INFORMATICA 4/91

PATTERN RECOGNITION USING KOHONEN MAPS AND MULTI-LAYER PERCEPTRONS

Tjaša Meško
Factuly of Electrical Engeneering and
Computer Science
University of Ljubljana

Keywords: multi-layer perceptrons,
initialization, a Kohonen map, clustering

Multi-layer perceptrons (MLPs) are now widely used for pattern recognition tasks such as speech recognition, handwritten character recognition, face recognition, etc. They were proven to generalize well to unseen data. Another kind of neural networks, namely, self-organizing feature maps have also been applied occasionally in pattern classification, but they were not that successful. In this study it is investigated how self-organizing feature maps could be useful in combination with MLPs as a tool for initializing the weights of a MLP. The purpose of the research was to reduce the amount of supervised training which is required to train MLPs.

PREPOZNAVANJE VZORCEV S KOHONENOVO MAPO IN Z VEČNIVOJSKIMI PERCEPTRONI. Večnivojski perceptroni se v zadnjem času pogosto uporabljajo pri prepoznavanju vzorcev (na primer govora), prepoznavanju pisav, prepoznavanju obrazov in podobno. Druga vrsta nevronskih mrež, Kohonenove mape, so bile tudi občasno uporabljene pri reševanju podobnih nalog, vendar rezultati so bili slabši. V tem članku je obravnavana možnost inicializacije uteži skritega nivoja trinivojskega perceptrona. Namen te raziskave je skrajšati čas učenja perceptrona.

1 Kohonen Self-Organizing Feature Maps

The self-organizing map belongs to the category of neural networks that use unsupervised training. This means that each time a new input is presented to the map, the desired output is unspecified. This type of neural networks is used to perform data compression, such as vector quantization and as it will be explained later, to reduce the amount of supervised training.

A vector quantizer is a mapping, q, that assigns to each input vector $\overline{x} = (x_1, x_2, \ldots, x_N)$, a codebook vector $\overline{c}_i = (c_{i1}, c_{i2}, \ldots, c_{iN})$

$$\overline{c}_i = q(\overline{x})$$

drawn from a finite set of codebook vectors

$$Q = \{\overline{c}_1, \overline{c}_2, \ldots, \overline{c}_M\}$$

where M is the number of codebook vectors. The quantizer q is completely described by the set of codebook vectors and it divides the input vector space into clusters

$$C_i = \{\overline{x} : q(\overline{x}) = \overline{c}_i\} \quad (1)$$

of input vectors which are mapped onto the i^{th} codebook vector. The distortion caused by reproducing an input vector $\overline{x}$ by a codebook vector $\overline{c}_i$ is given by

*A young researcher, employed in PAREX, d.o.o.

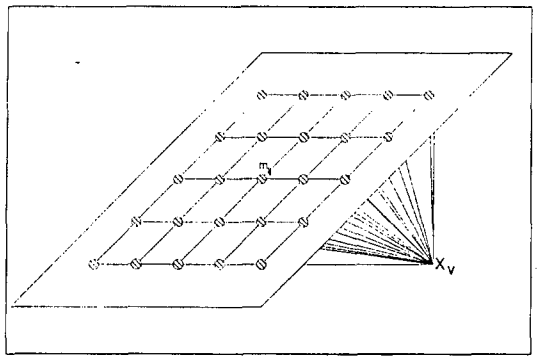

Figure 1: *Locations of map vectors in a square lattice. Vector $\overline{m}_{ij}$ can also be addressed as $\overline{c}_v$, where $v = (i-1)J + j$.*

$d(\overline{x}, \overline{c}_i)$, where d is assumed to be a Euclidean distance which is defined by equation 2.

$$d(\overline{x}, \overline{c}_i) = \sqrt{\sum_{n=1}^{N}(x_n - c_{in})^2} \qquad (2)$$

Kohonen's algorithm creates a vector quantizer by adjusting vectors which are typicaly arranged in a two-dimensional grid (usually a square or a hexagonal lattice). In this study square maps will be used (see figure 1) and the map vectors will be represented by their map coordinates (i, j). The vector at position (i, j) in the map will be addressed in two ways,

- $\overline{m}_{ij}$, $(i, j) = (1, 1), \ldots, (I, J)$
- $\overline{c}_v$, $v = 1, 2, \ldots, I \times J$

where $\overline{m}_{ij}$ equals $\overline{c}_{(i-1)J+j}$ and I, J are the map sizes.

The vector quantizer function $q(\overline{x})$ corresponding with a Kohonen map selects the codebook vector $\overline{c}_v$ which is closest to $\overline{x}$:

$$d(\overline{x}, \overline{c}_v) = \min_k d(\overline{x}, \overline{c}_k), k = 1, 2, \ldots, I \times J$$

In order to create the best codebook vectors, an iterative training is performed. During each iteration, a neighbourhood is defined around each vector of the map, as shown in figure 2. The neighbourhood NE(t) slowly decreases with time. At the beginning, the map vector components are initialized to small random values. Then, the following iterative procedure is applied whenever a new input vector $\overline{x}(t)$ is presented (t is the iteration number).

1. Compute the Euclidean distances $d(\overline{x}(t), \overline{c}_v(t))$, to all nodes $\overline{c}_v(t)$ according to the equation 2.
2. Select the node producing the minimum distance as the winning node $\overline{c}_v$.

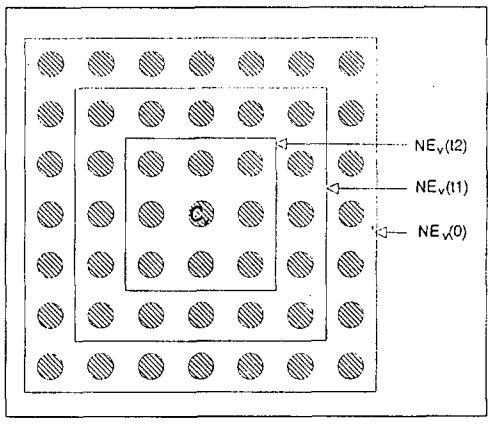

Figure 2: *Topological neighbourhood at different times as the feature map is formed. $NE_v(t)$ is the set of nodes considered to be in the neighbourhood of a node $\overline{c}_v$ at time t. The neighbourhood is decreased in size as time increases. In this example, $0 < t_1 < t_2$.*

3. Compute an updating factor $\alpha(t)$ and define a topological neighbourhood $NE_v(t)$.
4. Update the map vectors belonging to the topological neighbourhood of the winning node. For the adaptation of a map vector $\overline{c}_k$ the following formula is used,

$$c_{kn}(t+1) = c_{kn}(t) + \alpha(t)(x_n(t) - c_{kn}(t))$$

where $n = 1, 2, \ldots, N$.

The input vectors are taken from an input database, and are presented in a random order. The process is terminated as soon as the average distortion introduced by the vector quantization does not drop any more. Parameter $\alpha(t)$ is initialized to a value between zero and one, and is decreasing with time.

In this study an exponential rule for adapting α and for determining the topological neighbourhood [Brauer and Knagenhjelm, 1989] was used.

In order to use the Kohonen map for pattern classification, each map vector has to receive a label. The easiest way to achieve this is by applying the maximum a posteriory criterium: the labeled observations of a training set are presented to the map, and each node is labeled according to the number of observations of the different classes that were assigned to that node. Once the map is labeled, it can act as a pattern classifier.

The percentage of wrongly classified examples (the error rate) can be made as small as desired by introducing a large enough number of vectors. However, as it will be explained later, the training of a large map can be very time consuming and it is likely that such a map will not generalize properly to the unseen examples.

	6 × 6	9 × 9	12 × 12	15 × 15
train	74,55%	79,26%	81,02%	82,72%
test1	73,35%	77,59%	79,45%	80,79%
test2	75,47%	78,97%	79,85%	81,35%

Table 1: *Recognition rates on a BPC task, using different Kohonen maps.*

1.1 Results Using the Kohonen Map

The Kohonen map was tested as a pattern classifier on a Broad Phonetic Classification (BPC) task of Dutch spoken utterances. Here are the following five classes of the BPC task: *vowel*, *sonorant*, *fricative*, *burst* and *closure*. We have used a hand-labeled multi-speaker database of continuously spoken Dutch numerical strings, uttered by 30 different speakers (15 male and 15 female). The database consists of 300 different utterances (10 from each speaker). For training, 192 utterances from 24 speakers (8 from each speaker) are used and 108 utterances are used for testing. The test set is divided into 2 sets: *test1*, a multi-speaker test set (the remaining 48 utterances from the training speakers), and *test2*, a speaker independent test set (60 utterances from 6 new speakers). The speech signals were bandlimited to 4 kHz and sampled at a rate of 10 kHz. A 20-dimensional feature vector was extracted every 10 ms by means of an auditory model [Martens and Van Immerseel, 1990]. To take into account the dynamic nature of speech, several successive frames were presented simultaneously at the input of the map. In particular, we have used a feature vector consisting of three input frames: the first two frames were contextual frames located 40 and 20 ms ahead of the third frame which was the one to be classified. Therefore, each input vector consisted of 60 elements. The map vectors were labeled according to the maximum a posteriori criterium. The recognition rates for the different data sets are very similar, indicating an excellent generalization to unseen data. However, the feature-map cannot compete with a Multi-Layer Perceptron (MLP) trained by means of the back-propagation algorithm [Rumelhart et al., 1986]. Such a MLP obtains a recognition rate of 87,20% on the second test set [Depuydt et al., 1990]. The question was whether it would be possible to use the clusters obtained by the map to initialize the weights of a feed-forward net, and consequently to improve the supervised training time of that network.

2 Radial Basis Function Network

A traditional back-propagation network [Rumelhart et al., 1986] consists of nodes whose outputs are non-linear, differentiable squashing functions (typically sigmoid functions) f of the weighted sum of activations emerging from nodes on the previous layer. The output of node i is computed as follows,

$$o_i = f(\sum_j w_{ij} y_j + w_{i0})$$

where w_{ij} represents a connection weight, w_{i0} a bias variable, and y_j an output from a previous layer. The argument of f defines the following hyperplane in the input space:

$$\sum_j w_{ij} y_j + w_{i0} = 0$$

The hyperplanes defined by the different nodes constitute a set of class boundaries.

Searching for other ways of using the back-propagation algorithm, and thereby defining other kinds of class boundaries the idea of using *radial basis function* networks was introduced [Lowe, 1989]. The RBF network contains a hidden layer of m RBF units represented by the centres $\bar{c}_j$. The output layer consists of traditional summation units. Thus, the value of an output unit i is,

$$o_i(\bar{x}) = f(\sum_{j=1}^{m} \lambda_{ij} \Phi_j(\|\bar{x} - \bar{c}_j\|) + \lambda_{i0})$$

with f representing the sigmoid function, and Φ_j a RBF centered around a vector $\bar{c}_j$. Using the standard neural network terminology, the above formula for calculating the output of the i^{th} output-layer node in response to the p^{th} input pattern, o_{ip} can be stated as,

$$o_{ip} = f(\sum_{j=1}^{m} w_{ij} \Phi_j(\|\bar{x}_p - \bar{c}_j\|) + w_{i0})$$

In this study, a network consisting of a hidden layer of Gaussian nodes is considered, therefore, functions Φ_j are assumed to be,

$$\Phi_j(\|\bar{x}_p - \bar{c}_j\|) = \exp(-\sum_{n=1}^{N} \frac{(x_{pn} - c_{jn})^2}{2\sigma_{jn}^2})$$

where the parameters σ_{jn} represent the standard deviations of function Φ_j. The above network could be trained using the back-propagation algorithm with gradients:

$$\frac{\partial y_j}{\partial c_{jk}} = \frac{y_j(x_k - c_{jk})}{\sigma_{jk}^2}$$

$$\frac{\partial y_j}{\partial \sigma_{jk}} = \frac{y_j (x_k - c_{jk})^2}{\sigma_{jk}^3}$$

where $\Phi_j(\|\overline{x}_p - \overline{c}_j\|)$ is addressed as y_j. It was suggested that a network with RBF nodes could be trained by means of a layer-by-layer type of training (Gaussian nodes can be trained separately). In this case a computationally efficient way of determining the optimal weights to the output layer can be proposed [Renalds and Rohwer, 1989]. If y_{jp} is the output of RBF j given training input $\overline{x}_p$, then the output of output node i will be,

$$o_{ip} = f(\sum_j w_{ij} y_{jp} + w_{i0})$$

For optimal weights it holds that

$$E = 0.5 \sum_{ip} (o_{ip} - O_{ip})^2$$

is minimal, where O_{ip} is the desired value of the output node i. It can be shown [Renalds and Rower, 1989] that the optimal weights can be obtained as follows,

$$w_{ij}^* = \sum_k (\sum_p O_{ip} y_{kp}) M_{kj}^{-1} \qquad (3)$$

where M is the correlation matrix of the RBF outputs

$$M_{jl} = \sum_p y_{jp} y_{lp}$$

It is advised to use the pseudo-inverse of M to avoid possible singularities.

3 Initialization of Gaussian Nodes

With the error back-propagation algorithm, there is always a chance that the training gets trapped into a local minimum. The search of the optimal network configuration can also be very time consuming. Especially the determination of an optimal number of hidden units is a long process since the training has to be repeated for different amounts of hidden units. If there are too many hidden units, the generalization capabilities of the network might be reduced. If the number of hidden units is too low, the subspaces created by the network nodes cannot adequately model the class boundaries. Therefore, a dimensionality analysis of neural networks with one hidden layer of Gaussian hidden units and an output layer of conventional summation units was suggested [Weymaere and Martens, 1991]. The weights of the Gaussian nodes are initialized to values which are obtained by a modified k-means clustering algorithm [Wilpon and Rabiner, 1985] and the optimization procedure is performed in order to select the most effective set of Gaussian nodes. By performing a modified k-means clustering it is possible to obtain a fair description of the input data items by a limited number of clusters, each one being represented by a centre and a standard deviation vector. This parametric representation of the clusters can be used to initialize the hidden layer. The clustering can either be performed globally or per class. In the latter case [Weymaere and Martens, 1991], clusters are created for each class separately. During the k-means clustering the centre and standard deviation components are obtained for each class separately and for each number of clusters (1,..., predefined maximum). The cluster members tend to converge to that part of the input space that is covered by the examples of that class. Then the optimization procedure is carried out to select the optimal number of clusters for each class. The selected clusters are used to initialize the hidden layer in an almost optimal way.

We wondered whether it would also be possible to initialize the Gaussian nodes from the clusters obtained by a Kohonen map. A problem could be that the map was trained using all examples of the BPC task, and consequently that the map vector distribution was dominated by the dominating class (the vowels). In order to perform a clustering per class, we would have to construct five different maps (one for each class). Another problem is that by retaining only a few clusters from the map, only parts of the input space will remain well modelled. This is different in case of k-means clustering where the smaller cluster configurations (less centres) are determined to provide the best coarse representation of the data distribution over the entire input space. Due to the two problems stated above, there was doubt about the sensibility of using the Kohonen map cluster centres for the initialization of the Gaussian nodes. A property attributed to the Kohonen map is that it is not as much affected by noise and training inconsistencies as the k-means clustering is. Furthermore, the amount of nodes representing the different classes seems to be proportional to the number of examples of these classes in the training database. The map vectors tend to have the same distributions as the training database samples which is not the case for the clusters obtained by the k-means clustering algorithm.

The k-means clustering procedure (or the Kohonen map training) starts by selecting a small subset of the full training database (e.g. 50 times the number of clusters that are to be created). Clusters can be created globally or per class. In the former case an evaluation set is constructed which reflects the same a priori class probabilities as the full training database. In the latter case subsets of the different class sets are created. These sets (in case of global clustering there is

only one) are used to determine the centres and standard deviations of the candidate Gaussian units to be derived from the clusters.

The computation of the standard deviations assigned to the map vectors is performed as follows:

1. Compute exact cluster centres. In fact, the Kohonen map vectors divide the input space into clusters (see equation 1), but a map vector is not necessarily the centre of gravity of the cluster members.

2. Compute the standard deviations of the projections of the cluster members on the main axes.

3. Multiply all computed standard deviations by the same factor, which is determined in such a way that about 80% of the cluster members were located inside Gaussian output ellipsoid corresponding to a 0.5 output.

Once the cluster centres and the standard deviations are known, the optimization procedure [Weymaere and Martens, 1991] can be carried out. The optimization process is performed in n_p parallel paths. The parameter n_p is fixed in the beginning of the optimization procedure.

3.1 Results on the BPC Task

In order to evaluate the results obtained by initializing the network's hidden layer starting from a Kohonen map, the recognition rates of the obtained networks were compared to those of the corresponding networks which were initialized starting from the k-means clustering algorithm. Two comparisons were made: global clustering and clustering per class. (The results of the latter clustering method, using k-means clustering are reported in [Weymaere and Martens, 1991]). All the experiments were performed with n_p fixed to 3.

- The global k-means clustering and the 6×6 Kohonen map training were performed on a database consisting of 1800 samples. The 1800 samples reflected the same class probabilities as the full training database. The recognition rates of the initialized networks (as a function of the number of hidden nodes) are depicted on figure 3. The Kohonen map training took 77 minutes of CPU-time while k-means clustering took 75 minutes.

- In case of a clustering per class, 9 clusters per class (a 3×3 map in case of Kohonen clustering) were computed. In both cases a set of 500 examples of the same class was used for creating the clusters of that class. The recognition rates of the initialized networks, as a function of the number of hidden units, are shown in figure 4. The training of the Kohonen maps now took about 9 minutes, while the k-means clustering took no more than 2:20 minutes of CPU-time.

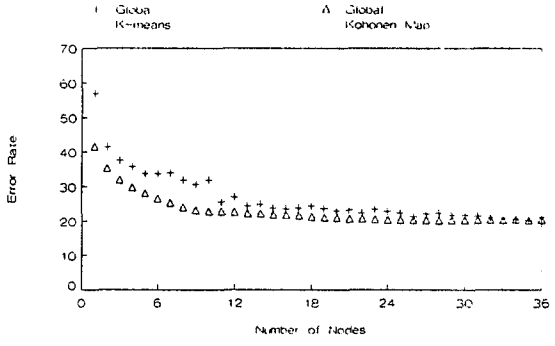

Figure 3: *Comparison of the recognition rates of networks using 36 global clusters (created by k-means algorithm/Kohonen map) to initialize Gaussian nodes.*

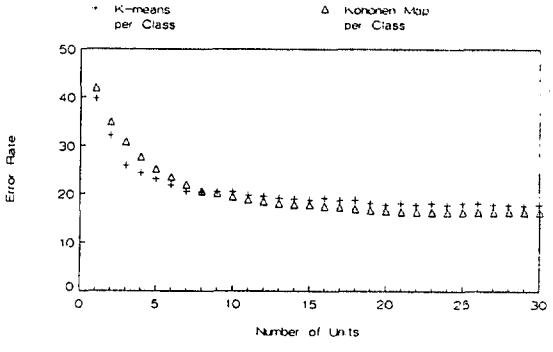

Figure 4: *Comparison of the recognition rates using 9 clusters per class for k-means clustering and for Kohonen map in order to initialize the Gaussian nodes.*

The Kohonen maps of sizes 12×12, 9×9 and 7×7 (which were already trained see subsection 1.1) were used to create clusters for the initialization of the Gaussian nodes. The parameter n_p was fixed to three and the maximum number of nodes was 60. In table 2 the recognition rates obtained by using three Kohonen maps for 30 and 60 hidden units are presented. The general conclusions are:

1. Using k-means clustering or Kohonen map clustering, it is possible to initialize MLPs with a Gaussian hidden layer to a near optimum point in

	144 clusters	81 clusters	49 clusters
30 nodes	83,8%	84,1%	83,0%
60 nodes	85,4%	84,4%	-

Table 2: *Recognition rates on a BPC task for different numbers of clusters which initialize Gaussian nodes.*

the weight space. The performance of the initialized network can be as large as 85,4% which is not far from the optimum performance of 89,20%, obtained with traditional 3-layer MLP trained with EBP.

2. The results obtained by clustering per class are substantially better than those obtained by global clustering (higher recognition rates and considerably less CPU-time).

3. The initial network performances obtained with k-means clustering and Kohonen map training are essentially the same, be it that k-means clustering is computationally more efficient. This superior computational efficiency is mainly devoted to the fact that all inter-sample distances required for k-means clustering can be computed (and stored) in advance. However, this advantage is lost as soon as the cluster datasets become larger.

4 Further training of the net

Once the Gaussian nodes are initialized, further training of the net can be carried out using the gradient descent method. The problem is that it is difficult to select proper learning rates and proper smoothing factors (for the adaptation of the output nodes' weights and for the adaptation of the Gaussian nodes' weights). If the choice of these parameters is inadequate, training can easily lead to a sub-optimum. Very few tests were run until now and the obtained results could still be improved.

The network with Gaussian units whose weights were obtained by our optimization procedure was tested on the three databases (training database, test set1, test set2). The weights to the output units were obtained according to equation 3. The optimization procedure used bpc_so as the training set. Networks with 20 and 30 Gaussian nodes obtained from a Kohonen map of 9×9 were used in this test. The network whose hidden layer consisted of 20 Gaussian nodes was trained for 8 cycles with a batch size of 1500 (this means that weights are adapted after 1500 input examples are presented). The network whose hidden layer consisted of 30 Gaussian nodes was trained for 13 cycles with the same batch size and for 20 additional cycles with a batch size of 3600 and a smaller learning rate. The results are presented in tables 3 and 4.

5 Conclusion

Experiments were carried out to investigate the capabilities of self-organizing feature maps (Kohonen maps) in a speech pattern recognition task. The conclusions of these experiments is that a labeled Kohonen map cannot compete with a feed-forward MLP trained on the same amount of labeled training examples.

	before training	after training
train	82,48%	88,70%
test1	80,58%	86,71%
test2	82,17%	86,94%

Table 3: *Recognition rates on a BPC task, using the network with 30 Gaussian nodes.*

	before training	after training
train	81,58%	87,27%
test1	80,21%	85,28%
test2	81,80%	85,50%

Table 4: *Recognition rates on a BPC task, using the network with 20 Gaussian nodes.*

Afterwards, it was investigated how Kohonen maps could be used to initialize the weights of a 3-layer MLP with a hidden layer of Gaussian units. It was found that initialization to a near optimum point in the weight space is feasible, especially if one starts from a set of small Kohonen maps each derived from examples of a particular output class. It was verified that Kohonen map clustering is a sensible alternative to k-means clustering, a technique which was introduced by Weymaere and Martens (1991) for the initialization of Gaussian networks.

The EBP-training of the initialized Gaussian networks lead to essentially the same recognition rates as were obtained with standard MLPs [Depuydt et al., 1990]. However, the initialization algorithm yields three major advantages:

1. The danger of getting trapped into a local minimum is reduced.

2. The required dimension (size) of the network can be determined without the need for EBP-training.

3. The training of the initialized network takes much less CPU-time than the traditional EBP-training of randomly initialized MLPs.

6 Acknowledgement

This study was made in Electronics Laboratory, University of Ghent in a periode from 1.2.1991 until 1.7.1991. I would like to thank my mentor Prof. Dr. Ir. Pipan from the University of Ljubljana, Faculty of Electrical Engineering and Computer Science for financial support. I am grateful to Prof. Dr. Ir. Vanwormhoudt for letting me work in his laboratory and

enabling me to use all its facilities. Special thanks are due to Dr. Ir. J.P. Martens for putting me on the right track, for correcting this report and for giving many useful advices. Thanks are also due to Ir. N. Weymaere for advising me on the literature, allowing me to use his programs and explaining me matters that I was not yet familiar with.

References

[1] Brauer,P. and Knagenhjelm,P. (1989). "Infrastructure in Kohonen Maps", Proc. IEEE ICASSP, Glasgow, May 1989, Vol. 1, pp. 647-650.

[2] Depuydt,L., Martens,J.P., Van Immerseel, L. and Weymaere, N. (1990). "Improved Broad Phonetic Classification and Segmentation with a Neural Network and a New Auditory Model", ICSLP, Vol. 2, pp. 1041-1044.

[3] Kohonen,T. (1989). "Self-Organisation and Associative Memory", Heiderberg, Springer Verlag, 3^{rd} Edition.

[4] Kohonen,T. (1990). "The Self-organizing Map", Proc. IEEE, vol. 78, no. 9, Sept. 1990, pp. 1464-1480.

[5] Lowe,D. (1989). "Adaptive Radial Basis Function Nonlinearities and the Problem of Generalization", IEE Conference on Artificial Neural Networks, 16-18 October.

[6] Martens,J.P. and Van Immerseel,L. (1990). "An Auditory Model Based on the Analysis of Envelope Patterns", ICASSP90, 402-406.

[7] Powell,M.J.D. (1985): "Radial Basis Functions for Multy-variable interpolation: a review", IMA Conference on algorithms for the approximation of functions and data, RMCS, Shrivenham.

[8] Renalds,S. and Rohwer,R. (1989). "Phoneme Classification Experiments Using Radial Basis Functions", IJCNN Washington.

[9] Rumelhart,D., Hinton,G. and Williams,R. (1986). "Parallel distributed processing: exploration in the microstructure of cognition", Vol. 1, MIT Press.

[10] Weymaere,N. and Martens,J.P. (1991). "A Fast and Robust Algorithm for Feed-forward Neural Networks", To appear in Neural Networks.

[11] Wilpon,J. and Rabiner, L. (1985). "A Modified K-means Clustering Algorithm for Use in Isolated Word Recognition", IEEE Transactions on Acustics, Speech and Signal Processing, Vol. 33, pp. 587-594.