Information Capabilities Analysis of Recurrent Neural Network Creating Energy Function Constructively

Božovský, P.
1995

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Information Capabilities Analysis of Recurrent Neural Network Creating Energy Function Constructively

Petr Božovský
Department of Computer Science, Charles University
Malostranské nám. 25, 118 00 Prague 1, Czech Republic

Dušan Húsek
Institute of Computer Science , Academy of Science CR
Po vodárenskou věží 2, 182 00 Prague 8, Czech Republic

Technical report No. 627

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244 fax: (+422) 8585789
e-mail: husek@uivt1.uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Information Capabilities Analysis of Recurrent Neural Network Creating Energy Function Constructively

Petr Božovský
Department of Computer Science, Charles University
Malostranské nám. 25, 118 00 Prague 1, Czech Republic

Dušan Húsek
Institute of Computer Science , Academy of Science CR
Po vodárenskou věží 2, 182 00 Prague 8, Czech Republic [1]

Technical report No. 627

Abstract

The Hebbian learning rule is an example of simple and quick unsupervised learning. It provides easy additional learning. However, it does not make sure the pattern states learnt to be stable. Moreover, there is created a large number of spurious stable states that worsen behavior of the neural network as an associative memory.

We propose a new method of creating an energy function for which the state vectors that are to be memorized are weighted. These vectors are proven to be stable including given neighborhood of them in sense of the Hamming distance. This method is polynomial. It also provides an additional learning. The problem of the weights tolerance is also solved.

Keywords

Recurrent neural network, Hebbian learning
Hopfield network, Energy function

[1] This paper was supported under grant No.201/94/0729 of Czech Grant Agency

1 Preliminaries

Assume that $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^r \in \{-1, 1\}^n$ are r different n-dimensional binary vectors. We wish to store this set of states in neural network consisting of n neurons. Hebbian learning is one of the learning algorithms that allow simple computer implementation. The interconnection synaptic strengths matrix $\boldsymbol{T} = (T_{ij})$ can be described by the following rule

$$\begin{aligned} T_{ij} &= \sum_{s=1}^{r} x_i^s x_j^s, \qquad \text{for } i \neq j, \\ T_{ii} &= 0, \end{aligned} \tag{1.1}$$

where $i, j \in \{1, \ldots, n\}$ and x_i^s is the i-th component of the vector $\boldsymbol{x}^s$. The thresholds θ_i of the neurons are zero.

In order to learn a new vector $\boldsymbol{x}$ the relation (1.1) can be written in the form of an incremental learning rule

$$\Delta T_{ij} = x_i x_j, \qquad i \neq j.$$

Since $T_{ii} = 0$ it is easy to show that the network reaches stable state at finite time.

Hebbian learning has several disadvantages. Some of the vectors learnt under this algorithm need not be stable. Moreover, there exist stable spurious states (phantoms) close to these vectors. It is difficult to determine the attraction area of stable states. Unlearning [2] improves the network behavior but the choice of unlearning coefficients is not evident.

2 Analysis

We propose a new method of Hebbian learning by constructing the energy function directly based on the required information storage capabilities. The vectors stored are weighted in order to make them stable and to make the attraction areas approximately the same. Learning rule is of the form

$$\begin{aligned} T_{ij} &= \sum_{s=1}^{r} \alpha_s x_i^s x_j^s, \qquad \text{for } i \neq j, \\ T_{ii} &= 0, \end{aligned} \tag{2.1}$$

where $i, j \in \{1, \ldots, n\}$ and $\alpha_1, \ldots, \alpha_r \geq 0$ and

$$\sum_{s=1}^{r} \alpha_s = 1. \tag{2.2}$$

The coefficients α_s are determined in the way we describe herein after.

As a step toward finding α_s let us investigate the conditions that have to be satisfied. The state $\boldsymbol{x}^k$ is stable if and only if for all $i \in \{1, \ldots, n\}$

$$\mathrm{sgn}(\sum_{j=1}^{n} T_{ij} x_j^k - \theta_i) = \mathrm{sgn}(x_i^k).$$

Since $x_i^k \neq 0$ we obtain the condition

$$(\sum_{j=1}^{n} T_{ij} x_j^k - \theta_i).x_i^k > 0.$$

Some calculations yield the relation

$$(\sum_{j=1}^{n} T_{ij} x_j^k - \theta_i).x_i^k = \sum_{s=1}^{r} \alpha_s x_i^s x_i^k \beta_{sk} - 1 - \theta_i x_i^k > 0, \tag{2.3}$$

where $\beta_{sk} = \sum_{j=1}^{n} x_j^s x_j^k$ corresponds with the Hamming distance between the vectors $\boldsymbol{x}^s$ and $\boldsymbol{x}^k$,

$$\beta_{sk} = n - 2\mathrm{ham}(\boldsymbol{x}^s, \boldsymbol{x}^k).$$

Since $x_i^s x_i^k \beta_{sk}$ is a constant for all i, s, k the system (2.3) can be written as a linear programming problem. Hence, the problem of learnt vectors stability is of the polynomial complexity.

We have tried to solve the system (2.3) for some small examples. For the system of upto ten different randomly chosen 50-dimensional vectors we obtained 502 constraints with 111 nonnegative variables. We used several sets of such vectors and no problem was detected during the computation. Since we had a very simple solving procedure we could not try to solve larger problems. It is remarkable that the matrix of the system (2.3) is very sparse. Using nonnegative variables we find that there are only $nr(r+2+1)$ nonzero values in the matrix with $nr(2n+r+1)$ values. In case of $r << n$ it is about $\frac{r}{2n}$ of the whole matrix (in our case 10%).

In order to investigate the network behavior at neighborhoods of the learnt vectors let t denote the maximum size of their attraction areas. As we do not know whether the inverses of $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^r$ will be learnt (as it happens in classical Hebbian learning) we obtain

$$|\beta_{sk}| \leq n - 4t,$$

and thus $t \leq n/4$. This condition was also used in [2].

Assume that $\boldsymbol{V}$ differs from $\boldsymbol{x}^k$ in just ρ components $d_1, \ldots, d_\rho$ where $1 \leq \rho \leq t$. We require the network in state $\boldsymbol{V}$ to reach the state $\boldsymbol{x}^k$. It can be fulfilled by the following condition

$$(\sum_{j=1}^{n} T_{ij} V_j - \theta_i).x_i^k > 0 \qquad \text{for all } i \in \{1, \ldots, n\}. \tag{2.4}$$

If it holds the input to the neuron i is of the same sign as the i-th element of the closest vector learnt and when updating one of the neurons $d_1, \ldots, d_n$ the Hamming distance between $\boldsymbol{V}$ and $\boldsymbol{x}^k$ decreases.

After some calculations the system (2.4) can be expressed as

$$\sum_{s=1}^{r} \alpha_s x_i^s x_i^k [\beta_{sk} - 2 \sum_{j \in \{d_1,\ldots,d_\rho\}} x_j^s x_j^k] - \theta_i x_i^k - V_i x_i^k > 0, \tag{2.5}$$

that has to hold for all $k \in \{1,\ldots,r\}$, $i \in \{1,\ldots,n\}$, $\rho \in \{0,\ldots,t\}$ and different $d_1,\ldots,d_\rho \in \{1,\ldots,n\}$.

Since $\sum_{j\in\{d_1,\ldots,d_\rho\}} x_j^s x_j^k$ can be any value from $\{-t,\ldots,t\}$ we can find lower bound of the left-hand side of (2.5) and write

$$\ldots \geq \sum_{s=1}^{r} \alpha_s (x_i^s x_i^k \beta_{sk} - 2t) - \theta_i x_i^k - 1 > 0, \qquad \text{for all } k,i. \tag{2.6}$$

This system does not depend on the choice of $d_1,\ldots,d_\rho$. Moreover, this estimation does not present additional restriction because for all $s,k \in \{1,\ldots,r\}$ there exists such a state $\boldsymbol{V}$ that the relation (2.6) follows from (2.5). It implies that the system (2.5) is equivalent to (2.6).

In order we could formulate (2.6) as a linear programming problem we need to add a new variable, say $\varepsilon \in \boldsymbol{R}$. For these purposes we denote $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_r)$, $\boldsymbol{n}_i^k = (n_{i1}^k,\ldots,n_{ir}^k)$, where

$$n_{is}^k = (x_i^s x_i^k \beta_{sk} - 2t).$$

Now the system (2.6) can be expressed as

$$\boldsymbol{\alpha}.\boldsymbol{n}_i^k - \theta_i x_i^k > 1. \tag{2.7}$$

It can be easily shown that (2.7) holds if there exists $\varepsilon > 0$ such that

$$\boldsymbol{\alpha}.\boldsymbol{n}_i^k - \theta_i x_i^k \geq 1 + \varepsilon |\boldsymbol{n}_i^k|, \tag{2.8}$$

further, if (2.8) holds for some $\boldsymbol{\alpha}^*$ and ε then the constraints (2.7) hold for all $\boldsymbol{\alpha}$ such that $|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*| < \varepsilon$.

The linear programming problem can be stated as the maximization of ε that is to be accomplished subject to a set of $nr+2$ linear constraints (2.2) and (2.8) among the variables $\alpha_1,\ldots,\alpha_r$, $\theta_1,\ldots,\theta_n$, ε.

We have shown that when we find a solution of linear programming problem

$$\begin{aligned}
&\textbf{max } \varepsilon \\
&\sum_{s=1}^{r} \alpha_s (x_i^s x_i^k \beta_{sk} - 2t) - \theta_i x_i^k - \varepsilon \Big(\sum_{s=1}^{r} (x_i^s x_i^k \beta_{sk} - 2t)^2 \Big)^{1/2} \geq 1, \\
&\sum_{s=1}^{r} \alpha_s = 1 \qquad\qquad \text{for all } k \in \{1,\ldots,r\}, i \in \{1,\ldots,n\}, \\
&\alpha_1,\ldots,\alpha_r \geq 0,
\end{aligned} \tag{2.9}$$

then the vectors learnt are stable and have attraction areas at least of the size t, where we choose

$$t \leq \ (n \ - \max_{s \neq k} |\beta_{sk}|)/4.$$

Further we provide the ε-inaccuracy of all α_s.

In global the network constructed by the described method cannot oscillate because $T_{ii} = 0$. Obviously, when we start from the t-neighborhood of some learnt vector we find that the network state goes directly to the closest stable vector, i.e. in this case it is sufficient to update each neuron only once.

Additional learning of a new vector $\boldsymbol{x}^{r+1}$ is also possible. When we set

$$\begin{aligned} \alpha_s^{\text{new}} &= \alpha_s^{\text{old}}, \qquad s = 1, \ldots, r, \\ \alpha_{r+1} &= 0 \end{aligned}$$

and ε^{new} small (negative) enough to satisfy the additional constraints we obtain a feasible solution of (2.9). Then we can process the maximization.

3 Results and Future Work

We solved (2.9) for the particular case described above. In various cases the solution was found for $t = 5$ upto $t = 10$ with the ε-tolerance 0.03-0.1. Results obtained were related to the choice of vectors learnt.

In future we want to use a modification of the linear programming network described in [4] to solve the problem (2.9).

Bibliography

[1] Hopfield J.: *Neural networks and physical systems with emergent collective computational abilities*, Proc. Natl. Acad. Sci. USA, Vol.79, pp. 2554-2558 (1982)

[2] Hopfield J., Feinstein D., Palmer R.: *'Unlearning' has a stabilizing effect in collective memories*, Nature 52, pp. 158-159 (1983)

[3] Li J., Michel A., Porod W.: *Analysis and synthesis of a class of neural networks: variable structure systems with infinite gain*, IEEE TCS 36, pp. 713-731 (1989)

[4] Tank D., Hopfield J.: *Simple "Neural" Optimization Networks*, IEEE TCS CAS-33, pp. 533-541 (1986)