Neuromata

Contents

1 Computational power of neural networks

Even if neural networks are computational models motivated by ideas about brain functioning, their computational power and efficiency is studied in the framework of computer science. It means to compare the computational power of neural networks to the traditional models of computation, such as finite automata and Turing machines. In this chapter the neural networks will work as neural acceptors of languages (over binary alphabet). There are known two types of input protocol. By finite neural networks, it is only one neuron as an input neuron for an input word (the input neuron reads sequentially bits of the input word). For this type of the input protocol there are considered discrete or analog model of network. In the second type of protocol, an sequence of neural networks exists, for each length of the input there is exactly one neural network with the competent number of neurons. In both cases, the state of output neuron specifies if the input word belongs to the language or not. We consider first type of input protocol and show some theoretical results from this point of view.

2 Automata and regular languages

We recall some basic notion from the automata theory and the language theory on the base Hopcroft [?]. An alphabet is a finite set of symbols. A string over alphabet Σ is a finite-length sequence of symbol from Σ. The length of a string s, denoted $|s|$, is the total number of symbols in s. The empty string, denoted by ϵ, is the string with no symbols. If s and t are strings, then the concatenation of s and t is the string st.

Definition 1: A language over an alphabet Σ is a set of strings over Σ. Let L_1 and L_2 be two languages. The language $L_1.L_2$, called the concatenation of L_1 and L_2, is $\{st|s \in L_1, t \in L_2\}$. Let L be the language. Then define $L^0 = \{\epsilon\}$ and $L^n = L.L^{n-1}$ for $n \geq 1$. The iteration of L, denoted $L^\star$, is the language $L^\star = \bigcup_{n=0}^{\infty} L^n$. Similarly the positive iteration $L^+ = \bigcup_{n-1}^{\infty} L^n$. $\odot$

Definition 2: The (deterministic) finite automaton is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where $Q \neq \emptyset$ is a finite set of automaton states, Σ is an input alphabet (in our case $\Sigma = \{0, 1\}$, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state of the automaton, and $F \subseteq Q$ is a set of accepting states.

The generalized transition function $\delta^* : Q \times \Sigma^* \to Q$ of the automaton is defined in the following way:

1. $\delta^*(q, \epsilon) = q$ for $q \in Q$,
2. $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$ for $q \in Q, w \in \Sigma^*, a \in \Sigma$. $\odot$

It was shown by Alon that every m-state deterministic finite automaton can be realized as discrete neural net with $O(m^{\frac{3}{4}})$ neurons and that at least $\Omega((m \log m)^{\frac{1}{3}})$ neurons are necessary for such construction. This upper and lower bound was improved by showing that $O(m^{\frac{1}{2}})$ neurons suffice and the most of the finite automata require $\Omega(m^{\frac{1}{2}})$ neurons when the values of weights in the network are polynomial in the network size.

Definition 3: The set $\mathcal{RE}$ of regular expressions over an alphabet $\Sigma = \{0, 1\}$ is defined as the minimal language over an alphabet $\{0, 1, \emptyset, \epsilon, +, ., \star, (,)\}$ satisfying the following conditions:

1. $\emptyset, \epsilon, 0, 1 \in \mathcal{RE}$
2. if $\alpha, \beta \in \mathcal{RE}$ then $(\alpha + \beta), (\alpha.\beta), \alpha^\star \in \mathcal{RE}$.

The set $\mathcal{RL} = \{[\alpha] | \alpha \in \mathcal{RE}\}$ is the set of regular languages $[\alpha]$ which are denoted by regular expressions α as follows:

1. $[\emptyset] = \emptyset$, $[\epsilon] = \{\epsilon\}$, $[0] = \{0\}$, $[1] = \{1\}$,
2. if $\alpha, \beta \in \mathcal{RE}$ then $[\alpha + \beta] = [\alpha] \cup [\beta], [\alpha.\beta] = [\alpha].[\beta], [\alpha^\star] = [\alpha]^\star$.

The regular expression α^+ corresponding to $[\alpha]^+$ is define by following expression, $[\alpha^\star] = [\epsilon + \alpha^+]$. $\odot$

We remember known the result from Hopcroft and Ullman [?]: **A language L is regular iff it is recognised by some finite automaton $\mathcal{A}$.**

3 Simulation of finite automata by the neural networks

We introduce the construction with $2m + 1$ neurons as has been investigated by Alon and all. [1]. This construction is very simple and shows the very used method for next construction of neural networks.

3.1 The construction of neural networks

Let $\mathcal{A}$ is the finite automaton. We describe an architecture and weight values of the constructed neural network.

- **Architecture**
  Each state of $\mathcal{A}$ will be represented by two neurons. The state $q \in Q$ is represented by the neurons $(q, 0)$ and $(q, 1)$ and we built the neural network in such way that at time t neuron (q, i) fires if and only if the original automaton $\mathcal{A}$ at time t is in state q and receives input i. The construction is illustrated in Example 1. In addition to the input neuron and the $2m$ neurons obtained as above there is a $2m + 1$-th neuron called the output neuron.

- **Weight values**
  For any two states q_j and q_k, $w_{(q_j,i),(q_k,0)} = w_{(q_j,i),(q_k,1)} \in \{0, 1\}$, and this equals 1 if and only if in the original finite state machine state q_j with input i leads to the next state q_k. For inputs, $w_{0,(q,0)} = -1$ and $w_{0,(q,1)} = +1$ for all q. For outputs, $w_{(q,i),2m+1} = \tau(q, i)$ (τ is output function). All weights not yet mentioned are equal to zero. The thresholds are defined by $c_{(q,1)} = 2, c_{(q,0)} = 1$ and $c_{(2m+1)} = 1$. It is clear that if at step $t = 0$ exactly one of the neurons $(q, i), q \in Q, i = 0, 1$, fires then at any step $t \geq 1$ exactly one of those neurons fires and the dynamics is exactly the same as that for the finite automaton. $\odot$

Example 1.
Let $\mathcal{A}_1 = (Q, \Sigma, \delta, q_0, F)$ is the finite automaton to recognize the language which contains all words and only the words with the even number of zeros and the even number of ones. The automaton is in Figure 1. The automaton has four states: q_1, q_2, q_3, q_4. The initial state is q_1.

We describe the using of the states:

The automaton is situated in state

$-q_1$ – iff the current input word has the even number of 0 and the even number of 1,

$-q_2$ – iff the current input word has the even number of 0 and the odd number of 1,

$-q_3$ – iff the current input word has the odd number of 0 and the even number of 1,

$-q_4$ – iff the current input word has the odd number of 0 and the odd number of 1.

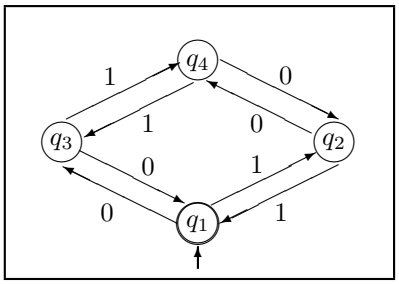

Fig. 1: The finite automaton for language of all words with the even number of 0 and the even number of 1.

The transition function is defined

$$\delta(q_1, 0) = q_3, \delta(q_2, 0) = q_4, \delta(q_3, 0) = q_1, \delta(q_4, 0) = q_2, \tag{1}$$

$$\delta(q_1, 1) = q_2, \delta(q_2, 1) = q_1, \delta(q_3, 1) = q_4, \delta(q_4, 1) = q_3, \tag{2}$$

The neural network to the automaton $\mathcal{A}$ is in Figure 2.

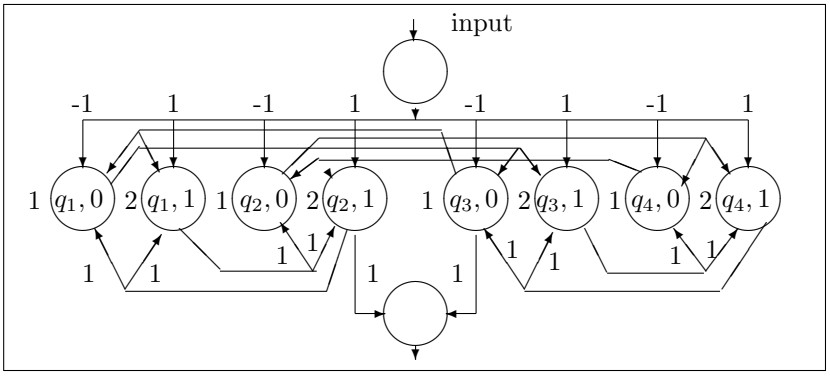

Fig. 2: The neural network to the finite automaton in the example.

The construction shows that each language recognized by the finite automaton (a regular language) can be recognised by the neural network and the number of the neural network is depending on the number of the automaton state. The opposite question we try to answer in the next section.

4 Neuromata - the acceptors of languages

There are known the following three results [8]:

1. Any language $L = L(\mathcal{N})$ recognized by a neural network (an acceptor) $\mathcal{N}$ is regular.
2. For every regular language L denoted by a regular expression $L[\alpha]$ there exists a neural acceptor $\mathcal{N}$ of the size $O(|\alpha|)$ such that L is recognized by $\mathcal{N}$.
3. There exists regular languages $L_n = [\alpha_n], n \geq 1$ such that any neural acceptor N that recognizes the language $L_n = L(N)$ requires at least $\Omega(|\alpha_n|)$ neurons.

We will try to explain the above described results. For the first type of input protocol we define the finite discrete neural acceptor. It is very closed to the finite automaton and it will be called *neuromaton*.

Definition 4: A neural acceptor *(shortly, a neuromaton)* is a 7-tuple $\mathcal{N} = (V, inp, out,$
$E, w, h, s_{init})$, where V is the set of n neurons including the input neuron $inp \in V$, and the output neuron

$out \in V$, $E \subseteq V \times (V - inp)$ is the set of edges, $w : E \longrightarrow Z$ (Z is the set of integers) is the weight function (we use the abbreviation $w(< i, j >) = w_{ij}$), $h : V - \{inp\} \longrightarrow \{0, 1\}$ is the threshold function (the abbreviation $h(i) = h_i$ is used), and $s_{init} : V - \{inp\} \longrightarrow \{0, 1\}$ is the initial state of the network. The graph (V, E) is called the architecture of the neural network N and $n =\mid V \mid$ is the size of the neuromaton. The number of bits that are needed for the whole neuromaton representation (especially for the weight and threshold functions) is called the descriptional complexity of neuromaton. $\odot$

For every regular language $L \in \mathcal{RL}$ denoted by regular expression $L = [\alpha]$ there exists a neuromaton $\mathcal{N}$ of the size $O(|\alpha|)$ such that $L = L(\mathcal{N})$ is recognized by $\mathcal{N}$. For language $L = [\alpha]$, we construct a neuromaton $\mathcal{N}_\alpha = (V, inp, out, E, w, h, s_{init}$ of size $O(|\alpha|$ so that $L = L_\alpha$.

The construction of a neuromaton

The construction is described in Šíma [8]. We first build an architecture (V, E) of the neural network $\mathcal{N}_\alpha$ recursively with respect to the structure of the regular expression α. For that purpose we define the sequence of graphs $V_k, E, k, k = 0, \ldots, p$ where (V_0, E_0) has one vertex corresponding to the whole expression α which is recursively partitioned into shorter regular subexpressions, so that (V_p, E_p) have vertices of the type 0 or 1 according to the expression α.

1. $V_0 = \{s, \alpha, o\}, E_0 = \{[s, \alpha], [\alpha, o]\}$
2. Assume that $V_k, E_k, 0 \leq k < p$ have already been constructed and $\beta \in V_k$ is a subexpression of α different from 0 or 1. Hence, besides the empty language and the empty string, the regular expression β can denote union, concatenation, or iteration of subexpressions in β. We will construct a new graph (V_{k+1}, E_{k+1}). We remove the vertex β and add new vertices and we get the new graph. One of the new vertex we can identify as β. The substitutions:
   - β is $\emptyset$: $V_{k+1} = V_k - \{\beta\}$, $E_{k+1} = E_k - \{[x, \beta], [\beta, y] \in E_k\}$.
   - β is ϵ: $V_{k+1} = V_k - \{\beta\}$, $E_{k+1} = (E_k - \{[x, \beta], [\beta, y] \in E_k\}) \cup \{[x, y] | [x, \beta], [\beta, y] \in E_k - \{[\beta, \beta]\}\}$.
   - β has the form $\beta + \gamma$: $V_{k+1} = V_k \cup \{\gamma\}$, $E_{k+1} = E_k \cup \{[x, \gamma], [\gamma, y] | [x, \beta], [\beta, y] \in E_k\} \cup \{[\gamma, \gamma] | [\beta, \beta] \in E_k\}$.
   - β has the form $\beta.\gamma$: $V_{k+1} = V_k \cup \{\gamma\}$, $E_{k+1} = (E_k - \{[\beta, y] \in E_k\}) \cup \{[\beta, \gamma]\} \cup \{[\gamma, y] | [\beta, y] \in E_k\}$.
   - β has the form β^+: $V_{k+1} = V_k, E_{k+1} = E_k \cup \{[\beta, \beta]\}$.

This construction is finished after $p = O(|\alpha|)$ steps when V_p containes only subexpressions 0 or 1. Then we define the network architecture in the following way:

$$V = V_p \cup \{inp\}, E = E_p \cup \{[inp, \beta] | \beta \in V_p - \{s, o\}\}. \tag{3}$$

For $i \in V$ denote by $d(i) = |\{j \in V_p | [j, i] \in E\}|$. Now we can define the weight function w and the threshold function h:

- $i \in V$ is the neuron of the type 1: $w_{ij} = 1$ for $[j, i] \in E_p$ and $w_{i,inp} = d(i), h_i = d(i) + 1$.
- $i \in V$ is the neuron of the type 0: $w_{ij} = 1$ for $[j, i] \in E_p$ and $w_{i,inp} = -d(i), h_i = 1$.
- $s \in V$: $h_s = 1$.
- $o \in V$: $w_{o,j} = 1$ for $[j, o] \in E_p, h_o = 1$.

The initial state is defined as $s_0(i) = 0$ for $i \in V_p - \{s\}$ and $s_0(s) = 1$. The set V contains three special neurons inp, s, o as well as, others neurons of the type 1 or 0 — one for each subexpression 1 or 0 i α; hence $|V| = O(|\alpha|)$. $\odot$

An example of the neuromaton to the language $[(1(0 + 0(1 + 0)))^\star]$ is in Figure 3.

The third result shows the lower bounds for the number of neurons that, in worst case, are necessary for the recognition of regular languages which are described by regular expressions of the length n. As a consequence, it follows that the above construction of the neuromaton is size-optimal. The lower bounds should be proved for the set of languages:

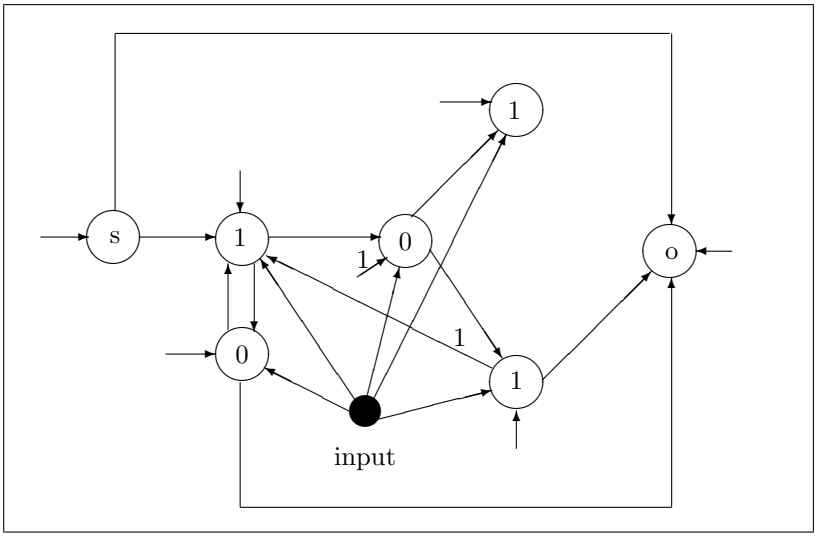

Fig. 3: Neuromaton for regular language $[(1(0+0(1+0)))^\star]$.

$$L_n = [(10 + 1(\epsilon + 0)10 + (1(\epsilon + 0))^2 10 + \ldots + (a(\epsilon + 0))^{n-3}10 + (1(\epsilon + 0))^{n-1}(1 + 0))^\star] \quad (4)$$

$$\Pi_k = \left[(1(\epsilon + 0))^k\right], \quad P_n = \cup_{k=0}^{n-1}. \quad (5)$$

It is clear that $P_n, n \geq 1$ is the set of prefixes for the language L_n. The regular expression which defines the language L_n is in fact of $O(n^2)$ length. It is possible to construct a regular language α_n of linear length denoting the same language L_n. The languges Π_n and P_n are used by the construction. And as the result of our consideration and computations can be proved that every neuromaton $\mathcal{N}$ recognizing the language L_n requires at least $\Omega(n)$ neurons.

Discussion. Siegelmann and Sontag [7] proved that one may simulate all Turing machines by recurrent neural first-order networks, it means the finite size networks which consist of interconnections of synchronously evolving neurons. Each neuron updates its state by applying a "sigmoidal" function to the linear combinations of the previous states of all neurons. In particular, one can simulate any multi-stack Turing machine in real time and there is the net with 886 neurons which computes a universal partial-recursive functions.

References

[1] Alon, N., Dewdney, A. K., Teunis, J. O.: Efficient Simulation of Finite Automata by Neural Nets, Journal of ACM, Vol. 38, No. 2, April 1991, pp.495-514.

[2] Hassoun, M. H.: Fundamentals of artificial neural networks. MIT Press, Cambridge, 1995, pp. 511.

[3] Hopfield, J. J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the Nat. Acad. of Sciences, USA, Vol. 81, pp. 3088-3092.

[4] Orponen, P.: Computational complexity of neural networks: A survey, NeuroCOLT Tech. Report Series, NC-TR-94-010, Royal Holloway University of London, 1994, pp. 20.

[5] Siegelmann, H. T., Sontag, E. D.: Analog computation via neural networks, Theoretical Computer Science 131, Elsevier, 1994, p. 331-360.

[6] Siegelmann, H. T.: Computation beyond the Turing limit, Science, 1995 , Vol. 268 , p. 545-548.

[7] Siegelmann, H. T., Sontag, E. D.: On the computational power of neural nets, Journal of computer and system sciences, Vol. 50, No. 1, 1995, p. 132-150.

[8] Šíma, J., Neruda, R.: Theoretical Questions on Neural networks, MatfyzPress, Prague, 1996.