Application of Fast Orthogonal Search for the Design of RBFNN

W. Ahmed, D. M. Hummels and M. T. Musavi
Department of Electrical and Computer Engineering
University of Maine, Orono, Maine 04469

***Abstract*— This paper addresses the use of a fast orthogonalization process to find the nodes of an RBF network which produce the best match to a target function. Several applications of RBF networks using this fast orthogonal search technique have been investigated and a classification problem is presented. The problem involves classification of human chromosomes, which is a highly complicated 30 dimensional and 24 class problem. Experimental results show that the fast orthogonal search technique not only outperforms the traditional technique, but it also uses much less time and effort.**

I. Introduction

The growth of neural networks has been heavily influenced by the Radial Basis Function (RBF) neural networks. The application of the RBF network can be found in pattern recognition [1], function approximation, signal processing and more [2]. The two most important parameters of a RBF node, the center and the covariance matrix, have been researched throughly [1]. One issue addresses by these researchers is the reduction of the number of nodes. This reduction involves clustering of the input samples without any consideration of the target function, or the convergence of the weights. The weights (the most significant component of any neural network) of the RBF network were left untouched by most of the researchers. The authors have developed a fast orthogonal search technique which will find a set of most significant nodes and their weights for a given network, using a technique which considers both the structure of the input parameter space and the target function to which the network will be trained [2]. This paper reviews the fast orthogonal search algorithm, and illustrates an application of the technique to the problem of classifying human chromosomes.

II. Radial Basis Function Neural Network

A. RBF structure

The RBF Neural Network gained its popularity for its simplicity and speed. RBF is a simple feed forward neural network with only one hidden layer, and an output layer. The hidden layer consists of a set of neurons or nodes with radial basis functions as the activation function of the neuron. A Gaussian density function is the most widely used activation function and assumed throughout this paper. The output layer is a summing unit, which adds up all of the weighted output of the hidden layer.

The output of the RBF network is given by

$$\hat{y} = f(\vec{x}) = \sum_{k=1}^{N} w_k \phi_k(\vec{x}), \tag{1}$$

where

$$\phi_k(\vec{x}) = (2\pi)^{-p/2} |\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\vec{x}-\vec{c}_k)\Sigma_k{}^{-1}(\vec{x}-\vec{c}_k)^T}. \tag{2}$$

Above, N is the number of network nodes, p is the dimensionality of the input space $\vec{x}$, and w_k, $\vec{c}_k$, and Σ_k represent the weight, center, and the covariance matrix associated with each node.

In supervised learning, if $(\vec{x}, y)$ is a input output pair, where $\vec{x}$ is the input and y is the desired output, then the network should learn the mapping function f, where $y = f(\vec{x})$. The network output can be written using the following matrix form,

$$\hat{\vec{y}} = \Phi \vec{w}, \tag{3}$$

where $\hat{\vec{y}}$ is an M dimensional vector (M is the number of samples), $\vec{w}$ is the N dimensional weight vector. Each column of the Φ matrix contains the output of a node for all M samples.

The problem of finding the network weights reduces to finding the vector $\vec{w}$ which makes the network output $\hat{\vec{y}}$ as close as possible to the vector of desired network outputs $\vec{y}$. Generally, $\vec{w}$ is determined by finding the least square (LS) solution to

$$\Phi \vec{w} = \vec{y}. \tag{4}$$

The method for finding the solution to (4) depends in large part on the structure of the network being designed. If number of nodes is equal to number of training samples (i.e. $N = M$) then the weights are given by $\vec{w} = \Phi^{-1}\vec{y}$ provided that Φ is nonsingular. Often, the number of nodes is much less than the number of training samples. In this case the system of equations (4) is overdetermined, and no exact solution exists. Various alternative methods of finding the weights in this case are discussed in the following section.

B. Solving for the Weights

Finding the weights for the network comes down to finding least squares solution to equation 4. Cholesky decomposition (CD), singular value decomposition (SVD) and orthogonal decomposition are most widely used methods to find the LS solution [6]. These procedures are computationally expensive and have a variety of problems [2]. A more useful approach is to use the orthogonal basis vectors to choose a set of nodes which reduces some error criteria for solving the least squares problem. Chen [3] presents one such algorithm, derived from the Gram-Schimidt procedure, to select the most 'significant' nodes one by one. In [4] a similar algorithm was also presented to select basis function one by one, with emphasis on the characterization of nonlinear systems with random inputs.

The importance of choosing the nodes one by one is significant in several aspects. Selection of nodes one by one provides an insight to the approximation problem, and the network structure. This insight can be used to further modify the network. This selection procedure will also let us meet some physical limitations. A reduction of nodes will provide a corresponding reduction of connections which can be very useful for hardware applications. The desired number of nodes can be easily chosen just by looking at the error behavior.

The orthogonal search method of Chen [3] is very useful, but not so practical. The computational complexity of the procedure is not generally practical for networks of reasonable size. However, the above algorithm may be shown to be extremely redundant. In the next section an efficient algorithm will be presented to perform orthogonalization procedure without explicitly calculating the orthogonal set.

III. The Fast Orthogonal Search

In this section a simple fast algorithm will be presented to find a set of weights (by solving (4)) that are best for the given network. The procedure may be shown to be a computationally efficient procedure for implementing the orthogonal search technique of [3, 4]. Like the orthogonal search technique, during each iteration of the algorithm a set of candidate nodes will be considered to identify which node will provide the best improvement to the approximation of $\vec{y}$. This node will be added to the network, and the procedure continues until either an error criterion is met or the number of nodes in the network reaches a desired value. For the mathmatical development of the algorithm pleases see [5].

The following algorithm presents the appropriate steps to implement the technique.

1. Store all the node outputs in the set $\{\vec{\phi}_j\}$. and initialize the following variables:

$$\begin{aligned} \alpha_i &= \vec{\phi}_i^T \vec{y}, \\ \xi_i^2 &= \vec{\phi}_i^T \vec{\phi}_i, \\ x_i &= [0 \times 1 \ \ vector], \\ U &= [0 \times 0 \ \ matrix], \end{aligned} \tag{5}$$

here, $i = 1, 2, \cdots, N$. Also set

$$Error = \vec{y}^T \vec{y},$$
$$Number\_node\_selected = 0.$$

2. The iteration begins here.
Find the maximum value of α_i^2/ξ_i^2 for all i. Let's say the maximum is at $i = k$.

3. Set U, an upper triangular matrix as,

$$\tilde{U} = \begin{bmatrix} U & \vec{x}_k \\ 0 & \xi_k \end{bmatrix}. \tag{6}$$

4. Update $\vec{x}_i$, which forms part of orthogonal basis for the i^{th} node, by finding β_i first,

$$\beta_i = \frac{1}{\xi_k}(\vec{\phi}_k^T \vec{\phi}_i - \vec{x}_k^T \vec{x}_i) \tag{7}$$

giving,

$$\tilde{\vec{x}}_i = \begin{bmatrix} \vec{x}_i \\ \beta_i \end{bmatrix}. \tag{8}$$

5. Update ξ_i, which is the diagonal term of U,

$$\tilde{\xi}_i^2 = \xi_i^2 - \beta_i^2. \tag{9}$$

6. Finally update α, which determines the next best node, by,

$$\tilde{\alpha}_i = \alpha_i - \frac{\alpha_k \beta_i}{\xi_k}. \tag{10}$$

In the above, from step 4 to step 6 updating is only required when $i \neq k$.

7. Keep repeating from step 4 through step 6 until all the nodes in the set $\{\vec{\phi}_j\}$ have been updated.

8. Delete the k^{th} node from the set $\{\vec{\phi}_j\}$.

9. Increment the *Number_node_selected* by one, and set $Error = Error - \alpha_k^2/\xi_k^2$. If *Error* is less than some error threshold, or the *Number_node_selected* is equal to the desired number of nodes then go to step 10, otherwise go through the steps 2 - 8 again.

10. At this stage we have a U matrix giving the Cholesky decomposition of $\Phi^T\Phi$

$$\Phi^T\Phi = U^T U. \tag{11}$$

Φ is formed only from the set of nodes that reduce the sum squared error of $\hat{\vec{y}}$. We can use the equation above in the normal equation, and then solve for the weights of the selected nodes by the method of forward substitution, and backward substitution [2].

The algorithm presented in this section not only implements the technique to find the weights of a given network, but also allows the user to select the number of nodes. This selection has physical significance since there may exist hardware or software limitations on implementing nodes. If the number of nodes is not an issue than one may be able to find a better network by choosing a sum square error threshold. Also we can look at how the error is behaving as the nodes have been added to the network. This provides an indication of whether addition of a node really makes a difference or not.

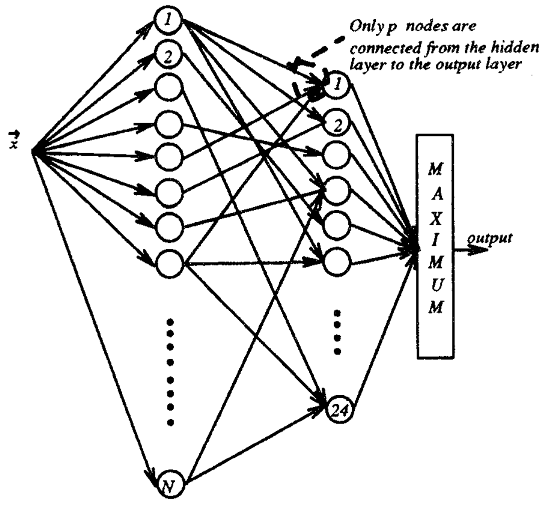
IV. A pattern Recognition Application

A complicated and challenging application of RBF will be discussed in this section. "Karyotyping", the classification of the chromosome in a metaphase into the 24 normal classes has been a very important issue in the medical field for many many years. Classification of chromosomes involves finding a good set of features to describe a chromosome, and a classification technique to identify the chromosomes using the features.

A. RBF Network for Chromosome Classification

The problem of karyotyping involves classifying the chromosome of 30 features (for the data base used here – Copenhagen [7, 8]) into 24 different classes. The chromosomes in a cell consists of 22 pairs of autosomes, one of each pair inherited of each parent, and two sex chromosomes (an X and Y for male, and two X's for female).

Figure 1 illustrates the RBF network for chromosome classification. For the standard RBF networks, all of the N nodes of the hidden layer would be connected to all of the 24 nodes of the output layer. The fast orthogonal search will be able to reduce the insignificant connections and nodes. The reduction of the nodes can be of very large scale for a multiclass problem like the chromosome classification.



Only p nodes are connected from the hidden layer to the output layer
$\vec{x}$
1
2
N
1
2
24
MAXIMUM
output

Figure 1: Network structure for the RBF Network for Chromosome Classification.

B. Results and Analysis

Each pattern of this database is an autosome, the X sex chromosome, or the Y sex chromosome. Each pattern consists of a set of 30 different features, which are the measurements of the normalized area, size, density, normalized convex hull perimeter, normalized length, area, centromeric index, mass centromeric index, length centromeric index, the weighted density distribution density, and others [7].

The RBF neural network was trained with 1000 training patterns. The initial nodes of the network were placed on first 500 of these 1000 patterns. The covariance of each node was chosen to be diagonal as in [2]. The fast orthogonal search was used to only find the best 40 nodes per class and their weights. The search technique successfully found the desired nodes. Figure 2 shows the training error for class 1 as each node was added in. Notice here that only 40 out of 500 RBF nodes are connected to each output node. By looking at figure 2, we see that the training error for class 1 has leveled off by the introduction of the 40^{th} node, implying that introducing another node may not improve the performance at all. Similar conclusion can be made for training other classes.

After training the network, the network was tested by a test set of 3000 patterns different from the training set. The percent error for this test set was 3.84%, which was lower than the ones presented by [7, 8].

The number of initial nodes from which to select a set of nodes of the network can influence the performance. Figure 3 illustrates the results of some simple tests where

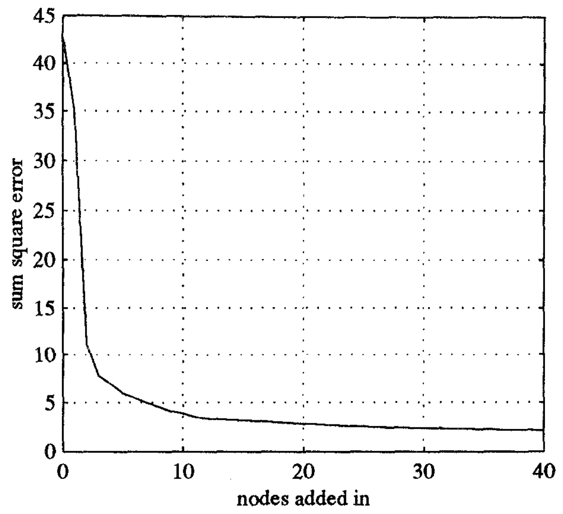


Figure 2: The sum squared error for training class 1 of the Chromosome problem.

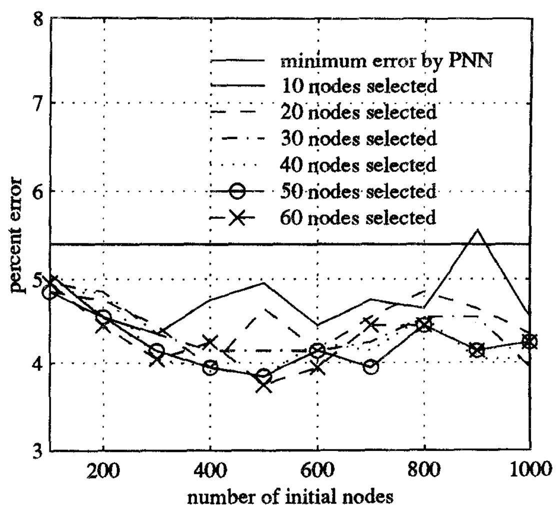


Figure 3: The percent error for the network for different number of the initial node selection. PNN performance at 5.4% is as reported in [8].

the number of initial node assignment was changed for the training procedure described above. The plots in figure 3 gives the percent error (y-axis) for a network constructed by selecting a set of nodes per class from a larger set of initial nodes (x-axis). The test was performed to select 10, 20, 30, 40, 50 and 60 nodes per class from the initial sets. The percentage error was generally lower than the error given by [8].

V. Conclusion

This paper presented a method which provides a simple way to find the most significant nodes of the network and their weights. The technique of fast orthogonal search is implementable using a simple 10 step algorithm. Traditional approaches require significantly more computations. The provided solution is the best to match the target function in a least squares sense. The approach gives a clear indication of the number of nodes to be used. Application of the technique to the problem of chromosome classification demonstrates that the orthogonal search technique may give better performance than that of the other approaches.

References

[1] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris, D.M. Hummels, "On the Training Algorithm for Radial Basis Function Classifiers," *Neural Networks*, vol. 5, pp. 595-603, 1992.

[2] W. Ahmed, D.M. Hummels, M.T. Musavi, "Fast Orthogonal Search for raining Radial Basis Function Neural Networks," submitted to *IEEE Transactions on Neural Networks*,

[3] S. Chen, C.F.N. Cowan, P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks" *IEEE Transactions on Neural Networks*, vol. 2, No. 2, pp. 302-309, Mar. 1991.

[4] M.J. Korenberg, L.D. Paarmann, "Orthogonal Approaches to Time-Series Analysis and System Identification," *IEEE SP Magazine*, July 1991, pp. 29-43.

[5] W. Ahmed, Fast Orthogonal Search for raining Radial Basis Function, M.S Thesis, University of Maine, Orono, Maine, August 1994.

[6] L.L. Scharf, *Statistical Signal Processing, Detection, Estimation, and Time Series Analysis*, Addision-Wesley Inc., 1991.

[7] J. Piper, E. Granum, "On Fully Automatic Feature Measurement for Banded Chromosome Classification", *Cytometry* vol. 10 pp. 242-255, 1989.

[8] W. P. Sweeney Jr., M.T. Musavi, J.N. Guidi "Classification of Chromosomes Using Probabilistic Neural Network", *Cytometry* vol. 16 pp. 17-24, 1994.