Binary Classification by SVM Based Tree Type Neural Networks

Jayadeva[1], Alok Kanti Deb[1], and Suresh Chandra[2]
[1]Department of Electrical Engineering, [2]Department of Mathematics
Indian Institute of Technology, Hauz Khas, New Delhi-110016.
e-mail: jayadeva@ee.iitd.ac.in, alokkanti@hotmail.com, chandras@maths.iitd.ac.in

Abstract- A technique for building a multilayer perceptron classifier network is presented. Initially, a single perceptron tries to correctly classify as many samples as possible. Misclassified samples are taken care of by adding as bias the output of upto two neurons to the parent neuron. The final classification boundary between the two disjoint half spaces at the output of the parent neuron is determined by a maximum margin classifier type SVM applied jointly to the training set of the parent neuron along with the correcting inputs from its child neuron(s). The growth of a branch in the network ceases when the terminal neuron is able to correctly classify all samples from its training set. No a priori assumptions need to be made regarding the number of neurons in the network or the kernel of the SVM classifier. Examples are presented to illustrate the effectiveness of the technique.

I. INTRODUCTION

Feed-forward multi-layer networks have been applied successfully to both function approximation and pattern classification. However, there is in general no clear way to choose the best neural network architecture for a given application. A network which is too small is unable to adequately learn a problem, while unduly large networks tend to overfit the data. Considerable effort has been applied to the development of techniques for incrementally modifying a network model so that it can accomplish a given task. These techniques fall into two categories, viz. pruning techniques and constructive methods. Pruning techniques begin with a larger trained network and then remove network connections and/or hidden units of lesser importance, by following some performance criterion. This not only results in a reduction in model complexity but also a decrease in the overfitting that occurs with large networks. Some well known pruning techniques are described in [1,2]. Constructive methods begin with a small network and gradually increase its size when necessary, in order to complete the learning task. Some constructive techniques for developing neural network architectures can be found in [3-7]. The basic objective in these techniques is to reduce the complexity of the model and also to satisfy some performance criterion.

Multilayer networks are able to represent arbitrarily complex decision regions, but there are no clear guidelines for choosing the number of neurons needed to accomplish a classification task or the best network architecture. For a given problem, most learning algorithms for multilayer networks generally build networks either by adding more layers of processing units, or by adding more neurons with a fixed number of hidden layers. It is also not desirable to run a network construction algorithm until all the training samples have been correctly classified. Insisting on 100% correct classification on the training samples may result in overfitting the data and poor generalization.

In this paper, we present a constructive neural network algorithm for addressing the binary nonlinear classification problem by using a tree-structured neural network in the multi-dimensional input space.

A. Pyramidal Delayed Perceptron

Given two classes of data, one of the approaches to construct a classifier is to build the topology of the network like a pyramid from the top, known as a "pyramidal delayed perceptron" [3]. In this approach, we first try to solve the classification problem by using a single neuron. The desired input-output relations are framed as inequalities. The patterns that are not correctly classified are corrected by adding a suitable bias to the neuron; the bias is the weighted output of a perceptron trained in the same way as the parent neuron. The separating plane obtained for classifying the patterns is obtained by solving an objective function formulated using the L_1-norm. Since this may not maximize the classification margin, the final classification boundary for the samples learnt by a parent neuron and its child neuron/neurons is obtained by maximizing the margin of classification between the two classes of samples. This is done by using a Support Vector Machine (SVM) which minimizes the L_2-norm of the weight vector.

B. Support Vector Machines

Support vector machines (SVMs) have been proposed as a powerful pattern classification technique [8-10]. The most popular SVM is the "maximum margin classifier", which aims at minimizing an upper bound on the generalization error through maximizing the margin between two disjoint half spaces [9,11]. These half spaces might be the original input spaces for a linear classification problem or the higher dimensional feature space for a nonlinear classification problem. The maximal margin classifier represents the classification problem as a convex optimization problem: minimizing a quadratic function under linear inequality constraints. Given a linearly separable training set, $S = \{(\mathbf{X}^i, y^i), i = 1,2,...,N; \mathbf{X}^i \in \Re^n; y^i \in \{-1,1\}\}$ the hyperplane $\mathbf{w}$ that solves the optimization problem,

$$\text{(QPP)} \qquad \underset{\mathbf{w},b}{\text{minimize}} < \mathbf{w} \cdot \mathbf{w} > \tag{1}$$

$$\text{subject to} \quad y^i(< \mathbf{w} \cdot \mathbf{X}_i > + b) \geq 1; i = 1,2,...,N \tag{2}$$

realizes the maximal margin hyper plane with geometric

margin, $\gamma = 1/\|\mathbf{w}\|_2$. For a proof, refer to [9,11]. The solution to the optimization problem (1)-(2) is obtained by solving its dual, which is given by

$$\text{maximize} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha^i \alpha^j < \mathbf{X}^i \cdot \mathbf{X}^j >$$

$$\text{subject to} \quad \sum_{i=1}^{N} y^i \alpha^i = 0$$

$$\alpha^i \geq 0, i = 1,2,...,N \tag{3}$$

If the parameter $\boldsymbol{\alpha}^*$ is the solution of the above optimization problem, then the weight vector $\mathbf{w}^* = \sum_{i=1}^{N} y^i \alpha^{i*} \mathbf{X}^i$ realizes the maximal margin hyperplane with geometric margin $\gamma = 1/\|\mathbf{w}^*\|_2$. As b does not appear in the dual formulation, the value of b^* is found from the primal constraints:

$$b^* = -\frac{\max_{y^i=-1}(< \mathbf{w}^* \cdot \mathbf{X}^i >) + \min_{y^i=1}(< \mathbf{w}^* \cdot \mathbf{X}^i >)}{2} \tag{4}$$

The optimal hyperplane can be expressed in the dual representation in terms of the subset of parameters:

$$\begin{aligned} f(\mathbf{X}, \boldsymbol{\alpha}^*, b^*) &= \sum_{i=1}^{N} y^i \alpha^{i*} < \mathbf{X}^i \cdot \mathbf{X} > + b^* \\ &= \sum_{i \in sv} y^i \alpha^{i*} < \mathbf{X}^i \cdot \mathbf{X} > + b^* \end{aligned} \tag{5}$$

The value of the Lagrangian multiplier α^{i*} associated with each sample $\mathbf{X}^i$ signifies the importance of the sample in the final solution. Samples having a substantial non-zero value of the Lagrangian multiplier constitute the support vectors for the two classes.

For the data $S = \{(\mathbf{X}^i, y^i), i = 1,2,...,N; \mathbf{X}^i \in \Re^n; y^i \in \{-1,1\}\}$ that is linearly separable in the feature space implicitly defined by the kernel $K(x,z)$ the following quadratic optimization problem need to be solved to realize the maximal margin hyperplane in the feature space,

$$\text{maximize} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha^i \alpha^j K(\mathbf{X}^i, \mathbf{X}^j)$$

$$\text{subject to} \quad \sum_{i=1}^{N} y^i \alpha^i = 0$$

$$\alpha^i \geq 0, i = 1,2,...,N \tag{6}$$

The maximal margin hyperplane in the feature space obtained by solving (6) can be expressed in the dual representation as,

$$f(\mathbf{X}, \boldsymbol{\alpha}^*, b^*) = \sum_{i=1}^{N} y^i \alpha^{i*} K(\mathbf{X}^i, \mathbf{X}) + b^* \tag{7}$$

The performance of a SVM depends to a great extent on the choice of the kernel function to transform a data from input space to a higher dimensional feature space [12,13]. The relation between the SVM kernel method and standard regularization theory was explained by Smola, Schölkopf and Müller [12]. The choice of kernel function is data dependent and there are no definite rules governing its choice that might yield a satisfactory performance. An information-geometric method for modifying a kernel to improve the performance was proposed by Amari and Wu [10].

In this work we try to address the nonlinear classification problem by using a tree-structured neural network in the input space. The samples to be learnt are presented to a perceptron, and the input weights to this neuron are obtained by using a linear programming approach. The weights thus obtained shall linearly classify a part or the whole of the data set presented to it. A maximal margin classifier is then found for the data set which was linearly separable. The part of the data set that was misclassified is corrected by adding appropriate biases to the neuron. The biases are generated by using additional perceptrons. In this way, the network grows as a pyramid, while the input weights of each neuron of the network are found such that the neuron acts like a SVM maximizing the classification margin. The remainder of the paper is organized as follows. Section II discusses the general binary classification problem. Section III describes how the binary classification problem can be formulated in a linear programming framework. It also discusses a structured algorithm used for building a tree structured neural network. Section IV contains experimental results. Section V is devoted to concluding remarks.

II. PROBLEM FORMULATION

Neural network learning can be considered as a function approximation problem where the objective is to learn an unknown function $f: \Re^n \to \Re$ (or a good approximation of it) from a set, $S = \{(\mathbf{X}, y) \mid \mathbf{X} \in \Re^n, y \in \Re\}$ of input-output pairs. Different constructive neural network learning algorithms for function approximation generally adopt a greedy strategy to add new neurons to the network so as to minimize the residual error as much as possible. Pattern classification is a special case of function approximation where the function's output y is restricted to one of M ($M \geq 2$) discrete values (or classes) i.e. it involves a real to M-ary function mapping. In our work for the binary classification problem (M=2), it is sufficient for each output to be binary valued (with outputs 0 and 1), which can be easily implemented by the hard-limiting activation function. The task is to obtain a set of weights $\mathbf{w}$ such that

$$\sum_{d=0}^{n} w_d x_d^{(i)} = \begin{cases} \geq \Delta, \text{if } y^i = 1 \\ \leq -\Delta, \text{if } y^i = 0 \end{cases}, \; x_0^{(i)} = 1; i = 1,2,...,N \qquad (8)$$

where i denotes the i-th sample, and Δ is a positive quantity introduced for better separation of the data. In this paper, we show the capability of learning the data by a tree-structured neural network where each neuron is a perceptron acting as a maximal margin classifier for the data linearly separable by it. Thus, in the whole network, each perceptron acts as a maximal margin classifier type of SVM.

III. BUILDING THE NEURAL NETWORK

The network grows one neuron at a time. Given an input-output data set to be learnt, a perceptron should satisfy all the inequalities in (8) for each input. Following Martinelli et.al. [3], the inequalities in (8) can be reduced to the following equalities,

$$\sum_{d=0}^{n} \left(w_d^{(+)} - w_d^{(-)}\right) X_d^i - \left(s_i^{(+)} - s_i^{(-)}\right) = \Delta, \text{if } y^i = 1;\ i \in \text{Class 1} \qquad (9a)$$

$$\sum_{d=0}^{n} \left(w_d^{(+)} - w_d^{(-)}\right) X_d^j + \left(s_j^{(+)} - s_j^{(-)}\right) = -\Delta, \text{if } y^j = 0;\ j \in \text{Class 0} \qquad (9b)$$

where $w_k^{(+)}, w_k^{(-)}, s_i^{(+)}, s_i^{(-)}, s_j^{(+)}, s_j^{(-)}$ are non-negative variables and $X_0^i = X_0^j = 1$. The p-th constraint is satisfied if,

$$s_p^{(-)} = 0, \text{ for } p \in \{1,2,...,m+k\} \qquad (10)$$

where m is the number of samples of Class 1, k is the number of samples of Class 0, and $m+k$ = N is the total number of samples.

Therefore, given a set of input-output data pairs, the obvious objective is to minimize the number of inequalities that are not satisfied. This can be formulated as the following linear programming problem (LPP),

$$\text{(LPP)} \qquad \text{Minimize} \frac{1}{m}\sum_{i=1}^{m} s_i^{(-)} + \frac{1}{k}\sum_{j=1}^{k} s_j^{(-)} \qquad (11)$$

subject to the following constraints,

$$\sum_{d=0}^{n} \left(w_d^{(+)} - w_d^{(-)}\right) X_d^{(i)} = \Delta + \left(s_i^{(+)} - s_i^{(-)}\right), \text{if } y^i = 1 \qquad (12a)$$

$$\sum_{d=0}^{n} \left(w_d^{(+)} - w_d^{(-)}\right) X_d^{(j)} = -\Delta - \left(s_j^{(+)} - s_j^{(-)}\right), \text{if } y^j = 0 \qquad (12b)$$

for i = 1,2,...,m; j=1,2,...,k, where m is the number of samples of Class 1, k is the number of samples of Class 0, and $X_0^i = X_0^j = 1$.

The choice of objective function helps to avoid the computation of trivial solution and ensures better convergence [14].

The LPP is solved to obtain a separating plane that tries to classify as many samples as possible. The N samples can be classified into the four classes of Table I on the basis of the output. Classes C_1, C_2, C_3 and C_4 contain N_1, N_2, N_3, and N_4 samples, respectively. Note that $N_1+N_2+N_3+N_4$ = N = total number of samples.

Table I: CLASSIFICATION OF SAMPLES BASED ON OUTPUT			
Class	**Actual output**	**Desired output**	**No. of Samples**
C_1	0	0	N_1
C_2	1	1	N_2
C_3	0	1	N_3
C_4	1	0	N_4

Depending on the mismatch between the actual output and the desired output for the samples presented to the parent neuron, two neurons A and B are added to the parent neuron as shown in figure 1 to take care of the samples belonging to classes C_3 and C_4 respectively.

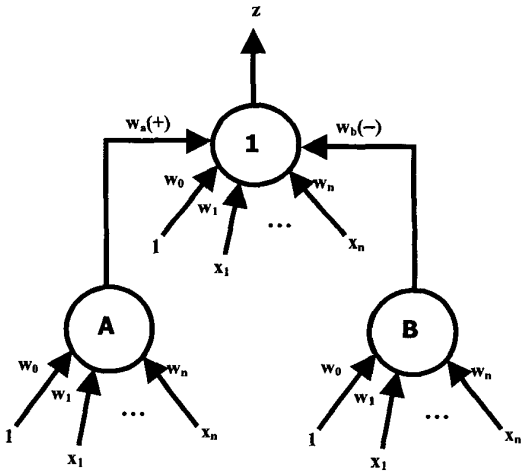

Fig 1. The Parent neuron and its two child neurons A and B

Neuron A contributes a positive input for samples belonging to class C_3, thereby making the total input to the parent neuron for those samples positive. Neuron B contributes a negative input for samples belonging to class C_4 thereby making the total input to the parent neuron for those samples negative. The desired output for the two new perceptrons are shown in Table II.

Table II: REQUIRED OUTPUT OF NEURON A AND B		
Class	**Output of A**	**Output of B**
C_1	0	Don't care
C_2	Don't care	0
C_3	1	0
C_4	0	1

If the outputs of neuron A and B for the i-th sample are denoted by x_a^i and x_b^i respectively, the constraints needed to

be satisfied by the parent neuron, and the desired output of A and B together are given by

$$\sum_{d=0}^{n} w_d x_d^i + w_a x_a^i + w_b x_b^i > \Delta; y^i = 1; i \in Class\,1 \qquad (13a)$$

$$\sum_{d=0}^{n} w_d x_d^i + w_a x_a^i + w_b x_b^i < -\Delta; y^i = 0; i \in Class\,0 \qquad (13b)$$

We have applied the maximum margin classifier type of SVM to this linearly separable data set by considering the parent neuron with the inputs from its child nodes. Its solution yields the input weight of the parent neuron **w**, as well as weights of the interconnections from neuron A and B, *viz.* w_a and w_b. The network grows until a neuron can learn its whole training set without any classification error. The steps of the algorithm are given below.

Learning Algorithm for a Neuron:

Input: $S = \left\{\left(\mathbf{X}^i, y^i\right) \;\middle|\; \mathbf{X}^i \in \Re^n, y^i \in [0\,1]; i = 1,2,\ldots,N\right\}$

Δ = Margin of separation = 0.1(say)

Step 1: Use the input attributes to formulate the equalities as in eq (9a) and eq (9b) in terms of the different components of **w** and suitable slack and surplus variables, all unrestricted in sign.

Step 2: Choose a suitable objective function in terms of the slack variables of the equalities.

Step 3: Apply the simplex algorithm until no exchange of variables is possible. Obtain the connection weights from the final simplex table.

Step 4: Using this weight compute the actual output of a perceptron for all samples.

Step 5: Classify the samples into categories C_1, C_2, C_3 and C_4 as described in Table I, by comparing their desired output with their actual output.

Step 6: If there is no sample is in C_3 and C_4, go to Step 9.

Step 7:

a) If any sample is in C_3, a type-A neuron needs to be added. Develop the training set for A by considering the desired output for samples from different classes according to Table II.

b) If any sample is in C_4, a type-B neuron needs to be added. Develop the training set for B by considering the desired outputs for samples from different classes according to Table II.

Step 8: If either A or B or both need to be added, augment the input dimension of the vertex neuron by considering the desired outputs of A or B or both, as obtained in step 7, as input to the parent neuron.

Step 9: Apply the maximum margin classifier to the augmented input-desired output set if vertex neuron has A or B or both, or to the original input-output relation if the vertex neuron has no child. Obtain the solution.

Step 10: If there is no sample in C_3 or in C_4, the solution yields the weight vector of the vertex neuron else the solution yields the input weight vector of the vertex neuron as well as the connecting weights from neurons A and B.

END

Time Complexity and Convergence issues

While accomplishing a learning task the time required to build a network of this type depends on the size of the dataset and also on the dimensionality of the data. The size of the training set for the neurons keeps on reducing as the network grows and the growth of a branch ceases when the terminal neuron of the branch is able to learn its whole training set. This ensures convergence of the algorithm.

IV. EXPERIMENTAL RESULTS

The performance of our method was tested by implementing the algorithm on both synthetic and real data sets.

i)An Artificial Classification Problem

We first consider the following synthetic two-dimensional data set. Samples belonging to Class 1 are located at the points with co-ordinates (1,8), (4,5), (4,4), (1,1), (6,5), (6,4), (10,8), (10,1), while samples belonging to Class 0 are located at the points with co-ordinates (2,6), (2,3), (8,6), (8,3). Figure 2 shows the SVM based tree type perceptron network having 6 nodes that learns the data. Figure 3 shows the points in two

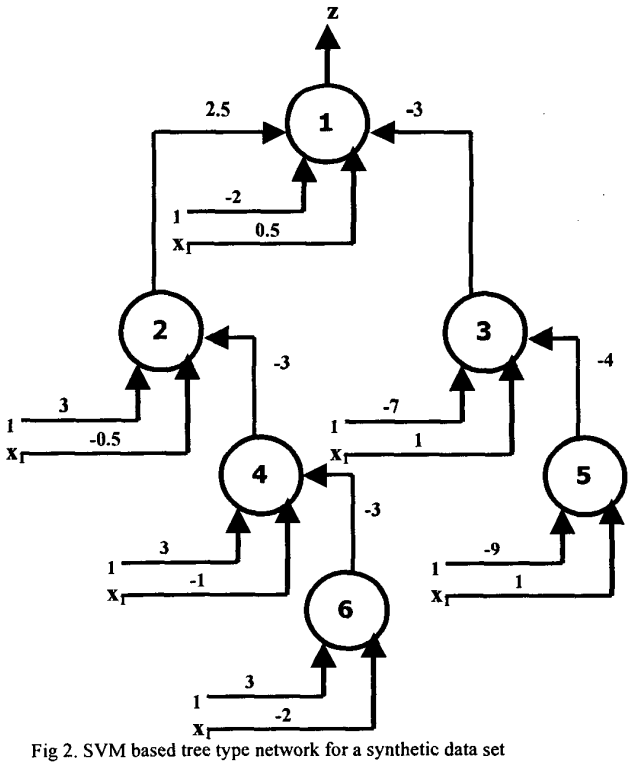

Fig 2. SVM based tree type network for a synthetic data set

dimensions and the decision boundaries learnt by the network.

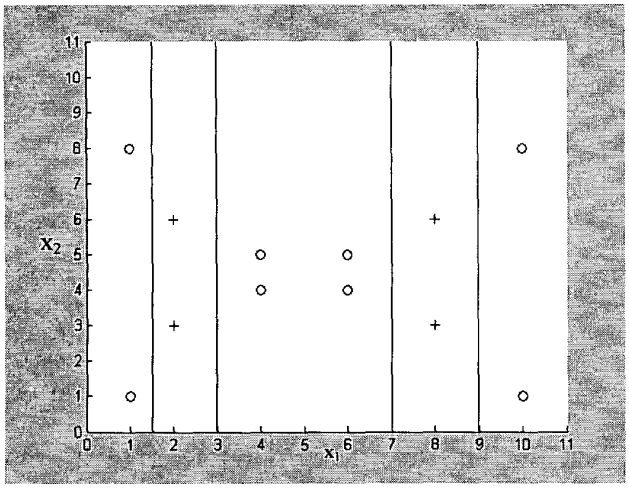

Fig 3. Decision boundaries generated by the SVM based tree type network for the synthetic data set of Example 1.

ii) Classification of Checkers Data

The synthetic data set used was the checkers data set [15], where the data is created by randomly generating a set of 465 points on a 4 × 4 checkers grid. Figure 4 shows the decision regions as learnt by the network. The data was learnt by a tree-type network having 189 nodes spread over 13 layers.

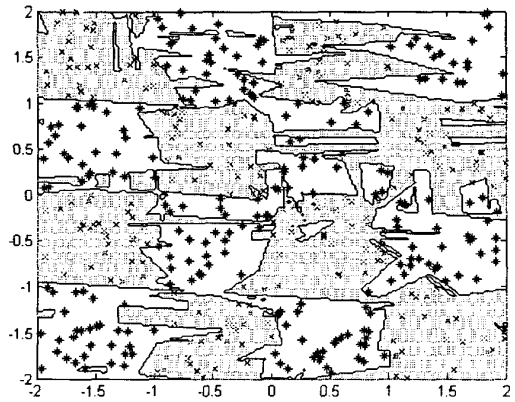

Fig 4. Decision region learnt by the SVM based tree type network for the checkers data

iii) Real Data Sets

There has been considerable debate among neural network researchers for adopting standard procedures for testing the relative merits of different learning algorithms [16,17]. The proposed algorithm was also tested by using some real datasets for binary classification. These were obtained from the *UCI* Machine Learning database [18]. The datasets chosen were the breast cancer-Wisconsin data (699 samples, 9 continuous input attributes) and the heart-statlog data (270 samples, 13 continuous input attributes). Our algorithm was applied to the datasets and evaluated using ten -fold cross-validation.

K-fold Cross Validation

Cross validation is a very popular technique for estimating the generalization error of any learning scheme and there are several versions. In each trial of K-fold cross validation, the training set is randomly split into K mutually exclusive subsets (folds) of approximately equal size. The neural network is trained using K−1 of the folds and tested on the fold left out. The mean of the test error over K trials gives an estimate of the expected generalization error.

Results

Ten-fold cross validation was applied to our algorithm and the training and testing accuracy obtained. The learning accuracy on the training set was expectedly higher than the predictive accuracy on the test set, with the training set accuracy being very close to 100 %. Table III shows a comparison of the results obtained by the proposed algorithm with the constructive feedforward neural network approach of Setiono [19] on the same datasets.

TABLE III: COMPARISON OF RESULTS

	Our Result		**Setiono's Result [19]**	
Name of dataset	Training accuracy (%)	Testing accuracy (%)	Training accuracy (%)	Testing accuracy (%)
Breast cancer-W	99.59±0.27	95.32±0.52	97.47±0.06	96.58±0.24
Heart-statlog	99.074±0.64	77.67±1.27	94.08±1.35	77.56±1.00

V. CONCLUSION

We have proposed an algorithm for constructing a tree-type neural network for solving binary classification problems. Each neuron in the network works like a maximum margin classifier for the samples correctly learnt by it. The samples not separable at a node are taken care of by adding child neurons to the node in question. The network growth stops when there are no classification errors. This approach obviates the need for an a priori choice of the number of neurons, number of hidden layers, or the kernel function in a SVM.

Acknowledgement: The authors would like to acknowledge the critical comments of Prof. M. Gopal and Prof. S.C. Dutta Roy. This work was done in the P.G. Electronics Laboratory, Department of Electrical Engineering, IIT, Delhi, India.

REFERENCES

[1] Y.L Cun, J.S. Denker and S.A. Solla, "Optimal Brain Damage," *Advances in Neural Information Processing Systems(NIPS)*, Vol 2., pp. 598-605, 1990.

[2] B. Hassibi and D.G Stork, "Second-order derivatives for network pruning: Optimal Brain surgeon," *Advances in Neural Information Processing Systems (NIPS)*, Vol. 5, pp. 164-171, 1993.
[3] G. Martinelli, L.P. Ricotti, S. Ragazzini and F.M. Mascioli,"A Pyramidal Delayed Perceptron", *IEEE Transactions on Circuits and Systems*, Vol. 37:9, pp. 1176-1181, 1990.
[4] F.M.F. Mascioli and G. Martinelli, "A Constructive Algorithm for Binary Neural Networks: The Oil-Spot Algorithm," *IEEE Transactions on Neural Networks*," Vol. 6, No. 3, pp. 794-797, May 1995.
[5] M. Muselli, "On Sequential Construction of Binary Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 6., No. 3, pp. 678-690, May 1995.
[6] S. Young and T. Downs, "CARVE-A Constructive Algorithm for Real Valued Examples," *IEEE Transactions on Neural Networks*, Vol. 9, No, 6, pp. 1180-1190, Nov 1998.
[7] R. Parekh, J. Yang and V. Honavar, "Constructive Neural Network Learning Algorithms for Pattern Classification", *IEEE Transactions on Neural Networks*, Vol .11, No.2, pp. 436-451, 2000.
[8] B.Schölkopf, K.K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers," *IEEE transactions on Signal Processing*, Vol. 45, No. 11, pp. 2758-2765, Nov 1997
[9] V. Vapnik, "*Statistical Learning Theory*," Wiley, 1998.
[10] S. Amari and S. Wu, " Improving support vector machine classifiers by modifying kernel functions", *Neural Networks*, 12, pp. 783-789, 1999.
[11] N. Cristianini and J. Shawe-Taylor, "*An Introduction to Support Vector Machines and other kernel based learning methods*," Cambridge University Press, 2000.
[12] A. J. Smola, B. Schölkopf, K. R. Müller, "The connection between regularization operators and support vector kernels", *Neural Networks*, 11, pp. 637-649, 1998.
[13] A.J. Smola and B. Schölkopf, "On a Kernel-Based Method for Pattern Recognition, Regression, Approximation, and Operator Inversion", *Algorithmica*, 22, pp. 211-231, 1998.
[14] K.P. Bennett and O.L. Mangasarian, "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets," *Optimization Methods and Software*, 1, pp. 23-34, 1992.
[15] S.S. Keerthi, S.K. Shevade, C. Bhattacharya, and K.R.K. Murthy (1999, Mar) A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. Intell. Syst. Lab., Dept of Comput. Sci. Automat., Indian Inst. Sci., Bangalore, Karnataka, India.
Available on-line at http://guppy.mpe.nus.edu.sg/~mpessk
[16] L. Prechelt, "A Quantitative Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice," *Neural Networks*, Vol. 9, No.3, pp. 457-462, 1996.
[17] L. Prechelt,"Proben 1-A Set of Neural Network Benchmark Problems and Benchmarking Rules," *Technical Report* 21/94, Universität Karlsruhe, Germany.
[18] C.L Blake and C.J. Merz,"UCI Repository for machine learning databases" Irvine, CA: University of California, Department of Information and Computer Science.
On-line at http://www.ics.uci.edu/~mlearn/MLRepository.html
[19] R. Setiono, "Feedforward Neural Network Construction using Cross Validation," *Neural Computation*, Vol. 13, No. 12, pp. 2865-2877, December 2001.