In *Proceedings of the Institute of Acoustics*, Volume 14, Part 6, pp 343 – 350, 1992.

# APPLICATION OF AN ARCHITECTURALLY DYNAMIC NETWORK FOR SPEECH PATTERN CLASSIFICATION

*Visakan Kadirkamanathan & Mahesan Niranjan*

Cambridge University Engineering Department

SUMMARY

We have previously adopted a function estimation approach to the problem of sequential learning with neural networks and derived a network that grows with increasing observations. This network is a single hidden layer Gaussian radial basis function (GaRBF) network with a single output unit. On receiving new observation, the network adds a new hidden unit or adapts the existing parameters by the Kalman filter.

In this paper, we extend the network to have multiple output units. By choosing to adapt only the linear coefficients (hidden - output layer weights), considerable memory savings can be achieved for the resulting Kalman filter than if the parameters of the basis functions were also adapted. Results for this network are presented for the Peterson-Barney vowel classification problem in which the observations are presented sequentially and only once. The performance is comparable to that achieved by some of the standard block estimation methods. This approach can also be viewed as an alternate method of arriving at an approximate complexity of the network required to solve a given problem, eliminating the need for an a priori selection of network size.

Acknowledgements:
This research is supported by an SERC grant GR/H80217

# APPLICATION OF AN ARCHITECTURALLY DYNAMIC NETWORK FOR SPEECH PATTERN CLASSIFICATION

Visakan Kadirkamanathan & Mahesan Niranjan

Cambridge University Engineering Department, Cambridge CB2 1PZ, UK.

## 1. INTRODUCTION

Speech pattern classification can be loosely split into two stages. The first stage involves the appropriate selection of features of the different speech patterns and the second involves the use of a classifier to from class boundaries in the feature space. Neural networks have emerged as a class of powerful nonlinear classifiers and have performed well on classifying static speech patterns [1], [2]. One of the problems associated with using neural networks is the selection of the size of the network for a given classification problem.

The neural network classifier is built by posing the task as a function interpolation or approximation problem. The function estimation approach is therefore the appropriate framework to analyse neural network learning where the task is to estimate or approximate the underlying function from given data. We have adopted this approach to solve the problem of sequential learning with neural networks and derived a network that modifies its architecture dynamically [3], [4]. This network resembles the form of the resource allocating network (RAN) [5] and is also similar to the restricted Coulomb energy (RCE) model of classification [6].

The architecturally dynamic network presented here is a Gaussian radial basis function (GaRBF) network that begins with no hidden units and grows by adding hidden units based on the novelty of the present observation. It learns from each observation sequentially and the estimate of the underlying mapping is formed such that it attempts to combine the past and present information optimally. The network was originally formulated for a single output network and used in solving function approximation and time-series prediction problems [3], [4].

In this paper, we extend the network to include multiple outputs. We also incorporate an efficient (memory requirements-wise) algorithm based on the Kalman filter to train the growing network. We apply this network to the problem of pattern classification. Results on the classification of the Peterson-Barney vowel data are presented in which the classes overlap considerably. We show that the dynamic network attains a performance comparable to other reported results with fixed size networks and nearest neighbour methods.

## 2. THE ARCHITECTURALLY DYNAMIC NETWORK

The basis for the architecturally dynamic network lies in the function space approach to sequential learning with neural networks [3], [4] which reduces to the resource allocating network (RAN) [5]. The network presented here extends the network considered in [3], [4], [5] by consisting of multiple output units and the appropriate modification of the growth criteria. Since each output unit maps a function, the function space approach can be readily used. The only difference being the basis functions common to all output mappings.

The network maps an $M$-dimensional input $\mathbf{x} \in \Re^M$ to an $L$-dimensional output $\mathbf{y} \in \Re^L$. The network output response to an input pattern is a linear combination of the hidden unit responses given by,

$$\mathbf{F}(\mathbf{x}) = \mathbf{W}\Phi(\mathbf{x}) \tag{1}$$

where $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_k(\mathbf{x}), \ldots, \phi_K(\mathbf{x})]^T$ is the vector formed by the responses of the hidden units to the input pattern. The $\mathbf{W} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_l^T, \ldots, \mathbf{w}_L^T]^T$ is the weight matrix of the hidden to output layer weights or coefficients. Hence, the $l^{\mathrm{th}}$ output unit response is given by,

$$f_l(\mathbf{x}) = \mathbf{w}_l^T \Phi = \sum_{k=1}^{K} w_{lk}\phi_k(\mathbf{x}) \tag{2}$$

The $k^{\mathrm{th}}$ hidden unit response is given by the Gaussian radial basis function (GaRBF),

$$\phi_k(\mathbf{x}) = \exp\left\{-\frac{1}{\sigma_k^2}\|\mathbf{x} - \mathbf{u}_k\|^2\right\} \tag{3}$$

where $\mathbf{u}_k$ is the unit centre or mean of the Gaussian and $\sigma_k$ is the spread of the neighbourhood or width of the Gaussian. Platt [5] describes the operation of a hidden unit as storing a local region in the input space – the neighbourhood of $\mathbf{u}_k$. Hence, the $\mathbf{u}_k$ may be viewed as stored patterns. The weights of the hidden – output layer, coefficients $\mathbf{w}_l$, determines the contribution of each hidden unit to a particular output unit.

The network begins with no hidden units. The observations are in the form $(\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{y}_n$ is the target output pattern associated with the input pattern $\mathbf{x}_n$. As observations are received the network grows by storing some of them by adding new hidden units. The decision to store an observation $(\mathbf{x}_n, \mathbf{y}_n)$ depends on its novelty, for which the following two conditions must be met:

$$\|\mathbf{x}_n - \mathbf{u}_{nr}\| \quad > \quad \epsilon_n \tag{4}$$

$$e_n = \quad > \quad e_{\min} \tag{5}$$

where $\|.\|$ is the $L^2$-norm or Euclidean norm, $\epsilon_n$, $e_{\min}$ are thresholds, $\mathbf{u}_{nr}$ is the nearest stored pattern to $\mathbf{x}_n$ in the input space, *viz.*,

$$\mathbf{u}_{nr} = \arg \min_{\mathbf{u}_{k=1,\ldots,K}} \|\mathbf{x}_n - \mathbf{u}_k\| \tag{6}$$

and $e_n$ is the largest error in the output units between the network response and the target,

$$e_n = \max\{e_{n_1}, \ldots, e_{n_l}, \ldots, e_{n_L}\} \tag{7}$$

where $\mathbf{e}_n$ is the output error vector given by,

$$\mathbf{e}_n = \mathbf{y}_n - \mathbf{F}(\mathbf{x}_n) \tag{8}$$

The first criterion says that the input must be far away from stored patterns and the second criterion says that the error in atleast one of the network output units between the response and the target must be significant. The value $e_{\min}$ is chosen to represent the desired accuracy of the network at all output units. The distance $\epsilon_n$ represents the scale of resolution in the input space in which the approximation is done.

When a new hidden unit ($K^{\text{th}}$) is added to the network, the parameters or weights associated with this unit are assigned as follows:

$$[w_{1K}, \ldots, w_{LK}]^T = \mathbf{e}_n \tag{9}$$

$$\mathbf{u}_K = \mathbf{x}_n \tag{10}$$

$$\sigma_K = \kappa \|\mathbf{x}_n - \mathbf{u}_{nr}\| \tag{11}$$

where $\kappa$ is an overlap factor which determines the overlap of the responses of the hidden units in the input space. The value for the width $\sigma_K$ is based on a nearest neighbour heuristic. The addition of a GaRBF hidden unit centred on an observation is similar in spirit to the nearest neighbour and Parzen window methods [7] in which all input observations are retained.

The network begins with $\epsilon_n = \epsilon_{\text{max}}$, the largest scale of interest, typically the size of the entire input space of non-zero probability density [5]. The distance $\epsilon_n$ is decayed exponentially as,

$$\epsilon_n = \max\{\epsilon_{\text{max}}\gamma^n, \epsilon_{\text{min}}\} \tag{12}$$

where $0 < \gamma < 1$ is a decay constant. The value for $\epsilon_n$ is decayed until it reaches $\epsilon_{\text{min}}$. The lower bound ensures that the number of hidden units allocated are also bounded above.

The growth pattern of the RAN and the dynamic architecture network depends critically on $\gamma$ which influences the rate of growth and on $e_{\text{min}}$ which determines the final size of the network together with $\epsilon_{\text{min}}$. These parameters have to be chosen *a priori* and hence the performance of the network depends crucially on their appropriate selection. The exponential decaying of the distance criterion allows fewer basis functions with large widths (smoother basis functions) initially and with increasing number of observations, more basis functions with smaller widths are allocated to fine tune the approximation.

When the observation $(\mathbf{x}_n, \mathbf{y}_n)$ does not satisfy the novelty criteria, the existing $(K-1)$ hidden unit network parameters $\mathbf{c} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_L^T, \mathbf{u}_1^T, \ldots, \mathbf{u}_{K-1}^T, \sigma_1, \ldots, \sigma_{K-1}]^T$, are adapted by the extended Kalman filter (EKF) algorithm. An alternative to the EKF would be to use the LMS algorithm as suggested by Platt [5] for increased computational speed. However, RAN with EKF exhibits fast convergence and also results in a network with smaller number of hidden units [3], [4]. In the next section, we shall develop an algorithm that compromises between using EKF algorithm to adapt the parameters to achieve fast convergence at the expense of added memory requirements and computational complexity, and using the LMS algorithm for fast computation at the expense of slow convergence.

## 3. THE ADAPTATION ALGORITHM

The EKF algorithm requires an additional memory for the error covariance matrix (square symmetric) with storage requirements of $\frac{1}{2}P(P+1)$ where $P$ is the number of parameters being adapted. If we choose to adapt all the parameters $\mathbf{c}$, then $P = K(L + M + 1)$.

Given a parameter vector $\mathbf{w}$, the EKF algorithm obtains the posterior estimate $\mathbf{w}^{(n)}$ from its prior estimate $\mathbf{w}^{(n-1)}$ and its prior error covariance estimate $\mathbf{P}_{n-1}$ as follows (see [8]):

$$\mathbf{w}^{(n)} = \mathbf{w}^{(n-1)} + e_n\mathbf{k}_n \tag{13}$$

where $e_n = y_n - f(\mathbf{x}_n)$ is the prediction error and $\mathbf{k}_n$ is the Kalman gain vector given by,

$$\mathbf{k}_n = \left[R_n + \mathbf{a}_n^T \mathbf{P}_{n-1} \mathbf{a}_n\right]^{-1} \mathbf{P}_{n-1} \mathbf{a}_n \tag{14}$$

where $\mathbf{a}_n = \nabla_w f(\mathbf{x}_n)$ is the gradient vector and $R_n$ is the variance of the measurement noise. The error covariance matrix is updated by,

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T\right] \mathbf{P}_{n-1} \tag{15}$$

$\mathbf{I}$ being the identity matrix. The rapid convergence of the EKF algorithm may prevent the model from adapting to future data. To avoid this problem, a random walk model is often used [9] where the covariance matrix update becomes

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T\right] \mathbf{P}_{n-1} + Q_0 \mathbf{I} \tag{16}$$

The parameter $Q_0$ is a scalar which determines the allowed random step in the direction of the gradient vector.

Since the hidden to output layer weights $\mathbf{w}_l$ are independent of each other, the estimate error covariance matrix will contain a lot of 0 values and is a sparse matrix. Further, since the hidden unit responses are common to all the weight vectors $\mathbf{w}_l$, the error covariance matrix of the estimates for $\mathbf{w}_l$ will be the same for all $l$. Hence by choosing to adapt the weight matrix $\mathbf{W}$ instread of the entire set of parameters, we can reduce this memory requirement to $K \times K$ even though $P = KL$.

Applying the EKF algorithm to the network in which we adapt only the hidden to output layer weights, provides the following adaptation algorithm:

$$\begin{aligned}
\mathbf{W}^{(n)} &= \mathbf{W}^{(n-1)} + \mathbf{e}_n \mathbf{k}_n^T && (17)\\
\mathbf{k}_n &= \left[R_n + \mathbf{\Phi}_n^T \mathbf{P}_{n-1} \mathbf{\Phi}_n\right]^{-1} \mathbf{P}_{n-1} \mathbf{\Phi}_n && (18)\\
\mathbf{P}_n &= \left[\mathbf{I} - \mathbf{k}_n \mathbf{\Phi}_n^T\right] \mathbf{P}_{n-1} + Q_0 \mathbf{I} && (19)
\end{aligned}$$

where $\mathbf{\Phi}_n = [\phi_1(\mathbf{x}_n), \ldots, \phi_K(\mathbf{x}_n)]^T$ is the hidden unit response vector to the input $\mathbf{x}_n$.

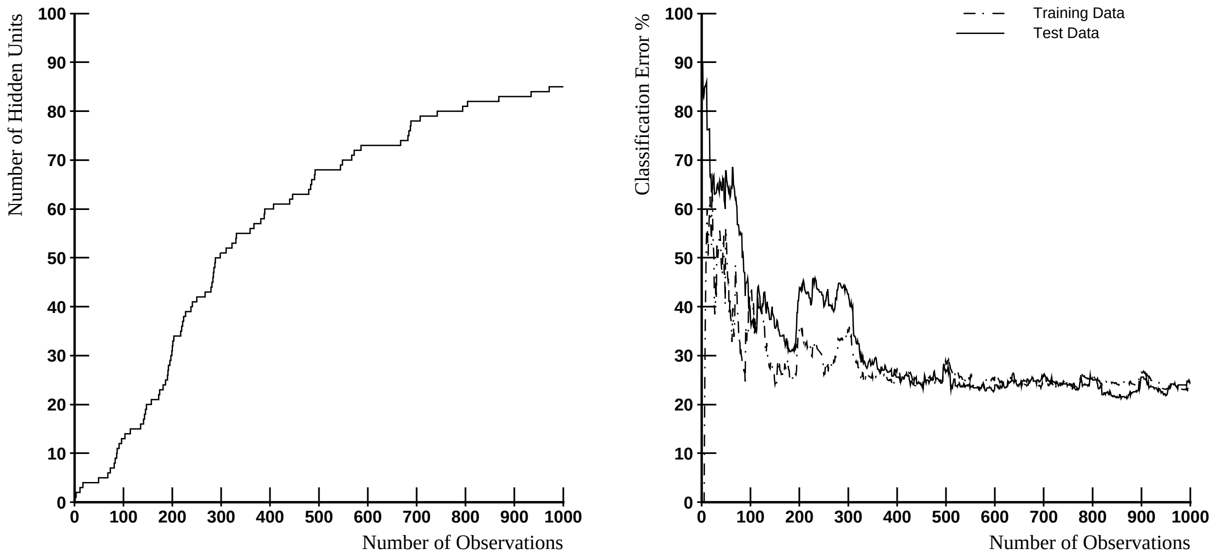
The error covariance matrix $\mathbf{P}_n$ is a $K \times K$ positive definite symmetric matrix, where $K$ is the number of hidden units in the network. Whenever a new hidden unit is allocated the dimensionality of $\mathbf{P}_n$ increases by 1 and hence, the new rows and columns must be initialised. Since $\mathbf{P}_n$ is an estimate of the error covariance of the parameters, we choose,

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & P_0 \end{pmatrix} \tag{20}$$

where $P_0$ is an estimate of the uncertainty in the initial values assigned to the weights, which in our case is also the variance of the observation $\mathbf{y}_n$.

## 4. SPEECH PATTERN CLASSIFICATION

The speech pattern classification task we have chosen here is the benchmark problem of classifying Peterson – Barney vowel data which contain the first and second formant frequencies for 10 spoken vowel classes, spoken by male, female and child American speakers. The training and test data consists of 1000 and 500 observations respectively, the test data being independent of the training data. This data contains classes which overlap considerably and hence is a difficult classification problem. The training data is presented to the dynamic architecture network one by one and only once. Figure 1 shows the growth of the network and the classification error percentage on the test and training data. The curve for the training data represents the classification error on only those observations that have already been presented for training.




(a) Growth Pattern

(b) Classification Error Pattern

Figure 1: The growth pattern and the classification error percentage with increasing number of observations.

The final network had 85 hidden units to achieve a performance with around 24% classification error on the test data. Figure 1 shows that the error tailing off after the network had used 350 observations for learning when the number of hidden units in the network was only around 55. This demonstrates the redundancy of the hidden units prevalent in the final network and the need for modification of the growth criterion for classification problems which in the network is based strictly on interpolation errors rather than on classification errors. The evolution of the class boundaries with increasing number of hidden units and hence with increasing number of observations are shown in figure 2. The observations used in learning the the boundaries are also shown. It illustrates the essential feature of the dynamic network which attempts to use the present data be consistent with that of the past from which it already has learned a representation.

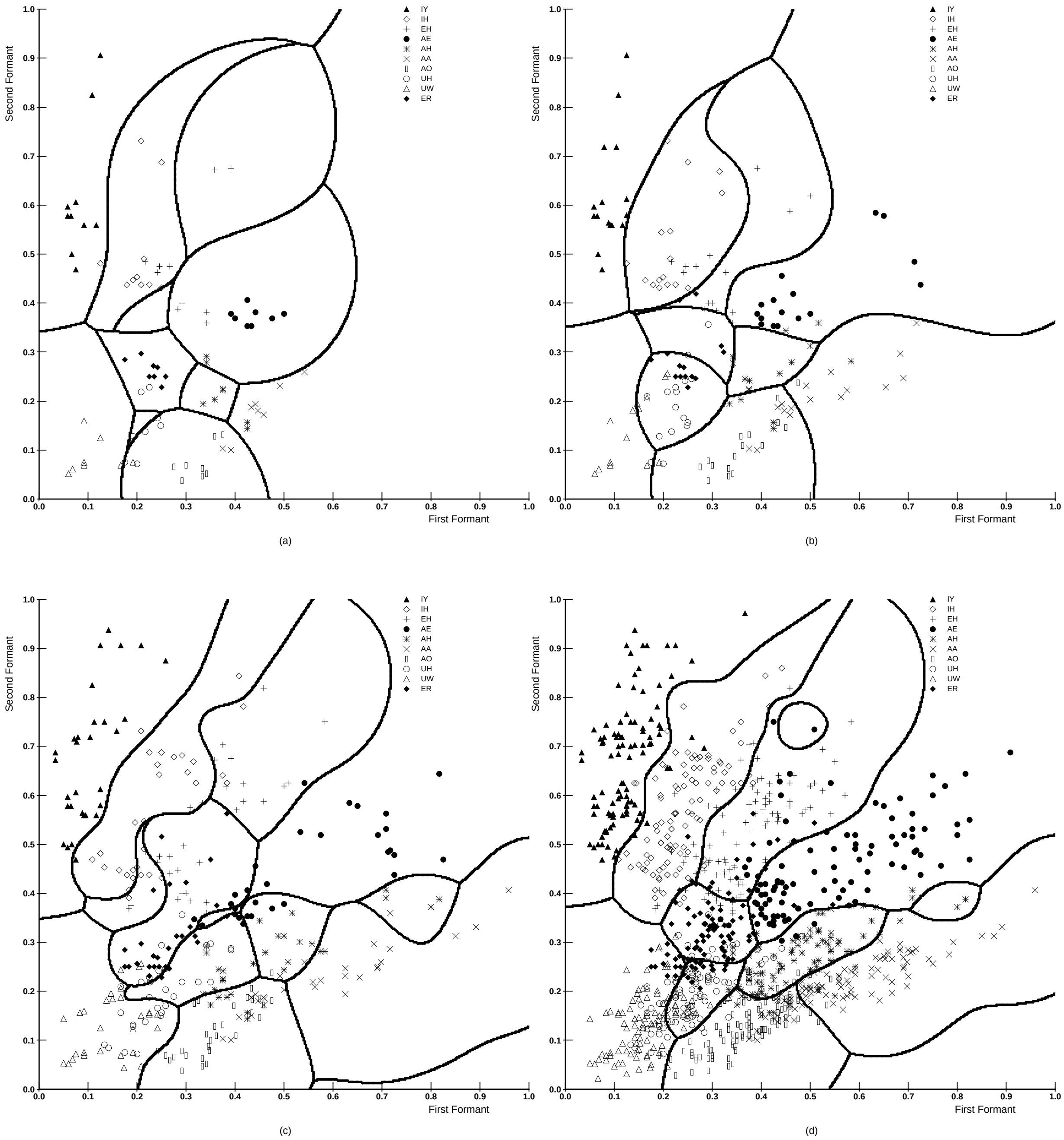



Figure 2: Evolution of the class boundary of the dynamic architecture network with increasing number of hidden units K after seeing N observations. (a) K=10, N=87 (b) K=20, N=157 (c) K=50, N=298 (d) K=85, N=1000.

We also tested Platt's RAN on the above data with each observation presented only once. At the end of training, the RAN had formed 157 hidden units but its classification error percentage on the training data was around 36% and on the test data was 39%. This clearly demonstrates the superior performance of the dynamic architecture network over the RAN which was also observed in time-series prediction results [4].

The 24% classification error in the network is comparable to the simple and more complex methods reported on the same but reduced size dataset. The data set used in our experiments contained child speakers which adds to the difficulty of the classification problem than those in [1]. It is also important to note that the results reported in [1] and [10] are obtained by using the data for many iterations or using them *en-bloc*. Reported results include 22% error with a Gaussian RBF type network by Bridle [10], in which a softmax function is used at the output and the parameters are estimated by minimising an entropy measure. We give some of the results reported by Lowe [1] and the results of our network in table 1.

| Classification Method | % Error (Train data) | % Error (Test data) |
|---|---|---|
| Gaussian Classifier | 23.37 | 25.15 |
| Nearest Neighbour | 0.00 | 22.52 |
| Gaussian RBF (36 hidden) | — | 19.82 |
| Thin plate spline RBF (32 hidden) | — | 18.92 |
| Dynamic network (85 hidden) † | 23.42 | 24.60 |

Table 1: Results from standard classifiers, fixed size networks (from [1]) and the dynamic architecture network (†Training and test data set was more than twice that used for other methods and the observations were also presented only once in the sequential learning procedure). DCM stands for distance to class mean.

Generally, the nearest neighbour methods give good classification performance. Their weakness lies in the run time required to find the best class for the given input pattern and the need to store the entire set of data. The fixed size RBF networks store only a sub-set of the input patterns, learning the input to output mapping, and its run time to find the best class is also faster. However, they require the entire data set to be available for training. The dynamic network goes one step further by training sequentially on each observation (seen only once) and chooses to store an input pattern if it is considered novel.

The architecturally dynamic network can be viewed as a combination of the nearest neighbour and the function interpolation methods. It reaches a size that is smaller than what would be required in the nearest neighbour method but is larger than the minimal network that would be arrived at if the data were used *en-bloc*. However, finding this minimal network with fixed size networks in practice requires many network configurations to be experimented and can be quite cumbersome. The network presented here provides a trade-off by alleviating this difficulty while achieving the same level of performance with a larger number of units.

## 5. CONCLUSION

We have presented an architecturally dynamic network for solving interpolation and pattern classification problems. A fast and efficient Kalman filter based adaptation algorithm has been presented for this network. The network learns sequentially and sees an observation only once. The classification performance of this network was comparable to other block estimation based results. The advantages then are that learning is fast, memory requirements in terms of data storage are minimal and more importantly the problem of choosing the size of the network *a priori* is eliminated. For the network to be more efficient, pruning schemes must be introduced to remove redundant basis functions (hidden units) whose contribution is minimal.

## REFERENCES

[1] D. Lowe. Adaptive radial basis function nonlinearities and the problem of generalisation. In *Proc. IEE Conference on Artificial Neural Networks*, 1989.

[2] M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4, 275-289, 1990.

[3] V. Kadirkamanathan. *Sequential learning in artificial neural networks.* PhD Thesis, Cambridge University Engineering Department, 1991.

[4] V. Kadirkamanathan, M. Niranjan and F. Fallside. Models of dynamic complexity for time-series prediction. In *Proc. ICASSP*, San Francisco, 1992.

[5] J. C. Platt. A resource allocating network for function interpolation. *Neural Computation*, 3(2), 213-225, 1991.

[6] D. L. Reilly, L. N. Cooper and C. Elbaum. A neural model for category learning *Biological Cybernetics*, 45:35–41, 1982.

[7] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley, New York, 1973.

[8] J. V. Candy. *Signal processing: The model-based approach.* McGraw-Hill, New York, 1986.

[9] P. C. Young. *Recursive estimation and time-series analysis.* Springer-Verlag, Berlin, 1984.

[10] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fougelman-Soulie & J. Herault (eds.) *Neuro-computing: algorithms, architectures and applications*, Springer-Verlag, 1990.