Kapitola 1

1.1 Rekurentné neurónové siete a rekurentný back-propagation algoritmus

Odvodíme pravidlá pre výpočet aktivačných hodnôt neurónov a adaptáciu váh prepojení, ktoré spolu tvoria algoritmus nazývaný **rekurentný back-propagation algoritmus**, ktorého autormi sú Pineda a Almeida (1987, 1988). Jeho názov vyplýva z toho, že je vlastne rozšírením klasického back-propagation algoritmu na rekurentné siete, alebo aj naopak, klasický back-propagation je špeciálnym prípadom rekurentného back-propagation algoritmu pre siete, kde nie sú povolené rekurentné prepojenia medzi neurónmi. Rekurentný back-propagation algoritmus je možné použiť na ľubovoľnú plne rekurentnú sieť so reálnym vstupom, pričom predpokladáme, že sieť vždy dosiahne stabilný stav.

Na obr. 1.1 vidíme rekurentnú sieť pozostávajúcu zo siedmich neurónov, vstupnými neurónmi neuróny 1, 2 a 3, výstupné sú neuróny 6 a 7.

Zavedieme najprv niekoľko pojmov a označení:

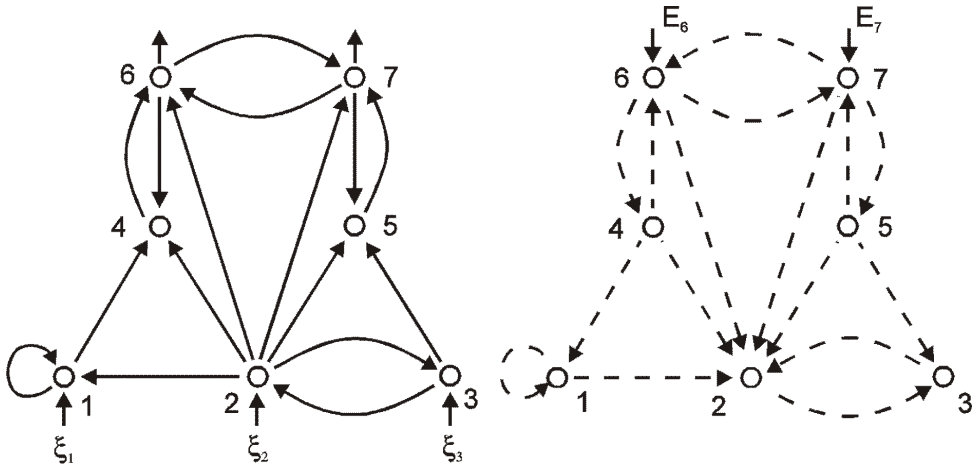
- Majme rekurentnú sieť pozostávajúcu z n neurónov s aktivačnými hodnotami $V_i, i = 1, \ldots, n$, každý z nich používa aktivačnú funkciu $g(x)$.
- Neuróny, ktoré prijímajú okrem signálov z ostatných neurónov aj vstupné signály z okolia, nazývame **vstupné neuróny**. Vonkajší vstup do i-teho neurónu označíme x_i. Pre neuróny, ktoré neprijímajú žiadne signály z okolia, definujeme pre jednoduchosť $x_i = 0$.
- Podobne **výstupné neuróny** majú hodnoty očakávané na výstupe označené ζ_i. Z každého neurónu vedú prepojenia do ostatných neurónov a doňho samotného. Prepojenia nemusia byť pritom symetrické.
- Váhu prepojenia z j-teho neurónu do i-teho neurónu označíme w_{ij}. V prípade, že v sieti dané prepojenie chýba, položíme $w_{ij} = 0$.

Rekurentný back-propagation algoritmus používa na výpočet aktivačných hodnôt jednotlivých neurónov pravidlo

$$V_i = g(h_i) = g\left(\sum_{j=0}^{n} w_{ij} V_j + x_i\right) \tag{1.1}$$

kde h_i je označenie pre celkový vstup do i-teho neurónu, $V_0 = -1, w_{i0}$ je prah.

Celkovú chybu na výstupe siete, ktorá vznikne porovnaním aktivačných hodnôt V_i výstupných neurónov a očakávaných hodnôt ζ_i, vyjadríme pomocou kvadratickej chybovej funkcie



Obr. 1.1: Rekurentná neurónová sieť a rekurentný back-propagation algoritmus

$$E = \frac{1}{2}\sum_{k=1}^{n} E_k^2 \tag{1.2}$$

kde

$$E_k = \begin{cases} \zeta_k - V_k & \text{ak k je výstupný neurón,} \\ 0 & \text{inak.} \end{cases} \tag{1.3}$$

Pomocou metódy najväčšieho spadu gradientu odvodíme pravidlo pre adaptáciu váh prepojení

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}} = -\eta \frac{\partial\left[\frac{1}{2}\sum_{k=1}^{n} {E_k}^2\right]}{\partial w_{pq}} = -\eta \cdot \tfrac{1}{2} \cdot 2 \sum_{k=1}^{n} E_k \frac{\partial E_k}{\partial w_{pq}} = \tag{1.4}$$

$$= -\eta \sum_{k=1}^{n} E_k \frac{\partial(\zeta_k - V_k)}{\partial w_{pq}} = \eta \sum_{k=1}^{n} E_k \frac{\partial V_k}{\partial w_{pq}} \tag{1.5}$$

Hodnotu E_k vypočítame jednoducho podľa (1.3) a na výpočet $\partial V_k / \partial w_{pq}$ použijeme rovnicu (1.1), z ktorej derivovaním získame

$$\frac{\partial V_i}{\partial w_{pq}} = \frac{\partial g(h_i)}{\partial w_{pq}} = g'(h_i)\frac{\partial h_i}{\partial w_{pq}} = g'(h_i)\frac{\partial[\sum_{j=1}^{n} w_{ij}V_j + x_i]}{\partial w_{pq}} = \tag{1.6}$$

$$= g'(h_i)\left[\delta_{ip}V_q + \sum_{j=1}^{n} w_{ij}\frac{\partial V_j}{\partial w_{pq}}\right] \tag{1.7}$$

kde δ_{ip} označuje tzv. *Kroneckerovo delta*, definované vzťahom

$$\delta_{ip} = \begin{cases} 1 & \text{ak } i = p, \\ 0 & \text{inak.} \end{cases} \tag{1.8}$$

Ak vo vzťahu (1.7) osamostatníme člen $\delta_{ip}g'(h_i)V_q$, dostaneme rovnicu

$$\begin{aligned} \delta_{ip}g'(h_i)V_q = & \frac{\partial V_i}{\partial w_{pq}} - \sum_{j=1}^{n} g'(h_i)w_{ij}\frac{\partial V_j}{\partial w_{pq}} = \\ = & \left[1 - g'(h_i)w_{ii}\right]\frac{\partial V_i}{\partial w_{pq}} + \sum_{j\neq i}\left[0 - g'(h_i)w_{ij}\right]\frac{\partial V_j}{\partial w_{pq}} = \\ = & \sum_{j=1}^{n}\left[\delta_{ij} - g'(h_i)w_{ij}\right]\frac{\partial V_j}{\partial w_{pq}} \end{aligned} \tag{1.9}$$

Definujme teraz maticu $\mathbf{L}$ typu $n \times n$ tak,že

$$L_{ij} = \delta_{ij} - g'(h_i)w_{ij} \tag{1.10}$$

Vzťah (1.9) môžeme teda napísať v tvare

$$\delta_{ip} g'(h_i) V_q = \sum_{j=1}^{n} L_{ij} \frac{\partial V_j}{\partial w_{pq}} \tag{1.11}$$

Rozlíšime dva prípady

$$\begin{array}{ll} i \neq p \Rightarrow \delta_{ip} = 0 & \text{a teda } \sum_{j=1}^{n} L_{ij} \frac{\partial V_j}{\partial w_{pq}} = 0 \\ i = p \Rightarrow \delta_{pp} = 1 & \text{a teda } \sum_{j=1}^{n} L_{pj} \frac{\partial V_j}{\partial w_{pq}} = g'(h_p) V_q \end{array} \tag{1.12}$$

Označme $\mathbf{V}$ maticu typu $n \times 1$

$$\mathbf{V}^T = \left(\frac{\partial V_1}{\partial w_{pq}}, \ldots, \frac{\partial V_n}{\partial w_{pq}} \right) \tag{1.13}$$

a $\mathbf{P}$ maticu typu $1 \times n$

$$\mathbf{P} = (L_{p1}, \ldots, L_{pn}) \tag{1.14}$$

Rovnica (1.12) má teda v maticovom zápise tvar

$$\mathbf{P}.\mathbf{V} = g'(h_p) V_q \tag{1.15}$$

Ak obidve strany vynásobíme zľava inverznou maticou $\mathbf{P}^{-1}$, dostaneme

$$\mathbf{V} = \mathbf{P}^{-1} g'(h_p) V_q \tag{1.16}$$

čo rozpísané po zložkách matice $\mathbf{V}$ dáva

$$\frac{\partial V_k}{\partial w_{pq}} = P_{1k}^{-1} g'(h_p) V_q \tag{1.17}$$

Ak si teraz uvedomíme, že pre každé $k = 1 \ldots n$ platí $P_{1k}^{-1} = L_{kp}^{-1}$, upravíme teraz rovnicu (1.17) do konečnej podoby

$$\frac{\partial V_k}{\partial w_{pq}} = L_{kp}^{-1} g'(h_p) V_q \tag{1.18}$$

Dosadením tohto vzťahu do pravidla pre adaptáciu váh (1.5) dostávame

$$\Delta w_{pq} = \eta \, g'(h_p) \sum_{k=1}^{n} E_k L_{kp}^{-1} V_q \tag{1.19}$$

Vzťah (1.19) je naším novým učiacim pravidlom. Môžeme ho vyjadriť aj v tvare

$$\Delta w_{pq} = \eta \delta_p V_q \tag{1.20}$$

pričom δ_p má v tomto prípade iný význam ako Kroneckerovo delta, platí

$$\delta_p = g'(h_p) \sum_{k=1}^{n} E_k {L^{-1}}_{kp} \tag{1.21}$$

Z pravidla (1.19) vidíme, že na výpočet prírastkov váh prepojení je potrebné v každom časovom kroku vypočítať inverznú maticu $\mathbf{L}^{-1}$, čo však vedie k väčším časovým a pamäťovým

nárokom algoritmu. Ukážeme si však, ako sa dá tento nedostatok aspoň čiastočne odstrániť. Ak zavedieme označenie

$$Y_p = \sum_{k=1}^{n} E_k L_{kp}^{-1} \tag{1.22}$$

alebo v maticovom zápise

$$\mathbf{Y} = \mathbf{E}.\mathbf{L}^{-1} \tag{1.23}$$

kde

$$\mathbf{Y} = (Y_1, \ldots, Y_n) \tag{1.24}$$

a

$$\mathbf{E} = (E_1, \ldots, E_n) \tag{1.25}$$

tak rovnicu (1.21) pre δ_p môžeme napísať v tvare

$$\delta_p = g'(h_p) Y_p \tag{1.26}$$

Ak invertujeme späť maticu $\mathbf{L}^{-1}$ v rovnici (1.23) dostaneme

$$\mathbf{Y}.\mathbf{L} = \mathbf{E} \tag{1.27}$$

Pre každé $i = 1 \ldots n$ teda platí

$$E_i = \sum_{p=1}^{n} Y_p L_{pi} \tag{1.28}$$

a po dosadení vzťahu (1.10) pre L_{pi}

$$E_i = Y_i - \sum_{p=1}^{n} g'(h_p) w_{pi} Y_p \tag{1.29}$$

Osamostatnením Y_i dostávame pravidlo pre výpočet Y_i

$$Y_i = \sum_{p=1}^{n} g'(h_p) w_{pi} Y_p + E_i \tag{1.30}$$

Ak porovnáme vzťahy (1.30) a (1.1), zistíme, že majú podobný tvar. To vedie k myšlienke, že hodnoty Y_i môže počítať sieť s rovnakou topológiou ako má pôvodná sieť, pričom všetky prepojenia budú mať opačný smer. Sieť počítajúca Y_i má teda n neurónov s prepojeniami, ktorých váhy v každom časovom kroku majú hodnoty $\overline{w}_{ip} = g'(h_p) w_{pi}$. Všetky neuróny sa riadia aktivačnou funkciou $\overline{g}(x) = x$ a do siete vstupujú hodnoty E_i - chyby pôvodnej siete.

Na obr. 1.1 vidíme rekurentnú sieť pozostávajúcu zo siedmich neurónov a k nej prislúchajúcu sieť propagujúcu chybu. V pôvodnej sieti sú vstupnými neurónmi neuróny 1, 2 a 3, výstupné sú neuróny 6 a 7. V sieti propagujúcej chybu sú neuróny 6 a 7 naopak vstupom a výstup tvoria vlastne všetky neuróny siete.

Ak porovnáme pôvodný algoritmus a jeho modifikáciu, ktorú sme práve uviedli, vidíme, že pre sieť pozostávajúcu z n neurónov výpočet inverznej matice $\mathbf{L}^{-1}$ použitím triviálneho algoritmu má časovú zložitosť rádovo $O(n^3)$, zatiaľ čo výpočet hodnôt Y_i trvá rádovo $O(n^2)$. To vedie k veľkej úspore času najmä pre siete s veľkým počtom neurónov. Ďalšou prednosťou použitia dvoch sietí - jednej na výpočet výstupných hodnôt a druhej na výpočet hodnôt Y_i potrebných pre adaptáciu váh pôvodnej siete - je fakt, že táto metóda vylučuje použitie nelokálnej operácie, akou je výpočet inverznej matice $\mathbf{L}^{-1}$.

1.1.1 Rekurentný Back-Propagation algoritmus

1. Inicializujeme váhy w_{ij} na malé náhodné hodnoty. Aktivačné hodnoty neurónov (okrem vstupných) vlastnej siete nastavíme na 0. Aktivačné hodnoty neurónov (okrem výstupných) chybu propagujúcej siete nastavíme na 0.

2. Na vstup pôvodnej siete prezentujeme tréningový príklad a vypočítame aktivačné hodnoty všetkých neurónov podľa pravidla

$$V_i = g(h_i) = g\left(\sum_{j=1}^{n} w_{ij} V_j + x_i\right)$$

3. Podľa hodnôt výstupných neurónov vypočítame chybu pôvodnej siete

$$E_i = \zeta_i - V_i$$

4. Nastavíme váhy siete propagujúcej chybu

$$\overline{w}_{ij} = g'(h_j) w_{ji}$$

5. Hodnoty E_i prezentujeme na vstup siete propagujúcej chybu a vypočítame výstupy Y_i použitím

$$Y_i = \overline{g}\left(\overline{h}_i\right) = \overline{g}\left(\sum_{j=1}^{n} \overline{w}_{ij} Y_j + E_i\right)$$

kde

$$\overline{g}(x) = x$$

6. Použitím hodnôt Y_i vypočítame prírastky váh prepojení v pôvodnej sieti

$$\Delta w_{ij} = \eta \delta_i V_j$$

kde

$$\delta_i = g'(h_i) Y_i$$

7. Adaptujeme váhy použitím

$$w_{ij}^{nov} = w_{ij}^{sta} + \Delta w_{ij}$$

8. Vyberieme nový tréningový príklad a postup opakujeme prechodom na 2. krok

1.2 Literatúra

[1] **J. Hertz, A. Krogh, R. G. Palmer:**
Introductin to the theory of neural computation
Addison-Wesley, Redwood City, California, 1991

[2] **S. Haykin:**
Neural Networks
Macmillan College Publishing Company, New York, 1994

[3] **W. Kinnebrock:**
Neuronale Netze
Oldenbourg Verlag Munchen Wien, 1994

[4] **L. Wu, M. Niranjan:**
Building Energy Data Analysis and Prediction with Recurrent Neural Nets
CUED/F-INFENG/TR 124, Cambridge, 1993