \tag{13.14}$$

kde $w_j \in \mathbb{R}$, α_j jsou afinní transformace, π_j projekce, γ Gaussova funkce a f_j jsou funkce z (13.12). Jedná se o funkce realizované sítí s n vstupy, h semi-lokálními jednotkami v první skryté vrstvě, q lineárními jednotkami ve druhé vrstvě a jednou semi-lokální jednotkou ve výstupní vrstvě.

13.5.2 Negativní výsledek

Věta 13.29 *Pro každé přirozené $n \geq 2$ platí, že $\mathcal{G}_n$ není hustá ani v $C(\mathcal{I}^n)$ ani v $\mathcal{L}_p(\mathcal{I}^n)$ pro $p \geq 1$.*

Důkaz: Naším cílem bude vzít libovolnou funkci z uzávěru $\mathcal{G}_n$ a ukázat, že není dostatečně obecná, aby reprezentovala všechny funkce v $C(\mathcal{I}^n)$ ani v $\mathcal{L}_p(\mathcal{I}^n)$.

Nechť tedy $g \in \overline{\mathcal{G}_n}$. Funkce g je pak limitou funkcí z $\mathcal{G}_n$:

$$g = \lim_{k \to \infty} f_k , \tag{13.15}$$

kde $f_k \in \mathcal{G}_n$ jsou tvaru:

$$f_k(x_1, \ldots, x_n) = \sum_{i=1}^{h_k} v_{ki} \left(\sum_{r=1}^{n} w_{kir} \gamma \circ \alpha_{kir}(x_r) \right) ,$$

kde $v_{ki}, w_{kir} \in \mathbb{R}$ a α_{kir} jsou afinní transformace na $\mathbb{R}$.

Označíme-li si

$$f_{kr} = \sum_{i=1}^{h_k} v_{ki} w_{kir} \gamma \circ \alpha_{kir} ,$$

můžeme f_k přepsat jako:

$$f_k = \sum_{r=1}^{n} f_{kr}(x_r) .$$

Jaký je geometrický význam tohoto zobecnění? Graf hodnot semi-lokální jednotky sestává z gaussovského vrcholu kolem centroidu, od nějž se táhnou zvýšené brázdy rovnoběžně s osami souřadnými. Právě tato rovnoběžnost je výsledkem složení afinní transformace s projekcí. Uvažujeme-li obecné afinní transformace, dostaneme brázdy, které mohou jít všemi směry. Tyto úvahy sumarizuje následující důsledek.

Důsledek 13.30 *Pro každé přirozené $n \geq 2$ platí, že $\mathcal{G}_n^1$ je hustá v $C(\mathcal{I}^n)$ i v $\mathcal{L}_p(\mathcal{I}^n)$ pro $p \geq 1$.*

Další možností, jak dosáhnout univerzální aproximace sítí se semi-lokálními jednotkami, je zvětšit počet vrstev sítě tak, abychom dostali funkce odpovídající třídě $\mathcal{G}_n^2$ funkcí definovaných dle (13.14). Následující dvě věty se týkají právě takových funkcí. K důkazu využijeme výsledků z předchozích podkapitol, konkrétně Kolmogorovovu větu a větu o univerzální aproximaci RBF sítěmi.

Věta 13.31 *Pro každé přirozené $n \geq 2$ platí, že $\mathcal{G}_n^2$ je hustá v $C(\mathcal{I}^n)$.*

Důkaz: Princip důkazu je následující: libovolnou spojitou funkci vyjádříme dle Kolmogorovovy věty pomocí funkcí jedné proměnné, které následně všechny aproximujeme gaussovskými semi-lokálními funkcemi jedné proměnné. Pro jednodimenzionální vstupy jsou semi-lokální jednotky totožné s RBF jednotkami (s gaussovskou aktivační funkcí), takže můžeme použít věty o univerzální aproximaci RBF sítí. Proveďme nyní celý postup podrobněji.

Buď $f \in C(\mathcal{I}^n)$ libovolná funkce, kterou můžeme reprezentovat jako (viz věta 13.3):

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi \left(\sum_{p=1}^{n} \lambda^{pq} \psi_q(x_p) \right) , \tag{13.18}$$

kde ϕ, ψ_q jsou spojité funkce jedné proměnné.

Vezměme interval $\langle a, b \rangle \subset \mathbb{R}$ dostatečně velký, aby platilo, že $\psi_q(\mathcal{I}) \subseteq \mathbb{R}$, $(q = 1, \ldots, 2n+1)$. Na tomto intervalu budeme nyní aproximovat funkci ψ funkcí realizovanou sítí se semi-lokálními jednotkami s jedním vstupem, tedy funkcemi $g : \langle a, b \rangle \to \mathbb{R}$ tvaru:

$$g = \sum_{i=1}^{h} w_i \gamma \circ \alpha_i , \tag{13.19}$$

kde $k \in \mathbb{N}$, $w_i \in \mathbb{R}$ a $\alpha_i(z) = a_i z + b_i$ jsou afinní transformace $\alpha_i : \langle a, b \rangle \to \mathbb{R}$.

Z věty 13.25 plyne, že množina všech funkcí g je hustá v $C(\langle a, b \rangle)$. Můžeme tedy vybrat takovou g, že pro každé $x \in \langle a, b \rangle$ platí:

$$|g(x) - \phi(x)| < \frac{\varepsilon}{2n(2n+1)} .$$

Definujme pro každé $r = 1, \ldots, n$ zobrazení $\nu_r : \mathbb{R} \to \mathbb{R}^n$ takové, že složení zobrazení $\pi_r \circ \nu_r$ dá identitu na $\mathbb{R}$. Jelikož platí (13.15), tak pro každé $x_r \in \mathbb{R}$ máme:

$$g \circ \nu_r(x_r) = \lim_{k \to \infty} f_k \circ \nu_r(x_r) = \lim_{k \to \infty} f_{kr}(x_r) + \delta_r ,$$

kde

$$\delta_r = \sum_{\substack{q=1 \\ q \neq r}}^{n} f_{kq} \circ \pi_q \circ \nu_r(x_r) .$$

Takže pro každé $r = 1, \ldots, n$ existuje funkce $g_r = \lim_{k \to \infty} f_{kr}$. Tedy

$$g(x_1, \ldots, x_n) = \sum_{r=1}^{n} g_r(x_r) . \tag{13.16}$$

Tím jsme dospěli ke konci, neboť jak v $C(\mathcal{I}^n)$, tak v $\mathcal{L}_p(\mathcal{I}^n)$ existují funkce, které nelze vyjádřit ve tvaru (13.16). Příkladem je funkce:

$$g(x_1, \ldots, x_n) = x_1 x_2 .$$

Dokázali jsme tedy, že $\mathcal{G}_n$ není hustá ani v $C(\mathcal{I}^n)$ ani v $\mathcal{L}_p(\mathcal{I}^n)$. □

13.5.3 Pozitivní výsledek

Síť složená z gaussovských semi-lokálních jednotek nemá tedy vlastnost univerzální aproximace. Podívejme se podrobněji na funkce z $\mathcal{G}_n$ a zamysleme se, proč tomu tak je. Vezměme tedy $f \in \mathcal{G}_n$:

$$f = \sum_{i=1}^{h} v_i \left(\sum_{r=1}^{n} w_{ir} \gamma \circ \alpha_{ir} \circ \pi_r \right) .$$

Položme $k = hr$, $u_j = v_i w_{ir}$ a $\beta_j = \alpha_{ir} \circ \pi_r$. Pak lze f přepsat na:

$$f = \sum_{j=1}^{k} u_j \gamma \circ \beta_j , \tag{13.17}$$

kde $k \in \mathbb{N}$, $u_j \in \mathbb{R}$ a β_j jsou speciální afinní transformace, které jsou vždy složením nějaké projekce s nekonstantní afinní transformací.

Kdybychom se vzdali tohoto omezení na transformace β_j a povolili všechny afinní transformace, dostaneme funkce třídy $\mathcal{G}_n^1$, což je vlastně (nyní již to můžeme prozradit) třída funkcí realizovaných perceptronovou sítí s jednou skrytou vrstvou a Gaussovou aktivační funkcí. Z výsledků podkapitoly 13.3 víme, že tyto sítě jsou univerzálními aproximátory.

kterou nazveme *řetězcové sítě*. Asi nás nepřekvapí fakt, že tyto sítě mají univerzální aproximační vlastnost, protože na perceptronové sítě s jednou vrstvou se můžeme dívat i jako na speciální případ kaskádových sítí s mnoha nulovými váhami. Zajímavý bude důkazový aparát, který použijeme — *řetězcové zlomky*, jež historicky patří do klasické části kalkulu matematické analýzy. V minulosti se využívaly při (ručním) počítání různých veličin a aproximaci funkcí k výpočtu tabulkových hodnot. První teoretické výsledky o řetězcových zlomcích se objevují už u Eulera a naopak soumrak této metody (alespoň jako prostředku k počítání hodnot funkcí) přichází s nástupem výpočetní techniky.

My využijeme některých teoretických výsledků o aproximaci funkcí řetězcovými zlomky. Jelikož existuje rozvinutá teorie řetězcových zlomků v oboru komplexních čísel, můžeme uvažovat kaskádové sítě s komplexními vstupy, výstupy i vahami. Zabývat se komplexními neuronovými sítěmi je samo o sobě zajímavým problémem z několika důvodů. Předně, některé praktické problémy, například z elektrotechniky, mohou být přirozeně vyjádřeny v komplexním oboru. Z teoretického hlediska představují komplexní čísla jisté zobecnění čísel reálných a proto i získané teoretické výsledky mohou být formulovány obecněji.

Problémem komplexního rozšíření neuronových sítí se zabývá několik prací, např. [113]. Situace je komplikovanější než v reálném oboru, protože není například jasné, jakou aktivační funkci zvolit. V komplexních číslech například neexistuje jednoznačný protějšek logistické sigmoidy. Často tak dochází k ad hoc volbě takové funkce, která je vhodná k řešení daného problému. Stejný postup zvolíme i my, když za aktivační funkci zvolíme co nejjednodušší racionální funkci.

13.6.2 Řetězcové zlomky a sítě

Budeme se tedy zabývat třídou funkcí s jednou skrytou vrstvou a jednou lineární výstupní jednotkou. Mezi jednotkami ve skryté vrstvě existují horizontální vazby, ale na rozdíl od obecné kaskádové sítě spojují vždy jen sousední jednotky. Navíc, uspořádáním jednotek je dána orientace jejich spojů, které míří vždy od i-té k $(i+1)$ jednotce (viz obr. 13.1).

Skryté jednotky počítají funkci ve tvaru:

$$o = \delta(z, y) = \frac{az + b}{cz + d + y}, \tag{13.20}$$

kde y je výstup z předchozí jednotky, z je vstupem sítě a a, b, c, d jsou komplexní parametry. Prvkům této podtřídy kaskádových sítí s aktivační funkcí tvaru (13.20) říkáme *řetězcové sítě*.

Dále, jelikož g je stejnoměrně spojitá, existuje $\delta \in \mathbb{R}$ takové, že:

$$|g(y) - g(z)| < \frac{\varepsilon}{2n(2n+1)},$$

pro všechna $y, z \in \langle a, b\rangle$, taková, že $|y - z| < \delta$.

Stejně, jako jsme aproximovali funkci ϕ, aproximujeme nyní každou z funkcí ψ_q. Vezměme tedy pro každé $q = 1, \ldots, 2n+1$ funkci h_q:

$$h_q = \sum_{j=1}^{h_q} v_{qj}\gamma \circ \beta_{qj},$$

takovou, že

$$|h_q(z) - \psi_q(z)| < \frac{\delta}{\lambda},$$

pro každé $z \in \mathcal{I}$, přičemž $\lambda = \max\{\lambda^{pq};\ p = 1, \ldots, n;\ q = 1, \ldots, 2n+1\}$.

Tedy platí:

$$\left| \sum_{q=1}^{2n+1} \sum_{i=1}^{h} w_i \gamma \circ \alpha_i \left(\sum_{j=1}^{h_q} \left(\sum_{p=1}^{n} \lambda^{pq} v_{qj} \gamma \circ \beta_{qj} \circ \pi_p \right) \right) \right| < \varepsilon.$$

Takže jsme aproximovali f s přesností ε pomocí sítě se $\left(\sum_{q=1}^{2n+1} h_q\right)$ gaussovskými jednotkami v první skryté vrstvě, $k(2n+1)$ lineárními jednotkami ve druhé (a jednou gaussovskou výstupní jednotkou). □

Následující věta vyslovuje stejné tvrzení vzhledem k prostoru $\mathcal{L}_p$. Je důsledkem právě dokázané věty a dalších vztahů prostorů C a $\mathcal{L}_p$, jako je například Luzinova věta (viz [242]).

Věta 13.32 *Pro každé přirozené $n \geq 2$ platí, že $\mathcal{G}_n^2$ je hustá v $\mathcal{L}_p(\mathcal{I}^n)$ pro $p \geq 1$.*

13.6 Kaskádové sítě

Tato sekce má za úkol ukázat jak použít různé nástroje matematické analýzy k důkazům aproximačních schopností speciálních neuronových sítí.

13.6.1 Úvod

S kaskádovými sítěmi jsme se již setkali v podkapitole 2.4, kde jsme se zabývali inkrementálním učícím algoritmem kaskádové korelace. V této podkapitole budeme vyšetřovat aproximační vlastnosti podtřídy kaskádových sítí,

Definujme nyní řetězcové zlomky a ukažme, že funkci realizovatelnou řetězcovou sítí můžeme přirozeně reprezentovat řetězcovým zlomkem.

Definice 13.36 Řetězcovým zlomkem *rozumíme následující schéma:*

$$b_0 + \cfrac{a_1(z)}{b_1(z)+\cfrac{a_2(z)}{b_2(z)+\cfrac{a_3(z)}{b_3(z)+\ddots}}} \tag{13.21}$$

kde a_i a b_i jsou komplexní funkce proměnné z.

Částečný řetězcový zlomek $P_n(z)$ *je řetězcový zlomek, který má jen konečně mnoho koeficientů a_i, b_i:*

$$P_n(z)=b_0+\cfrac{a_1(z)}{b_1(z)+\cfrac{a_2(z)}{b_2(z)+\cfrac{a_3(z)}{\ddots\,+\cfrac{\vdots}{b_{n-2}(z)+\cfrac{a_n(z)}{b_{n-1}+\frac{a_n(z)}{b_n(z)}}}}}}.$$

Existuje-li $lim_{n\to\infty}P_n(z)$ a je rovna v, říkáme, že řetězcový zlomek (13.21) má hodnotu v.

Tento odstavec uzavřeme pozorováním, ke kterému jsme směřovali a jež přímo vyplývá z předchozích definic:

Pozorování 13.37 *Každá funkce $f \in \mathcal{F}$ má tvar částečného řetězcového zlomku P_n, kde n odpovídá počtu jednotek ve skryté vrstvě.*

13.6.3 Řetězcové sítě a komplexní řady

V tomto odstavci uvedeme větu o univerzální aproximaci řetězcových sítí dokázanou Nerudou a Štědrým v [205], která ukazuje, že funkce realizovatelné řetězcovou sítí mohou s libovolnou přesností aproximovat každou meromorfní funkci. K důkazu použijeme dvou silných vět o rozvoji mocninných řad v řetězcové zlomky. Jako vždy v této části se zaměříme na hlavní myšlenky a techniky důkazu.

Věta 13.38 *Řetězcová síť může aproximovat každou meromorfní funkci s libovolnou přesností. Navíc, funkce ze třídy $\mathcal{E}$ mohou být řetězcovou sítí reprezentovány přesně.*

Důkaz: Budeme tedy dokazovat, že množina $\mathcal{F}$ je hustá v prostoru všech meromorfních funkcí se supremovou metrikou. Vezměme si libovolnou meromorfní funkci f a vyjádřeme si ji pomocí jejího Laurentova rozvoje:

$$f(z) = \sum_{i=-\infty}^{\infty} a_i z^i = \sum_{i=0}^{\infty}(c_i/z^{i+1}) + \sum_{i=0}^{\infty} c_i z^i\,. \tag{13.22}$$

Jak jsme se již zmínili, používáme jako aktivační funkce nejjednodušší racionální funkci, jež nám umožní snadnou aplikaci aparátu řetězcových zlomků. Tato funkce je v teorii komplexních čísel poměrně rozšířená, lze ji snadno vyčíslit na počítači, ale i přes svou jednoduchost nám umožňuje aproximovat i nespojitosti.

Abychom mohli formulovat výsledky, zavedeme si označení pro dvě třídy funkcí a několik dalších definic:

Definice 13.33 *Nechť $\mathcal{F}$ množinu všech funkcí $f : \mathbb{C} \to \mathbb{C}$ reprezentovatelných řetězcovou sítí s konečným počtem jednotek ve skryté vrstvě.*

Definice 13.34 *$\mathcal{E}$ budiž množina funkcí $f : \mathbb{C} \to \mathbb{C}$ tvaru: $\frac{p(z)}{q(z)}$, kde p a q jsou polynomy.*

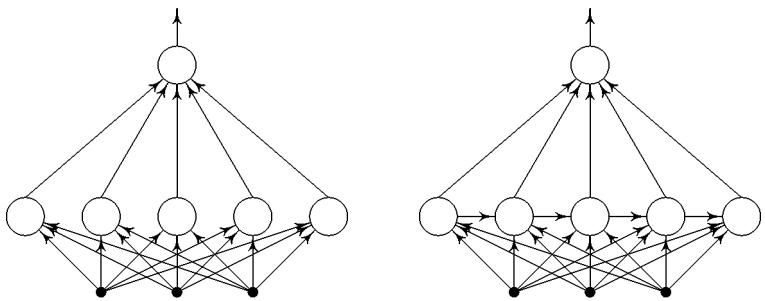
Definice 13.35 Homogenním polynomem *o n proměnných $H_n(x_1, \ldots, x_n)$ myslíme polynom tvaru:*

$$H_n(x_1, \ldots, x_n) = \sum_{j=1}^{m} a_j \prod_{i=1}^{n} x_i^{q_{ij}}$$

tak, že $\sum_{i=1}^{n} q_{ij}$ jsou totožné pro všechna $j = 1 \ldots m$. Označme $\mathcal{N}(H_n)$ množinu nulových bodů H_n.

Meromorfní funkce jsou takové komplexní funkce $f(z)$, jejíž jediné singularity jsou póly. Pro nás bude důležité, že každou meromorfní funkci lze napsat ve tvaru Laurentovy řady, což je nekonečná řada:

$$f(z) = \sum_{i=-\infty}^{\infty} a_i z^i;\ a_i \in \mathbb{C}$$



Obr. 13.1: Perceptronová a řetězcová síť

Potom lze sestrojit posloupnost polynomů ortogonálních vzhledem k posloupnosti $\{c_0, c_1, \ldots\}$.

Použitím předchozích vět můžeme vyjádřit celou Laurentovu řadu odpovídající naší meromorfní funkci f pomocí řetězcových zlomků a tím i řetězcovou sítí. Poslední technická podrobnost, kterou se nebudeme detailně zabývat spočívá v tom, že naše řada je obecná a obě věty 13.39 i 13.42 kladou jisté (slabé) podmínky na její koeficienty. To lze obejít rozepsáním řady pomocí tří řad, které tyto podmínky splňují. □

Dosud jsme se zajímali jen o řetězcové sítě s jednorozměrnými vstupy a výstupy, ale předchozí výsledek lze rozšířit i na obecné sítě s více vstupy. Použijeme k tomu techniku *redukce dimenze*, která spočívá v tom, že vícedimenzionální funkci nahradíme funkcemi jednodimenzionálními. Následující věta je jednoduchým důsledkem výsledku Vostrecova a Kreinese [279].

Věta 13.43 *Buď* $n \in \mathbb{N}$ *a* $W \subset \mathbb{C}^n$ *taková, že pro každý homogenní polynom* H_n *platí* $W \not\subset \mathcal{N}(H_n)$. *Pak existují spojité funkce* φ_{ik} *jedné proměnné takové, že množina funkcí* $\{\Phi_k(x);\ \Phi_k(x) = \sum_{i=1}^{k} \varphi_{ik}(w_i \cdot x);\ w_i \in W\}$, *je hustá v prostoru všech spojitých funkcí.*

Rozšíření do sítí s více výstupy, tedy na vektorové funkce $\mathbf{f} : \mathbb{C}^n \to \mathbb{C}^m$, lze jako obvykle provést jednoduše pomocí kartézského součinu jednorozměrných výstupních prostorů. Celým tímto postupem jsme se tedy dopracovali k univerzální aproximaci obecnými řetězcovými sítěmi.

13.6.4 Možnosti učení

Na rozdíl od jiných důkazů univerzální aproximace, které nejsou konstruktivní, můžeme předchozí postup využít i k návrhu alternativního učícího algoritmu pro řetězcové sítě. Tyto sítě můžeme učit různými způsoby. Například je možné odvodit pro ně variantu zpětného šíření chyby. Jediné, na co musíme pamatovat, je, aby výpočet bral v úvahu laterální vazby jednotek a probíhal skrytou vrstvu správným směrem.

Další zajímavější možností je inkrementální učící algoritmus, který postupně přidává do sítě další skryté jednotky. Jeden takový postup — Fahlmanův algoritmus kaskádové korelace — jsme si již popsali v podkapitole 2.4. Lze si představit i jiný postup, který více odpovídá řetězcovým zlomkům, takže přidání skryté jednotky je vlastně rozšířením zlomku o další člen, zatímco ostatní členy zůstávají stejné. Pokud by náš algoritmus byl spolehlivý, pak by posloupnost takto sestrojených řetězcových zlomků konvergovala k aproximované funkci. (Představme si například nerealistický případ, kdy je aproximovanou funkcí mocninná řada. Pak lze použít přímo Eulerovy

Řadu jsme rozdělili na dva členy podle znaménka exponentu i. Nyní ukážeme, že obě sumy můžeme vyjádřit pomocí řetězových zlomků. Využijeme k tomu dvou fundamentálních vět o rozvoji mocninných řad. První z nich je Eulerova identita, která ukazuje, jak rozepsat tu část Laurentova rozvoje, jež má kladné koeficienty.

Věta 13.39 *Mají-li výrazy na obou stranách smysl, platí následující rovnost:*

$$\sum_{i=0}^{\infty} c_i z^i = \cfrac{c_0 + c_1 z}{1 - \cfrac{c_2 z}{c_1 + c_2 z - \cfrac{c_1 c_3 z}{\ddots \quad \vdots \\ c_2 + c_3 z - \ldots \\ - \frac{c_{i-2} c_i z}{c_{i-1} c_i z} - \ddots}}}$$

Mocninnou řadu se zápornými exponenty lze také rozepsat ve tvaru řetězcového zlomku. Abychom to ukázali, zavedeme jednu další definici a bez důkazů prezentujeme dvě věty z monografie [280].

Definice 13.40 *Posloupnost polynomů* $B_p(u)(p \leq m)$ *nazveme* ortogonální vzhledem k posloupnosti $c = c_0, c_1, \ldots$, *je-li*

$$\int B_p(u) B_q(u) d\phi_c(u) \begin{cases} = 0 \ ; p \neq q, p, q \leq m \\ \neq 0 \ ; p = q < m \end{cases}$$

kde:

$$\int \sum_{i=0}^{n} a_i u^i d\phi_c(u) = \sum_{i=0}^{n} k_i c_i \, .$$

Věta 13.41 *Mocninnou řadu* $\sum_{i=0}^{\infty}(c_i/z^{i+1})$ *lze vyjádřit řetězcovým zlomkem právě tehdy, když lze sestrojit posloupnost polynomů ortogonálních vzhledem k posloupnosti koeficientů* $c_0, c_1, \ldots, c_n, \ldots$.

Věta 13.42 *Buďte* $\sum_{i=0}^{\infty}(c_i/z^{i+1})$ *mocninná řada a* Δ_p *determinanty:*

$$\Delta_p = \begin{vmatrix} c_0 & c_1 & \ldots & c_p \\ c_1 & c_2 & \ldots & c_{p+1} \\ & \vdots & & \\ c_p & c_{p+1} & \ldots & c_{2p} \end{vmatrix} ,$$

takové, že :

1. *buď* $\Delta_p \neq 0$ *pro všechna* p,
2. *nebo existuje číslo* m *takové, že:* $\Delta_p \neq 0$ *pro* $p \leq m$ *a* $\Delta_p = 0$ *pro* $p > m$.

identity k výpočtu koeficientů ve zlomku.) V každém případě ale můžeme pro nalezení dalších koeficientů použít gradientní algoritmus.

Poslední alternativou učení řetězcových sítí, kterou uvedeme, je postup vycházející z Viskovantovova výsledku o vyjádření racionálních funkcí řetězcovými zlomky [280]. Podle následujícího vzorce můžeme inkrementálně sestrojit řetězcový zlomek, známe-li koeficienty dané racionální funkce:

$$\frac{a_{10}+a_{11}x+a_{12}x^2+\ldots+a_{1n}x^n+\ldots}{a_{00}+a_{01}x+a_{02}x^2+\ldots+a_{0n}x^n+\ldots}=\frac{a_{10}}{a_{00}+\frac{a_{20}x}{a_{10}+\frac{a_{30}x}{a_{20}+\ldots}}}\,, \tag{13.23}$$

kde

$$a_{mn}=a_{m-1,0}a_{m-2,n+1}-a_{m-2,0}a_{m-1,n+1}\,. \tag{13.24}$$

Triviálním využitím tohoto výsledku je naučit se na základě tréninkové množiny racionální funkci a tu pak pomocí (13.23) vyjádřit jako řetězcový zlomek, potažmo síť. Není ovšem třeba používat jako mezičlánku racionální funkci, neboť lze odvodit algoritmus, který bude nastavovat přímo váhy řetězcové sítě podle rekurentního vztahu (13.24).

13.6.5 Diskuse

V této podkapitole jsme tedy ukázali, že množina funkcí realizovatelných řetězcovými sítěmi v $\mathbb{C}$ je hustá v prostoru meromorfních funkcí. Podíváme-li se zpět do oboru reálných čísel, zjistíme, že jsme touto sítí schopni aproximovat poměrně velkou množinu funkcí zahrnující dokonce funkce s nespojitostmi druhého druhu.

Aktivační funkce našich komplexních jednotek je vzhledem k řetězcovým zlomkům ta nejjednodušší možná — jde o podíl lineárních funkcí. Z teorie aproximace funkcí řetězcovými zlomky je známo, že použití vyšších polynomů může v některých případech výrazně urychlit konvergenci přibližného řešení. Použití složitějších aktivačních funkcí je otázkou rovnováhy mezi složitostí výpočtu uvnitř jedné jednotky a počtem jednotek v síti.

Aproximace funkcí pomocí perceptronů nebo RBF sítí se může zdát geometricky intuitivní, což neplatí u řetězcových sítí, kde racionální komplexní funkce spolu s inkrementálním skládáním hodnot geometrickou představu hatí. Na druhou stranu, z vlastností řetězcových zlomků plyne, že získaná aproximace je optimální vzhledem k počtu drahých strojových operací, tj. násobení a dělení.

Kapitola 14

Funkční ekvivalence a genetické učení

Jak jsme viděli v předchozí kapitole, otázka univerzální aproximace byla intenzivně studována několik let a pro většinu důležitých modelů dopředných sítí byla úspěšně vyřešena. Na tento fakt se můžeme dívat ze dvou pohledů. Za prvé víme, že perceptronové, RBF a další sítě jsou skutečně obecný výpočetní prostředek ve smyslu, schopnosti aproximovat libovolně přesně každou spojitou funkci. Na druhou stranu, jednou z motivací pro zkoumání univerzální aproximace byla i snaha získat kritérium, které by umožnilo určit, zda daná architektura je lepší či horší než jiná.

To je jeden z důvodů, proč se hledají další, jemnější, kritéria posuzování jednotlivých architektur. Typickými kandidáty na taková měřítka jsou velikost sítě (daná počtem jednotek a velikostí parametrů) a rychlost učení. Jednou z možností, jak zrychlit učící proces, je omezit prohledávaný parametrický prostor odstraněním redundancí. Toho lze dosáhnout tak, že se omezíme na minimální podprostory váhového prostoru, které obsahují jednoho zástupce od všech *funkčně ekvivalentních* váhových vektorů (obr. 14.1), tj. vektorů parametrů, které určují stejnou vstupně–výstupní funkci neuronové sítě.

Hecht-Nielsen [106] poukázal na to, že charakterizace funkčně ekvivalentních parametrizací sítě může zrychlit učící proces. První výsledky o funkční ekvivalenci se týkaly perceptronových sítí s hyperbolickým tangentem jako aktivační funkcí (Sussmann [262], Chen a kol. [133]). V poslední době se objevily práce o perceptronových sítích s obecnějšími aktivačními funkcemi (Albertini a Sontag [4], Kůrková a Kainen [165], Kainen a kol. [144]). Kůrková a Neruda [166] studovali problém funkční ekvivalence pro RBF sítě s Gaussovou aktivační funkcí.

V následujících podkapitolách si zavedeme potřebné pojmy a ukážeme hlavní výsledky o funkční ekvivalenci pro perceptronové a RBF sítě. Dále si na příkladu RBF sítí vyložíme, jak využít funkční ekvivalenci k popisu tzv. *kanonických parametrizací* sítě. Potom si stručně ukážeme, co to jsou genetické učící algoritmy a jak je lze modifikovat, abychom využili kanonických parametrizací k urychlení učení.

14.1 Pojmy

V následujícím budeme hovořit o perceptronových sítích s n vstupy, jednou skrytou vrstvou a jednou lineární výstupní jednotkou. Takové sítě počítají funkce $f : \mathbb{R}^n \to \mathbb{R}$ tvaru:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i \psi(\mathbf{v}_i \mathbf{x} + b_i)\,, \tag{14.1}$$

kde $k \in \mathbb{N}$ je počet jednotek ve skryté vrstvě, $w_i, b_i \in \mathbb{R}$, $\mathbf{v}_i \in \mathbb{R}^n$ a $\psi : \mathbb{R} \to \mathbb{R}$ je aktivační funkce.

Dále se budeme zabývat RBF sítěmi s n vstupy, skrytou vrstvou RBF jednotek a jednou lineární výstupní jednotkou. Tyto sítě reprezentují funkce $f : \mathbb{R}^n \to \mathbb{R}$ tvaru:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i \psi\left(\frac{\rho(\mathbf{x}, \mathbf{c}_i)}{b_i}\right)\,, \tag{14.2}$$

kde $k \in \mathbb{N}$ je počet jednotek ve skryté vrstvě, $w_i, b_i \in \mathbb{R}$, ρ je metrika na $\mathbb{R}^n$ a $\psi : \mathbb{R} \to \mathbb{R}$ aktivační funkce (obvykle je metrika indukována normou: $\rho(x, y) = ||x - y||$).

Posloupnost všech parametrů sítě, nazveme parametrizací. Každou parametrizaci určujeme vzhledem k aktivační funkci a počtu vstupů, u RBF jednotek ještě navíc k metrice, kterou měříme vzdálenost ve vstupním prostoru.

Definice 14.1 *Nechť $n \in \mathbb{N}$, $\psi : \mathbb{R} \to \mathbb{R}$, ρ metrika na $\mathbb{R}^n$.*

- Parametrizace perceptronové sítě *vzhledem k (ψ, n) je posloupnost* $\mathbf{Q} = (w_i, \mathbf{v}_i, b_i;\ i = 1, \ldots, k)$ *s významem symbolů dle (14.1).*
- Parametrizace RBF sítě *vzhledem k (ψ, n, ρ) je posloupnost* $\mathbf{P} = (w_i, \mathbf{c}_i, b_i;\ i = 1, \ldots, k)$ *s významem symbolů dle (14.2).*

Je jasné, že každá parametrizace $\mathbf{Q}$, resp. $\mathbf{P}$, určuje jednoznačně vstupně–výstupní funkci perceptronové, resp. RBF, sítě podle vztahu (14.1), či (14.2).

Věta 14.8 *Je-li aktivační funkce ψ asymptoticky konstantní a má vlastnost jednoznačné parametrizace vzhledem k 1 pro perceptronové sítě, pak má vlastnost jednoznačné parametrizace vzhledem k libovolnému přirozenému číslu.*

Důkaz této věty využívá vlastností z afinní geometrie a je uveden v [165].

Když je vlastnost jednoznačné parametrizace vzhledem k 1 porušena (pro nekonstantní funkce), pak lze ψ vyjádřit jako

$$\psi(t) = \sum_{i=1}^{k} w_i \psi(v_i t + b_i) + c\,, \tag{14.3}$$

kde $w_i \neq 0$, $v_i \neq 0$ a když $i \neq j$, pak buď $v_i \neq v_j$, nebo $b_i \neq b_j$ a navíc, je-li $b_i = 0$, pak $v_i \neq 1$.

Definice 14.9 *Je-li (14.3) splněna pro $k = 1$, nazýváme ψ* soběafinní. *Je-li (14.3) splněna pro $k \geq 2$, nazýváme ψ* afinně rekurzivní.

Následující důsledek charakterizuje funkce, jež mají vlastnost jednoznačné parametrizace perceptronových sítí vzhledem k jakémukoliv n.

Důsledek 14.10 *Nechť ψ je omezená, nekonstantní a asymptoticky konstantní aktivační funkce. Není-li ψ ani soběafinní ani afinně rekurzivní, má vlastnost jednoznačné parametrizace pro perceptronové sítě vzhledem k libovolnému $n \in \mathbb{N}$.*

Řada funkcí, včetně logistické sigmoidy či Gaussovy funkce, není afinně rekurzivní, takže takřka splňují nutné podmínky pro vlastnost jednoznačné parametrizace perceptronové sítě. Pro přesné popsání situace je však nutno rozšířit pojem záměnné ekvivalence tak, aby bral v úvahu i možnost změny znamének, která u perceptronových sítí hraje roli. Je-li např. aktivační funkce sudá (jako Gaussova funkce), nebo lze-li ji posunutím převést na lichou funkci (to je případ logistické sigmoidy), získáme další triviální případy funkční ekvivalence (např. pro Gaussovu funkci γ platí: $\gamma(x) = \gamma(-x)$). Podrobně jsou tyto případy popsány v [165].

14.2.1 Parametrizace RBF sítí

Pro RBF sítě nebyl dokázán žádný obdobný výsledek jako je věta 14.8, proto se nemůžeme omezit na vyšetřování funkcí v jedné dimenzi. Přesto i zde existují věty o jednoznačné parametrizaci vzhledem k libovolné dimenzi. Uvedeme si dva podobné výsledky, které hovoří o RBF sítích s Gaussovou radiální funkcí a metrikou odvozenou od skalárního součinu (typickým představitelem je euklidovská metrika), nebo maximovou metrikou.

Definice 14.2 *Říkáme, že dvě RBF parametrizace $\mathbf{P}$ a $\mathbf{P}'$ jsou* funkčně ekvivalentní, *realizují-li stejnou vstupně–výstupní funkci.*

Stejně se funkční ekvivalence definuje i pro perceptronové parametrizace. Povšimněme si, že počet skrytých jednotek se u $\mathbf{P}$ a $\mathbf{P}'$ může obecně lišit. Nejjednodušším typem funkční ekvivalence je permutace skrytých jednotek, která odpovídá permutování sčítanců sumy (14.1) nebo (14.2).

Definice 14.3 *Dvě RBF parametrizace jsou* záměnně ekvivalentní, *když $k = k'$ a existuje permutace π množiny $\{1, \ldots, k\}$ taková, že $w_i = w'_{\pi(i)}$* a *$b_i = b'_{\pi(i)}$* a *$\mathbf{c}_i = \mathbf{c}'_{\pi(i)}$, pro každé $i \in \{1, \ldots, k\}$.*

Analogická definice opět platí i pro perceptronové parametrizace.

V dalším se budeme zabývat jen takovými parametrizacemi, které nemají nějaké redundantní členy, budeme jim říkat *redukované parametrizace.*

Definice 14.4 *Parametrizace perceptronové sítě je* redukovaná, *když pro každé $i \in \{1, \ldots, k\}$ $w_i \neq 0$ a pro každé $i, j \in \{1, \ldots, k\}$ z $i \neq j$ vyplývá, že buď $\mathbf{v}_i \neq \mathbf{v}_j$, nebo $b_i \neq b_j$ a existuje nejvýše jedno i takové, že $\mathbf{v}_i = \mathbf{0}$.*

Definice 14.5 *Parametrizace RBF sítě je* redukovaná, *když pro každé $i \in \{1, \ldots, k\}$ $w_i \neq 0$ a pro každé $i, j \in \{1, \ldots, k\}$ z $i \neq j$ vyplývá, že buď $\mathbf{c}_i \neq \mathbf{c}_j$, nebo $b_i \neq b_j$.*

14.2 Parametrizace perceptronových sítí

Naším cílem bude zkoumat vztah mezi funkční a záměnnou ekvivalencí redukovaných parametrizací. Konkrétně nás bude zajímat, zda funkční ekvivalence implikuje záměnnou ekvivalenci (opačná implikace je triviální).

Definice 14.6 *Nechť $n \in \mathbb{N}$. Funkce ψ má vlastnost* jednoznačné parametrizace *vzhledem k n, platí-li pro každé dvě redukované parametrizace perceptronových sítí vzhledem k (ψ, n) (resp. RBF sítí vzhledem k (ψ, n, ρ)), že funkční ekvivalence implikuje záměnnou ekvivalenci.*

Definice 14.7 *Funkci $f : \mathbb{R} \to \mathbb{R}$ nazveme* asymptoticky konstantní, *má-li konečné limity v $+\infty$ i v $-\infty$.*

Pro perceptronové sítě lze dokázat následující větu o redukci dimenze, která ukazuje, že pro třídu asymptoticky konstantních funkcí stačí vyšetřovat vlastnosti sítě v jedné dimenzi.

Důkaz věty 14.11:

Podle lemmatu 14.13 stačí dokázat, že žádná redukovaná parametrizace negeneruje nulovou funkci. Pro spor předpokládejme opak:

$$\sum_{i=1}^{k} w_i \exp\left(-\frac{\rho(\mathbf{x},\mathbf{c}_i)^2}{b_i^2}\right) = \sum_{i=1}^{k} w_i \exp\left(-\frac{||\mathbf{x}-\mathbf{c}_i||^2}{b_i^2}\right) = 0 \qquad (14.4)$$

pro všechny $\mathbf{x} \in \mathcal{R}^n$.

Dále můžeme předpokládat, že $1 = b = \max\{b_i; i = 1,\ldots,k\}$ (jinak můžeme změnit měřítko) a že $\{b_i; i = 1,\ldots,k\}$ jsou uspořádány tak, že $1 = b_1 = \ldots = b_m > b_i$ pro všechny $i = m+1,\ldots,k$ a nakonec, že $\mathbf{c}_1 = 0$ (jinak můžeme posunout počátek soustavy souřadnic).

Vynásobíme (14.4) výrazem $\exp\left(||\mathbf{x}-\mathbf{c}_1||^2\right)$ a dostaneme

$$0 = w_1 + \sum_{i=2}^{m} \hat{w}_i \exp\left(2\mathbf{x}\cdot\mathbf{c}_i\right) + \\ + \sum_{i=m+1}^{k} \hat{w}_i \exp\left(||\mathbf{x}||^2\left(1-\frac{1}{b_i^2}\right) - 2\mathbf{x}\cdot\left(-\frac{\mathbf{c}_i}{b_i^2}\right)\right)$$

kde $\hat{w}_i = w_i \exp\left(-\frac{||\mathbf{c}_i||^2}{b_i^2}\right)$.

Jelikož pro každé $i = m+1,\ldots,k$ $1 - \frac{1}{b_i^2} < 0$, tak pro všechny $\mathbf{u} \in \mathcal{R}^n$ platí následující rovnost:

$$\lim_{t\to\infty} \sum_{i=m+1}^{k} \hat{w}_i \exp\left(||t\mathbf{u}||^2\left(1-\frac{1}{b_i^2}\right) - 2t\mathbf{u}\cdot\left(-\frac{\mathbf{c}_i}{b_i^2}\right)\right) = 0\,.$$

Tedy pro každé $\mathbf{u} \in \mathcal{R}^n$:

$$-w_1 = \lim_{t\to\infty} \sum_{i=2}^{m} \hat{w}_i \exp\left(2t\mathbf{u}\cdot(\mathbf{c}_i)\right)\,. \qquad (14.5)$$

Zvolme mezi $\{\mathbf{c}_2,\ldots,\mathbf{c}_m\}$ vektor $\mathbf{c}_j$ s nejmenší normou $||\mathbf{c}_j||$, tj. $||\mathbf{c}_j|| \geq ||\mathbf{c}_i||$ pro každé $i = 2,\ldots,m$. Dle lemmatu 14.14 pak pro každé $i \neq j$ $\mathbf{c}_i\cdot\mathbf{c}_j < \mathbf{c}_j\cdot\mathbf{c}_j$.

Dosadíme $\mathbf{u} = \mathbf{c}_j$ do (14.5) a dostáváme

$$-w_1 = \lim_{t\to\infty} \sum_{i=2}^{m} \hat{w}_i \exp\left(2t\mathbf{c}_j\cdot\mathbf{c}_i\right)\,. \qquad (14.6)$$

Připomeňme, že skalární součin $\cdot$ na $\mathbb{R}^n$, indukuje normu $||\cdot||$ na $\mathcal{R}^n$ předpisem

$$||\mathbf{x}||^2 = \mathbf{x}\cdot\mathbf{x}; \quad \mathbf{x} \in \mathcal{R}^n\,.$$

ρ pak označuje metriku odvozenou z normy $||\cdot||$:

$$\rho(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||; \quad \text{pro } \mathbf{x},\mathbf{y} \in \mathcal{R}^n\,.$$

Maximová metrika ρ_M je definována jako:

$$\rho_M(\mathbf{x},\mathbf{y}) = \max_{i=1,\ldots,n} |x_i - y_i|\,.$$

Věta 14.11 *Nechť $n \in \mathbb{N}$, ρ metrika indukovaná skalárním součinem, $\gamma(t) = \exp(-t^2)$. Pak γ má vlastnost jednoznačné parametrizace příslušné RBF sítě vzhledem k n.*

Věta 14.12 *Nechť $n \in \mathbb{N}$, ρ_M maximová metrika, $\gamma(t) = \exp(-t^2)$. Pak γ má vlastnost jednoznačné parametrizace příslušné RBF sítě vzhledem k n.*

Jelikož oba důkazy jsou podobné, uvedeme pouze důkaz první věty. Využijeme k němu následujících dvou lemmat.

Lemma 14.13 *Buď $n \in \mathbb{N}$, ρ metrika na $\mathcal{R}^n$ a $\varphi : \mathcal{R}_+ \to \mathcal{R}$ funkce. Když neexistuje redukovaná RBF parametrizace vzhledem k (φ, n, ρ) generující konstantní nulovou funkci, pak má φ vlastnost jednoznačné parametrizace vzhledem k n.*

Důkaz: Budeme postupovat sporem. Předpokládejme, že dvě různé redukované RBF parametrizace určují stejnou funkci:

$$\sum_{i=1}^{k} w_i \varphi\left(\frac{\rho(\mathbf{x},\mathbf{c}_i)}{b_i}\right) = \sum_{i=1}^{k'} w'_i \varphi\left(\frac{\rho(\mathbf{x},\mathbf{c}'_i)}{b'_i}\right)\,.$$

Odečtěme pravou stranu rovnice od levé a dejme dohromady členy se stejnými středy a šířkami. Tak vznikne redukovaná parametrizace, která generuje konstantní nulovou funkci, což je spor. □

Druhé lemma vyslovuje jednoduché tvrzení z lineární algebry. Uvedeme ho zde bez důkazu.

Lemma 14.14 *Buď ρ_E norma indukovaná skalárním součinem $\cdot$ na $\mathcal{R}^d$, $\{\mathbf{c}_i; i = 1,\ldots,k\}$ buď množina různých nenulových vektorů v $\mathcal{R}^n$. Nechť $j \in \{1,\ldots,k\}$ je takové, že platí $||\mathbf{c}_j|| \leq ||\mathbf{c}_i||$ pro každé $i = 1,\ldots,k$. Potom $\mathbf{c}_i\cdot\mathbf{c}_j < \mathbf{c}_j\cdot\mathbf{c}_j$ pro každé $i \neq j$.*

Předchozí věty 14.11 a 14.12 tedy říkají, že ke každé parametrizaci existuje kanonická parametrizace určující stejnou funkci. Chen a Hecht-Nielsen [134] navrhli studovat podprostory váhového prostoru obsahující právě jednoho reprezentanta každé třídy funkčně ekvivalentních parametrizací, nazývají je *minimální prohledávací množiny.*

Důsledek 14.16 *Pro každé $n \in \mathbb{N}$, pro každou metriku ρ na $\mathcal{R}^n$ indukovanou skalárním součinem nebo maximovou metriku ρ_M tvoří množina kanonických parametrizací minimální prohledávací množinu.*

Abychom mohli předchozí výsledky využít při řešení praktických problémů, potřebujeme učící algoritmus, který by uměl pracovat pouze na kanonických parametrizacích. Bohužel, obvyklé gradientní metody toho nejsou schopny, protože analytické řešení, které získáme v každé iteraci algoritmu, nemůže být principiálně omezeno na určitou podmnožinu váhového prostoru. Naštěstí se takový algoritmus dá najít, a to v rodině genetických algoritmů, se kterými se stručně seznámíme v následující podkapitole.

14.3 Genetické algoritmy

Genetické algoritmy (GA) představují stochastickou prohledávací metodu, která je inspirována genetickými principy jako je přirozený výběr, křížení a mutace. I když se první pokusy o modelování evoluce na počítači objevily už v padesátých letech, za objevitele genetických algoritmů tak, jak je známe dnes, je považován Holland, který položil základy této disciplíny v sedmdesátých letech [118, 119].

14.3.1 Základy

GA se nejvíce používají k řešení problémů optimalizace. Mnoho problémů (např. učení neuronové sítě) může být formulováno jako hledání minima či maxima funkce v závislosti na jejích parametrech. V řeči GA většinou mluvíme o *účelové funkci.* Jelikož parametrický prostor je typicky příliš velký pro exhaustivní prohledávání, používáme nějakou stochastickou techniku pro nalezení přibližného řešení. Gradientní metody například reprezentují hledání lokálního minima či maxima pomocí jednoho zpřesňujícího se řešení.

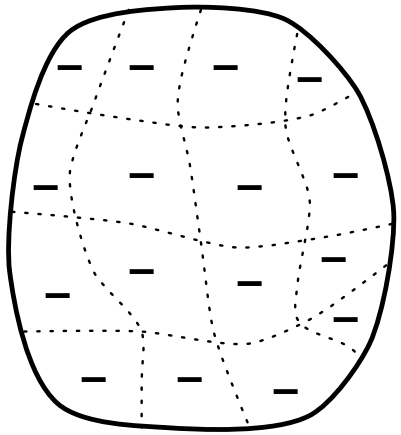
Genetické algoritmy představují jiný přístup, který využívá celou *populaci* prozatímních řešení, jež paralelně procházejí parametrický prostor a navzájem se ovlivňují a modifikují pomocí *genetických operátorů.* Tímto mechanismem, nazývaným *implicitní paralelismus*, se dosahuje jakéhosi synergetického efektu, takže populace jedinců najde správné řešení rychleji, než

Vynásobíme (14.6) výrazem $\exp\left(-2t\|\mathbf{c}_j\|^2\right)$:

$$-\lim_{t\to\infty} w_1 \exp\left(-2t\|\mathbf{c}_j\|^2\right) = \hat{w}_j + \lim_{t\to\infty} \sum_{\substack{i=2\\ i\neq j}}^{m} \hat{w}_i \exp\left(2t(\mathbf{c}_j\cdot\mathbf{c}_i - \mathbf{c}_j\cdot\mathbf{c}_j)\right) .$$

Jelikož oba výrazy $-\|\mathbf{c}_j\|^2$ i $(\mathbf{c}_j\cdot\mathbf{c}_i - \mathbf{c}_j\cdot\mathbf{c}_j)$ jsou záporné (první zřejmě, druhý podle lemmatu 14.14), limity na obou stranách se rovnají 0. Tedy $\hat{w}_j = w_j e^{-\|\mathbf{c}_j\|^2} = 0$. Čili $w_j = 0$, což je ale ve sporu s naším předpokladem, že jde o redukovanou parametrizaci. □

Nyní využijeme předchozích vět k tomu, abychom popsali kanonickou reprezentaci sítě realizující danou funkci. Jelikož jediné transformace parametrů, které nemění danou funkci, jsou výměny jednotek ve skryté vrstvě, musíme zvolit zástupce každé ze tříd parametrizací s permutovanými skrytými jednotkami (obr. 14.1).



Obr. 14.1: Třídy funkční ekvivalence a kanonické parametrizace

Jednou z možností je uspořádat váhové vektory odpovídající jednotlivým skrytým jednotkám vzestupně v lexikografickém uspořádání. Reprezentujme parametrizaci $\{w_i, k_i, \mathbf{c}_i; i = 1,\ldots,k\}$ jako vektor $\mathbf{P} = \{\mathbf{p}_i,\ldots\mathbf{p}_k\} \in \mathcal{R}^{k(n+2)}$, kde $\mathbf{p}_i = \{w_i, b_i, c_{i1},\ldots,c_{in}\} \in \mathcal{R}^{n+2}$ jsou parametry příslušející i-té skryté jednotce. Nechť $\prec$ označuje lexikografické uspořádání na $\mathcal{R}^{n+2}$, tj. pro $\mathbf{p},\mathbf{q} \in \mathcal{R}^{d+2}$ platí: $\mathbf{p} \prec \mathbf{q}$, existuje-li index $m \in \{1,\ldots,d+2\}$ tak, že $p_j = q_j$ pro $j < m$ a $p_m < q_m$.

Definice 14.15 *Parametrizaci RBF sítě $\mathbf{P} = \{\mathbf{p}_i,\ldots\mathbf{p}_k\}$ nazveme* kanonickou, *platí-li*

$$\mathbf{p}_1 \prec \mathbf{p}_2 \prec \ldots \prec \mathbf{p}_k . \qquad (14.7)$$

Počáteční populace Počáteční populaci m jedinců

$$\mathcal{P}_0 = \{\mathbf{P}_1, \ldots, \mathbf{P}_m\},$$

ze které začíná vlastní učící proces vytváříme typicky náhodně.

Účelová funkce V případě neuronové sítě máme požadované chování jedince určeno tréninkovou množinou. Jejím průchodem vypočítáme chybu sítě, tj. rozdíl mezi požadovanými a skutečnými odpověďmi sítě, a na jejím základě vypočítáme hodnotu účelové funkce jako:

$$\mathcal{G}(\mathbf{P}) = C - \sum_{t=1}^{k} ||y^{(t)} - d^{(t)}||^2, \tag{14.8}$$

kde $y^{(t)}$ (resp. $d^{(t)}$) je skutečný (resp. požadovaný) výstup sítě po předložení vzoru $\mathbf{x}^{(t)}$, a $C \in \mathbb{R}$ je maximální chyba v populaci (nebo její odhad).

Operátor selekce Operátor selekce slouží k vybrání jedince z populace v závislosti na jeho hodnotě účelové funkce. Selekce modeluje darwinistický přirozený výběr tak, že vybírá jedince náhodně s pravděpodobností úměrnou hodnotě jeho účelové funkce. Pravděpodobnost výběru p_l získáme znormováním hodnot účelové funkce $\mathcal{G}(P_l)$ pro $l = 1, \ldots, r$:

$$p_l = \frac{G(\mathbf{P}_l)}{\sum_{r=1}^{m} G(\mathbf{P}_r)}. \tag{14.9}$$

Tomuto způsobu realizace selekce se říká *ruletová selekce*, protože si můžeme představit, že ji realizujeme pomocí ruletového kola, které rozdělíme na výseče odpovídající jednotlivým jedincům. Velikost každé výseče je přímo úměrná hodnotě p_l odpovídajícího jedince. Existují i jiné způsoby, jak vybírat jedince z populace, například různé turnaje.

Navíc se v praxi často používá heuristický *elitářský mechanismus*, který zaručí, že určité procento nejlepších jedinců je vždy reprodukováno do nové populace. Tím je zajištěno, že maximum hodnot účelové funkce v populaci s časem neklesá.

Operátor křížení Křížení kombinuje dva vybrané jedince a na jejich základě vytvoří dva nové jednice. Nejčastějším typem křížení je křížení *jednobodové*, kdy se vybere náhodná pozice, na níž se jedinci rozdělí, a jejich části se navzájem vymění. Někdy se používají i vícebodová křížení a jejich limitním případem je *uniformní* křížení, u něhož se pro každou základní jednotku jedince (tj. buď bit, nebo číslo, v závislosti na způsobu zakódování) náhodně určí, do kterého z nových jedinců se zařadí. Výhodou uniformního křížení je, že u něj nezáleží na pořadí, v jakém jsou informace v jedinci zakódovány.

kdyby prohledávali prostor izolovaně. Výsledek [118] ukazuje, že populace GA o n jedincích pracuje zhruba tak rychle jako n^3 izolovaných hledačů.

Goldberg ve své knize [78] formuloval základní rozdíly mezi GA a klasickou optimalizací:

1. GA pracuje s řetězcem zakódovaných hodnot parametrů, ne s parametry samotnými.
2. GA prohledává prostor parametrů vycházejíc z populace počátečních bodů, ne z jednoho bodu.
3. GA využívá pouze informace poskytované účelovou funkcí, ne derivace nebo další doplňující znalosti.
4. GA používá pravděpodobnostní mechanismus hledání.

Popsat GA znamená definovat pět následujících částí:

- genetickou reprezentaci potenciálních řešení (zakódování parametrů)
- způsob vytváření počáteční populace
- účelovou funkci
- genetické operátory, které vytvářejí novou populaci
- hodnoty některých parametrů GA (velikost populace, pravděpodobnosti operátorů, ...)

14.3.2 Genetické učení neuronových sítí

Ukažme si nyní ve stručnosti jednoduchý GA, který bude učit neuronovou síť. Na začátku vytvoříme počáteční populaci jedinců a v každém kroku algoritmu vybíráme jedince z populace pomocí operátoru selekce, případně je modifikujeme operátory křížení a mutace a vytváříme novou populaci. To opakujeme tak dlouho, dokud se v populaci nevyskytne jedinec s dostatečně dobrou hodnotou své účelové funkce. Schematicky je jednoduchý GA znázorněn na obr. 14.2. Popišme nyní jeho základní části.

Zakódování parametrů Neuronová síť je určena hodnotami svých vah, takže jedinec GA odpovídá parametrizaci $\mathbf{P}_i$ neuronové sítě. Jde tedy o řetězec zakódovaných vah. Zakódování může být binární, nebo lze za základní jednotku považovat jedno racionální číslo ve strojové reprezentaci. Lze si též představit zakódování, které nemá pevnou délku a tak se může měnit i topologie sítě, ale pro jednouchost se takovou variantou nebudeme zabývat.

Čili sousedé $\mathbf{p}_{s-1}$ a $\mathbf{p}_{s+1}$ určují meze, v nichž náhodně pozměníme $\mathbf{p}_s$. Výsledná parametrizace má tedy tvar:

$$\mathbf{P}' = (\mathbf{p}_1, \ldots, \mathbf{p}_{s-1}, \mathbf{p'}_s, \mathbf{p}_{s+1}, \ldots, \mathbf{p}_k)\,,$$

kde $\mathbf{p'}_s$ je modifikovaný $\mathbf{p}_s$.

14.4.3 Křížení

Křížení v našem případě vytváří pouze jednoho potomka $\mathbf{P}' \in \mathcal{P}_{i+1}$, na základě dvou parametrizací $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)$ a $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_k)$ z $\mathcal{P}_i$

Nejprve náhodně vybereme pozici křížení $s \in \langle 1, k\rangle$ a pak zjistíme, který ze dvou možných potomků splňuje podmínku kanoničnosti. Konkrétně to znamená, že ověříme, zda platí

$$\mathbf{p}_s \prec \mathbf{q}_{s+1}\,. \tag{14.10}$$

Pokud ano, přiřadíme $\mathbf{P}'$ vektor

$$\mathbf{P}' = (\mathbf{p}_1, \ldots, \mathbf{p}_s, \mathbf{q}_{s+1}, \ldots, \mathbf{q}_k)\,.$$

V opačném případě (podmínka (14.10) neplatí), bude $\mathbf{P}'$ mít tvar:

$$\mathbf{P}' = (\mathbf{q}_1, \ldots, \mathbf{q}_s, \mathbf{p}_{s+1}, \ldots, \mathbf{p}_k)\,.$$

14.4.4 Diskuse

Na rozdíl od standardního GA není časová složitost operací křížení a mutace konstantní. V našem případě zabere každé křížení $O(k)$ operací a mutace $O(k+n)$ operací (kde k je počet skrytých jednotek, n dimenze vstupu).

Na druhou stranu, redukce prohledávacího prostoru je mnohem větší. V případě RBF sítí, kde funkční ekvivalence odpovídá permutaci jednotek ve skryté vrstvě, pracuje algoritmus jen nad $\frac{1}{k!}$ celého parametrického prostoru (neb permutací k prvků je právě $k!$). To dává naději, že dojde i ke zrychlení učícího procesu. Praktické pokusy skutečně ukazují, že kanonický algoritmus je průměrně dvakrát rychlejší než standardní GA. Zatím chybí porovnání s gradientními metodami učení, ale dle podobných výsledků z aplikací GA se zdá, že rychlost genetického učení není výrazně vyšší než jiných metod, a že v závislosti na aplikacích se dosahuje odlišných výsledků. Na druhou stranu, GA jsou vhodnější pro paralelní implementaci než algoritmy typu zpětného šíření, protože vyžadují mnohem méně komunikace.

V kontextu genetických algoritmů spadá popsaný kanonický GA do oblasti *evolučního programování* [195], což je obecnější přístup než klasické GA v tom smyslu, že se uvažují různé způsoby zakódování parametrů a v závislosti na nich i příslušné modifikace tradičních genetických operátorů.

Operátor mutace Zatímco selekce modeluje mechanismus přirozeného výběru a křížení se snaží ze dvou jedinců vytvořit výměnou jejich genetického materiálu jedince kvalitnější, mutace vnáší do populace náhodné změny. Typicky se mutace provádí řádově méně často než křížení. Mutaci realizujeme tak, že náhodně zvolíme pozici v jedinci a danou základní jednotku (bit, číslo, atp.) náhodně změníme. Význam mutace tkví v tom, že dokáže vytvářet zcela nové hodnoty parametrů a zabraňuje tak uvíznutí celého procesu v oblasti lokálního extrému optimalizované funkce.

14.4 Kanonický genetický algoritmus

Navážeme nyní na předchozí odstavec, kde jsme si velmi stručně ukázali, jak funguje standardní GA při učení neuronových sítí. Popíšeme, jak vypadá *kanonický GA*, který dokáže využívat specifikace kanonických parametrizací z podkapitoly 14.2. Ukážeme si variantu algoritmu pro RBF sítě.

Základ kanonického GA je totožný s tím, co jsme popsali výše. Opět operujeme nad parametrizacemi sítě, které chápeme jako vektory racionálních čísel (nesestupujeme tedy na úroveň bitů). Naší snahou je pracovat pouze s kanonickými parametrizacemi. Na začátku tedy vygenerujeme populaci kanonických parametrizací, jejíž členy pak modifikujeme pomocí selekce, křížení a mutace, přičemž musíme zaručit, aby tyto operace zachovávali kanoničnost parametrizace. Jelikož selekce jen vybírá jedince z populace, zaměříme se na inicializaci, křížení a mutaci.

14.4.1 Inicializace

Při vytváření počáteční populace musíme vygenerovat jen kanonické parametrizace. Čili musíme zajistit, aby pro každou parametrizaci $\mathbf{P}_l \in \mathcal{P}$ platilo:

$$\mathbf{p}_{l,s} \prec \mathbf{p}_{l,s+1};\ s = 1, \ldots, k;\ l = 1, \ldots, m\,.$$

Při vytváření parametrizace postupujeme pro $s = 2, \ldots, k$ a dbáme, aby se $\mathbf{p}_s$ generovalo náhodně, ale následovalo v lexikografickém uspořádání $\mathbf{p}_{s-1}$.

14.4.2 Mutace

Mutace u kanonického algoritmu operuje na dvou úrovních. Nejprve se díváme na parametrizaci $\mathbf{P}$ jako na vektor $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$ a náhodně vybereme pozici $s \in \langle 1, k\rangle$ určující kandidáta $\mathbf{p}_s$ na mutaci.

Jelikož musíme zachovat kanoničnost, nelze $\mathbf{p}_s$ měnit náhodně. Musíme zajistit, aby stále platilo, že

$$\mathbf{p}_{s-1} \prec \mathbf{p}_s \prec \mathbf{p}_{s+1}\,.$$

```
var Stará, Nová: Populace N Jedinců;
    Otec, Matka, Syn, Dcera: Jedinci;
    i: integer;
    p, max: real;
begin
(*inicializace*)
    for i:= 1 to N do
    begin
            Otec:= random Jedinec;
            p:= účelová-funkce(Otec);
            dej (Otec,p) do Staré
    end;
    repeat
(* další populace *)
            max:= 0;
            vyber Otce ze Staré;
            vyber Matku ze Staré;
            zkřiž Otce a Matku do Syna a Dcery;
            mutuj Syna;
            mutuj Dceru;
            p:= účelová-funkce(Syn);
            dej (Syn,p) do Nové;
            if p>max then max:= p;
            p:= účelová-funkce(Dcera);
            dej (Dcera,p) do Nové;
            if p>max then max:= p;
    until max<koncové-kritérium
end.
```

Obr. 14.2: Schéma jednoduchého genetického algoritmu

Rejstřík pojmů a symbolů

$\emptyset$	prázdná množina
$a \in A$	a je prvkem množiny A
$A \subseteq B$	A je podmnožinou B (inkluze)
$A \subsetneqq B$	$A \subseteq B$ a $A \neq B$ (ostrá inkluze)
$A \cup B$	sjednocení množin A a B
$A \cap B$	průnik množin A a B
$A \setminus B$	rozdíl množin A a B
$A \Delta B$	symetrická diference $(A \setminus B) \cup (B \setminus A)$
$\bar{A}$	doplněk množiny A
$\mathcal{P}(A)$	potenční množina množiny A
$A \times B$	kartézský součin množin A a B
A^n	kartézská mocnina množiny A
$\|A\|$	počet prvků množiny A (kardinalita A)
$\mathbb{N}$	množina přirozených čísel
$\mathbb{Z}$	množina celých čísel
$\mathbb{Q}$	množina racionálních čísel
$\mathbb{R}$	množina reálných čísel
$\mathbb{R}^+$	množina kladných reálných čísel
$\mathbb{C}$	množina komplexních čísel
$\langle a, b \rangle$	uzavřený interval od a do b
(a, b)	otevřený interval od a do b
$\mathcal{I}$	interval $\langle 0, 1 \rangle$
$x \wedge y$	konjunkce x a y ($AND(x, y)$)
$x \vee y$	disjunkce x a y ($OR(x, y)$)
$\bar{x}$	negace x ($NOT(x)$)
$x \oplus y$	vylučovací disjunkce x a y ($XOR(x, y)$)
$\forall$	univerzální kvantifikátor
$\exists$	existenční kvantifikátor
$f : A \longrightarrow B$	zobrazení A do B
f^{-1}	inverzní zobrazení k f
$f \circ g$	složené zobrazení ($f(g(x))$)

$n!$	faktoriál čísla n
$\binom{n}{k}$	kombinační číslo
$\operatorname{abs} x = \lvert x \rvert$	absolutní hodnota z čísla x
$\lfloor x \rfloor$	dolní celá část čísla x
$\lceil x \rceil$	horní celá část čísla x
$\min A$	minimum množiny A
$\max A$	maximum množiny A
$\log x$	logaritmus o základu 2
$\ln x$	přirozený logaritmus
e^x	exponenciální funkce
$\operatorname{tgh} x$	hyperbolický tangens
$\lim_{n\to\infty} a_n$	limita posloupnosti a_n
$\frac{dy}{dx}$	derivace y podle x
$\frac{\partial y}{\partial x}$	parciální derivace y podle x
$\sum_{i=1}^{n} a_i$	součet čísel $a_1 + \ldots + a_n$
$\sum_{a\in A} a$	součet čísel z množiny A
$\prod_{a\in A} a$	součin čísel z množiny A
$\int_a^b f(x)dx$	určitý integrál funkce $f(x)$ v intervalu $\langle a, b\rangle$
$C(X)$	prostor spojitých funkcí nad kompaktní X
$\mathcal{L}_p$	prostor funkcí integrovatelných v $\lvert f\rvert^p$
$\mathcal{L}^\infty$	prostor funkcí omezených skoro všude

$\mathbf{x}$	vektor
$\mathbf{x} \cdot \mathbf{y}$	skalární součin vektorů $\mathbf{x}$ a $\mathbf{y}$ (resp. $\mathbf{xy}$)
$\lVert\mathbf{x}\rVert$	norma vektoru $\mathbf{x}$ (resp. funkce)
$\mathbf{A}$	matice
$\mathbf{A}^T$	transponovaná matice
$\mathbf{A}^{-1}$	inverzní matice
$\mathbf{A}^+$	pseudoinverze matice $\mathbf{A}$
$\mathbf{A}^\mathsf{H}$	Hermitovská matice
$\lvert\mathbf{A}\rvert$	determinant matice $\mathbf{A}$

$\rho(x, y)$	metrika
$\mathcal{T}$	topologie
$\overline{A}$	uzávěr množiny A

$\mathbf{P}\{A\}$	pravděpodobnost jevu A
$\mathbf{P}\{A\lvert B\}$	podmíněná pravděpodob. jevu A, pokud nastal jev B
$\mathbf{E}[X]$	střední hodnota náhodné veličiny X

e	prázdné slovo
$\mathbf{x}$	slovo (řetězec) jazyka
$\lvert\mathbf{x}\rvert$	délka slova $\mathbf{x}$
$\boldsymbol{\alpha} + \boldsymbol{\beta}$	součet regulárních výrazů $\boldsymbol{\alpha}$ a $\boldsymbol{\beta}$ (sjednocení)
$\boldsymbol{\alpha}\boldsymbol{\beta}$	spojení regulárních výrazů $\boldsymbol{\alpha}$ a $\boldsymbol{\beta}$
$\boldsymbol{\alpha}^n$	mocnina regulárního výrazu $\boldsymbol{\alpha}$
$\boldsymbol{\alpha}^\star$	iterace regulárního výrazu $\boldsymbol{\alpha}$
$\boldsymbol{\alpha}^+$	pozitivní iterace regulárního výrazu $\boldsymbol{\alpha}$
$[\boldsymbol{\alpha}]$	regulární jazyk popsaný regulárním výrazem $\boldsymbol{\alpha}$
$L(M)$	jazyk rozpoznávaný zařízením M

$g = O(f(n))$	$(\exists\varepsilon > 0)\,(\exists n_0)\,(\forall n \geq n_0)\, g(n) \leq \varepsilon f(n)$
$g = \Omega(f(n))$	$(\exists\varepsilon > 0)\,(\exists n_0)\,(\forall n \geq n_0)\, g(n) \geq \varepsilon f(n)$
$g = \Theta(f(n))$	$g = O(f(n))$ a $g = \Omega(f(n))$
P	třída jazyků rozpoznávaná v polynomiálním čase
NP	třída jazyků rozpoznávaná nedeterm. v polynom. čase
$PSPACE$	třída jazyků rozpoznávaná v polynomiálním prostoru

Literatura

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowksi. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, 1974.

[3] H. Aizenstein, L. Hellerstein, and L. Pitt. Read-thrice DNF is hard to learn with membership and equivalence queries. In *Proceedings of the* 33^{rd} *Annual Symposium on Foundations of Computer Science*, pages 523–532. IEEE Computer Society Press, 1992.

[4] F. Albertini and E. D. Sontag. For neural networks, function determines form. *Neural Networks*, 6:975–990, 1993.

[5] E. Allender. A note on the power of threshold circuits. In *Proceedings of the* 30^{th} *Annual Symposium on Foundations of Computer Science*, pages 580–584. IEEE Computer Society Press, 1989.

[6] L. B. Almeida. Backpropagation in perceptrons with feedback. In R. Eckmiller and Ch. von der Malsburg, editors, *Neural Computers*, pages 199–208, Neuss, 1987. Berlin: Springer-Verlag.

[7] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In M. Caudill and C. Butler, editors, *Proceedings of the IEEE First International Conference on Neural Networks*, volume II, pages 609–618, San Diego, 1987. New York: IEEE Press.

[8] N. Alon. Asynchronous threshold networks. *Graphs and Combinatorics*, 1:305–310, 1987.

[9] S.-I. Amari. Neural theory of association and concept formation. *Biological Cybernetics*, 26:175–185, 1977.

[10] S.-I. Amari. Topographic organization of nerve fields. *Mathematical Biology*, 42:339–364, 1980.

[11] D. J. Amit, H. Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55:1530–1533, 1985.

[12] D. J. Amit, H. Gutfreund, and H. Sompolinsky. Information storage in neural networks with low levels of activity. *Physical Review*, A 35:2293–2303, 1987.

[13] J. A. Anderson. A memory storage model utilizing spatial correlation functions. *Kybernetik*, 5:113–119, 1968.

[14] J. A. Anderson. Two models for memory organization using interacting traces. *Mathematical Biosciences*, 8:137–160, 1970.

[15] J. A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14:197–220, 1972.

[16] J. A. Anderson. A theory for the recognition of items from short memorized lists. *Psychological Review*, 80:417–438, 1973.

[17] J. Anděl. *Matematická statistika*. SNTL, Praha, 1985.

[18] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[19] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[20] D. Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the* 24^{th} *Annual ACM Symposium on the Theory of Computing*, pages 351–369. New York: ACM Press, 1992.

[21] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and Systems Sciences*, 18:155–193, 1979.

[22] M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge University Press, Cambridge, 1992.

[23] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal Algebraic and Discrete Methods*, 8:277–284, 1987.

[24] V. I. Arnold. On functions of three variables. *Dokl. Akad. Nauk SSSR*, 114:679–681, 1957.

[25] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proceedings of the* 27^{th} *Annual Symposium on Foundations of Computer Science*, pages 337–347. IEEE Computer Society Press, 1986.

[26] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity*, volume I. Springer-Verlag, Berlin, 1988.

[27] J. L. Balcázar, R. Gavaldà, and H. T. Siegelmann. Computational power of neural networks: A Kolmogorov-like complexity characterization. *To appear in IEEE Transactions of Information Theory*, 1996.

[28] J. L. Balcázar, M. Hermo, and E. Mayordomo. Characterizations of logarithmic advice complexity classes. In *IFIP Transactions on Information Processing*, volume I, pages 315–321. North-Holland, 1992.

[29] R. Batruni. A multilayer neural network with piecewise-linear structure and back-propagation learning. *IEEE Transactions on Neural Networks*, 2:395–403, 1991.

[30] E. B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation*, 2:248–260, 1990.

[31] E. B. Baum. A polynomial time algorithm that learns two hidden unit nets. *Neural Computation*, 2:510–522, 1990.

[32] E. B. Baum. Book reviews. *IEEE Transactions on Neural Networks*, 2:181–182, 1991.

[33] E. B. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2:5–19, 1991.

[34] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[35] P. Berman, I. Parberry, and G. Schnitger. A note on the complexity of reliability in neural networks. *IEEE Transactions on Neural Networks*, 3:998–1002, 1992.

[36] C. M. Bishop. Improving the generalization properties of radial basis function neural networks. *Neural Computation*, 3:579–588, 1991.

[37] A. L. Blum and R. Kannan. Learning an intersection of k halfspaces over a uniform distribution. In *Proceedings of the* 34^{th} *Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE Computer Society Press, 1993.

[38] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127, 1992.

[39] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[40] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.

[41] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[42] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[43] J. Bruck and J. W. Goodman. A generalized convergence theorem for neural networks and its applications in combinatorial optimization. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume III, pages 649–656, San Diego, 1987. New York: IEEE Press.

[44] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Blaisdell, New York, 1969.

[45] J. P. Cater. Successfully using peak learning rates of 10 (and greater) in back-propagation networks with the heuristic learning algorithm. In M. Caudill and C. Butler, editors, *Proceedings of the IEEE First International Conference on Neural Networks*, volume II, pages 645–651, San Diego, 1987. New York: IEEE Press.

[46] M. A. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:815–826, 1983.

[47] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the* 3^{rd} *Annual ACM Symposium on the Theory of Computing*, pages 151–158. New York: ACM Press, 1971.

[48] S. A. Cook. Towards a complexity theory of synchronous parallel computation. *L'Enseignement Mathématique*, 27:99–124, 1980.

[49] G. W. Cottrell, P. Munro, and D. Zipser. Image compression by back propagation: An example of extensional programming. Technical Report 8702, University of California at San Diego Institute for Cognitive Science, 1987.

[50] T. M. Cover. Capacity problems for linear machines. In L. Kanal, editor, *Pattern Recognition*, pages 283–289. Thompson Book Co., 1968.

[51] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with application in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326–334, 1965.

[52] G. Cybenko. Approximations by superpositions of a sigmoidal functions. *Mathematics Control Signals Systems*, 2:303–314, 1989.

[53] Y. Z. Cypkin. *Foundation of the theory of learning systems*. Academic Press, New York, 1973.

[54] J. Červenka. Konstruktivní učení vícevrstvé neuronové sítě. Diplomová práce, katedra informatiky, MFF UK, Praha, 1996.

[55] B. DasGupta and G. Schnitger. The power of approximating: A comparison of activation functions. In C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 615–622. San Mateo: Morgan Kaufmann, 1993.

[56] B. DasGupta and G. Schnitger. Analog versus discrete neural networks. *Neural Computation*, 8:805–818, 1996.

[57] B. DasGupta, H. T. Siegelmann, and E. Sontag. On the complexity of training neural networks with continuous activation functions. *IEEE Transactions on Neural Networks*, 6:1490–1504, 1995.

[58] D. Desieno. Adding a conscience to competitive learning. In *Proceedings of the International Conference on Neural Networks*, volume I, pages 117–124. New York: IEEE Press, 1988.

[59] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. G. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–261, 1989.

[60] B. Eisenberg and R. L. Rivest. On the sample complexity of PAC learning using random and chosen examples. In *Proceedings of the* 3^{rd} *ACM Conference on Computational Learning Theory*, pages 154–162. San Mateo: Morgan Kaufmann, 1990.

[61] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky et al., editors, *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, 1988.

[62] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 524–532, Denver, 1989. San Mateo: Morgan Kaufmann, 1990.

[63] L. V. Fausett. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice–Hall, New Jersey, 1994.

[64] P. Floréen and P. Orponen. On the computational complexity of analyzing Hopfield nets. *Complex Systems*, 3:577–587, 1989.

[65] P. Floréen and P. Orponen. Attraction radii in Hopfield nets are hard to compute. *Neural Computation*, 5:812–821, 1993.

[66] P. Floréen and P. Orponen. Complexity issues in discrete Hopfield networks. Research Report A–1994–4, Department of Computer Science, University of Helsinki, 1994.

[67] M. A. Franzini. Speech recognition with back propagation. In *Proceedings of the Ninth Annual Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1702–1703, Boston, 1987. New York: IEEE Press.

[68] M. Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2:198–209, 1990.

[69] K. Fukushima. Cognitron: A self-organizing multi-layered neural network. *Biological Cybernetics*, 20:121–136, 1975.

[70] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[71] A. R. Gallant and H. White. There exist a neural network that does not make avoidable mistakes. In *Proc. of the IEEE Second Conference on Neural Networks*, volume I, pages 657–664. SOS Printing, 1988.

[72] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[73] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[74] T. Geszti. *Physical Models of Neural Networks*. World Scientific, Singapore, 1990.

[75] F. Girosi and T. Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1:465–469, 1989.

[76] R. J. Glauber. Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4:294–307, 1963.

[77] G. H. Godbeer, J. Lipscomb, and M. Luby. On the computational complexity of finding stable state vectors in connectionist models (Hopfield nets). Technical Report 208/88, Department of Computer Science, University of Toronto, 1988.

[78] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison–Wesley, New York, 1989.

[79] P. Goldberg and M. Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the* 6^{th} *ACM Conference on Computational Learning Theory*, pages 361–369, Santa Cruz, 1993. ACM Press.

[80] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[81] L. M. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of Boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

[82] E. Goles-Chacc, F. Fogelman-Soulié, and D. Pellegrin. Decreasing energy functions as a tool for studying threshold networks. *Discrete Applied Mathematics*, 12:261–277, 1985.

[83] E. Goles-Chacc and S. Martínez. Exponential transient classes of symmetric neural networks for synchronous and sequential updating. *Complex Systems*, 3:589–597, 1989.

[84] E. Goles-Chacc and J. Olivos. The convergence of symmetric threshold automata. *Information and Control*, 51:98–104, 1981.

[85] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, 1984.

[86] T. Greville. Some applications of the pseudoinverse of a matrix. *SIAM Review*, 2:15–22, 1960.

[87] S. Grossberg. Pattern learning by functional-differential neural networks with arbitrary path weights. In K. Schmitt, editor, *Delay and functional differential equations and their applications*, pages 121–160. Academic Press, New York, 1972.

[88] S. Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[89] S. Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*. Reidel Press, Boston, 1982.

[90] B. Grunbaum. *Convex Polytopes*. John Wiley, London, 1967.

[91] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proceedings of the* 28^{th} *Annual Symposium on Foundations of Computer Science*, pages 99–110. IEEE Computer Society Press, 1987.

[92] A. Haken. Connectionist networks that need exponential time to stabilize. Department of Computer Science, University of Toronto, 1989.

[93] S. J. Hanson and L. Pratt. A comparison of different biases for minimal network construction with back-propagation. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 177–185, Denver, 1988. San Mateo: Morgan Kaufmann, 1989.

[94] R. Hartley and H. Szu. A comparison of the computational power of neural network models. In *Proceedings of the IEEE First International Conference on Neural Networks*, pages 15–22, San Diego, 1987. New York: IEEE Press.

[95] E. Hartman and J. D. Keeler. Predicting the future: Advantages of semilocal units. *Neural Computation*, 3:566–578, 1991.

[96] J. Håstad. On the size of weights for threshold gates. *SIAM Journal Discrete Mathematics*, 7:484–492, 1994.

[97] S. Haykin. *Neural Networks*. Macmillan College Publishing Company, New York, 1994.

[98] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.

[99] T. Hegedüs. Can complexity theory benefit from learning theory? In *Proceedings of the European Conference on Machine Learning*, volume 667 of *LNAI*, pages 354–359, Vienna, 1993. Berlin: Springer-Verlag.

[100] T. Hegedüs. Computational learning theory and neural networks: An introduction. In M. Bartošek, editor, *Proceedings of the SOFSEM winter school*, pages 77–95, Milovy, 1994.

[101] T. Hegedüs. On training simple neural networks and small-weight neurons. In J. Shawe-Taylor and M. Anthony, editors, *Proceedings of the* 1^{st} *Euro-Colt Conference*, pages 69–82, London, 1993. Oxford: Clarendon Press, 1994.

[102] T. Hegedüs and N. Megiddo. On the geometric separability of Boolean functions. IBM Research Report RJ–9147 (81262), IBM Almaden Research Center, San Jose, 1992.

[103] R. Hecht-Nielsen. Counterpopagation networks. *Applied Optics*, 26:4979–4984, 1987.

[104] R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the International Conference on Neural Networks*, volume 3, pages 11–14. IEEE Press, New York, 1987.

[105] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, California, 1990.

[106] R. Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[107] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Iber. Deutsch. Math. Verein*, 32:175–176, 1923.

[108] M. Hermo. On Kobayashi's compressibility of infinite sequences. Research Report LSI–93–36–R, Universitat Politècnica de Catalunya, 1993.

[109] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*, volume I of *Lecture Notes, Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley, California, 1991.

[110] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst, 1986. Hillsdale: Erlbaum.

[111] G. E. Hinton and T. J. Sejnowksi. Optimal perceptual inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 448–453, Washington, 1983. New York: IEEE Press.

[112] G. E. Hinton and T. J. Sejnowksi. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume I, pages 282–317. MIT Press, Cambridge MA, 1986.

[113] A. Hirose. Continuous complex-valued back-propagation learning. *Electronics Letters*, 20:1854–1855, 1992b.

[114] K. Hlaváčková and R. Neruda. Radial basis function networks. *Neural Network World*, 3(1):93–101, 1993.

[115] M. Hoehfeld and S. E. Fahlmann. Learning with limited numerical precision using the cascade-correaltion algorithm. Technical Report CMU–CS–91–130, Carnegie Mellon University, Pittsburgh, PA, 1991.

[116] K.-U. Höffgen. Computational limitations on training sigmoid neural networks. *Information Processing Letters*, 46:269–274, 1993.

[117] T. Hofmeister, W. Hohberg, and S. Köhling. Some notes on threshold circuits and multiplication in depth 4. *Information Processing Letters*, 39:219–226, 1991.

[118] J. H. Holland. *Adaptation in Natural and Adaptation Systems.* University of Michigan Press, Ann Arbor, 1975.

[119] J. H. Holland. *Hidden Order.* Addison-Wesley, Reading, MA, 1995.

[120] J. Hong. *Computation: Computability, Similarity and Duality.* Pitman Publishing, London, 1986.

[121] J. Hong. On connectionist models. Technical Report 87–012, Department of Computer Science, University of Chicago, 1987.

[122] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, pages 2554–2558, 1982.

[123] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Sciences*, volume 81, pages 3088–3092, 1984.

[124] J. J. Hopfield, D. I. Feinstein, and R. G. Palmer. "Unlearning" has a stabilizing effect in collective memories. *Nature*, 304:158–159, 1983.

[125] J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[126] B. G. Horne and D. R. Hush. On the node complexity of neural networks. *Neural Networks*, 7:1413–1426, 1994.

[127] K. Hornik. Approximation capabilities of multilayer feedfoward networks. *Neural Networks*, 4(2):251–257, 1991.

[128] K. Hornik, M. Stinchcombe, and H. White. Multi-layer feedfoward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[129] J. Hořejš. A view on neural networks paradigms development. *An ongoing survey starting in Neural Network World*, 1:61–64, 1991.

[130] J. Hořejš and O. Kufudaki. Počítače a mozek (neuropočítače). In *Sborník semináře SOFSEM*, Beskydy, 1988.

[131] A. K. Chandra, L. J. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13:423–439, 1984.

[132] Y. Chauvin. A back-propagation algorithm with optimal use of hidden units. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 519–526, Denver, 1988. San Mateo: Morgan Kaufmann, 1989.

[133] A. M. Chen, L. Haw-minn, and R. Hecht-Nielsen. On the geometry of feedforward neural network error spaces. *Neural Computation*, 5(6):910–927, 1993.

[134] A. M. Chen and R. Hecht-Nielsen. On the geometry of feedforward neural network weights spaces. In *Proceedings of the 2nd IEE Conference on Artificial Neural Networks*, pages 1–4. IEE Press, London, 1991.

[135] S. Chen, C. F. N. Cowan , and P. M. Grant Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural networks*, 2(2), March 1991.

[136] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[137] M. Chytil. *Automaty a gramatiky*, volume 19 of *Matematický seminář*. SNTL, Praha, 1984.

[138] N. Immerman and S. Landau. The complexity of iterated multiplication. In *Proceedings of the* 4^{th} *IEEE Structure in Complexity Theory Conference*, pages 104–111. IEEE Press, 1989.

[139] P. Indyk. Optimal simulation of automata by neural nets. In *Proceedings of the* 12^{th} *Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *LNCS*, pages 337–348. Berlin: Springer-Verlag, 1995.

[140] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[141] T. Jiang and R. Ravikumar. A note on the space complexity of some decision problems for finite automata. *Information Processing Letters*, 40:25–31, 1991.

[142] N. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11:68–85, 1975.

[143] J. S. Judd. *Neural Network Design and the Complexity of Learning.* MIT Press, Cambridge MA, 1990.

[144] P. Kainen, V. Kůrková, V. M. Kreinovich, and O. Sirisengtaksin. A new criterion for chooseing an activation function: Uniqueness leads to faster learning. *Neural Parallel and Scientific Computations*, 1993. in press.

[145] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[146] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[147] M. Karpinski and A. J. MacIntyre. Polynomial bounds for VC dimension of sigmoidal neural networks. In *Proceedings of the* 27^{th} *Annual ACM Symposium on the Theory of Computing*, pages 200–208. Las Vegas: ACM Press, 1995.

[148] J. Kilian and H. T. Siegelmann. Computability with the classical sigmoid. In *Proceedings of the* 6^{th} *ACM Conference on Computational Learning Theory*, pages 137–143, Santa Cruz, 1993.

[149] S. Kirkpatrick, C. D. Jr. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[150] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 3–41. Princeton University Press, Princeton NJ, 1956.

[151] T. Kohonen. Correlation associative memory. *IEEE Transactions on Computers*, C-21:353–359, 1972.

[152] T. Kohonen. *Associative Memory — A System Theoretical Approach.* Springer-Verlag, New York, 1977.

[153] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[154] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1984.

[155] T. Kohonen. Learning vector quantization. *Neural Networks*, 1:303, 1988.

[156] T. Kohonen. The self-organizing map. In *Proc. of the IEEE*, volume 78. September 1990.

[157] T. Kohonen. *Self-Organizing Map.* Number 30 in Springer Series in Information Sciences. Springer-Verlag, Berlin, 1995.

[158] T. Kohonen and K. Ruohonen. Representation of associated data by matrix operations. *IEEE Transactions on Computers*, C-22:701–702, 1973.

[159] P. Koiran and E. Sontag. Neural networks with quadratic VC dimension. NeuroCOLT Technical Report Series NC–TR–95–044, 1995.

[160] A. N. Kolmogorov. On the representations of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, 114:953–956, 1957.

[161] J. Komlós and R. Paturi. Convergence results in an associative memory model. *Neural Networks*, 1:239–250, 1988.

[162] A. H. Kramer and A. Sangiovanni-Vincentelli. Efficient parallel learning algorithms for neural networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 40–48, Denver, 1988. San Mateo: Morgan Kaufmann, 1989.

[163] V. Kůrková. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5:501–506, 1992.

[164] V. Kůrková. Universal approximation using feedforward neural networks with gaussian bar units. In *Proceedings of the ECAI'92*, pages 193–197. Wiley, 1992.

[165] V. Kůrková and P. Kainen. Functionally equivalent feedforward networks. *Neural Computation*, 6:543–558, 1993.

[166] V. Kůrková and R. Neruda. Uniqueness of the functional representations for the gaussian basis functions. In *Proceedings of the ICANN'94*, pages 474–477. London: Springer, 1994.

[167] R. E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7:18–20, 1975.

[168] K. J. Lang and E. B. Baum. Query learning can work poorly when a human oracle is used. In *Proceedings of the International Joint Conference on Neural Networks*, Beijing, 1992.

[169] M. Lepley and G. Miller. Computational power for networks of threshold devices in an asynchronous environment. Technical report, Department of Mathematics, MIT, 1983.

[170] M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.

[171] P. M. Lewis. A lower bound on the number of corrections required for convergence of the single threshold gate adaptive procedure. *IEEE Transactions on Electronic Computers*, C-15:933–935, 1966.

[172] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, New York, 1993.

[173] J.-H. Lin and J. S. Vitter. Complexity results on learning by neural nets. *Machine Learning*, 6:211–230, 1991.

[174] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communication*, COM-28:84–95, 1980.

[175] R. P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4:4–22, 1987.

[176] W. A. Little and G. L. Shaw. Analytical study of the memory storage capacity of a neural network. *Mathematical Biosciences*, 39:281–290, 1978.

[177] G. G. Lorentz. *Approximation of function.* Halt, Reinhart, and Winston, New York, 1966.

[178] L. Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the* 4^{th} *Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12. Winnipeg: Utilitas Mathematica Publishing, 1973.

[179] L. Lukšan. *Metody s proměnnou metrikou.* Academia, Praha, 1990.

[180] O. B. Lupanov. Implementing the algebra of logic functions in terms of bounded depth formulas in the basis $+, \star, -$. *Soviet Physics Doklady*, 6:107–108, 1961.

[181] O. B. Lupanov. Circuits using threshold elements. *Soviet Physics Doklady*, 17:91–93, 1972.

[182] W. Maass. Neural nets with superlinear VC-dimension. *Neural Computation*, 6:877–884, 1994.

[183] W. Maass, G. Schnitger, and E. D. Sontag. On the computational power of sigmoid versus Boolean threshold circuits. In *Proceedings of the* 32^{nd} *Annual Symposium on Foundations of Computer Science*, pages 767–776. IEEE Computer Society Press, 1991.

[184] W. Maass and G. Turán. On the complexity of learning from counterexamples. In *Proceedings of the* 30^{th} *Annual Symposium on Foundations of Computer Science*, pages 262–267. Los Angeles: IEEE Computer Society Press, 1989.

[185] A. J. MacIntyre and E. D. Sontag. Finiteness results for sigmoidal "neural" networks. In *Proceedings of the* 25^{th} *Annual ACM Symposium on the Theory of Computing*, pages 325–334, San Diego, 1993. New York: ACM Press.

[186] C. von der Malsburg. Self-organizing of orientation sensitive cells in striate cortex. *Kybernetik*, 14:85–100, 1973.

[187] S. Makram-Ebeid, J.-A. Sirat, and J.-R. Viala. A rationalized back–propagation learning algorithm. In *Proceedings of the International Joint Conference on Neural Networks*, volume II, pages 373–380, Washington, 1989. New York: IEEE Press.

[188] M. Marchand, M. Golea, and P. Ruján. A convergence theorem for sequential learning in two-layer perceptrons. *Europhysics Letters*, 11:487–492, 1990.

[189] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[190] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh. The capacity of the Hopfield associative memory. *IEEE Transactions on Information Theory*, 33:461–482, 1987.

[191] N. Megiddo. On the complexity of polyhedral separability. *Discrete Computational Geometry*, 3:325–337, 1988.

[192] M. Mézard and J.-P. Nadal. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A*, 22:2191–2204, 1989.

[193] H. N. Mhaskar and C. A. Micchelli. Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics*, 13:350–373, 1992.

[194] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Construct. Approx.*, 2:11–22, 1986.

[195] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, Berlin, second, extended edition, 1994.

[196] M. L. Minsky. *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem*. Ph. D. thesis, Princeton University, Princeton NJ, 1954.

[197] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge MA, 1969.

[198] J. Moody and C. Darken. Learning with localized receptive fields. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the Connectionist Models Summer School*. San Mateo: Morgan Kaufmann, 1989, 1988.

[199] J. Moody and C. Darken. Fast adaptive k-means clustering: some empirical results. In *Proc. IJCNN, San Diego '90*, volume 2, pages 233–238, 1990.

[200] S. Muroga. *Threshold Logic and its Applications*. Wiley–Interscience, New York, 1971.

[201] S. Muroga, I. Toda, and S. Takasu. Theory of majority decision elements. *J. Franklin Inst.*, 271:376–418, 1961.

[202] S. Muroga, T. Tsuboi, and C. R. Baugh. Enumeration of threshold functions of eight variables. *IEEE Transactions on Computers*, EC-19:818–825, 1970.

[203] B. K. Natarajan. On learning Boolean functions. In *Proceedings of the* 19^{th} *Annual ACM Symposium on the Theory of Computing*, pages 296–304. ACM Press, 1987.

[204] E. I. Nečiporuk. The synthesis of networks from threshold elements. *Problemy Kibernetiki*, 11:49–62, 1964.

[205] R. Neruda and A. Štědrý. Approximation capabilities of chain architectures. In *Proceedings of the ICANN'95*, volume I, pages 575–580. Paris: EC2 & Cie, 1995.

[206] J. von Neumann. The general and logical theory of automata. In L. A. Jeffress, editor, *Cerebral Mechanisms in Behavior*, pages 1–41. Wiley, New York, 1951.

[207] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 43–98. Princeton University Press, Princeton NJ, 1956.

[208] N. J. Nilsson. *Learning Machines*. McGraw-Hill, New York, 1965.

[209] P. Orponen. On the computational power of discrete Hopfield nets. In *Proceedings of the* 20^{th} *International Colloquium on Automata, Languages, and Programming*, volume 700 of *LNCS*, pages 215–226. Berlin: Springer-Verlag, 1993.

[210] P. Orponen. Computational complexity of neural networks: A survey. *Nordic Journal of Computing*, 1:94–110, 1994.

[211] P. Orponen. Computing with truly asynchronous threshold logic networks. *To appear in Theoretical Computer Science*, 178, 1997.

[212] I. Parberry. A primer on the complexity theory of neural networks. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Sourcebook*, pages 217–268. Elsevier, North-Holland, Amsterdam, 1990.

[213] I. Parberry. *Circuit Complexity and Neural Networks*. MIT Press, Cambridge MA, 1994.

[214] I. Parberry and G. Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36:278–302, 1988.

[215] I. Parberry and G. Schnitger. Relating Boltzmann machines to conventional models of computation. *Neural Networks*, 2:59–67, 1989.

[216] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

[217] J. Park and I. W. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 5:305–316, 1993.

[218] D. B. Parker. Learning-logic: Casting the cortex of the human brain in silicon. Technical Report TR–47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, 1985.

[219] E. Pelikán. On a neural network approach to the detection of characteristic events in signals. In *Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1251–1252, Orlando, Florida, 1991.

[220] P. Peretto. On learning rules and memory storage abilities of asymmetrical neural networks. *Journal de Physique Lettres*, 49:711–726, Paris, 1988.

[221] F. J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232, 1987.

[222] F. J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216–245, 1988.

[223] F. J. Pineda. Recurrent back-propagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1:161–172, 1989.

[224] N. Pippenger. On simultaneous resource bounds. In *Proceedings of the* 20^{th} *Annual Symposium on Foundations of Computer Science*, pages 307–311. IEEE Computer Society Press, 1979.

[225] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35:965–984, 1988.

[226] D. Plaut, S. Nowlan, and G. Hinton. Experiments on learning by back propagation. Technical Report CMU–CS–86–126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1986.

[227] T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78. September 1990.

[228] S. Poljak and M. Sura. On periodical behaviour in societies with symmetric influences. *Combinatorica*, 3:119–121, 1983.

[229] S. Porat. Stability and looping in connectionist models with asymmetric weights. *Biological Cybernetics*, 60:335–344, 1989.

[230] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J.C. Mason and M.G.Cox, editors, *Algorithms for Approximation*, pages 143–167. Oxford Science Publications, 1987.

[231] M. J. D. Powell. The theory of radial basis function approximation in 1990. In W. Light, editor, *Advances in Numerical Analysis*, pages 105–210. Oxford Science Publications, 1992.

[232] N. J. Redding, A. Kowalczyk, and T. Downs. Constructive higher-order network algorithm that is polynomial time. *Neural Networks*, 6:997–1010, 1993.

[233] N. P. Redkin. Synthesis of threshold element networks for certain classes of Boolean functions. *Kibernetika*, 5:6–9, 1970.

[234] J. H. Reif and S. R. Tate. On threshold circuits and polynomial computation. *SIAM Journal on Computing*, 21:896–908, 1992.

[235] F. Riesz and B. Sz-Nagy. *Functional Analysis*. Dover Publications Inc., New York, 1990.

[236] J. Riordan and C. E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21:83–93, 1942.

[237] R. Rohwer and B. Forrest. Training time-dependence in neural networks. In M. Caudill and C. Butler, editors, *Proceedings of the IEEE First International Conference on Neural Networks*, volume II, pages 701–708, San Diego, 1987. New York: IEEE Press.

[238] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[239] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, Washington DC, 1962.

[240] A. Roy, L. S. Kim, and S. Mukhopadhyay. A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. *Neural Networks*, 6:535–545, 1993.

[241] V. Roychowdhury, K.-Y. Siu, and A. Orlitsky. *Theoretical Advances in Neural Computation and Learning*. Kluwer Academic Publishers, MA, 1994.

[242] W. Rudin. *Functional Analysis*. McGraw-Hill, New York, 1973.

[243] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume I, pages 318–362. MIT Press, Cambridge MA, 1986.

[244] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition I & II*. MIT Press, Cambridge MA, 1986.

[245] A. Sakurai. Tighter bounds of the VC-dimension of three layer networks. In *Proceedings of World Congress on Neural Networks*, volume 3, pages 540–543, 1993.

[246] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

[247] J. E. Savage. Computational work and time on finite machines. *Journal of the ACM*, 19:660–674, 1972.

[248] R. Scalettar and A. Zee. Emergence of grandmother memory in feedforward networks: Learning with noise and forgetfulness. In D. Waltz and J. A. Feldman, editors, *Connectionist Models and Their Implications: Readings from Cognitive Science*, pages 309–332. Ablex, Norwood, 1988.

[249] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[250] C. E. Shannon. A universal Turing machine with two internal states. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 157–165. Princeton University Press, Princeton NJ, 1956.

[251] J. F. Shepanski and S. A. Macy. Teaching artificial neural systems to drive: Manual training techniques for autonomous systems. In D. Z. Anderson, editor, *Proceedings of the Neural Information Processing Systems Conference*, pages 693–700. New York: American Institute of Physics, 1988, 1987.

[252] M. Schmitt. Lower bounds on identification criteria for perceptron-like learning rules. NeuroCOLT Technical Report Series NC–TR–96–024, 1996.

[253] H. T. Siegelmann. On the computational power of probabilistic and faulty neural networks. In S. Abiteboul and E. Shamir, editors, *Proceedings of the* 21^{st} *International Colloquium on Automata, Languages, and Programming*, volume 820 of *LNCS*, pages 23–34. Berlin: Springer-Verlag, 1994.

[254] H. T. Siegelmann and E. D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131:331–360, 1994.

[255] H. T. Siegelmann and E. D. Sontag. Computational power of neural networks. *Journal of Computer System Sciences*, 50:132–150, 1995.

[256] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*. Pergamon Press, New York, 1990.

[257] M. Sipser. Borel sets and circuit complexity. In *Proceedings of the* 15^{th} *Annual ACM Symposium on the Theory of Computing*, pages 61–69. ACM Press, 1983.

[258] K.-Y. Siu, V. P. Roychowdhury, and T. Kailath. Computing with almost optimal size threshold circuits. In *Proceedings of the International Symposium on Information Theory*, Budapest, 1991.

[259] K.-Y. Siu, V. P. Roychowdhury, and T. Kailath. Depth–size tradeoffs for neural computation. *IEEE Transactions on Computers*, 40:1402–1412, 1991.

[260] D. A. Spielman. Computing arbitrary symmetric functions. Technical Report TR–906, Department of Computer Science, Yale University, 1992.

[261] D. A. Sprecher. On the structure of continuous of several variables. *Transactions of the American Mathematical Society*, 115:340–355, 1965.

[262] H. J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–594, 1992.

[263] J. Šíma. Loading deep networks is hard. *Neural Computation*, 6:842–850, 1994.

[264] J. Šíma. Hopfield languages. In M. Bartošek, J. Staudek, and J. Wiedermann, editors, *Proceedings of the SOFSEM Seminar on Current Trends in Theory and Practice of Informatics*, volume 1012 of *LNCS*, pages 461–468, Milovy, 1995. Berlin: Springer-Verlag.

[265] J. Šíma. Neural expert systems. *Neural Networks*, 8:261–271, 1995.

[266] J. Šíma. Back-propagation is not efficient. *Neural Networks*, 9:1017–1023, 1996.

[267] J. Šíma and J. Wiedermann. Neural language acceptors. In *Proceedings of the 2^{nd} International Conference Developments in Language Theory*, pages 430–439, Magdeburg, 1995. Singapore: World Scientific, 1996.

[268] J. Šíma and J. Wiedermann. Theory of neuromata. 1996.

[269] F. Tanaka and S. F. Edwards. Analytic theory of the ground state properties of a spin glass: I. Ising spin glass. *Journal of Physics F*, 10:2769–2778, 1980.

[270] G. Tesauro. Scaling relationships in back-propagation learning: Dependence on training set size. *Complex Systems*, 1:367–372, 1987.

[271] G. Tesauro and B. Janssens. Scaling relationships in back-propagation learning. *Complex Systems*, 2:39–44, 1988.

[272] A. J. Ticknor and H. Barrett. Optical implementations of Boltzmann machines. *Optical Engineering*, 26:16–21, 1987.

[273] V. V. Tolat and B. Widrow. An adaptive 'broom balancer' with visual inputs. In *Proceedings of the International Conference on Neural Networks*, volume II, pages 641–647. New York: IEEE Press, 1988.

[274] G. Turán. Lower bounds for *PAC* learning with queries. In *Proceedings of the 6^{th} ACM Conference on Computational Learning Theory*, pages 384–391, Santa Cruz, 1993. New York: ACM Press.

[275] L. G. Valiant. A theory of the learnable. *Communication of the ACM*, 27:1134–1142, 1984.

[276] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[277] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Application*, 16:264–280, 1971.

[278] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59:257–263, 1988.

[279] B. A. Vostrecov and M. A. Kreines. Approximation of continuous functions by superposition of plane waves. *Doklady Akademii Nauk SSSR*, 140(2), 1961.

[280] H. S. Wall. *Analytic Theory of Continued Fractions*. Van Nostrand, New York, 1948.

[281] W. Wee. Generalized inverse approach to adaptive multiclass pattern classification. *IEEE Transactions on Computers*, C-17:1157–1164, 1968.

[282] I. Wegener. *The Complexity of Boolean Functions*. Wiley–Teubner, Stuttgart, 1987.

[283] I. Wegener. Unbounded fan-in circuits. In *Advances in the Theory of Computation and Computational Mathematics*. 1990.

[284] G. Weisbuch and F. Fogelman-Soulié. Scaling laws for the attractors of Hopfield networks. *Journal de Physique Lettres*, 46:623–630, Paris, 1985.

[285] R. S. Wenocur and R. M. Dudley. Some special Vapnik-Chervonenkis classes. *Discrete Mathematics*, 33:313–318, 1981.

[286] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph. D. thesis, Applied Mathematics, Harvard University, Cambridge, MA, 1974.

[287] B. Widrow. Generalization and information storage in networks of ADALINE "neurons". In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems*, pages 435–461. Spartan Books, Washington DC, 1962.

[288] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, volume 4, pages 96–104. IRE, New York, 1960.

[289] J. Wiedermann. O výpočtovej sile neurónových sietí a príbuzných výpočtových systémov. In *Sborník semináře SOFSEM*, pages 73–78, Beskydy, 1988.

[290] J. Wiedermann. Complexity issues in discrete neurocomputing. *Neural Network World*, 4:99–119, 1994.

[291] H. Wiklicky. The neural network loading problem is undecidable. In J. Shawe-Taylor and M. Anthony, editors, *Proceedings of the 1^{st} Euro-Colt Conference*, pages 183–192, London, 1993. Oxford: Clarendon Press, 1994.

[292] D. J. Willshaw and C. von der Malsburg. How patterned neural connections can be set up by self-organization. *Proc. R. Soc. Lond. B*, 194:431–445, 1976.

[293] S. Yajima and T. Ibaraki. A lower bound on the number of threshold functions. *IEEE Transactions on Electronic Computers*, EC-14:926–929, 1965.