Real Time Dynamic Neural Network (RTDNN)

Álvaro Sánchez Miralles, Miguel Ángel Sanz Bobi, *Member, IEEE* [1]

***Abstract*—This paper describes a new neural network able to adapt itself, both its parameters and its structure, to a data set in real-time conditions. The adaptation is based on a non-supervised learning procedure. The new neural network can automatically create interconnections between neurons using any generic activation function. Still another important feature of this new neural network is the use of few neurons to make a good prediction using a reduced number of examples. This is relevant in order to make fast calculations using few resources in real-time applications. Some examples using this neural network are included in order to demonstrate its good performance. These examples use elliptical Gausian functions as domains for the neurons.**

***Index terms*—real-time neural network, neural network self-adaptation, dynamic neural network (DNN), topologies representing network (TRN), elliptical Gaussian domain of neurons, radial basis function network (RBFN), probability density function (PDF)**

I. Introduction

As is already known, there are many algorithms based on non-supervised learning for the training of neuronal networks. Their applications are diverse, such as clustering [1], density function estimation [2], aid to the supervised learning [3] [*4*] and networks for topology representation [*5*].

Some of these algorithms are "*k-means"* [*6*] and "*Neural Gas"* [*7*]. They require the making of an *a priori* decision concerning the number of neurons that the neural network will use before its training. Therefore, they have little ability for adaptation under unknown scenarios for the designers.

An alternative could be neural networks that dynamically change their structure such as "*Growing Neural Gas"* [8], "*Growing Cell Structures"* [*9], "Dynamic TopologyRrepresenting Networks"* [*10*]. Also, they can create interconnections between neurons. This type of neural networks are referred to as "*topology-representing networks"*. However, they have several weak points: the training is controlled by a great number of parameters that can be changed, they need a large data set to calculate an acceptable solution and most importantly, they generate complex neural networks with a large number of neurons. In counterpart, their algorithms are prepared to run in real time and are able to adapt their structure under new situations.

The previously mentioned types of algorithms can be used to aid the supervised learning process of RBFN networks [3] [4], where the centers of the radial units are fixed by these algorithms, the width of the radial unit is calculated through heuristic rules and finally an optimization algorithm is applied. The weak point is to calculate the width of the radial unit through heuristic rules, which sometimes entails falling into a local optimum.

On the other hand there are learning algorithms for the estimation of density functions (FDP) [11]. These are based on the optimization of the logarithmic probability of a data set. Usually they need to limit the maximum number of neurons to be used, in order to prevent a perfect learning situation by a continuous addition of neurons with the consequent lost of efficiency and generalization capability. Some papers [12] recommend the use of a function that weighs the quality of the model and the cost to add a new neuron. However, this is a laborious process since there are problems that require a greater number of neurons, due to the complexity of the data distributions.

In this paper a new Real-Time Dynamic Neural Network (RTDNN) is proposed which is able to adapt, both its parameters and its structure, to a data set in real-time conditions without previous knowledge. The training process is non-supervised and it would be based on existing soft competitive learning techniques [5]. It does not require a great number of samples to carry out a good adaptation. The RTDNN consists of a reduced number of neurons with generic activation function. It is mandatory that this activation function be convex and with axial symmetry in respect to its own axes (they do not have to coincide with the co-ordinate axes). An example of such a function is the elliptical Gaussian function that

[1] Universidad Pontificia de Comillas
Escuela Técnica Superior de Ingeniería- ICAI
Instituto de Investigación Tecnológica
Santa Cruz de Marcenado 26, 28015 Madrid, SPAIN. alvaro@iit.upco.es

will be used in the examples included in this paper. This type of activation function allows to obtain a good adaptation, requiring few neurons and great generalization capability. In addition to this, the neurons can be interconnected to each other. It facilitates the creation of clusters and the representation of topologies. The RTDNN is designed to be adapted with each sample that is collected. All these characteristics of the RTDNN are useful for:

- Applications running in real-time.
- Applications using an unknown size of the data set.
- Applications without knowledge available.
- Applications for clustering and construction of topological maps.
- Estimation of density functions.
- Applications of data mining and case based reasoning.

This paper is organized in the following sections. First, the RTDNN is described, including its structure and its main parameters. The next section presents the algorithm used by the RTDNN for training. Finally, some examples about the performance of the RTDNN are included.

II. Description of the RTDNN

The objective of this section is the description of the RTDNN. A special notation will be used in order to simplify the RTDNN description where its structure is presented in Figure 1.

A. RTDNN Notation

M: Workspace dimension.

R : Neural network workspace of dimension M.

R_i : Workspace of the neuron i.

I : Number of neurons in the neural network.

NN_I : Set of I neurons. Each element of this set is called nn_i.

N : Number of sample vectors of dimension M each one, that belong to the workspace R.

$\mathbf{X}_N$: Set of N sample vectors of dimension M, that belong to the workspace R. Each sample vector is called x_i.

K_i : Number of sample vectors of dimension M, that belong to the workspace R_i.

$\mathbf{X}^i_{K_i}$: Set of K_i sample vectors of dimension M, that belong to the workspace R_i . Each sample vector of this set is called $\mathbf{x}^i_j$ with components x^i_{jk} .

$\boldsymbol{\mu}^i_{K_i}$: Mean vector of the K_i sample vectors of dimension M, that belong to the workspace R_i.

$\boldsymbol{\Omega}^i_{K_i}$: Covariance matrix of the K_i sample vectors of dimension M, that belongs to the workspace R_i. This matrix has dimension MxM. Each element of this matrix is called $v^i_{(x,y)\cdot K_i}$; being *x* the row and *y* the column of the matrix to which the element belongs.

R_e : Input region.

H : Number of sample vectors of dimension M that excite the neural network.

$\mathbf{X}^e_H$: Set of H sample vectors of dimension M that excite the neural network; in another words they belong to the input region. This is the input data set to the neural network. Each sample vector of this data set is called $\mathbf{x}^e_i$.

$\boldsymbol{\mu}^e_H$: Mean vector of the H sample vectors of dimension M that excite the neural network.

$\boldsymbol{\Omega}^e_H$: Covariance matrix of the H sample vectors of dimension M that excite the neural network.

B. RTDNN Description

Let be $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^N$ a sample vector set belonging to R with $R \subset \Re^M$ and $\mathbf{x}_i = \{x_{ij}\}_{j=1}^M$. The information included in this vector will be represented by the following neural network $NN_I = \{nn_i\}_{i=1}^I$ that is a RTDNN, see Figure 1.

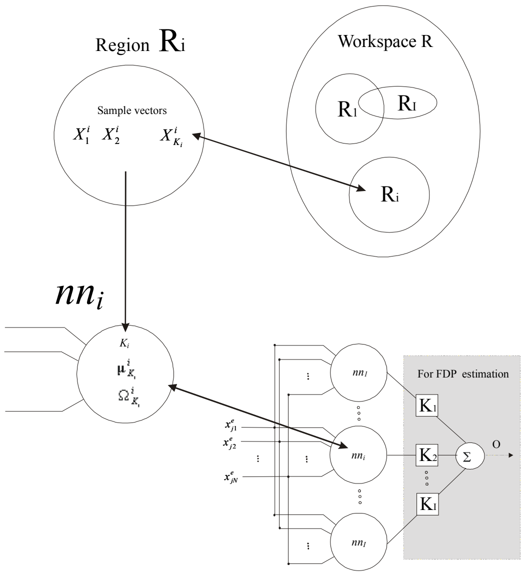

Figure 1: RTDNN structure

Each neuron *nn_i* in the RTDNN is specialized in a particular variable region of the workspace R_i with $R_i \subset R$. The original sample vectors included in this region,

$\mathbf{X}_{K_i}^i = \{\mathbf{x}_j^i\}_{j=1}^{K_i} / \mathbf{x}_j^i \in R_i$ will be characterized by the following RTDNN parameters :

- The number of sample vectors that belongs to region K_i
- One vector $\boldsymbol{\mu}_{K_i}^i$ representative obtained by the estimation of the mean values of the sample vectors of that region.

$$\boldsymbol{\mu}_{K_i}^i = \sum_{j=1}^{K_i} \frac{\mathbf{x}_j^i}{K_i}$$

- A set of quadratic distances in respect to the mean in each region $\boldsymbol{\Omega}_{K_i}^i$ such that

$$\boldsymbol{\Omega}_{K_i}^i = \begin{bmatrix} v_{(1,1)\cdot K_i}^i & \cdots & v_{(1,M)\cdot K_i}^i \\ \cdots & \cdots & \cdots \\ v_{(M,1)\cdot K_i}^i & \cdots & v_{(M,M)\cdot K_i}^i \end{bmatrix}$$

where

$$v_{(x,y)\cdot K_i}^i = \frac{\sum_{j=1}^{K_i} (\mathbf{x}_{x\cdot j}^i - \boldsymbol{\mu}_{x\cdot K_i}^i)\cdot(\mathbf{x}_{y\cdot j}^i - \boldsymbol{\mu}_{y\cdot K_i}^i)}{K_i}$$

According to the RTDNN parameters, all the information of a particular region could be substituted at an instant of time by M^2+M+1 parameters. They have the following meaning:

M: the M dimensional position of the i region $\boldsymbol{\mu}_{K_i}^i$
1: the number of samples in that region
M^2 : the M^2 parameters that measure the shape and size of the region.

These parameters suggest the use of activation functions with axial symmetry in respect to their main axes of information in the neurons. These axes could be or could not be the cartesian coordinates in the domain. This is a special RTDNN feature, instead of activation functions with total symmetry used by some neural networks. Although the number of parameters used in each region is greater than that required by other types of simpler activation functions, it allows for a lower number of neurons to make a good adaptation with reduction of time and resources required.

On the other hand the neurons can be interconnected to each other so that each neuron can have some neighboring neurons. These connections are quantified indicating the force of the same ones. In this manner, the interdependence of the neurons can be quantified to each other. This is an important advantage of the RTDNN.

III. RTDNN TRAINING

In this section the process followed to adapt the parameters of the RTDNN will be described. This learning procedure is based on the automatic fitting of the RTDNN parameters using a set of recursive calculations. According to these parameters, the excitation of each neuron can be measured and also, fusion of two regions dominated by two near neurons can be decided or not. All this information guides the RTDNN training process allowing for a dynamic and fast adaptation of the neural network structure and its parameters.

The structure of this section is the following. First, the recursive equations previously mentioned will be presented, next the measurement of the excitation of each neuron will be described, and also, the procedure to fuse regions belonging to near neurons. Finally, the RTDNN algorithm itself will be presented.

A. Recursive equations for estimation of the RTDNN parameters

The dynamic self-adaptation of the RTDNN is the result of a process of recursive cycles of reevaluation of its main parameters. The process requires a very short time for calculations and adaptation of the RTDNN structure to the new data coming during the training phase.

The RTDNN adapts each time that new examples excite it. This is based on a set of recursive equations that allow for a quick updating of its parameters. Next the set of recursive equations will be presented.

A.1 Number of examples that represent a neuron in a region i

The number of examples that represent a neuron of the region i is calculated as the known number of examples which are represented in the region i plus the number of new examples (n) falling in the region i:

$$K_i = K_i + n \qquad (1)$$

A.2 Mean value of k examples in a region i

The mean value of k examples in a region i is calculated using the mean values coming from the k-n examples existing in the region i and the n new examples falling in the region i.

Let this be the mean value of k examples:

$$\mu_k = \sum_{j=1}^{k} \frac{x_j}{k} \qquad (2)$$

and making some calculations

$$k \cdot \mu_k = \sum_{j=0}^{k-n} x_j + \sum_{j=1}^{n} x_j = (k-n)\cdot\mu_{k-n} + n\cdot\mu_n$$

finally a recursive equation is obtained:

$$\mu_k = \frac{k-n}{k}\cdot\mu_{k-n} + \frac{n}{k}\cdot\mu_n \qquad (3)$$

A.3 Quadratic distances of k examples in a region i

The quadratic distances of k samples in a region i can be obtained using the mean values and the quadratic distances of k-n examples and n examples in the region i.

Let

$$v_{(x,y)\cdot k} = \frac{\sum_{i=1}^{k}(x_{x\cdot i} - \mu_{x\cdot k})\cdot(x_{2i} - \mu_{2k})}{k} =$$

$$= \frac{\sum_{i=1}^{k}(x_{x\cdot i}\cdot x_{y\cdot i} - x_{x\cdot i}\cdot\mu_{y\cdot k} - x_{x\cdot i}\cdot\mu_{y\cdot k} + \mu_{x\cdot k}\cdot\mu_{y\cdot k})}{k}$$

This can be simplified taking into account equation (2):

$$v_{(x,y)\cdot k} = \frac{\sum_{i=1}^{k}(x_{x\cdot i}\cdot x_{y\cdot i})}{k} - \frac{\mu_{y\cdot k}\cdot\sum_{i=1}^{k} x_{x\cdot i}}{k} - \frac{\mu_{x\cdot k}\cdot\sum_{i=1}^{k} x_{y\cdot i}}{k} + \mu_{x\cdot k}\cdot\mu_{y\cdot k} =$$

$$= \frac{\sum_{i=1}^{k}(x_{x\cdot i}\cdot x_{y\cdot i})}{k} - \mu_{x\cdot k}\cdot\mu_{y\cdot k}$$

This can be developed as follows:

$$k\cdot v_{(x,y)\cdot k} = \sum_{i=1}^{k-n}(x_{x\cdot i}\cdot x_{y\cdot i}) + \sum_{j=1}^{n}(x_{x\cdot j}\cdot x_{y\cdot j}) - k\cdot\mu_{x\cdot k}\cdot\mu_{y\cdot k}$$

$$k\cdot v_{(x,y)\cdot k} = (k-n)\cdot(v_{(x,y)\cdot k-n} + \mu_{x\cdot k-n}\cdot\mu_{y\cdot k-n}) + \\ + n\cdot(v_{(x,y)\cdot n} + \mu_{x\cdot n}\cdot\mu_{y\cdot n}) - k\cdot\mu_{x\cdot k}\cdot\mu_{y\cdot k}$$

Finally, the following recursive equation allows for the estimation of the quadratic distances of k examples:

$$v_{(x,y)\cdot k} = \frac{k-n}{k}\cdot(v_{(x,y)\cdot k-n} + \mu_{x\cdot k-n}\cdot\mu_{y\cdot k-n}) + \\ \frac{n}{k}\cdot(v_{(x,y)\cdot n} + \mu_{x\cdot n}\cdot\mu_{y\cdot n}) - \mu_{x\cdot k}\cdot\mu_{y\cdot k} \quad (4)$$

B. Excitation of a neuron in the RTDNN

The neurons in the RTDNN are excited by a set of sample vectors $\mathbf{X}_H^e = \{\mathbf{x}_i^e\}_{i=1}^{H}$. These samples belong to the input region R_e, which is characterized by the same parameters as the other regions: the number of input sample vectors H, the mean vector of input examples $\boldsymbol{\mu}_H^e$ and a set of quadratic distances with respect to the average $\boldsymbol{\Omega}_H^e$ in the region. The diagonal parameters of the matrix $\boldsymbol{\Omega}_H^e$ must be less than a minimum resolution called v_{max}^e. If the set of excitation samples were a single sample (H=1), they would be characterized by the average $\boldsymbol{\mu}_H^e$ and the diagonal matrix $\boldsymbol{\Omega}_H^e$ with all elements equal to v_{max}^e.

The excitation degree of the i-th neuron is measured using the following equation:

$$exc_i = \sqrt{\frac{1}{(\boldsymbol{\mu}_H^e - \boldsymbol{\mu}_{K_i}^i)^T \cdot (\Omega_{K_i}^i)^{-1} \cdot (\boldsymbol{\mu}_H^e - \boldsymbol{\mu}_{K_i}^i)}} \quad (5)$$

This excitation is labeled according to the following criteria:

- Maximum excitation: If $exc_i >= exc_{max}$
- Minimum Excitation: If $exc_{max} > exc_i >= exc_{min}$.
- Non excitation: If $exc_i < exc_{min}$.

In the case of the use of an elliptical Gaussian activation function in the neurons, the following criteria can be taken:

- $exc_{min} = 1/3$, since below this excitation level is 1% of the data, and it is considered that the sample is outside the neuron.
- $exc_{max}=1$, since above this excitation level is 68% of the data, and any sample that has maximum excitation reinforces the logarithmic probability of the neuron, the standard deviation of the same diminishes.

Conceptually, the neuron excitation is a measure of the inverse of the normalized distance between the center of neuron and the mean of the input samples. It is used a normalized distance because the measure of excitation must be independent of the neuron size. In that way it could be obtained a measure of the neuron size using this normalized distance in order to control the growing of the neuron. Next it is explained how to obtain this measure.

Let be a neuron nn_i and one M-dimensional point **y** (**y** could be any point in the workspace, such as $\boldsymbol{\mu}_H^e$ o $\boldsymbol{\mu}_{K_i}^i$). It could be defined the distance between the neuron and the point **y**, which is denoted by $v_{K_i}^i(\mathbf{y})$, as

$$\mathbf{v}_{K_i}^i(\mathbf{y}) = \frac{(\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)^T \cdot (\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)}{(\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)^T \cdot (\boldsymbol{\Omega}_{K_i}^i)^{-1} \cdot (\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)}$$

$v_{K_i}^i(\mathbf{y})$ is called equivalent quadratic distance of the neuron nn_i from **y**. The relationship between the excitation and $v_{K_i}^i(\mathbf{y})$ is as follows:

$$\frac{1}{exc_i} = \frac{distance(\mathbf{y}, \boldsymbol{\mu}_{K_i}^i)}{\sqrt{\mathbf{v}_{K_i}^i(\mathbf{y})}} = \frac{\sqrt{(\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)^T \cdot (\mathbf{y} - \boldsymbol{\mu}_{K_i}^i)}}{\sqrt{\mathbf{v}_{K_i}^i(\mathbf{y})}}$$

The graphical interpretation of this concept is applied to a bidimensional Gaussian activation function in Figure 2. As it is showed in the figure the measure is the variance of the Gaussian in the direction formed by the center of the neuron $\boldsymbol{\mu}_{K_i}^i$ and the point **y**, which is a measure of the neuron size in that direction.

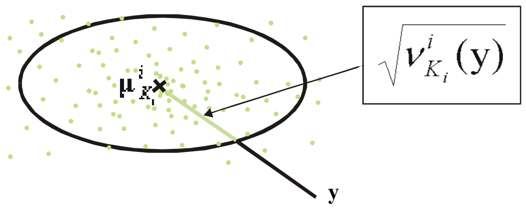

Figure 2: $v_{K_i}^i(\mathbf{y})$ interpretation

C. Fusion of neurons in the RTDNN

The fusion of neurons plays an important role inside the procedure of the RTDNN training. It allows for the reduction of the number of neurons by fusion of several ones, and for the saving of memory, resources and time of calculations. This process is oriented to be applied during the training phase of the RTDNN, but it is also possible to apply it to neural networks already trained. It is important to enhance its simplicity and efficiency.

The fusion process is based on looking for two neurons which are specialized in regions of the workspace very similar, if they are found. Then, it fuses these two neurons in a new one that is specialized in a region of the workspace resulting from the union of the previous two regions. Often this fusion cannot be carried out since the region corresponding to the resulting neuron is too great, in which case the fusion is not carried out. The parameter that delimits the size of the neurons is named v_{max}. This parameter regulates the number of neurons in the neural network according to the accuracy required for the characterization of the information included in the different examples. In the case that the number of neurons is great because v_{max} is small, it is best to apply the fusion process with a greater v_{max}. This will allow for the reduction of the number of neurons. This parameter v_{max} is unidimensional, however it acts in all the dimensions. In order to apply this concept, the equivalent quadratic distance from **y** is used so that $v^{i}_{K_i}(\mathbf{y}) < v_{max}$. In the case of fusion of two neurons nn$_i$ and nn$_j$, it is verified that the resulting neuron nn$_f$ fulfils

$$v^{f}_{K_f}(\boldsymbol{\mu}^{i}_{K_i}) < v_{max} \text{ and}$$
$$v^{f}_{K_f}(\boldsymbol{\mu}^{j}_{K_j}) < v_{max} \quad (6)$$

Several steps must be followed in the fusion process. Following is a complete list of these steps:

Let a set of neurons $NNF \subset NN$ be proposed to be fused:

1. Two close neurons are selected $nn_i, nn_j \in NNF$
2. It is estimated
$$d = \frac{\sqrt{(\boldsymbol{\mu}^{j}_{K_j} - \boldsymbol{\mu}^{i}_{K_i})^T \cdot (\boldsymbol{\mu}^{j}_{K_j} - \boldsymbol{\mu}^{i}_{K_i})}}{\sqrt{v^{i}_{K_i}(\boldsymbol{\mu}^{j}_{K_j})} + \sqrt{v^{j}_{K_j}(\boldsymbol{\mu}^{i}_{K_i})}}$$
3. If $d < \frac{exc_{min}}{2}$, the neurons are fused applying equations (1)(3)(4). If the neurons are similar ($v^{i}_{K_i}(\boldsymbol{\mu}^{j}_{K_j}) \approx v^{j}_{K_j}(\boldsymbol{\mu}^{i}_{K_i})$), means that one neuron center excite the other;
$$\sqrt{\frac{1}{(\boldsymbol{\mu}^{j}_{K_j} - \boldsymbol{\mu}^{i}_{K_i})^T \cdot (\Omega^{i}_{K_i})^{-1} \cdot (\boldsymbol{\mu}^{j}_{K_j} - \boldsymbol{\mu}^{i}_{K_i})}} > exc_{min}$$
4. If equation (6) is not fulfilled, the fusion can not be done.
5. Return to point 1 until every combination of pairs of neurons belonging to NNF has been tried.

In order to increase the speed of the algorithm and to obtain better solutions, it is suggested to use, in the third point, an equation that depends on the number of examples that the neuron represents (K$_i$) and depending on a low number of examples, to increase the possibility of fusion.

D. Training algorithm of the RTDNN

The training algorithm is based on what is exciting to the neural network, and on observing the neurons that surpass a certain level of excitation (exc$_{min}$). In the case that there is no neuron that surpasses this minimum level of excitation, this means that there is a region non-modeled by the network and therefore a neuron is added. On the other hand if there is some neuron that surpasses the minimum level of excitation, its parameters are updated. This updating is not always possible since it is not allowed that the neurons include a very great region of space, in which case a new neuron is added. According to the number of neurons growing, they are merged with others. The fusion algorithm between neurons is one of the key parts of the training. This algorithm tries to fuse two neurons when it detects that the same ones specialize in space regions that are very similar.

The RTDNN parameters controlling the training procedure are v^{e}_{max} and v_{max}. v^{e}_{max} represents the resolution of the neural network. This parameter must only be considered when the training of the neural network is off-line with the disordered data. At the time of applying the algorithm in real time it is not necessary to fix v^{e}_{max} since it is already fixed by its own effect from the sampling of the application.
The parameter v_{max} represents the accuracy of the neural network. When v_{max} is greater, the accuracy of the neural network will be greater, but the capability of generalization will be smaller. This parameter must be fixed beforehand in any case and can be changed during the network training. It is recommended to use a small value of v_{max} at the beginning of the training (this can slow down the training), so that the network does not lose great accuracy, and to use a big value of v_{max} at the end of the training.

The training algorithm of the RTDNN consists of the following steps:

1. Let H sample vectors $\mathbf{X}^{e}_{H} = \{\mathbf{x}^{e}_{i}\}^{H}_{i=1}$ that excite the neural network. Estimate $\Omega^{e}_{H}, \boldsymbol{\mu}^{e}_{H}$, If there are not any samples left, then go to point 10.
2. The excitation due to the neural network inputs is calculated for each neuron as such:
$$exc_i = \sqrt{\frac{1}{(\boldsymbol{\mu}^{e}_{H} - \boldsymbol{\mu}^{i}_{K_i})^T \cdot (\Omega^{i}_{K_i})^{-1} \cdot (\boldsymbol{\mu}^{e}_{H} - \boldsymbol{\mu}^{i}_{K_i})}}$$
3. Select the most excited neuron, which has an excitation level that surpasses exc$_{max}$. This neuron will be called nn_{d_1} ($nn_{d_i} / exc_{d_i} > exc_{max}$). If there is no any neuron go to point 5.
4. Update the neuron parameters according to the equations (1)(3)(4). Go to point 1

5. Select the neurons which have an excitation level greater than the exc_{min} threshold. This set of neurons will be called D2. $D2 = \{d2_i\}_{i=1}^{ND} / exc_{d2_i} > exc_{min}$. If there is no neuron, go to point 9.
6. Select the neurons from set D2 that could grow up: $D3 \subset D2, D3 = \{d3_i\}_{i=1}^{ND3} / v_{K_{d3_i}}^{d3_i}(\mu_H^e) < v_{max}$ If there are no neurons, then go to point 8.
7. Update the neuron parameters of set D3 according to the equations (1)(3)(4). Next it should be propose a fusion among the neurons belonging to set D2. Make all neurons belonging to D2 neighbors among themselves. Go to point 1.
8. Add one neuron with the entries parameters but influenced by the neuron more excited of the set D2 (any reasonable criterion is right, it is optional). Make all neurons belonging to D2 neighbors among themselves. Go to point 1.
9. Add one neuron with the entries parameters. Go to point 1.
10. A fusion should be proposed between all neurons.

Anything previously forgotten in the network knowledge could be introduced. It is only necessary to substitute the equation (1) for the (7) one:

$$K_i = \frac{x-H}{x} K_i + H \quad (7)$$

where x is the number of samples that the neuron needs to remember beginning at the last one.

IV. RTDNN Performance

Next some results showing the RTDNN performance will be presented. These results are based on two-dimensional examples for easier understanding. They use elliptical Gaussian as activation function in the neurons of the RTDNN.

The training consists of cycles where 5 examples by each one are taken (that is to say, H=5). From the parameters that regulate the training pays attention to $v_{max} = 1$. v_{max}^e is not necessary to fix, as was previously commented.

The next subsections include three cases of the RTDNN application to different data sets. These cases are explained using graphs where:

- the data to fit are drawn with dots in a light color.
-the neurons are represented by the center with a dark circumference and by the average distance of the samples to the center in all directions of the neuron work space (equivalent to the standard deviation in one dimension) with dots in a dark color.
- the connections between neurons are represented by straight lines.

A. Case 1

This case is based on the RTDNN adaptation to some examples obtained from a noisy hyperbolic tangent. Figure 3 shows the RTDNN resulting after the training process when v_{max} = 1. According to point 10 in section II.D, if $v_{max} \rightarrow \infty$, less neurons will be required saving some memory resources, this is represented in Figure 4.

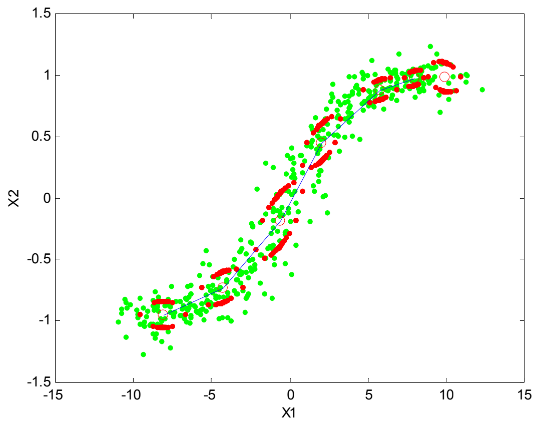

Figure 3. Adaptation to Tanh using v_{max} = 1

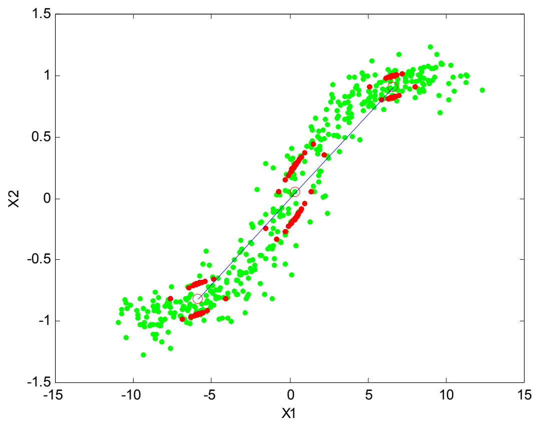

Figure 4: Adaptation to Tanh using $v_{max} \rightarrow \infty$

Using a hyperbolic tangent function with a higher slope in order to obtain examples to train a RTDNN, two clusters of neurons are discovered . They are shown in Figure 5.

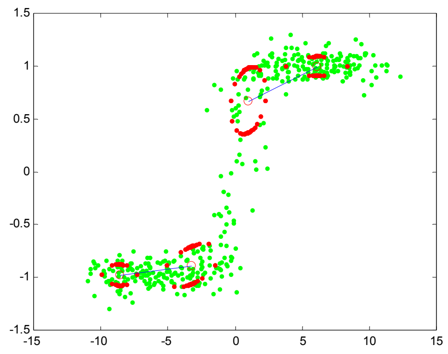

Figure 5: final adaptation

B. *Case 2*

This case shows successive training cycles during the RTDNN adaptation to a data set distributed following a semi circumference. Figure 6 shows the 13th training where the fitting is not good yet. .Figure 7 shows the 22nd training cycle and now shown. In this cycle the RTDNN adaptation is better.

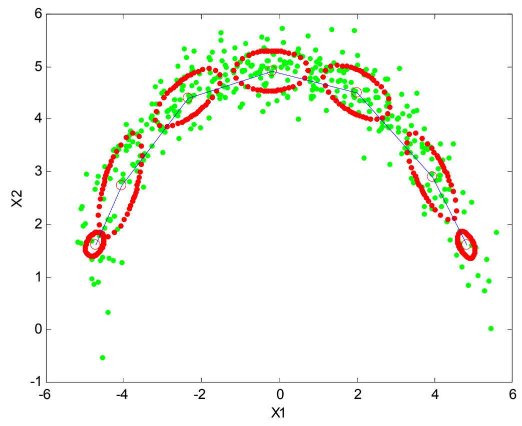

Figure 8 shows the last RTDNN training cycle. Figure 9 shows the reduction of information using $V_{max} \rightarrow \infty$.

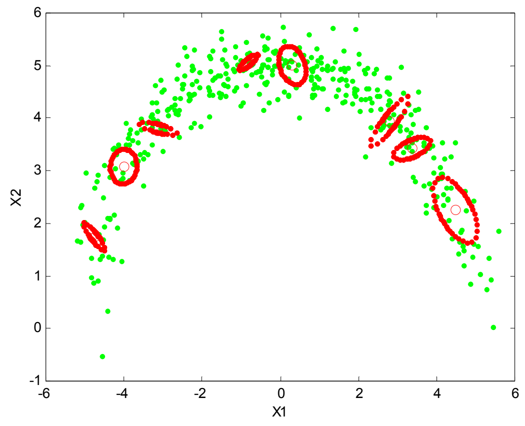

Figure 6: 13nd RTDNN training cycle

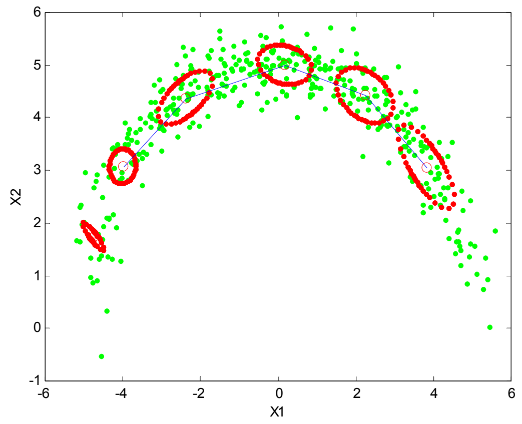

Figure 7: RTDNN 22nd training cycle

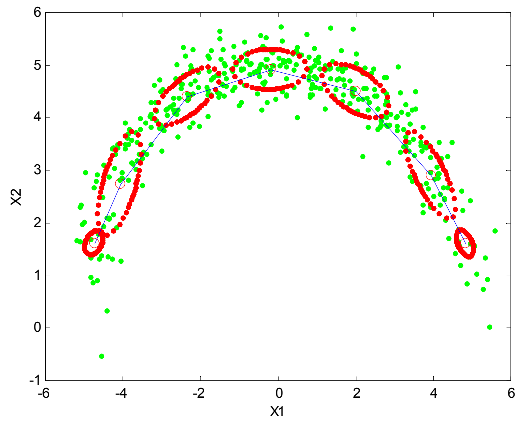

Figure 8: last RTDNN training cycle

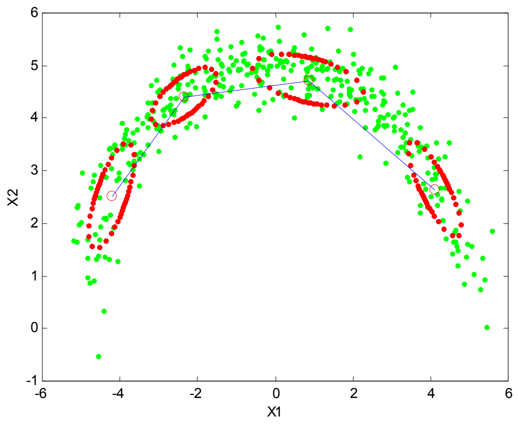

Figure 9: last RTDNN training cycle using $V_{max} \rightarrow \infty$

C. *Case 3*

This case illustrates how a RTDNN can estimate a probability density function. The input space is generated from an hyperbolic sine function. Figure 10 shows the final RTDNN adaptation, while Figure 11 represents the three dimensional probability density estimation.

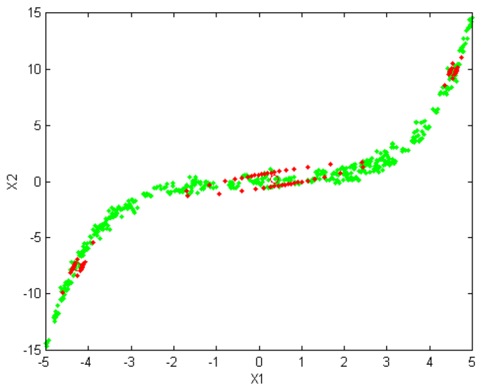

Figure 10: final RTDNN adaptation to Sinh

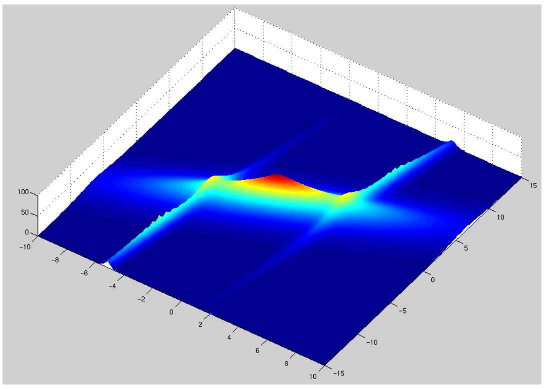

Figure 11: probability distribution

D. *Case 4*

Finally, Figure 12 shows a last example that illustrates a more complicated workspace. As it can be observed a good adaptation is also obtained.

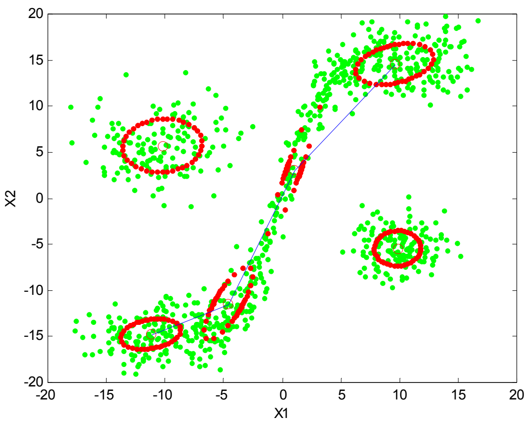

Figure 10 : final adaptation

V. CONCLUSIONS

A new neural network named Real Time Dynamic Neural Network (RTDNN) with generic activation function has been set out. It is able to adapt itself, both parametrically and structurally, to the data necessities, which allows to work with a data set of indeterminate and changing size.

In addition to this, a training algorithm has been developed which allows for the adaptation of the network on line and in real time.

It must be noted that real time is a concept that depends on application and on computation machines. This article has proposed some recursive equations and a very optimized training method in order to apply the RTDNN to real time and on-line applications.

In future studies we will apply the RTDNN to environment modeling for the guidance of an AGV (Auto Guided Vehicle).

Furthermore, some experiments have been done with a neural network with Gaussian activation function in its neurons. The application to another type of activation function is possible in the same manner as described in this article.

VI. REFERENCES

[1] A. K. Jain and R.C. dubes. Algorithms for clustering data. Prentice Hall, 1988.
[2] A. Muñoz San Roque. Aplicación de técnicas de redes neuronales artificiales al diagnóstico de procesos

industriales. Tesis doctoral. Universidad Pontificia de Comillas. Madrid. 1996.

[3] J. E. Moody and C.Darken. Fast learning in networks of locally-tuned processing units. Neural Computation, 1989.

[4] B. Fritzke. Fast learning with incremental RBF networks. Neural Processing Letters,1(1):2-5,1994.

[5] T. M. Martinetz and K. J. Schulten. Topology representing networks. Neural Networks,1994.

[6] J. MacQueen. Some methods for classification and analysis of multivariate observations. Volume 1 of Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, pages 281-297, Berkeley, 1967. University of California Press.

[7] T. M. Martinetz and K. J. Schulten. A “neural-gas” network learns topologies. In T.Kohonen, K.Mäkisara, O.Simula, and J.Kangas, editors, Artificial Neural Networks, pages 397-402. North-Holland,Amsterdam,1991.

[8] B. Fritzke. A growing neural gas network learns topologies. In G.Tesauro, D.S. Touretzky, and T.K. Leen, editors, Advances in Neural Information Processing Systems 7, pages 625-632. MIT Press, Cambridge MA, 1995.

[9] B. Fritzke. Growing cell structures-a self organizing network for unsupervised and supervised learning. Neural Networks,7(9):1441-1460,1994.

[10] J. Si, S. Lin, M.-A. Vuong. Dynamic topology representing networks. Neural Networks 13 (2000) 617-627.

[11] K. A. Gernoth, J. W. Clark. Neural Networks that learn to predict probabilities: global models of nuclear stability and decay”, 1995.

[12] J. Barry Gomm, Ding Li Yu. Selecting Radial Basis Function Network Centers with Recursive Orthogonal Least Squares Training. IEEE Transactions on Neural Networks. Volume 11, NO. 2, Pages 306-314. March 2000.