Revista Mexicana de Física **39**, *No. 2 (1993) 235–249*

A new kind of neural network based on radial basis functions

M. SERVÍN AND F.J. CUEVAS
Centro de Investigaciones en Optica, A.C.
Apartado postal 948, León, Gto., México

Recibido el 16 de octubre de 1991; aceptado el 12 de enero de 1993

ABSTRACT. We have derived a new kind of neural network using normalized radial basis functions, "RBF", with the same classifying properties as if it were built up using sigmoid functions. This equivalence is mathematically demonstrated. In addition, to this, we also show that the proposed network is equivalent to a gaussian classifier. The network does not require any computing learning time to build a classifier. This network has been compared with well known adaptive networks, such as backpropagation and linear combination of generalized radial basis functions (GRBF's). Its adapted forms are presented to see how the classifying regions and boundaries among the supplied examples are formed. This neural network can be made to have identical classifying properties, as the nearest neighborhood classifier "NNC". In the case of having many examples per class, fewer centers can be found using vector quantizing "VQ" techniques as done in Kohonen's network. Finally, this neural system can also be used to approximate a smooth continuous function, given sparse examples.

RESUMEN. Hemos derivado una nueva clase de redes neuronales usando funciones de base radial normalizadas, "FBR", con las mismas propiedades de clasificación de una red construida con funciones sigmoides. Esta equivalencia es demostrada matemáticamente. Además, mostramos que la red propuesta es equivalente a un clasificador gaussiano. La red no requiere tiempo de aprendizaje para construir el clasificador. Esta red ha sido comparada con redes adaptivas conocidas, como el modelo de retro-propagación y una combinación lineal de funciones generalizadas de base radial. Se presenta cómo se forman las regiones de clasificación, dado un conjunto de ejemplos de entrenamiento. Esta red neuronal puede ser modelada para tener las mismas propiedades de clasificación de un clasificador del vecino más cercano. En el caso de tener varios ejemplos por clase, los centros pueden encontrarse usando técnicas de cuantificación de vectores tal como se usa en la red de Kohonen. Finalmente, este sistema neuronal también puede ser usado para aproximar una función suave continua, dados puntos dispersos.

PACS: 87.22.J

1. INTRODUCTION

Approximation and classification are currently the two most important tasks that neural networks deal with successfully. Since the work of Rumelhart *et al.* [1] on multilayer feedforward neural networks and sigmoid based networks of the backpropagation kind have been the most commonly used neural model for these tasks. Recently Poggio and Girossi [2], emphasized that generalized radial basis functions (GRBF) have many interesting properties to be considered seriously as more suitable basis functions to approximate and learn smooth continuous functions from sparse data. The two main learning procedures mentioned by Poggio and Girossi are: the simple gradient descent on the network's parameter space for on-line learning (used for backpropagation, too), and the Moore-Penrose pseudoinverse.

However, given a classifying task to be performed, backpropagation classifiers need in general, fewer neurons to classify an input set of feature vectors **X** in the required classes than a linear combination of GRBF's. Let us take for example, the simplest case of classifying the entire feature space into two sharply separated regions using a plane, only one high gain sigmoid will be needed to fulfill the above requirement. However, the number of neurons required using a linear combination of GRBF's to solve the same classifying task could be too high. That is one reason why most classifying problems have generally been solved using sigmoidal based classifiers.

Unfortunately, the backpropagation algorithm for training these sigmoid based networks is quite slow. Moreover, in general, there is no way to know in advance the number of sigmoids the neural system will require to fulfill a given classifying task. The number of hidden neurons is made large enough to allow for such uncertainties, therefore often constituting a waste of computing power and overfitting.

The network described in this work is based on RBF's centered at the data points when the number of templates per class is not very large. Moreover it is possible to have fewer RBF's than examples using vector quantizing "VQ" techniques [10,11] to distribute their centers among the data. The approximation and classification of the proposed network are equivalent to those based on sigmoid functions. In other words, the feature space can be filled by back to back hyperpolygons enclosing a unique class forming a Voronoi tessellation map [3]. The only difference is that, for sigmoid networks, several hyperplanes (neurons) are needed to build up a closed convex region.

There is yet another interesting property of the proposed neural network, using smooth RBF neurons. Making those RBF neurons less selective, a smooth continuous function will pass through or near the given set of sparse examples.

The learning procedure can be regarded from two points of view. The classical one, where the examples arrive stochastically in time, so the network learns the given mapping on-line, and the off-line learning, where the complete set of examples and their correct classification are given at once.

In the earlier years of neural networks [4], on-line learning was an important motivation for the development of a neural network theory. The on-line learning method is dynamic, that is, the network dynamically wires itself to reduce the error between its current output and the desired one, often in real time. Due to this, the most widely used method of learning has been: simple gradient descent. On the other hand, when we have off-line learning, that is, when it is possible to wait until several examples and their correct classification has been given, then gradient descent is not generally the best learning procedure. The reason for this, is the fact of having all the data at once, we implicitly have much more information, so it is then possible to use this information surplus to get much faster learning algorithms because we can see the future, the present, and the past simultaneously. Classical approximation methods can be used to achieve such improved learning: the Moore-Penrose pseudoinverse [2], or the conjugate gradient descent. The proposed network described in this work, is built off-line. In the pattern recognition field it is very often possible to have several representative templates in advance along with their corresponding correct classifications. In this case, classical approximation and well known classifying techniques [10] can be used to find the appropriate classifying network much faster than simple gradient descent.

2. Theoretical basis for the proposed neural network

Our starting point to develop a network in which the processors work toward a common goal is statistical mechanics "SM". In SM every individual (*i.e.* atoms, molecules or neurons) will change its state to reduce an energy function called the free energy, F, of the system [5], until it reaches its minimum. This state can change stochastically [6] as normally occurs in nature, or using a deterministic dynamic system to find local minima close enough to the global one to be considered as acceptable solutions to the problem at hand.

Because of the simple form of the internal energy proposed for the classifying task, the minimum of the proposed free energy can be found in just one iteration, that is, no dynamical system is required to find it.

The free energy F of a thermodynamic system at constant temperature can be expressed as [5]

$$F = E - TS, \tag{1}$$

where E, the internal energy, translates into mathematical terms the kind of task we are dealing with. In pattern recognition, the Euclidean distance in the feature space can be used as a measure of similarity between the incoming data to be classified and the templates used to train the network. The parameter T is the absolute temperature and it gives us an indication of how well the task represented by E can be achieved. The last term is called the entropy S of the system.

Let us suppose we choose M real valued n-dimensional vectors $\mathbf{A}_i = (a_{1i}, a_{2i}, \ldots, a_{ni})$ as being our set of templates. These templates are chosen to be a complete and representative set of examples in order to guarantee (like any other pattern recognition system) a reasonably good performance of the system. Additionally, we associated M real valued classes $\mathbf{Y} = (y_1, y_2, \ldots, y_m)$, with each of the M templates. We assume we are given, or we can calculate an estimate of the spread-out of a given class around its template. This class-region spread will be expressed as $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$.

Using the above defined terms we can write a free energy for the classifying system as

$$F(\mathbf{X}) = \sum_{i=1}^{M} \frac{(\mathbf{X} - \mathbf{A}_i)^T(\mathbf{X} - \mathbf{A}_i)}{\sigma_i} P(\mathbf{X} \in \mathbf{A}_i) + T \sum_{i=1}^{M} P(\mathbf{X} \in \mathbf{A}_i) \log P(\mathbf{X} \in \mathbf{A}_i), \tag{2}$$

providing that

$$\sum_{i=1}^{M} P(\mathbf{X} \in \mathbf{A}_i) = 1, \tag{3}$$

where $P(\mathbf{X} \in \mathbf{A}_i)$ is the probability that gives us a measure to decide whether an observed feature vector $\mathbf{X}$ belongs to the region surrounding the template $\mathbf{A}_i$.

In fact, the term in the internal energy expression which multiplies $P(\mathbf{X} \in \mathbf{A}_i)$, can have many other mathematical forms with the only condition of being a monotonic increasing function of the distance to the template. Changing this isotropic function the network will be based on different RBFs. Furthermore, we can even find a representation based on non-RBFs by using an anisotropic monotonic increasing function. In consequence, the network can be based on a variety of functions. The one presented here is only one type among an infinite number of different possibilities.

The class to which the observed feature vector $\mathbf{X}$ will be assigned, is given by the following mathematical expected value:

$$y(\mathbf{X}) = \sum_{i=1}^{M} y_i P(\mathbf{X} \in \mathbf{A}_i). \tag{4}$$

Now we have to find the expression for $P(\mathbf{X} \in \mathbf{A}_i)$, which minimizes Eq. (2) subject to the normalizing, condition given by Eq. (3). Doing this we obtain

$$P(\mathbf{X} \in \mathbf{A}_i) = \frac{\exp\left[-\frac{1}{T\sigma_i}(\mathbf{X} - \mathbf{A}_i)^T(\mathbf{X} - \mathbf{A}_i)\right]}{\sum_{j=1}^{M} \exp\left[-\frac{1}{T\sigma_j}(\mathbf{X} - \mathbf{A}_j)^T(\mathbf{X} - \mathbf{A}_j)\right]}. \tag{5}$$

Finally, Eqs. (4) and (5) constitutes our classifying system.

Having chosen (or estimated from the examples) the relative region's hypervolumes σ_i's, a low value for the T parameter will force the network to behave similarly to a multilayer high gain sigmoid based neural network, so sharp classifying boundaries are formed with the region's hypervolumes, which are proportional to σ_i. On the other hand, a high value of this parameter T will make the network behave as formed by a linear combination of broad Gaussians centered at the examples, so a smooth approximation curve is then obtained. The topology of the network is shown in Fig. 1.

If the network is used as a classifier, then the user should set the temperature parameter T to the lowest possible value to get the sharpest boundaries among the classes.

The ambiguity in choosing the temperature parameter T arises when dealing with smooth approximation, because for an infinite T, the approximating function becomes a constant real value equal to the average value of the sparse examples (infinite smoothness). So an additional criterion should be supplied to find the "best" value for T to set the amount of desired smoothness. Due to the wide range of applications this approximation can be used for, no particular criterion has been given and the user should find the most appropriate one to find the "best" T depending upon its particular application.

Making all the σ_i's equal, and using a very low value for the T parameter, the classifying properties of the network become those of a NNC because it divides the feature space exactly in the same way as a NNC, so any figure of merit, limitation or application reported concerning the NNC, will directly be applied to this network. Also the feature space segmentation will be that of a Voronoi tessellation map.

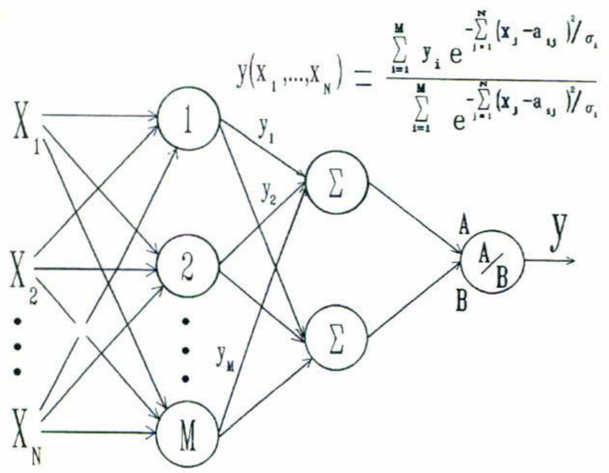
$y(x_1,\dots,x_N) = \frac{\sum_{i=1}^{M} y_i e^{-\sum_{j=1}^{N}(x_j-a_{ij})^2/\sigma_i}}{\sum_{i=1}^{M} e^{-\sum_{j=1}^{N}(x_j-a_{ij})^2/\sigma_i}}$
X_1
X_2
X_N
1
2
M
y_1
y_2
y_M
Σ
Σ
A
B
A/B
y

FIGURE 1. Topology of the proposed neural network. Each circle in the first hidden layer represents neurons having gaussian kind of response centered at the training examples.

In the case of clustered, noisy data, we can reduce network size by finding the mean and deviation for clusters formed by the feature vectors and assign one neuron per cluster. For a given class there may be several clusters that can be sufficiently close to be considered as connected and these connections can make up class regions with complicated forms. Having this in mind, we can rewrite Eq. (5) as function of these local averages and variances as

$$Y(\mathbf{X}) = \frac{\sum_{i=1}^{M} y_1 \exp\left[-\frac{1}{T\sigma_i}(\mathbf{X}-\mathbf{E}_{\mathbf{A}_i})^T(\mathbf{X}-\mathbf{E}_{\mathbf{A}_i})\right]}{\sum_{i=1}^{M} \exp\left[-\frac{1}{T\sigma_i}(\mathbf{X}-\mathbf{E}_{\mathbf{A}_i})^T(\mathbf{X}-\mathbf{E}_{\mathbf{A}_i})\right]}, \tag{6}$$

where $E_{\mathbf{A}_i}$ and σ_i are given as

$$\mathbf{E}_{\mathbf{A}_i} = \frac{1}{|N_i|}\sum_{k\in N_i} \mathbf{A}_k, \tag{7}$$

$$\sigma_i = \frac{1}{|N_i|}\sum_{k\in N_i} (\mathbf{A}_k - \mathbf{E}_{\mathbf{A}_i})^T(\mathbf{A}_k - \mathbf{E}_{\mathbf{A}_i}), \tag{8}$$

where $\mathbf{A}_k$ represents the templates belonging to cluster i. $|N_i|$ is the number of examples of cluster i, which belong to class y_i. As shown from Eqs. (7) and (8), $\mathbf{E}_{\mathbf{A}_i}$ is the center of mass for cluster i and σ_i gives us a measure of how spread those examples are from its center of mass.

Another alternative (which works much better) to find less Gaussian centers than examples, is obtained by using vector quantizing (VQ) techniques. This technique has been used by Kohonen [9] in a self-organizing neural network. The application and consequences of this techniques coupled with the neural model herein described is now developing and will be published in a future paper.

3. A COMMENT ON THE SMOOTH FUNCTION APPROXIMATION PROBLEM

So far we have concentrated on the classifying problem, now we briefly mention (without any formal proof whatsoever) how the proposed network can be used to approximate a smooth function which passes through or near the data points.

The easiest way of approximating a smooth function from examples $(\mathbf{A}_i, y_i)$ using the proposed network, is, as mentioned before, by making the T parameter relatively large. This simple procedure gives reasonable approximations as shown in the experimental result sections. Another way to proceed is to add another term to the internal energy expression which encodes this smoothness condition. This additional term is called a regularization term [2]. This procedure gives, in general, far better smooth approximation. We do not follow this approach but it is worth mentioning because it is the root of a more general way of approximating smooth functions to sparse data [2].

4. EQUIVALENCE BETWEEN NORMALIZED RBF NETWORKS AND SIGMOID BASED ONES

We show in this section how, for a two class problem, the region boundary is formed between the two classes using Eqs. (4) and (5), and the condition needed to reduce it to a network based on sigmoid function with linear argument.

If we have two n-dimensional real valued template vectors $\mathbf{A} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{B} = (b_1, b_2, \ldots, b_n)$, which belong to the real valued classes y_1 and y_2 respectively, the proposed neural network has the following input-output relationship:

$$y(\mathbf{X}) = \frac{y_1 \exp[-g_1(\mathbf{X}-\mathbf{A})^T(\mathbf{X}-\mathbf{A})] + y_2 \exp[-g_2(\mathbf{X}-\mathbf{B})^T(\mathbf{X}-\mathbf{B})]}{\exp[-g_1(\mathbf{X}-\mathbf{A})^T(\mathbf{X}-\mathbf{B})] + \exp[-g_2(\mathbf{X}-\mathbf{A})^T(\mathbf{X}-\mathbf{B})}, \qquad (9)$$

where we have made $g_i = 1/(T\sigma_i)$ for $i = \{1, 2\}$. When both gains g_1 and g_2 are equal, the proposed network can algebraically be transformed into

$$y(\mathbf{X}) = y_1 + \frac{y_2 - y_1}{1 + \exp\left[-2g_1(\mathbf{B}-\mathbf{A})^T\left(\mathbf{X} - \frac{\mathbf{A}+\mathbf{B}}{2}\right)\right]}, \qquad (10)$$

which is a sigmoid function with linear argument. The boundary between the classes is an hyperplane halfway and perpendicular to the line connecting both templates. For example, in a two dimensional feature space $(\mathbf{X},\mathbf{Y})$, if we have two templates at $\mathbf{A} = (a_1, a_2)$ and $\mathbf{B} = (b_1, b_2)$, according to Eq. (10) the boundary will lie on the following straight line:

$$\mathbf{Y} = \frac{a_1 - b_1}{b_2 - a_2}\mathbf{X} + \frac{b_1^2 - a_1^2 + b_2^2 - a_2^2}{2(b_2 - a_2)}. \qquad (11)$$

When the gains g_1 and g_2 are different Eq. (10) becomes

$$y(\mathbf{X}) = y_1 + \frac{y_2 - y_1}{1 + \exp[(g_2 - g_1)\mathbf{X}^T\mathbf{X} - 2(g_2\mathbf{B} - g_1\mathbf{A})^T\mathbf{X} + g_2\mathbf{B}^T\mathbf{B} - g_1\mathbf{A}^T\mathbf{A}]}. \qquad (12)$$

The argument of the sigmoid function looks like a complicated quadratic hypersurface but it is in fact a hypersphere. To show it in a simple way let us suppose that the two feature vectors are: $\mathbf{A} = (0, 0, \ldots, 0)$ and $\mathbf{B} = (b, 0, \ldots, 0)$, that is, $\mathbf{A}$ is situated at the origin and $\mathbf{B}$ at a distance b from $\mathbf{A}$. Furthermore let us consider $(g2/g1) = \eta$ as the spread ratio between the two classes. Using those values in the sigmoid argument of Eq. (12), the boundary will lie within the set of points $\mathbf{X}$ which satisfy

$$\left(\mathbf{X} - \frac{\eta b}{\eta - 1}\mathbf{I}\right)^T \left(\mathbf{X} - \frac{\eta b}{\eta - 1}\mathbf{I}\right) - \frac{\eta b^2}{(\eta - 1)^2} = 0; \quad \mathbf{I} = (1, 1, \ldots, 1). \tag{13}$$

As can be analyzed from Eq. (13), if the spreading ratio is set to $\eta > 1$ then the hypersphere will enclose the $\mathbf{B}$ example. Otherwise with $\eta < 1$ the hypersphere will enclose the $\mathbf{A}$ template. If $\eta = 1$, then the infinite radius hypersphere becomes the separating hyperplane discussed above. Additionally, if $\eta \gg 1$ then the hypersphere surrounding the $\mathbf{B}$ example collapses at the template's point $\mathbf{B}$. That means, having the extreme case of $\eta = \infty$ the only feature vectors assigned to the y_2 region will be those exactly equal to $\mathbf{B}$. On the other hand if $\eta \ll 1$, the hypersphere collapses in example $\mathbf{A}$.

If this problem is solved in two dimensions using sigmoid network with backpropagation, we would need at least three sigmoid neurons to build a closed region. Furthermore, if the number of dimensions of the feature space grows, the number of hyperplanes needed to build a convex closed region will grow proportionally. Consequently, the number of backpropagation neurons depends also on the dimensions of the feature space for a similar classifying task.

From this discussion it can be seen that the proposed model is in fact a generalization of the commonly used sigmoid when the linear term in its argument is replaced by a quadratic function, and by using isotropic Gaussians the boundaries formed between the classes are hyperspheres. Using another RBFs or non-RBFs the model can give rise to other boundary functions (closed or open) such as ellipses, parabolas, etc., with probable additional interesting properties not yet explored.

We conclude this section by recalling that if all the σ_i's are set to an equal value and $T \to 0$, then the network proposed can be switched into a network based on high gain linear argument sigmoid functions. Its properties, merits and limitations are the same as for a NNC. Otherwise, if the σ_i's are different, the network is still switcheable into a sigmoid based network but now the boundaries among the classes are portions of hyperspheres and its classifying properties are like as for a MLC system. This will be discussed in the next section.

5. Relation of the purposed model to Gaussian classifiers

In this section, it is shown that the neural network proposed is mathematically equivalent to a Gaussian maximum likelihood classifier "GMLC" [8]. Moreover, introducing additional prior knowledge (the prior probability of the templates) the presented network can be transformed into a Gaussian estimator.

Let us suppose we have M templates $\mathbf{A}_i$ in an n-dimensional feature space. Each time we sample the outside world (which is our source of vectors $\mathbf{X}$) a feature vector $\mathbf{X}$ is generated. Having this input vector $\mathbf{X}$, we have to look at the stored templates $\mathbf{A}_i$, and choose the template which resembles the most. This selection is done according to some prior knowledge of how these $\mathbf{X}$ are generated. Once we have chosen the most similar template $\mathbf{A}_i$, then we can associate to the $\mathbf{X}$ the class y_i to which the template $\mathbf{A}_i$ belongs.

One way of specifying the mechanism used by the source to generate those $\mathbf{X}$, is estimating or assuming prior probabilities. The conditional prior probability, $p(\mathbf{X} \mid A_i)$, gives us a measure in the sense that: having observed a feature vector $\mathbf{X}$, a noisy process from $\mathbf{A}_i$ was generated. The other one is the prior probability $p(\mathbf{A}_i)$, which gives us a measure of the relative frequency of sampling a noisy version of $\mathbf{A}_i$ among the other $M-1$ possibilities.

Those two probabilities are commonly used to find the posterior probability, which gives us a similarity measure to choose $\mathbf{A}_i$ given $\mathbf{X}$. Using the Bayes' formula, this probability reads as

$$p(\mathbf{A}_i \mid \mathbf{X}) = \frac{p(\mathbf{A}_i)p(\mathbf{X} \mid \mathbf{A}_i)}{\sum_{j=1}^{M} p(\mathbf{A}_j)p(\mathbf{X} \mid \mathbf{A}_j)}, \quad (1 \le i \le M), \tag{14}$$

and its class will be given by the following expected value:

$$y(\mathbf{X}) = \sum_{i=1}^{M} y_i p(\mathbf{A}_i \mid \mathbf{X}). \tag{15}$$

Many probability distributions are represented by smooth continuous functions, in consequence Eq. (14) gives us continuous real numbers in the range $(0, 1)$. To choose the most probable one, we have to select the $p(\mathbf{A}_i \mid \mathbf{X})$ with the highest value. Once this selection has been done, the remaining probabilities are set to 0, and the selected one to 1. In this way Eq. (15) will give us only a possible class, and not a weighted average of them.

If the distribution of the feature vectors $\mathbf{X}$ are Gaussians (or assumed to be Gaussians) with variance equal σ_i and centered at the templates $\mathbf{A}_i$, and having the same chance of sampling any $\mathbf{A}_i$, then, the resulting classifier given by Eqs. (14)-(15) looks similar to the proposed network, with the exception that the temperature parameter T is set to one. Also the network as stated by Eq. (14) and Eq. (15) is the same as the one reported by Moody and Darken [9], which performs a soft competition among the classes. In other words, it approximates a smooth surface, so no sharp classifying is obtained (the "winner takes all" process is not achieved).

Adding the temperature parameter T to Eq. (14) as done in the proposed model, then the maximum selection process is achieved simultaneously to the process of computing $p(\mathbf{A}_i \mid \mathbf{X})$. In this way a GMLC is obtained.

Moreover as mentioned before, the proposed network will perform in parallel a Gaussian maximum likelihood classification of the input vector $\mathbf{X}$. To show it, in a simple way, let's

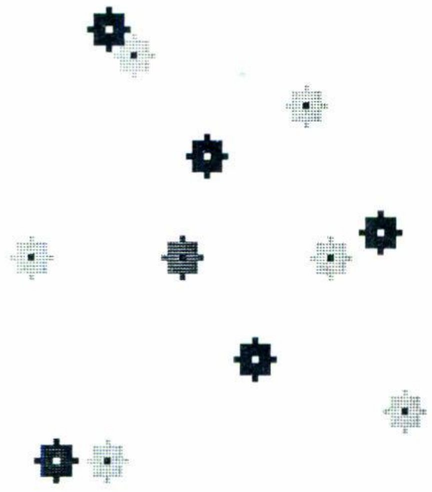

FIGURE 2. Set of representative templates used to build the classifying neural network. The gray level of the small circles represents one of four different classes of the problem. The dot in their center is the exact templates coordinates in the feature space.

return to our simple two class problem of Section 4. For this classifying task, we can write the likelihood ratio as

$$l(\mathbf{X}) = \frac{p(\mathbf{X} \mid \mathbf{A})}{p(\mathbf{X} \mid \mathbf{B})} = \exp[(g_2 - g_1)\mathbf{X}^T\mathbf{X} - 2(g_2\mathbf{B} - g_1\mathbf{A})^T + g_2\mathbf{B}^T\mathbf{B} - g_1\mathbf{A}^T\mathbf{A}], \quad (16)$$

where the decision is taken according to [8].

1. Assign $\mathbf{A}$ if $l(\mathbf{X}) > 1$,
2. Assign $\mathbf{B}$ if $l(\mathbf{X}) < 1$,
3. Make an arbitrary decision at the boundary: $l(\mathbf{X}) = 1$.

The decision boundary (statement 3 above) is the same hypersphere as the boundary formed by the proposed network. This section has shown that, if the probability of selecting an $\mathbf{A}_i$ is not evenly distributed, the Bayesian Gaussian classifier gives a better performance. Otherwise the Gaussian-Bayesian classifier is reduced to a GMLC with the same properties as the proposed neural system.

6. EXPERIMENTAL RESULTS

The proposed network has been tested to show its classifying, interpolation and approximation capabilities. Fig. 2 shows a two dimensional feature region at which the 12 examples used to build the proposed network are shown. The gray level of the circles, which surround each template represents one possible class. The example coordinates and their corresponding classes were generated at random inside the region.

After feeding the templates' coordinates and their corresponding classes, into the proposed network [Eqs. (4) and (5)], the network creates the classifying topography shown at Fig. 3. The parameter T has been set to a small value and the σ_i's have been chosen all

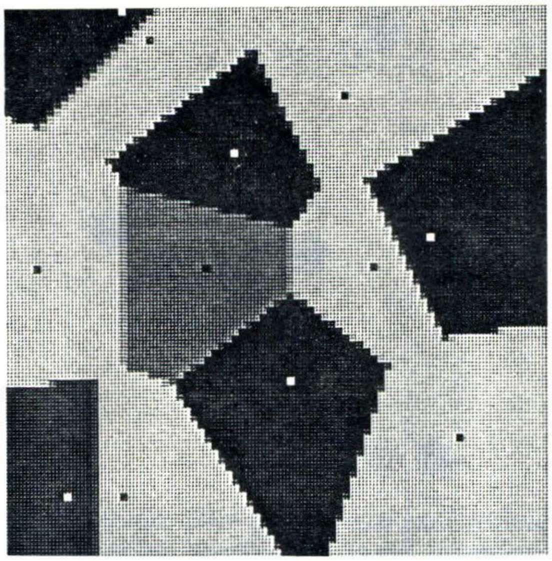

FIGURE 3. Sharp region boundaries formed by the proposed neural network given the classifying task shown at Fig. 2. The variance from the templates have been set equal to 1. The temperature parameter T has been set to a low value to obtain sharp classifying boundaries.

equal to one. Fig. 3 shows how sharp boundaries between the classes are formed and those boundaries fall half way between the examples. This classifier has the same behavior as the NNC, but with substantially faster computing capabilities when a fully parallel hardware or a vector based digital machine is used. Fig. 4 shows the same classifying problem but now the σ_i's of the white regions have been increased so its classifying area has grown and the region boundaries are now arc segments belonging to the boundaries circles.

Fig. 5 shows the same examples and classes shown in Fig. 2, but there the parameter T has been increased. Their deviations from their templates have been taken equal to one. It can now be seen that a smooth curve passing through or near the examples has been formed. This is almost the kind of smooth functions that would be obtained if we had trained a RBF's kind of network centered at these examples.

7. COMPARISON TO SIGMOID AND GAUSSIAN ADAPTIVE CLASSIFIERS

The most traditional approaches in adaptive neural networks have also been tested using the same classifying problem for comparison purposes. We have used the following approximations for the sigmoidal or backpropagation neural network and for the Gaussian one:

$$\bar{y}(x_1, x_2) = \sum_{i=1}^{50} h_i \exp\left[-\sum_{j=1}^{2} \frac{(x_j - c_{ij})^2}{\sigma}\right], \tag{17a}$$

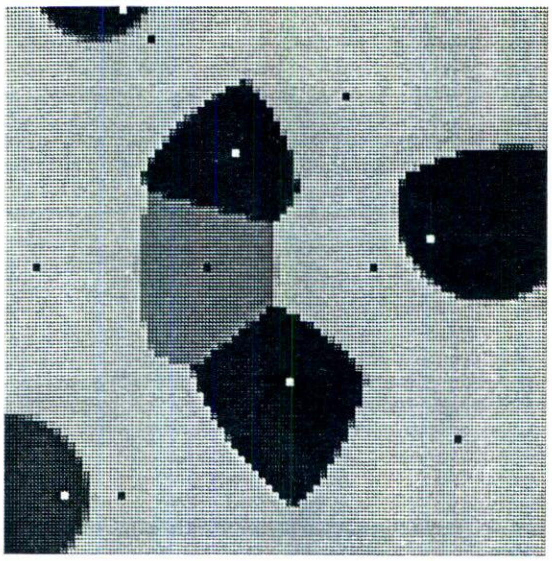

FIGURE 4. Same classifying task shown in Fig. 2, but in this case the variations for the templates assigned to the brighter class has been doubled, so its region class has grown.

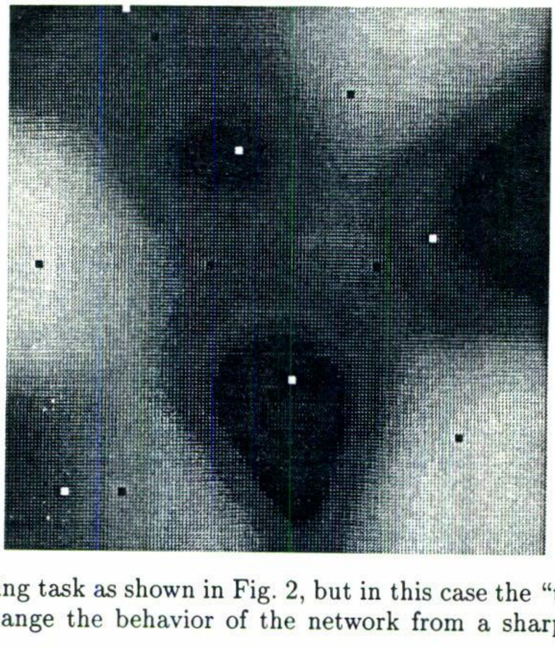

FIGURE 5. Same classifying task as shown in Fig. 2, but in this case the "temperature" parameter T has been raised, to change the behavior of the network from a sharp classifier to a smooth approximator.

$$\bar{y}_k(x_1, x_2) = \sigma_k\left(\sum_{i=1}^{15} h_{ik}\sigma_i\left(\sum_{j=1}^{2} x_j w_{ijk} - c_{jk}\right)\right) \quad k \in (0,1), \tag{17b}$$

(18)

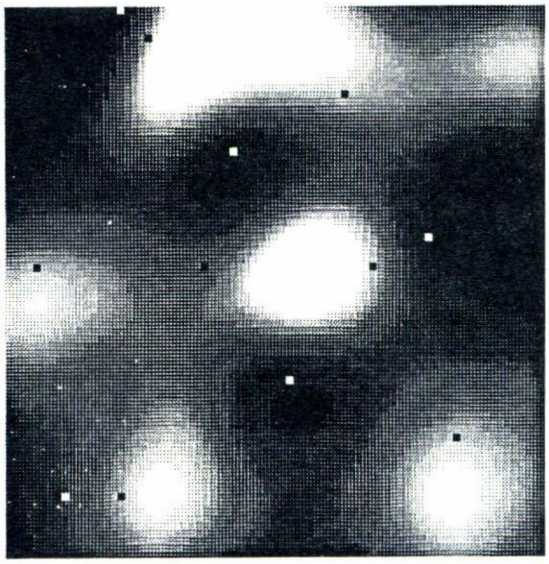

FIGURE 6. Classifying topography created by the network based on Gaussian functions when the templates and classes in Fig. 2 where supplied to the adaptive network. It can be seen how some well classified templates are in the raising edge of a basis function making even very close observed feature vectors be missclassified.

and the error has been defined as:

$$E = \sum_{k=1}^{12} \left(\bar{y}(a_{1k}, a_{2k}) - y_k\right)^2, \tag{19}$$

as in the proposed network the classes' numeric values were chosen to be $0, 1, 2$ and 3. The number of neurons used for the GRBF network was 50, and 15 for the sigmoid based one. The equations of motion for the adaptive systems were the simple gradient descent over the error surface E, in the network's parameter space, that is

$$h_i = -\eta \frac{\partial E}{\partial h_i}, \qquad c_{ij} = -\eta \frac{\partial E}{\partial c_{ij}}, \tag{20}$$

for the GRBF network and for the sigmoid network

$$h_{ik} = \eta \frac{\partial E}{\partial h_{ik}}, \quad w_{ijk} = \eta \frac{\partial E}{\partial w_{ijk}}, \quad c_{ik} = \eta \frac{\partial E}{\partial c_{ik}}, \tag{21}$$

where η is the rate of convergence or gain for the deterministic adaptive neural systems.

The above adaptive networks were tested solving the classifying task shown at Fig. 2. These networks have created the regions and boundaries shown at Figs. 6 and 7 for the Gaussian and the sigmoid networks respectively. The error over the 12 examples given by Eq. (18) has been almost zero after the adaptive process.

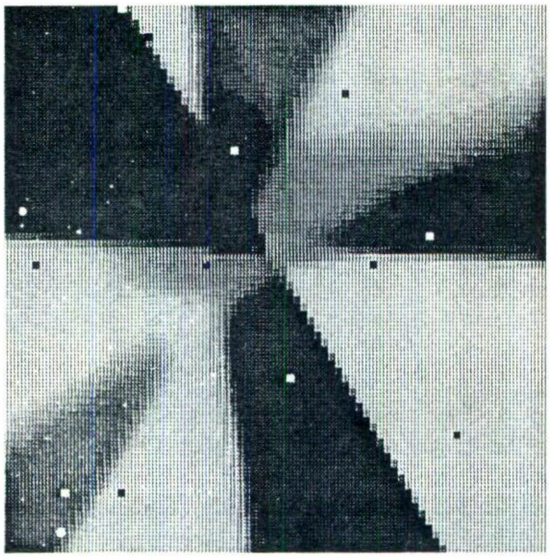

FIGURE 7. Classifying topography created by the backpropagation kind of neural network when the templates and classes shown at Fig. 2 were supplied to the adaptive network. It can be seen how the classification topography has completely changed with reference to the Gaussian based network and the proposed one. The same problem that some templates are on the raising edge of the basis functions is also presented in this case.

From these Figures mainly two things can be observed: firstly the created regions are sometimes not sharply classified, especially in the Gaussian case, unless many Gaussian neurons were used; secondly, quite often the class regions are not plateau having the same gray level. Frequently some well classified examples may fall on the raising edge of a basis function making even very close input feature vectors to be missclassified, as can be seen from Figs. 6 and 7. What is normally done in such systems to improve the regions plateau is to provide the neural system with many examples per class around its average to spread out the class' region.

Finally, before leaving this section it is worthwhile showing the class regions created by the NNC using the same classifying task, so we can compare it with the topography created by the proposed network when all σ_i's are equal. These class regions are shown in Fig. 8. It can be observed how the class boundaries are formed in the same way as the proposed network with the exception that, the nearest neighborhood classifier has the sharpest boundaries among the classes, given that it is not a continuous function as the proposed one.

8. CONCLUSIONS

A new kind of network for classification and approximation based on RBFs has been obtained. Their capabilities and the condition needed to switch it into a linear argument, sigmoid based network has been considered. Comparison of well known approaches such

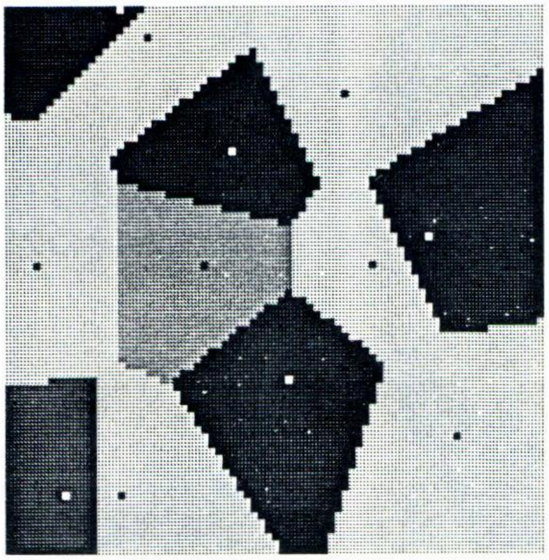

FIGURE 8. Classifying topography created by the nearest neighborhood classifier for the classifying task shown in Fig. 2. It can be observed how the separating boundaries between the examples are exactly the same as the ones created by the proposed neural network shown in Fig. 3. The sole difference is that these boundaries are even sharper, given that the nearest neighborhood classifier is not a continuous function in the feature space.

as backpropagation, GRBF's, NNC, Bayesian classifier and MLC, have also been pointed out.

The main advantage of this approach is that having the examples and its correct classification in advance, they can be fed into the network just as they are, and no gradient descent training or matrix inversion are needed to build up the right classifying network, that is, the network requires no learning. If we have many noisy examples, statistical reduction can be used to find fewer centers and their associated deviations per class. These two numbers are fed into the reduced network. Moreover it has been shown how we can continuously change the behavior of the network from a sharp classifier to a smooth function approximator by changing just one parameter, which is the value of the absolute temperature T of the system.

This network can also learn on-line under gradient descent. Its behavior, capabilities and learning rates are now being researched.

ACKNOWLEDGEMENTS

We wish to acknowledge Dr. J.L. Marroquin's useful suggestions throughout the development of this work. We also with to acknowledge Prof. Stephen Grossberg's fruitful comments to improve the presentation of this work.

REFERENCES

1. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation", in *Parallel distributed processing: explorations in the microstructures of cognition* **1**, D.E. Rumelhart and J.L. McClelland (eds.) Cambridge, MA, MIT Press (1986) 318.
2. T. Poggio and F. Girossi, "A theory of networks for approximation and learning", in *MIT A.I. Memo* **1140** (1989), C.B.I.P. paper No. 31.
3. J.O. Murphy, "Nearest neighbor pattern classification perceptrons", in *Proceedings of the IEEE* **78** (1990) 1595.
4. R. Rosenblatt, *Principles of neurodynamics*, New York, Spartan Books (1959).
5. G.E. Hinton, *Neural Computing* **1** (1989) 143.
6. S. Kirpatrick, C.D. Gelatt Jr. and M.R. Vecchi, *Science* **220** (1983) 671.
7. J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", in *Proceedings of the National Academy of Science USA* **79** (1982) 2554.
8. J.T. Tou, R.C. Gonzalez, *Pattern recognition principles*, Massachusetts, Addison-Wesley Publishing Company (1974).
9. J. Hertz, Krogh A., Palmer, R.G. *Introduction to the theory of neural computation*, California, Addison-Wesley Publishing Company (1991).
10. J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding", in *Proceedings of the IEEE* **73**, 11 (Nov. 1985) 1551.
11. R. Duda, P. Hart, *Pattern classification and scene analysis*, Wiley-Interscience Publication (1973).