A STATISTICAL CONVERGENCE APLICATION FOR THE HOPFIELD NETWORKS

Víctor Giménez-Martínez, Gloria Sánchez–Torrubia, Carmen Torres–Blanc

Abstract: When Recurrent Neural Networks (RNN) are going to be used as Pattern Recognition systems, the problem to be considered is how to impose prescribed prototype vectors $\xi^1,\xi^2,...,\xi^p$, as fixed points. The synaptic matrix W should be interpreted as a sort of sign correlation matrix of the prototypes, In the classical approach. The weak point in this approach, comes from the fact that it does not have the appropriate tools to deal efficiently with the correlation between the state vectors and the prototype vectors The capacity of the net is very poor because one can only know if one given vector is adequately correlated with the prototypes or not and we are not able to know what its exact correlation degree. The interest of our approach lies precisely in the fact that it provides these tools. In this paper, a geometrical vision of the dynamic of states is explained. A fixed point is viewed as a point in the Euclidean plane R^2. The retrieving procedure is analyzed trough statistical frequency distribution of the prototypes. The capacity of the net is improved and the spurious states are reduced. In order to clarify and corroborate the theoretical results, together with the formal theory, an application is presented

Keywords: Learning Systems, Pattern Recognition, Graph Theory, Recurrent Neural Networks.

1. Introduction

As is well known, a RNN is a discrete time, discrete-valued dynamic system which at any given instant of time *t* is characterized by a binary *state vector* $x(t)=\left[x_1(t),...,x_i(t),...,x_n(t)\right]\in\{1,\ -1\}^n$. The behavior of the system is described by a *dynamic equation* of the type

$$x_i(t+1)=Sgn\left[\sum_{j=1}^{n}w_{ij}x_j(t)-\theta_i\right]\quad i=1,2,...,n \tag{1}$$

A point x is a fixed point if all its components remain unchanged when (1) is applied. The aim is to get the network parameters, namely the synaptic matrix W and the threshold vector θ, for which the prototype vectors $\xi^1,\xi^2,\ ...,\xi^p$, are fixed points. In our approach $x(t)=\{0,1\}^n$ and, as the components of $x(t)$ may only be zero or one, we will refer to them as the null and unit components in $x(t)$. Associated to the network there will be a complete graph G, with n vertices $\{v_1,...,v_n\}$, and *one* bi-directional *edge* a_{ij} for every possible pair of different vertices (1). Initially, at the *training stage* a null value w_{ij} is assigned to every edge a_{ij} in the graph; afterwards, when ξ^μ is presented to the net; the weight w_{ij} is updated by:

$$\Delta w_{ij}=\begin{cases}+1 & if\quad \xi_i^\mu=\xi_j^\mu=1,\ i\neq j,\\ -1 & if\quad \xi_i^\mu=\xi_j^\mu=0,\ i\neq j,\\ 0 & otherwise.\end{cases} \tag{2}$$

This idea may be much more easily understood using the next *graphical interpretation* of the training algorithm: At the first step, a null value is assigned to all the edges a_{ij}, then, when a learning pattern ξ^μ is acquired by the net, it is superposed over the graph *G.* The components $\{\xi_1^\mu,...,\xi_n^\mu\}$, are going to be mapped over the vertices $\{v_1,...,v_n\}$ of *G.* This mapping may be interpreted as a *coloring* of the edges in *G*, in such a way that, if $\xi_i^\mu=\xi_j^\mu=1$, the edge a_{ij} (whose ending vertices are v_i and v_j) will be colored with a certain color, for

example *red*. On the other hand, if $\xi_i^\mu = \xi_j^\mu = 0$, then a_{ij} will be colored with a different color, as for example *blue*. The rest of the edges in *G* remain uncolored. Once this coloring has been done, the value assigned over the, also *complete*, graph of *red* edges are positively reinforced and the value assigned over the edges of the *blue* graph are negatively reinforced. The value over the rest of the edges remains unchanged. Once the pattern ξ^μ is acquired, the colors are erased and we repeat the same color assignation with the next pattern to be acquired by the net, and so on. When every vector in the training pattern set has been integrated in the net, the training stage is finished, the *resulting graph G* has become edge-valued and its weight matrix is the *synaptic matrix W* of the net.

2. Parameters of the Net

According with the theorem proved in [1], If any set $\xi^1, \xi^2, \ldots, \xi^p$ of prototype vectors are acquired by the net, then for any possible four different components $"i", "j", "r"$ y $"s"$, then the relation: $w_{ij} + w_{rs} = w_{is} + w_{rj}$ is satisfied, and solving the system

$$\{w_{ij} = p_i + p_j,\ i \neq j \tag{3}$$

(in *n* unknown $p_1, p_2, \ldots, p_n$) a solution and only a solution is obtained [1].. The *training algorithm* could be revisited in order to obtain the *weight vector* $\vec{p}$ without the necessity of obtaining the *weight matrix W* first and then solving the system (3). In the *graphical interpretation* of the training algorithm, we may consider $\{p_i, p_j\}$ as the ending vertices of the generic edge a_{ij}. Just when the pattern ξ^μ has been acquired, if a_{ij} has been colored by red, its weight w_{ij} has been incremented by one. As $w_{ij} = p_i + p_j$, we may consider that p_i and p_j have both been incremented by *"½"*. If n_1 and n_2 are the number of *unit* and *null* components of ξ^μ and if $\xi_i^\mu = 1$, then obviously the number of red edges with one end in p_i is equal to $(n_1 - 1)$. Consequently just when the pattern ξ^μ has been acquired p_i has been incremented by $½(n_1 - 1)$. In the same way, it could be proved that if $\xi_i^\mu = 0$, then p_i is incremented by $-½(n_0 - 1)$. The training algorithm in (3) could then be designed as follows:

$$\Delta p_i = \begin{cases} \frac{1}{2}(n_1 - 1) & \text{if } \xi_i^\mu = 1 \\ -\frac{1}{2}(n_0 - 1) & \text{if } \xi_i^\mu = 0 \end{cases} \tag{4}$$

(n_1 and n_0 are the number of *unit* and *null* components of ξ^μ). When all the learning patterns have been acquired, the training is finished. Considering ½ a scale factor and since the inner product $\xi^\mu \cdot \xi^\mu$ is equal to n_1 and the inner product $\bar{\xi}^\mu \cdot \bar{\xi}^\mu$ is equal to n_0,, it can be interpreted that when the learning pattern ξ^μ is acquired the *weight vector* $\vec{p}$ is modified as follows:

$$\vec{p} \leftarrow \vec{p} + (\xi^\mu \cdot \xi^\mu - 1) \cdot I - (\bar{\xi}^\mu \cdot \bar{\xi}^\mu - 1) \cdot I \text{, where } I \text{ is the unitary vector } (1,1,..,1) \tag{5}$$

The above expression realizes the *updating* of the weights p_i, for *i* from *1* to *n*, when ξ^μ is acquired. The computational *time* of the training algorithm, is then highly optimized.

2.1. Energy

The state vector *x* at time *t* could also be interpreted as a *coloring* of the edges in *G,* but now this coloring is going to be used to retrieve the stored data. If the graph *G* is colored with the coloring associated with the pattern $x(t)$, it is easy to understand (taking into account how the training algorithm was designed), that the bigger the summation of all the edges in the *red graph* and the lower the summation of all the edges in the *blue graph* are,

then the more correlated the pattern $x(t)$ must be, with those that were used during the training stage. So, if *W* is the weight matrix of *G*, the *energy point EP* of the net is defined as a pair of numbers. The first of them represents the summation of all the values on the edges of the *red* graph, and the second one represents the same summation, but on the *blue* ones. So if *G* is colored with the color associated to *x(t)*, then *{I (t), O(t)}* may be defined as the pair of quadratic forms:

$$\begin{cases} I(t) = \frac{1}{2} x(t) \cdot W \cdot x(t)^t \\ O(t) = \frac{1}{2} \overline{x}(t) \cdot W \cdot \overline{x}(t)^t \end{cases} \tag{6}$$

If n_1 is the number of *unit* components of $x(t)$ and n_0 is the number of the *null* ones (in other words n_1 is the *Hamming distance* from *x(t)* to the *zero* vector). By other hand, it is obvious [1], that if $\{I_i(t), O_i(t)\}$, is the *EP*, when $W = (w_{ij})$, is the matrix with all its values equal to zero except those in file or the row *I*, then

$$I_i(t) \begin{cases} n_1 - 1 & \text{if } x_i(t) = 1 \\ 0 & \text{if } x_i(t) = 0 \end{cases}, \quad \text{and} \quad O(t) \begin{cases} n_0 - 1 & \text{if } x_i(t) = 0 \\ 0 & \text{if } x_i(t) = 1 \end{cases} \tag{7}$$

So, if $i_1, i_2, ..., i_{n_1}$ and $j_1, j_2, ..., j_{n_0}$ are the places where the *unit* and *null* components of *x(t)* are respectively located, the equations (13) could be written as

$$I(t) = (n_1 - 1)\left(p_{i_1} + ... + p_{i_{n_1}}\right) \quad \text{and} \quad O(t) = (n_0 - 1)\left(p_{j_1} + ... + p_{j_{n_0}}\right) \tag{8}$$

Which means that

$$\frac{I(t)}{(n_1 - 1)} = \left(p_{i_1} + ... + p_{i_{n_1}}\right) \quad \text{and} \quad \frac{O(t)}{(n_0 - 1)} = \left(p_{j_1} + ... + p_{j_{n_0}}\right) \text{ and} \tag{9}$$

in other words

$$\frac{I(t)}{(n_1 - 1)} + \frac{O(t)}{(n_0 - 1)} = K \qquad \text{being } K = (p_1 + ... + p_n) \tag{10}$$

As all *state vectors* $x(t)$ with the same *Hamming distance "i"* to the zero vector contains the same numbers n_1 and n_0 of *unit* and *null* components, the *energy points* $\{I(t), O(t)\}$, associated to all of them, will be placed in the same line r_i $r_{i,}$ whose equation expressed in *(x,y)* is

$$r_i \equiv (n_0 - 1)x + (n_1 - 1)y - (n_0 - 1) \cdot (n_1 - 1) \cdot K = 0 \tag{11}$$

In other words, all the *EP´s* associated with state vectors with the same number of *unit* components are placed in the same line of the *energy field*, and the equation of this line is the one represented in (11). In this way the *state vector space* is classified in as many classes as the dimension *n* of the space.

2.2. Dynamics

On the other hand, and as we said in the introduction, the nature of the algorithm here proposed let to know how the value of $x(t)$ affects the whole energy of the state $x(t)$. We may define the relative weight of the neuron i when the net is in state x(t) as the contribution of this neuron to the component $I(t)$, if $x_i(t) = 1$; or as the contribution of this neuron to the component $O(t)$, if $x_i(t) = 0$. So, if $x_i(t) = 1$, we define the relative weight $w(t)$ of the neuron i when the net is in state $x(t)$ as:

$$w_i(t) = \frac{1}{n_1 - 1} + \frac{n_1 - 2}{n_1 - 1} \cdot \frac{p_i}{p.x(t)} = \frac{1}{x(t) \cdot x(t) - 1} + \frac{x(t) \cdot x(t) - 2}{x(t) \cdot x(t) - 1} \cdot \frac{p_i}{p.xx(t)} \quad (12)$$

If in time *t* the state vector $x(t)$ is in class $[j]$, then for any *i* from *1* to *n*, the dynamic equation is defined as

$$x_i(t+1) = f_h\left[f_b(x_i(t)) \ . \ (w_i(t) - \theta_j)\right] \quad (13)$$

where f_h is the Heaviside step function and f_b is the function defined as $f_b(x) = 2x - 1$, which achieves the transformation from the domain $\{0,1\}$ to the domain $\{1,-1\}$

It can also be stated that the sum of the relative weights $w_i(t)$ for the unit components of $x(t)$ is equal to *2*. The same could be proved for the null components. We have then that the relative weight vector $w(t)$ associated to any state vector $x(t)$ may also be interpreted as a sort of frequency distribution of probabilities [2]. The reason is that

$$\sum_{i=1}^{n} w_i(t) = 4 \quad \Rightarrow \quad \sum_{i=1}^{n} \tfrac{1}{4} w_i(t) = 1 \quad (14)$$

For any relative weight vector $w(t)$ The "uniform distribution vector" would be the one with all its components equal to $4/n$. For any state $x(t)$ we could then define its deviation $D(x(t))$ as

$$D(x(t)) = \sqrt{\sum_{i=1}^{n}\left[x_i(t) - \frac{4}{n}\right]^2} \quad (15)$$

The deviation of a given vector to the prototypes has been used for avoiding the parasite fixed points.

3. Application

We take, as an example for validating the performance of the algorithm we propose, the problem of the recognition of the Arabian digits as the prototype vectors:

Where the dimension *n*, of the pattern space is *28*, and

$$\begin{cases} \xi^1 = [0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1] \\ \xi^2 = [1,1,1,1,0,0,0,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,0,0,0,1,1,1,1] \\ . \\ . \\ \xi^{10} = [1,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,,1,1,1,1] \end{cases}$$

and $p = 1/14\ \{53, 25, 25, 53, 11, -73, -73, 39, 11, -73, -73, 39, 39, 25, 25, 67, -17, -73, -73, 53, -17, -73, -73, 53, 11, 11, 11, 67\}$. In figure 1 the reader may see the energy lines and theirs associated PE´s. The Arabian digits are in this way placed on the lines: r_7, r_{16}, r_{16}, r_{13}, r_{16}, r_{15}, r_{10}, r_{20}, r_{15}, r_{18}. The associated PE´s are $1/7\{1113,-3710\}, 1/7\{3420,-2508\}$, $1/7\{4470,-3278\}, 1/7\{3210,-3745\}, 1/7\{4050,-2970\}, 1/7\{2821,-2418\}, 1/7\{2133,-4029\}, 1/7\{5548,-2044\}, 1/7\{4095,-3510\}, 1/7\{4539,-2403\}$,

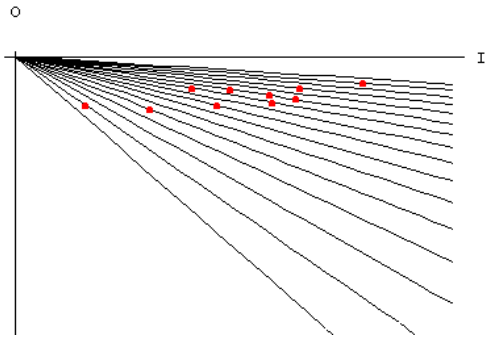

Figure 1 Arabian Digits Projections

The problem now is how to obtain in an adaptive way the capacity parameters $\theta_1, \theta_2, ..., \theta_{28}$, in order to obtain the Arabian digits as fixed points with the least number of parasitic points as possible.

When the dynamic equation in (13) is considered, a point $x(t)$ whose energy projection belongs to the r_j line, is a fixed point if, and only if, the (capacity) parameter θ_j is an upper bound for all the relative weights $w_i(t)$ associated to the components of $x(t)$. Once the training has finished, the relative weight vector of the prototypes could then be calculated. If the energy projection of the prototype ξ^μ belongs to r_j and the largest of the components of $w_i(t)$ is taken as θ_j: it is clear that the prototype ξ^μ will be a fixed point. But the problem is how to avoid that points with high degree of correlation with a prototype but with all its relative weights components lower than the capacity parameter to skip away from this prototype. The idea proposes in this paper, made use of the deviation defined in (15). When, in time *t*, the dynamic equation is applied to a component of the vector $x(t)$, this component will change its state not only if the relative weight $w_i(t)$ is lower that the capacity parameter of its class. The deviation of the new state, in the case of change of sate, must be similar to the deviation of the prototypes in the new class. The degree of similarity may be measured by a coefficient . The coefficient is handled in a dynamical way (the more is the time the higher is the coefficient).

Besides the weight vector, there are other set of parameters of the net. For every one class r_i, the capacity parameter i and the deviation of the prototypes in this class are obtained. So the algorithm control not only if the new state is strongly correlate with some prototype in its class, the algorithm also control that the components in the new state must, with a high degree of probability, be placed in similar places as some prototype of the class. We have applied with to our example, obtaining that almost all the points inside a neighborhood of radius 1, of the prototypes, are attracted by these prototypes. The 10 Arabian digits are fixed points of the system, and almost all the 28 neighbor of any one of them were attracted by its attractor prototype. In figure 2, the number of points inside a neighborhood of radius 1, of the prototypes are expressed.

$$\begin{cases} \boxed{24 \to 1} & \boxed{23 \to 2} & \boxed{25 \to 3} & \boxed{22 \to 4} & \boxed{27 \to 5} \\ \boxed{22 \to 6} & \boxed{25 \to 7} & \boxed{25 \to 8} & \boxed{22 \to 9} & \boxed{21 \to 0} \end{cases}$$

Figure 2. Prototypes belonging also to r_{15}

4. Conclusion

The weight parameters in the *Hopfield* network are not a free set of variables. They must fulfill a set of constrains which have been deduced trough a new re-interpretation of the net as *Graph Formalisms*. Making use of this constrains the *state-vector* has been classified in *n* classes according to the *n* different possible distances from any of the state-vectors to the *zero* vector. The $(n \times n)$ matrix of weights may also be reduced to a *n*-vector of

weights. In this way the computational time and the memory space, required for obtaining the weights, is optimized and simplified. The degree of correlation from a pattern with the prototypes may be controlled by the dynamical value of two parameters: the capacity parameter θ which is used for controlling the capacity of the net (it may be proved that the bigger is the θ_j component of θ, the lower is the number of fixed points located in the r_j energy line) and the parameter μ which measures the deviation to the prototypes. A typical example has been exposed, the obtained results have proved to improve the obtained when the classical algorithm is applied.

Bibliography

[1] V. Giménez-Martínez, *A Modified Algoritm Hopfield Auto-Associative Memory with Improved Capacity*, IEEE Transactions on Neural Networks, (in press), 2000.

[2] N. K. Bose and P. Liang. *Neural Network Fundamentals with Graphs, algorithms and Applications.* McGraw Series in Electrical and Computer Engineering, 1996.

[3] V. Giménez-Martínez, P. Gómez-Vilda, E. Torrano and M. Pérez-Castellanos, *A New Algorithm for Implementing a Recursive Neural Network,* Proc. of the IWANN´95 Torremolinos, Málaga, Spain, June, pp. 252-259, 1995.

[4] V. Giménez-Martínez, P. Gómez, M. Pérez- Castellanos, and E. Torrano, *A new approach for controlling the capacity of a Recursive Neural Network,* Proc of AMS´94. IASTED, Lugano, Suise, June, pp 90-93, 1994

Authors' Information

Víctor Giménez-Martínez – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: vgimenez@fi.upm.es

M Gloria Sánchez–Torrubia – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: gsanchez@fi.upm.es

Carmen Torres–Blanc – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: ctorres@fi.upm.es

NEURAL NETWORKS DIAGNOSTICS IN HOMEOPATH SYSTEM

Larysa Katerynych, Alexander Provotar

Abstract: We suppose the neural networks for solution the problem of the diagnostic in Homeopath System and consider the algorithms of the training.

Keywords: artificial intelligence, neural networks, training of neural networks, information granules.

Introduction

As a rule, as a consequence of the cerebrum study and mechanisms of its functioning there have been created new computer models, namely artificial neural networks (NN). The tasks of the office automation processes based upon the research in the sphere of the artificial intelligence (AI) are of current importance to present day. NN permit to solve applications such as pattern recognition, modeling, fast data conversion (parallel computational processes), identifications, management, and expert systems creation [Терехов, 2002, Барский, 2004].

Theoretically, NN can solve a wide frame of tasks in the specific data domain. (as it is the human brain model prototype), but it is still not practically possible to create the integrated universal NN for the specific data domain at present, since there is no integrated construction algorithm (functioning) of the NN. The moment to date the specific structure NN and with the defined learning algorithms are used for the solution of the concrete group of tasks out of the fixed data domain.