A NOVEL CONSTRUCTIVE NEURAL NETWORK THAT LEARNS TO FIND DISCRIMINANT FUNCTIONS

José L. Alba and Laura Docío
Departamento de Tecnologías de las Comunicaciones
Universidad de Vigo, Spain
Phone:34-86-812126, Fax:34-86-812116
e-mail:jalba@tsc.uvigo.es , ldocio@tsc.uvigo.es

ABSTRACT

This paper presents a novel architecture based on a constructive algorithm that allows the network to grow attending to both supervised and unsupervised criteria. The main goal is to end up with a set of discriminant functions able to solve a multi-class classification problem. The main difference with well-known NN-classificators lean on the fact that training is performed over labeled sets of patterns that we call *high-level-structures* (HLS). Every set contain patterns linked each other by some physical evidence, like neighbor pixels in a subimage or a time-sequence of frequency vectors in a speech utterance, but the membership of every individual pattern in the high-level-structure can not be so clear.
This architecture has been tested on a number of artificial data sets and real data sets with very good results. We are now applying the algorithm to classification of real images drawn from the DataBase created for the ALINSPEC project. [1]

1 INTRODUCTION

Since Moody introduced in 1988 [1] the concept of localized receptive field units as synaptic functions in a neural network architecture, a bunch of works have shown its capabilities to solve functional approximation and classification problems. The use of localized units combined with perceptron-like connections to end up with an hybrid architecture has appeared to be competitive with the more classical non-linear architecture composed by layers of sigmoidal functions (MLP). The training strategy is usually divided into two steps: an unsupervised algorithm (K-means [2], SOM [3]) to organize the hidden layer with LRF units, and a supervised one to train the connections to the output layer (gradient descent technique). So it has the capability to represent arbitrary functions, and can be trained more quickly.

The operation performed by every hidden node (i.e., the nonlinear activation function of each neuron) can be any decreasing function of the distance between the input pattern x and the LRF unit ω_j, with a single maximum when the distance equals to zero and dropping off to zero for large distances. The most widely accepted node equation takes the form of a Gaussian kernel:

$$u_j = e^{-\frac{|x-\omega_j|^2}{2\sigma_j^2}} \tag{1}$$

For this function, the receptive field takes an hyperspherical shape centered at the unit ω_j, where the effective radii σ_j can be calculated in different ways depending on the computational burden we can accept.
All the LRF units in the hidden layer are connected to every output units through a layer of perceptrons:

$$\Psi_k = \sum_{j=1}^{J} c_{jk} u_j \tag{2}$$

where c_{jk} is the connection between LRF unit j and output unit k.
In the standard training procedures the number of LRF is predefined and located by using any self-organizing algorithm, and the output connections are trained by a gradient-descent technique that minimizes an error function between the input pattern x_j and the desired associated target t_j. The MSE is the usual criterium to minimize, and this will tend to approximate the probability density function of every class given the input. Nevertheless, in most of the cases, the training set is not so huge to allow an accurate approximation at the whole input space, and then, the classification error does not run parallel to the MSE. The modifications of the standard architecture are then focused to the positioning of LRF units where they are really necessary to minimize the output error more than relying solely in the performance of the self-organized algorithm. The key idea is creating a unit when the pattern at hand is far away from any previously located unit (using an adaptively decreasing function of distance) and the output error is larger than a threshold. If the error doesn't surpass

[1]This system has been developed in connection with ALINSPEC (Automatic Inspection of Alimentary Products): A BRITE-EURAM project partially supported by EEC under contract number BRE2-CT92-0132.

this threshold, all the parameters of the network are updated (not only the perceptron weights) resulting in an adaptive tuning of units location. Fritzke developed an hybrid architecture following similar principles [4], using self-organizing criteria for updating the LRF units before computing any output error, and adding a new unit close to the old unit that most contributed to the accumulated output error.

All these variations of the original hybrid architecture for classification rely on the assumption that the problem is defined by the input-output training set (x_i, t_i) over which an error function should be minimized. The architecture we propose in this paper, that we called from now CMHNN (Constructive Modular Hybrid Neural Network) is based on the previous growing strategies, but with some fundamental differences:

- The classes are defined by a prior knowledge over a set of "connected" patterns, this means the pattern x_i is not labeled itself but belongs to a labeled higher level structure.
- The training method is Decision-Based [5] which implies that a set of inequalities of some discriminant functions govern the creation of units and training of weights to yield a correct classification.
- It has a hierarchical structure with two levels. The higher level consists of multiple subnets or modules (one per class), and the lower level consists of multiple subnodes in a module. The network output is the winner among all the modules, i.e. the module (class) with highest discriminant function.

In the next section we introduce the terminology of the network and its functionalities. At section 3 we will discuss the training procedures and issues related to convergence. Section 4 is dedicated to present some experiments over artificial and real data, and in section 5 some conclusions are presented.

2 TERMINOLOGY AND FUNCTIONALITY

We start this section defining the main parameters and functions involved in the CMHNN.

Every subset of patterns forming a higher level structure will be called $\mathcal{X}_i$ which is labeled as one out of M classes with $1 \leq i \leq I$ and I the number of training subsets. The input patterns will be represented by $x_{ij} \in \mathcal{X}_i$, $x_{ij} \in R^N$ with $1 \leq j \leq J_i$ and J_i the number of patterns in $\mathcal{X}_i$.

The hidden nodes will be denoted by $\omega_{lm} \in R^N$ with $1 \leq l \leq l_m$ and l_m the number of nodes in module m. Every LRF hidden unit will have a Gaussian transfer function u_{lm} as in (1) but using a normalized distance measurement and adding up the contribution of each pattern $x_{ij} \in V_{lm}(i)$, where $V_{lm}(i)$ is the "influence" domain of unit lm for HLS $\mathcal{X}_i$ defined as $V_{lm}(i) = \{x_{ij}/d_M(x_{ij}, \omega_{lm}) \leq d_M(x_{ij}, \omega_{tm}), \quad \forall\ tm \neq lm\}$ and $d_M(x, w)$ the Mahalanobis distance between pattern x_{ij} and node ω_{lm}, using the covariance matrix associated to node lm. (From now on we will omit the index i in $V_{lm}(i)$ to make the equations more clear)

$$u_{lm}(\mathcal{X}_i) = \frac{1}{\mathcal{C}_i(V_{lm})} \sum_{x_{ij} \in V_{lm}} e^{-\frac{1}{2} d_M(x_{ij}, w_{lm})} \tag{3}$$

with $\mathcal{C}_i(V_{lm})$ denoting the cardinality of set V_{lm}. Every hidden unit is connected to the output node of the module m with a strength c_{lm}. In this way we define the *discriminant function* for module m as

$$\Psi_m(\mathcal{X}_i) = f(\hat{c}) \sum_{l=1}^{l_m} c_{lm} u_{lm}(\mathcal{X}_i) \tag{4}$$

where $f(\hat{c})$ is a normalizing factor depending on the vector of 0œ[Cœconnections.

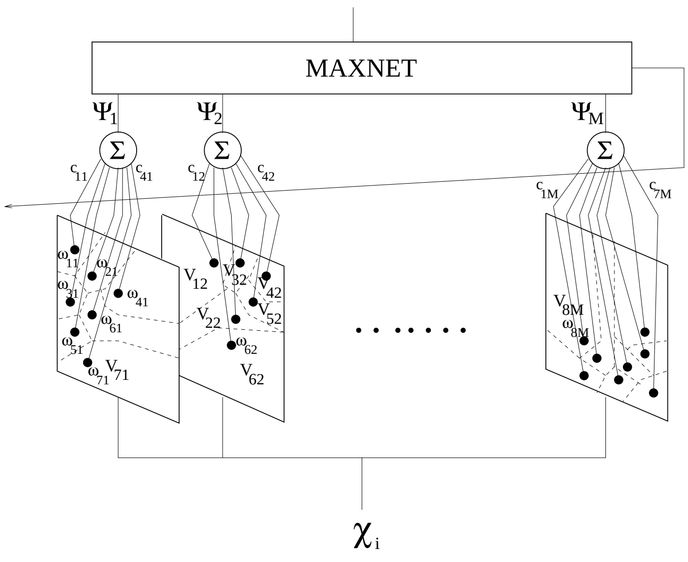

Figure 1: Global architecture of CMHNN

The contribution of every individual pattern belonging to some HLS is computed by means of a elliptical-basis function that introduces the concept of localized response into the algorithm. The whole HLS contribution for class m is locally integrated by means of the output of every unit u_{lm} and linearly combined through the connections c_{lm} in the discriminant function associated to the class m. The discriminant function with highest output value is selected as winner and its class is assigned to the input structure.

The underlying philosophy of this structure is that if some pattern is noisy or of unknown membership, it will be classified wrongly with high probability when considered alone, but putting together patterns that

share some kind of neighborhood by means of localized responses, the classification rate increases. Obviously this architecture is useful when our application allows to group patterns this way, or they are naturally structured.
The training procedure is in charge of creating and locating LRF units to tune every discriminant function while a perceptron-like rule adapts connection weights until some stop criterium is reached.

3 LEARNING IN CMHNN

The goal of the network is to classify the input structure of patterns as belonging to some of the M classes by finding the module that best matches it. In the learning process the modules grow by adapting themselves to the training structures and this is performed by means of two steps (hybrid training): 1) creating units *where* and *when* they are needed to ensure the correct classification of the input structure of patterns at hand, more than to reduce the MSE in the output; and 2) adapting the connections c_{lm} in each module to balance the contribution of every node.

3.1 Adding And Updating Nodes

Every time a new HLS $\mathcal{X}_i$ corresponding to class m is presented to the CMHNN, and the MAXNET outputs a wrong class we start an analysis of the module m that should have won to check out if it "needs" a new node. In order to decide if this will be inserted a measure of distortion among the modules with wrong outputs is computed, and if this distortion is bigger than a time-adaptive threshold a new node is created in the mass center of the patterns associated to the domain V_{lm} with highest quantization error, i.e.

$$\max_{1 \le l \le l_m} \{ \frac{1}{\mathcal{C}_i(V_{lm})} \sum_{x_{ij} \in V_{lm}} \frac{1}{2} (d_M(x_{ij}, w_{lm}) \} \qquad (5)$$

So, we use the supervised information of the classification assessment and a global distortion measurement to decide if some node is necessary and a local test of the quantization error (unsupervised) detects the region were such a node should be inserted.
After adding a node, this and all his neighbors should be updated. This is accomplished by a rule that moves the node towards the new center of the local distribution measuring distances between patterns and old nodes in every V_{lm}, and taking into account:

$$\Delta\omega_{lm} = \mu g(\Sigma_{lm}) \sum_{x_{ij} \in V_{lm}} e^{-k_{lm}/\lambda} (x_{ij} - \omega_{lm}) \qquad (6)$$

where μ is the convergence rate, $g(\Sigma_{lm})$ is a decreasing function of the dispersion of the data (we are using: $g(x) = 1 - exp(-x)$, k_{lm} is the number of units that are closer to the pattern x_{ij} than the unit under updating , and λ is a constant decay parameter [6].
A heuristic method is used to determine or to update the new covariance matrices Σ_{lm}.

$$\Delta\Sigma_{lm} = (1 - \epsilon)(\hat{\Sigma}_{lm} - \Sigma_{lm}) \qquad (7)$$

with $\hat{\Sigma}_{lm}$ the covariance of the input patterns within the structure at hand assigned to the node lm and Σ_{lm} the old one. The parameter ϵ weights the importance of the new distribution with respect to the previously learned one.
After the creation and adaptation of nodes, the connection weights are updated as explained in the next subsection.

3.2 Updating Connection Weights

The adaptation is based in the concept of reinforced and antireinforced learning (originated from the perceptron rule). This rule will tend to increment the discriminant function of the module that should have won and decrement those that had larger values [5]. Suppose that the i-th training structure $\mathcal{X}_i$ is known to belong to class m, but there are at least one module n such that:

$$\Psi_m(\mathcal{X}_i) < \Psi_n(\mathcal{X}_i) \qquad (8)$$

Using *(anti)reinforced learning*, the connections in module *(n)m* are updated following the (negative)positive direction of the gradient of $\Psi_{(n)m}$:

$$\hat{c}^{t+1}_{(n)m} = \hat{c}^{t}_{(n)m} \pm \eta \nabla \Psi_{(n)m}, \qquad (9)$$

with $\hat{c}_{(n)m}$ the vector of connections from module *(n)m* and η a positive learning rate.
The new discriminant function is calculated and if the structure is still misclassified we check again the global distortion criterion to add a new node and independently on this decision, another (anti)reinforced step is done to adapt the connections.
This process can be repeated until the HLS in the training set is correctly classified. After all the HLS have been presented, another epoch should be started to fine-tune parameters. These epochs are repeated until all the training HLS are correctly classified. This training process has been used for some tests we have done, but some effects must be commented about the dynamics of the algorithm.
It is possible to train the CMHNN in order to have all the HLS in the training set correctly classified, or stop the algorithm with another criterion, like the classification rate over a validation set. If the control parameters (distortion threshold, convergence rates, etc) are not properly set, the algorithm can suffer from the following effects:
(1) If we relax the condition for the creation of nodes and choose the criterion of all training HLS correctly classified, we can have a huge overfitting net with bad generalization.

(2) If the convergence rates decrease too fast or too slow, the classification error doesn't change after some epoch, or can oscillate like a perceptron output in a non-linearly-separable problem.

4 SIMULATION EXAMPLES

In this section, we demonstrate the performance of the CMHNN on artificial data sets and the well-known SATIMAGE and TEXTURE databases.

4.1 Artificial Data Sets Classification

The CMHNN performs very well with data sets drawn from Gaussian-mixture distributions. To show this we have created some HLS's containing mixtures of bidimensional Gaussian distributions with different mean vectors and covariance matrices. Furthermore, the classes are highly overlapped. The results have been very encouraging, because in every simulation the error rate obtained over test data sets was 0%.
The major advantage of CMHNN versus others neural networks is based in working with whole structures instead of working with each individual pattern within the structure. To verify this, we have generated two data sets containing each of them one single Gaussian distribution. After the creation of the modules we tested the network performance with structures with an increasing number of patterns. We have seen that the more number of patterns/HLS we use, the lower error rate we achieve. In the following table we show this:

Patt/HLS	1	2	4	8	10	20	40
Error (%)	30.5	28.9	23.4	14.4	9.0	7.0	0.0

4.2 Real Data Sets Classification

Texture Data: This database was generated for the Esprit project ELENA No. 6891.
Data are individually labeled as belonging to 11 different classes, so we need to group subsets of data (HLS) to apply our architecture. Obviously our criterion for grouping is only the membership to a specific class, because we don't know the 2-dimensional original image. We made 10 HLS/class with 50 40-dimensional patterns each, and the test error over a validation set of 4 HLS/class was 0%. Every module finished with 2-4 nodes.
LANDSAT Satellite Data: This database was in use in the European StatLog Project.
Data are individually labeled as belonging to 7 different classes. We made 15 HLS/class with variable number (50-75) of 4-dimensional patterns each, and the test error over a validation set of 5 HLS/class was 5%. Every module finished with 2-4 nodes.

4.3 Image Classification

The CMHNN is being tested with images drawn from a DataBase explicitly created in the context of the Alinspec project. The problem is the classification between healthy/defective samples of alimentary products (chickens meat). Adapting our nomenclature to this application, we have that $\mathcal{X}$ is an image or an arbitrary shaped subregion of it. Patterns x are vectors with chromatic and location-related coordinates. The objective is that every module learn the discriminant features of its assigned class.
At the moment of writing this paper, the results on the classification are worse than a MLP specifically prepared for this application, when we use our algorithm on a 1 pattern/HLS basis. The defective areas contain too many outliers and we are working towards making the architecture more robust to them.

5 CONCLUSIONS

A new neural architecture has been created that merges some largely known techniques for supervised and unsupervised training with a modular self-growing philosophy to build up discriminant functions. The beauty of CMHNN stems from the fact that individual patterns do not need to be labeled, but belonging to a higher level structure with known class-membership. The architecture is flexible enough to accommodate different problem requirements. We have obtained good results working with different real data sets, and the classification of images from the ALINSPEC project is still under research.

References

[1] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proceedings of the 1988 Conectionist Models Summer School* (D. Touretzky, G. Hinton, and T. e. Sejnowsky, eds.), (San Mateo), pp. 52–59, Morgan Kaufmann, 1988.

[2] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, March 1982.

[3] T. Kohonen, *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1988.

[4] B. Fritzke, "Growing cell structures - a self-organizing network for unsupervised and supervised learning," tech. rep., International Computer Science Institute, TR-93-026, Berkeley, CA, 1993.

[5] S. Kung and J. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Transactions on Neural Networks*, vol. 6, pp. 170–181, January 1995.

[6] T. Martinetz, S. Berkovich, and K. Schulten, ""neural-gas" network for vector quatization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, pp. 558–569, July 1993.