A NEW APPROACH FOR ELIMINATING THE SPURIOUS STATES IN RECURRENT NEURAL NETWORKS

Víctor Giménez-Martínez, Carmen Torres,
José Joaquín Erviti Anaut, Mercedes Perez-Castellanos

Abstract: As is well known, the Convergence Theorem for the Recurrent Neural Networks, is based in Lyapunov´s second method, which states that associated to any one given net state, there always exist a real number, in other words an element of the one dimensional Euclidean Space R, in such a way that when the state of the net changes then its associated real number decreases. In this paper we will introduce the two dimensional Euclidean space R^2, as the space associated to the net, and we will define a pair of real numbers (x, y), associated to any one given state of the net. We will prove that when the net change its state, then the product $x \cdot y$ will decrease. All the states whose projection over the energy field are placed on the same hyperbolic surface, will be considered as points with the same energy level. On the other hand we will prove that if the states are classified attended to their distances to the zero vector, only one pattern in each one of the different classes may be at the same energy level. The retrieving procedure is analyzed trough the projection of the states on that plane. The geometrical properties of the synaptic matrix W may be used for classifying the n-dimensional state-vector space in n classes. A pattern to be recognized is seen as a point belonging to one of these classes, and depending on the class the pattern to be retrieved belongs, different weight parameters are used. 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, Image Processing, Recurrent Neural Networks.

ACM Classification Keywords: I.2.6 Learning: Connectionism and neural nets; G.2.2. Graph Theory; I.4.0 Image processing software

1. Introduction

The problem to be considered when Recurrent Neural Networks (RNN) are going to be used as *Pattern Recognition* systems, is how to impose prescribed prototype vectors $\xi^1, \xi^2, ..., \xi^p$, of the space $\{-1,1\}^n$, as fixed points. In the classical approach, the synaptic matrix $W = (w_{ij})$ should be interpreted as a sort of sign correlation matrix of the prototypes. The element $w_{ij} \in W$, is going to represent some kind of relation between coincidences and not coincidences on the list of the components "i" and "j" for all the prototype vectors $\xi^1, \xi^2, ..., \xi^p$. The classical solution to impose fixed points by means of the synaptic matrix W is the *Hebb´s* law, which states that the synaptic weight w_{ij} should increase whenever neurons "i" and "j" have simultaneously the same activity level and it should decrease in the opposite case. As it was pointed out above, the prototype vector components must belong to the set $\{-1,1\}$; this fact is the cornerstone of the *Hebb´s* law mathematical interpretation. The reason is that when the prototype ξ^μ is stored, neurons "i" and "j" may receive a similar sign or not". The mathematical advantage of this interpretation lies in the fact that when the prototype ξ^μ is acquired, the synaptic weight w_{ij} should increase if neurons "i" and "j" receive a similar sign: in other words if $\xi_i^\mu . \xi_j^\mu$ is positive. On the other hand w_{ij} should decrease if $\xi_i^\mu . \xi_j^\mu$ is negative. The updating of the weights may be then expressed by, $\Delta w_{ij} = \xi_i^\mu . \xi_j^\mu$, in other words, when the sign

of the components "i" and "j" in the prototype ξ^{μ} are with similar sign, the weight w_{ij} is positively reinforce, otherwise do the same but in a negative sense. In general a positive learning parameter η may be used, and it can be state as the general training rule that the prototype ξ^{μ} is stored then $\Delta w_{ij} = \eta . \xi_i^{\mu} . \xi_j^{\mu}$ (being η a positive learning factor); which means that, $\Delta w_{ij} \in \{-\eta, \eta\}$. The synaptic matrix W should be interpreted as a sort of sign correlation matrix of the prototypes.

2. Training: Parameters of the Net

In our approach, instead of a matrix for storing the weights, a weight vector $\vec{p} = (p_1, p_2, ..., p_n)$ is going to be introduced. At the beginning $\vec{p} = (0,0,...,0)$. At time t, when the training pattern ξ^{μ} is acquired, the weight vector $\vec{p}$, will be updated by this very simple rule:

$$\vec{p} = \vec{p} + \xi^{\mu}$$

which means that, $\forall i, j = 1,..,n$ then,

$$\begin{cases} \text{If } \left(\xi_i^{\mu}=1 \text{ and } \xi_j^{\mu}=1\right) \text{ then } \left(p_i = p_i + 1 \text{ and } p_j = p_j + 1\right) \\ \text{If } \left(\xi_i^{\mu}=-1 \text{ and } \xi_j^{\mu}=-1\right) \text{ then } \left(p_i = p_i - 1 \text{ and } p_j = p_j - 1\right) \\ \text{If } \left(\xi_i^{\mu}=-1 \text{ and } \xi_j^{\mu}=1\right) \text{ then } \left(p_i = p_i - 1 \text{ and } p_j = p_j + 1\right) \\ \text{If } \left(\xi_i^{\mu}=1 \text{ and } \xi_j^{\mu}=-1\right) \text{ then } \left(p_i = p_i + 1 \text{ and } p_j = p_j - 1\right) \end{cases}$$

It is clear that in this way training is faster than in the classical procedure, and the knowledge stored in the net parameter is equivalent. The synaptic matrix $W = (w_{ij})$, may be rebuilt by doing,

$$w_{ij} = p_i + p_j, \quad \forall i, j = 1,..,n$$

Which is equivalent to state that when the prototype ξ^{μ} is acquired, the synaptic weight w_{ij} should increase if neurons "i" and "j" are in state 1, and will decrease if neurons "i" and "j" are in state -1. We may then consider that with this procedure, instead of storing some kind of sign correlation of the prototypes, like in the classical procedure was done, we are storing the correlation prototypes features. This method has also a very good property, and this is that, for any $i, j, r, s \in \{1,..,n\}$, then as,

$$p_i + p_j + p_r + p_s = p_i + p_s + p_r + p_j$$

one has that

$$w_{ij} + w_{rs} = w_{is} + w_{rj}$$

So, as the way the parameters are stores is related with the features of the pattern components, we are going to use in our approach the Boolean space $\{0,1\}^n$ instead of the bimodal space $\{-1,1\}^n$. The training procedure will be then defines by

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

Two important advantages may be extracted from the above approach. The first advantage is that the space required for storing the parameters is lesser than in the classical one: the parameters may be stored in the weight

vector, and then doing $w_{ij} = p_i + p_j$, to span it to the weight matrix if necessary. The second advantage is that the training may be much more easily understood using the next graphical interpretation of the training algorithm: A n complete graph G may be introduced associated with the net. 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 $W = \left(w_{ij}\right)$ of the net. Now if we define the basic matrix $U^k = (u_{ij}^k)$, where

$$\begin{cases} u_{ij} = 1 & \text{if} \quad (i = k \quad \text{xor} \quad j = k) \\ u_{ij} = 0 & \text{otherwise} \end{cases}$$

then, any synaptic matrix W is generated by the set of basic matrices $\left\{U^1, U^2, ..., U^n\right\}$. In other words,

$$W = p_1.U^1 + ... + p_n.U^n.$$

3. Recall

For recalling a pattern from the net, the net should be colored with the color associated with that pattern, which may be interpreted as if the net had in a certain state $x(t)$. Then, it is clear that, we may define the energy pair number $\left\{I(t), O(t)\right\}$, where

$$I(t) = \frac{1}{2}\left(x(t)\ .\ W\ .\ x(t)^t\right)$$

represents the sum of the values of all the parameters w_{ij}, associated with all the edges colored in red, and if $\bar{x}(t)$ is the symmetric vector of $x(t)$, then

$$O(t) = \frac{1}{2}\left(\bar{x}(t)\ .\ W\ .\ \bar{x}(t)^t\right)$$

represents the sum of the values of all the parameters w_{ij}, associated with all the edges colored in blue. Taking now into account that

$$W = p_1 \cdot U^1 + ... + p_n \cdot U^n$$

one has that

$$\frac{1}{2}\left(x(t)\ \cdot\ W\ \cdot x(t)^t\right) = \frac{1}{2}x(t)(p_1U^1 + .. + p_nU^n)x(t)^t = p_1\left[\frac{1}{2}x(t)\cdot U^1 \cdot x(t)^t\right] + ... + p_1\left[\frac{1}{2}x(t)\cdot U^n \cdot x(t)^t\right]$$

and

$$\frac{1}{2}\left(\bar{x}(t)\ \cdot\ W\ \cdot\ \bar{x}(t)^t\right) = \frac{1}{2}\bar{x}(t)(p_1U^1 + .. + p_nU^n)\overline{x(t)}^t = p_1\left[\frac{1}{2}\bar{x}(t)\cdot U^1 \cdot \bar{x}(t)^t\right] + .. + p_1\left[\frac{1}{2}\bar{x}(t)\cdot U^n \cdot \bar{x}(t)^t\right]$$

In other words, if $\forall i, j = 1,..,n$, we do

$$\begin{cases} I_i = \frac{1}{2} x(t) \cdot U^i \cdot x(t)^t \\ O_i = \frac{1}{2} \bar{x}(t) \cdot U^i \cdot \bar{x}(t)^t \end{cases}$$

then

$$\begin{cases} I(t) = p_1 \cdot I_1(t) + ... + p_n \cdot I_n(t) \\ \\ O(t) = p_1 \cdot O_1(t) + ... + p_n \cdot O_n(t) \end{cases}$$

but if n_1 is the number of unit components of $x(t)$ and if 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), it is obvious that

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

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 above equations could be written as

$$I(t) = (n_1 - 1)(p_{i_1} + p_{i_2} + ... + p_{i_{n_1}}) \text{ and } O(t) = (n_0 - 1)(p_{j_1} + p_{j_2} + ... + p_{j_{no}})$$

which means that

$$\frac{I(t)}{n_1 - 1} + \frac{O(t)}{n_0 - 1} = k \text{ (where } k = p_1 + ... + p_n \text{).}$$

In other words: all states $x(t)$ sharing the same distance "j" (where $j \in \{1,2,...,n\}$), will verify the before equation. The states' space could be then classified in the n classes $[1],[2],...,[j],...,[n]$. If the pair $\{I(t), O(t)\}$, where defined as the energy pair (PE), associated to $x(t)$, one could state that all the states in the same class, has theirs PE´s in the same energy line. On the other hand, 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* $w_i(t)$ of the neuron "i" when the net is in state x(t) as:

$$w_i(t) = \frac{\sum_{j=1}^{n} x_j(t) \cdot w_{ij}}{I(t)}$$

and taking into account that,

$$\sum_{j=1}^{n} x_j(t).w_{ij} = x_1(t)(p_i + p_1) + .. + x_i(t) \cdot 0 + .. + x_n(t)(p_i + p_n) =$$

$$(x_1(t) \cdot p_1 + ... + x_n(t) \cdot p_n) + p_i (x_1(t) + ... + x_n(t)) - 2 \cdot x_i(t).p_i = p \cdot x(t) + p_i(n_1 - 2)$$

and

$$I(t) = (n_1 - 1) \cdot p.x(t)$$

$w_i(t)$ may be represented as

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

and without difficult it could be also proved that if $x_i(t) = 0$, then

$$w_i(t) = \frac{1}{n_0 - 1} + \frac{n_0 - 2}{n_0 - 1} \cdot \frac{p_i}{p.\bar{x}(t)}$$

4. Dynamic Equation

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

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

where f_h is the Heaviside step function and f_b is the function defined as $f_b(x) = 2 \cdot x - 1$, which achieves the transformation from the domain $\{0,1\}$ to the domain $\{-1,1\}$. In other words: if $x(t)$ is in class $[j]$ and $x_i(t) = 1$ the above expression could be written as

$$x_i(t+1) = f_h\left(w_i(t) - \theta_j\right)$$

which states that if $w_i(t) < \theta_j$, then $x_i(t)$ changes its state from state $x_i(t) = 1$ to $x_i(t) = 0$; and otherwise $x_i(t)$ doesn't change its state. On the other hand, if $x(t)$ is in class $[j]$ and $x_i(t) = 0$, the above expression could be written as

$$x_i(t+1) = f_h\left(\theta_j - w_i(t)\right)$$

which states that if $w_i(t) < \theta_j$, then $x_i(t)$ changes its state from state $x_i(t) = 0$ to $x_i(t) = 1$; and otherwise $x_i(t)$ doesn't change its state. The value of θ_j is the j-th component of a n-dimensional vector $\bar{\theta}$ and is related with the class $[j]$ to which the state $x(t)$ belongs, and must not to be interpreted as a threshold for the "i" unit (which will be assumed to be zero, whenever this hypothesis is not critical for the results we will to establish). It is clear that the lower we set the value of θ_j, the more states in class $[j]$ will have theirs relatives weights greater than θ_j which means that more fixed points the class $[j]$ will have. The desirable values for the, so to be called, capacity vector parameter $\bar{\theta} = (\theta_1, ..., \theta_n)$, may be obtained in an adaptive way. 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 $w(t) = (w_1(t), w_2(t), ..., w_n(t))$ associated to any state vector $x(t)$ may also be interpreted as a sort of frequency distribution of probabilities. The reason is that

$$\sum_{i=1}^{n} w_i(t) = 4 \;\Rightarrow\; \sum_{i=1}^{n} \frac{1}{4} \cdot w_i(t) = 1$$

For any relative weight vector $w(t)$. The "uniform distribution vector" would be the one with all its components equal to $4/n$, which mean that the relative weight vector may be interpreted as a sort of frequency distribution of probabilities, this distribution may be considered as the relative weight vector associated to that state.

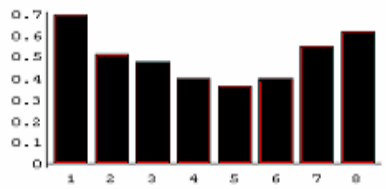

Figure 1 Relative Weight Vector associated to a certain state x, in a space of dimension 8.

5. Application

Our algorithm has been used in several applications. In this paper, 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:

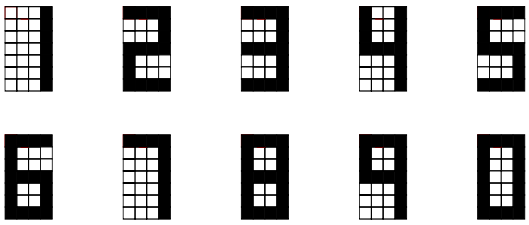

Figure 2 Arabian digits

Where the dimension n, of the pattern space is 28, and ξ^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], ξ^2 = [1,1,1,1,0,0,0,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,0,0,0,1,1,1,1,1], and so on. After training, the weight vector is 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}.

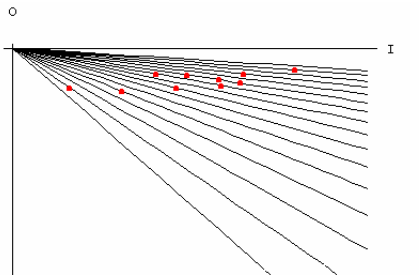

Figure 3 Arabian Digits Projections

In figure 3 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}. And 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}. 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 before dynamic equation 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 using. 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 a point 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. In our example, the number 2 is a prototype with 16 units components, in other words its energy projection is placed in the line r_{16} and the capacity parameter θ_{16} is equal to 0.0741306. If a little noise is added to the pattern (pattern in figure 4) and this noisily pattern (belonging to class r_{15}) is given for retrieving:

It may happen that the noisily pattern changes to other state quite different from its natural attractor (the number 2) The reason for that, is that the prototype number 6, see figure 5, belongs also to the class r_{15} and the stability condition in this class was set very high.

Figure 4 Noisily Prototype 2 belonging to r_{15}

Figure 5 Prototype 6 belonging to r_{15}

The question is how to get that only the second component changes its state, when the Noisily Prototype 2, is given for retrieving. In other words, how to get all the neighbors (inside a give radius) of a prototype to be attracted by this prototype, see figure 6. The idea, proposed in this paper, made use of the deviation defined in [Giménez 2000]. 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 μ. The coefficient μ is handled in a dynamical way (the more is the time the higher is the coefficient).

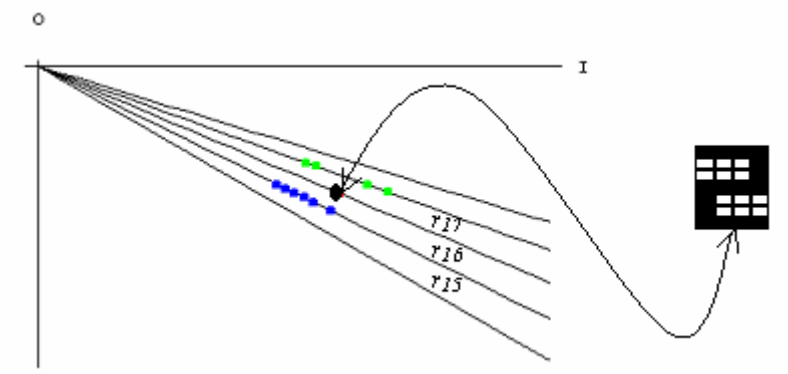

Figure 6 Neighbor of prototype 2 in r_{15} and r_{17}

Besides the weight vector, there is other set of parameters of the net. For every one class r_i, the capacity parameter θ_l 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 7, the number of points inside a neighborhood of radius 1, of the prototypes are expressed.

24	→	1	22	→	6
23	→	2	25	→	7
25	→	3	25	→	8
22	→	4	22	→	9
27	→	5	21	→	0

Figure 7 Prototype 6 belonging also to r_{15}

6. 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 an 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

[Hopfield 82.] J. J. Hopfield. Neural Networks and physical systems with emergent collective behavior. Proc. Natl. Acad. Sci. USA, 79:2554, 1982.

[Kinzel 85] W. Kinzel, Z. Phis, (B-Condensed Matter) Learning and pattern recognition in spin glass models, vol.60, pp.205-213, 1985

[Elice 87] R. Mc. Elice, E. Posner, E. Rodemich, and S. Venkatesh. The capacity of the Hopfield associative memory. IEEE Trans. On Information Theory, vol.IT-33. pp.461-482, 1987.

[Bose 96] N. K. Bose and P. Liang. Neural Network Fundamentals with Graphs, algorithms and Applications. McGraw Series in Electrical and Computer Engineering.1996.

[Giménez 97] V. Giménez-Martínez, M. Pérez-Castellanos, J. Ríos Carrión and F. de Mingo, Capacity and Parasitic Fixed points Control in a Recursive Neural Network, Lecture Notes in Computer Science, SPRINGER-VERLAG, 1997, pp.215-226

[Giménez 2000] V. Giménez-Martínez. A Modified Hopfield Algorithm Auto-Associative memory with Improved Capacity, IEEE Transactions on Neural Networks, vol.11,n.4, 2000, pp. 867-878.

[Giménez 2001] Giménez-Martínez V., Erviti Anaud J. and Pérez-Castellanos M.M, Recurrent Neural Networks for Statistical Pattern Recognition, Frontiers in Artificial Intelligence and Applications, Vol.69, part 1, n.3, 2001, pp.1152-1159.

[Giménez 2001] Giménez-Martínez V., Aslanyan L., Castellanos J.and Ryazanov V., Distribution functions as attractors for Recurrent Neural Networks Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications, vol.11, n.3, 2001, pp.492-497.

Authors' Information

V. Giménez-Martínez – Dep. de Matemática Aplicada.

C. Torres – Dep. de Matemática Aplicada.

J. Erviti Anaut – Dep. de Matemática Aplicada.

M. Perez-Castellanos – Dep de Arquitectura y Tecnología. de Sistemas

Facultad de Informática, U.P.M. Campus de Montegancedo s/n Boadilla del Monte, 28660 MADRID, SPAIN

e-mail: vgimenez@fi.upm.es