International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:2, 2007

SVM Based Model as an Optimal Classifier for the Classification of Sonar Signals

Suresh S. Salankar and Balasaheb M. Patre

***Abstract*—**Research into the problem of classification of sonar signals has been taken up as a challenging task for the neural networks. This paper investigates the design of an optimal classifier using a Multi layer Perceptron Neural Network (MLP NN) and Support Vector Machines (SVM). Results obtained using sonar data sets suggest that SVM classifier perform well in comparison with well-known MLP NN classifier. An average classification accuracy of 91.974% is achieved with SVM classifier and 90.3609% with MLP NN classifier, on the test instances. The area under the Receiver Operating Characteristics (ROC) curve for the proposed SVM classifier on test data set is found as 0.981183, which is very close to unity and this clearly confirms the excellent quality of the proposed classifier. The SVM classifier employed in this paper is implemented using kernel Adatron algorithm is seen to be robust and relatively insensitive to the parameter initialization in comparison to MLP NN.

***Keywords*—**Classification, MLP NN, backpropagation algorithm, SVM, Receiver Operating Characteristics.

I. Introduction

PATTERN recognition is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories). The goal of pattern-recognition is to build machines, called, classifiers, that will automatically assign measurements to classes. A natural way to make class assignment is to define the decision surface. The decision surface is not trivially determined for many real-world problems. The central problem in pattern-recognition is to define the shape and placement of the boundary so that the class-assignment errors are minimized. In classification problem, the task is to assign new inputs to one of a number of discrete classes or categories. Here, the functions that we seek to approximate are the probabilities of membership of the different classes expressed as functions of the input variables.

Neural networks have been employed efficiently as pattern classifiers in numerous applications [1]. These classifiers are non-parametric and make weaker assumptions on the shape of the underlying distributions of input data than traditional statistical classifiers. Therefore, they can prove more robust when the underlying statistics are unknown or the data are generated by a nonlinear system. The motivation behind the use of neural network classifiers in sonar systems is the desire to emulate the remarkable perception and pattern recognition capabilities of humans and animals, such as the powerful ability of dolphins and bats to extract detailed information about their environments from acoustic returns [2]-[4]. Automatic sonar signals classification is a domain very rarely explored by neural networks and therefore, there is a need to develop automatic intelligent knowledge extraction. In view of this, design of optimal classifier for classification of sonar signals has been investigated in this paper using neural network. Performance of neural network classifiers is seen to be affected by the choice of the parameters of the network architecture, training algorithm, and input signals, as well as parameter initialization [5], [6].

A neural network performs pattern recognition by first undergoing a training session, during which the network is repeatedly presented a set of input patterns along with the category to which each particular pattern belongs. Later, a new pattern is presented to the network that has not been seen before, but which belongs to the same population of patterns used to train the network. The network is able to identify the class of that particular pattern because of the information it has extracted from the training data. Pattern recognition performed by a neural network is statistical in nature, with the patterns being represented by points in a multidimensional decision space. The decision space is divided into regions, each one of which is associated with a class. The decision boundaries are determined by the training process. The construction of theses boundaries is made statistical by the inherent variability that exists within and between classes.

Optimal design of classifier is investigated using MLP neural network on sonar database [7], [8]. The task is to train a classifier to discriminate between sonar signals bounced off a metal cylinder and those bounced off a rough cylindrical rock. Using the first 104 instances for training, they found that MLP NN (one hidden layer with 12 neurons) trained with standard backpropagation algorithm attained an average of about 84.7 % accuracy on the remaining 104 test instances.

Suresh S. Salankar is with the Electronics & Telecommunication Department, B.D. College of Engineering, Sevagram, India 442102 (e-mail: salankar_ss@ rediffmail.com).

Dr. Balasaheb M. Patre is with the Instrumentation Department, S.G.G.S. Institute of Engineering & Technology, Nanded, India 431606 (e-mail: bmpatre@yahoo.com).

A new classification system based on statistical learning theory, called the Support Vector Machine (SVM) has recently been applied to the problem of signal classification [9]. The main idea behind this classification technique is to separate the classes with a surface that maximize the margin between them, using boundary data point to create the decision surface. We will restrict ourselves to the special case of pattern recognition, where the function is an indicator function. The proposed SVM classifier uses the idea of large margin classifiers for training. This decouples the capacity of the classifier from the input space and at the same time provides good generalization. This paper determines the optimal parameters of SVM-based classifier and compares the performance of MLP NN on the sonar data set collected from the published studies.

The paper is organized as follows. In Section II the data partition schemes of sonar data is given in order to design a classifier. Various important performance measures to assess estimated neural network model are described in section III. Section IV describes design of optimal MLP NN for the binary classification task. In section V, SVM-based classifier was designed and it was tested on the test data set to see how robust it was. The analysis of designed SVM-based classifier is carried out on the basis of their validation performance with respect to the performance measures such as MSE, NMSE (normalized mean square error), r (correlation coefficients), percent classification accuracy and area under ROC curve on the testing instances. Finally, the conclusions are discussed in section VI; with a recommendation to use the proposed SVM-based classifier.

II. Data Description

The sonar data is obtained by Terry Sejnowski, now at the Salk Institute and the University of California at San Deigo. The data set was developed in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center. This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [7]. The data set, "sonar.data", is in the standard CMU Neural Network Benchmark format. The sonar database constitutes 208 instances with 60 continuous-valued inputs and one output denoting the class of the instance. The first 104 samples (1:104) are used for training; the next 104 samples (105:208) for testing and classifier comparison purpose. The second data set is generated from the first one through swapping of training and testing exemplars. That is, the samples used for training in data set1 are now used as testing data and the samples used for testing in data set 1 are now used for the purpose of training. This data set is referred to as data set 2. The fallowing Table I highlights the data partition schemes employed in order to design a classifier. This data set can be used in a number of different ways to test learning speed, quality of ultimate learning, ability to generalize, or combinations of these factors.

TABLE I
DATA PARTITION SCHEME FOR NEURAL NETWORK BASED CLASSIFIER

Data Partition	Training instances	Testing instances
Set 1 (Normal tagging)	1:104 (104 samples) "Metal Cylinder"samples = 49 "Rock" samples= 55	105:208 (104 samples) "Metal Cylinder" samples= 62 "Rock" samples= 42
Set 2 (Reverse tagging)	105:208 (104 samples) "Metal Cylinder" samples= 62 "Rock" samples= 42	1:104 (104 samples) "Metal Cylinder" samples= 49 "Rock" samples= 55

III. Performance Measures

The estimated neural network models should be assessed on the basis of important performance measures. When used as a classifier, the MSE and NMSE of the neural network model on the test data set should be as low as possible The lower bound may be specified as an error threshold by the user. The correlation coefficient of the estimated model should ideally approach unity. The classification accuracies of the model should ideally approach 100 %. In addition, the area under ROC must come close to unity for reliable classification. In the following paragraphs, the performance measures used for validation of neural network model are explained.

A. MSE (Mean Square Error):

The formula for the mean squared error is:

$$MSE = \frac{\sum_{j=0}^{P}\sum_{i=0}^{N}\left(d_{ij} - y_{ij}\right)^2}{NP}$$

Where P = number of output processing elements, N = number of exemplars in the data set, y_{ij} = network output for exemplar i at processing element j, d_{ij} = desired output for exemplar i at processing element j.

B. NMSE (Normalized Mean Square Error):

The normalized mean squared error is defined by the following formula:

$$NMSE = \frac{P\,N\,MSE}{\sum_{j=0}^{P}\frac{N\sum_{i=0}^{N} d_{ij}^{\,2} - \left(\sum_{i=0}^{N} d_{ij}\right)^2}{N}}$$

Where P = number of output processing elements, N = number of exemplars in the data set, MSE = mean square error, d_{ij} = desired output for exemplar i at processing element j.

C. Correlation coefficient (r):

By definition, the correlation coefficient between a network output x and a desired output d is:

$$r = \frac{\frac{\sum_i \left(x_i - \bar{x}\right)\left(d_i - \bar{d}\right)}{N}}{\sqrt{\frac{\sum_i \left(d_i - \bar{d}\right)^2}{N}}\sqrt{\frac{\sum_i \left(x_i - \bar{x}\right)^2}{N}}} \quad \text{where} \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i$$

The size of the mean square error (MSE) can be used to determine how well the network output fits the desired output, but it doesn't necessarily reflect whether the two sets of data move in the same direction. For instance, by simply scaling the network output, we can change the MSE without changing the directionality of the data. The correlation coefficient (r) solves this problem. The correlation coefficient is confined to the range [-1, 1]. When r = 1 there is a perfect positive linear correlation between x and d, that is, they covary, which means that they vary by the same amount. When r = -1, there is a perfectly linear negative correlation between x and d, that is, they vary in opposite ways (when x increases, d decreases by the same amount). When r = 0 there is no correlation between x and d, i.e. the variables are called uncorrelated. Intermediate values describe partial correlations. For example, a correlation coefficient of 0.88 means that the fit of the model to the data is reasonably good.

D. Receiver Operating Characteristics (ROC):

Receiver Operating Characteristic (ROC) matrices are used to show how changing the detection threshold affects detections versus false alarms. If the threshold is set too high then the system will miss too many detection. Conversely, if the threshold is set too low then there will be too many false alarms. The percentage of detections classified correctly (Sensitivity or true positive rate) is plotted against the percentage of non-detections incorrectly classified as detections (i.e. false alarms or false positive rate) as a function of the detection threshold. ROC enables the user to evaluate a model in terms of the trade-offs between sensitivity and specificity. It is the best way to evaluate a detector. The performance of classification for test data set is assessed by calculating the area under the ROC curve (A_Z). It is noticed that the values for A_z range from 0.5 for chance to 1.0 for a perfect classifier.

E. Confusion Matrices:

A confusion matrix is a simple methodology for displaying the classification results of a network. The confusion matrix is defined by labeling the desired classification on the rows and the predicted classifications on the columns. For each exemplar, a 1 is added to the cell entry defined by (desired classification, predicted classification). Since we want the predicted classification to be the same as the desired classification, the ideal situation is to have all the exemplars end up on the diagonal cells of the matrix (the diagonal that connects the upper-left corner to the lower right).

IV. DESIGN OF A MLP-BASED CLASSIFIER

The configuration of the MLP NN is determined by the number of hidden layers, number of the neurons in each of the hidden layers, as well as the type of the activation functions used for the neurons. It has been proved that the performance of the network does not depend much on the type of the activation function (as long as it is non-linear), the choice of the number of hidden layers and the number of units in each of the hidden layers is critical. It has been established that an MLP NN that has only one hidden layer, with a sufficient number of neurons, acts as universal approximators of non-linear mappings [10]. Experimentally, it can be verified that the addition of extra hidden layer can enhance the discriminating ability of the NN model. However, it does so at the cost of the added computational complexity. The trade-off between accuracy and complexity of the model should be resolved carefully. In practice, it is very difficult to determine a sufficient number of neurons necessary to achieve the desired degree of approximation accuracy. Frequently, the number of units in the hidden layer is determined by trial and error. The possible parameter variations chosen for this MLP NN are listed in table II.

To determine the weight values, one must have a set of examples of how the outputs should relate to the inputs. The task of determining the weights from these examples is called *training* or *learning*, and it is basically a conventional estimation problem. That is, the weights are estimated from the examples in such a way that the network, according to some metric, models the true relationship as accurately as possible. When a NN has been trained, the next step is to evaluate it. This is done by a standard method in statistics called *independent validation*. This method divides the available data into a training set and a test set. The entire data set is usually randomized first. The training data are next split into two partitions; the first partition is used to update the weights in the network, and the second partition is used to assess (or cross-validate) the training performance. The test data are then used to assess how well the network has generalized. The learning and generalization ability of the

TABLE II
VARIABLE PARAMETERS OF MLP NN CLASSIFIER

Parameter	Typical Range
Number of hidden layers	(1,3)
Number of hidden neurons	(1,50)
Learning-rate parameter	(0,1)
Momentum constant	(0,1)
Transfer function in output layer	Tanh, lintanh, softmax, linear
Learning Rule	Momentum, conjugate-gradient, step, quick-propagation, delta bar delta, Levenberg Marquardt

estimated NN based classifier is assessed on the basis of certain performance measures such as MSE, NMSE, correlation coefficients, area under the ROC curve, and the rate of correct classification. Since it is very likely that one ends up in a "bad" local minimum, the network should be trained a couple of times (typically at least three times), starting from different initial weights. NeuroSolutions (version 5.0) and Neural Network Toolbox for MATLAB (version 7.0) are specifically used for obtaining results.

For classification, the output processing element must be nonlinear. Here, as the outputs are 1 and 0, the operating point of the hidden processing elements is normally driven to saturation. However, for comparison, linear transfer function is also considered in the output layer along with the other nonlinear transfer functions such as tanh, lintanh, and softmax. The softmax may be used as the output of any MLP to allow interpretation of the output as a probability, as normally is the case in classification. Since the ultimate objective of a pattern classifier is to achieve an acceptable rate of correct classification, this criterion is used to judge when the variable parameters of the MLP (used as a pattern classifier) are optimal.

Starting with a single hidden layer MLP NN as a classifier, issue is how to choose the number of hidden neurons. In a rigorous experimental study, the number of hidden neurons in the hidden layer is gradually increased from 1 to 50 and run the network several times for each with different weight-initializations for 1000 epochs. The process of training is closely monitored with an eye over crucial performance measures in order to judge the optimality of the classifier. Variation of mean MSE as a function of the number of PEs is shown in Fig. 1. For Pentium 4, 1.7 GHz, 768 MB RAM Computer: training of MLP NN continues at the rate of 78.33 (average value) epochs per second.

It is demonstrated that the best network should have 8 neurons in the hidden layer. In addition, the transfer function of neurons in hidden layers as well as output layer should be hyperbolic-tangent (tanh) and the network should be trained using step learning algorithm for the best performance. The optimal parameter settings for MLP NN based classifier are as fallows. PEs in Hidden layer =8, Transfer function of PEs in hidden layer and output layer =tanh, Learning rule in hidden and output layers = step.

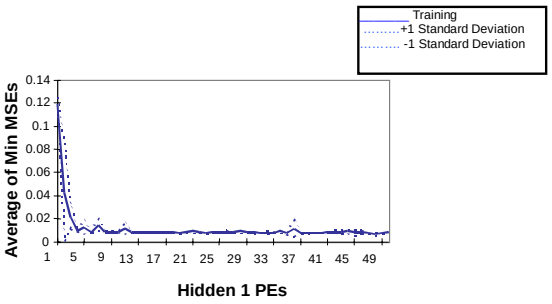

Fig. 1 Average of Minimum MSEs Vs Number of PEs in the Hidden Layer

As there are 60 numeric inputs and one symbolic output (translated into two numeric-valued outputs, where "Metal Cylinder" is 0 1 and "Rock" is 1 0) for the given system, the number of input and output processing elements is chosen as sixty and two, respectively. Once the design of the MLP classifier is finalized as (60-8-2), it is run at least three times with different initialization of connection weights. Table III displays the various important performance measures of MLP classifier on different datasets for the "Rock" as well as "Metal" instances.

To what extent the classifier is able to correctly classify the exemplars is the most important criterion for its proper evaluation. This is expressed as "% Correct" in the table. However, other performance measures such as MSE, NMSE, and correlation coefficients are included only as a matter of record, since a small MSE or NMSE does not necessarily imply good generalization (i.e., good performance with data not seen before). There are 104 instances in the testing data set out of which "Rock" instances are 42 and "Metal" instances are 62. It is noticed that the classifier recognizes two "Rock" instances as "Metal" ones entailing misclassification. However, it classifies 40 "Rock" instances as "Rock" only. Thus, the classification accuracy over "Rock" instances is 40 correct classifications out of 42 samples, that is, 95.238 %. Similarly, the classification accuracy over "Metal" instances are 53 correct classifications out of 62 instances amounting to

TABLE III
PERFORMANCE MEASURES OF MLP CLASSIFIER ON DIFFERENT DATA SETS

S. N.	Data set	Performance measures of MLP Classifier							
		"Rock" instances				"Metal" instances			
		% Correct	MSE	NMSE	r	% Correct	MSE	NMSE	r
1	Testing Set 1	95.238	0.09014	0.374398	0.805246	85.4838	0.0904363	0.3756371	0.807992
2	Training Set 1	100	0.006985	0.029012	0.986715	100	0.004791	0.0199001	0.99089
3	Testing Set 2	67.272727	0.1660514	0.666424	0.644374	87.755	0.166857	0.6696576	0.644712
4	Training Set 2	100	0.006182	0.025676	0.987664	100	0.0057032	0.023688	0.9885588

85.4838 %. In order to confirm whether the proposed model is really consistently capable of near optimum classification, different data partition as in Set 2 (reverse tagging order) is used to train the classifier. The classification accuracy over "Rock" instances is 37 correct classifications out of 42 samples, that is, 67.272727%. Similarly, the classification accuracy over "Metal" instances is 43 correct classifications out of 62 instances amounting to 87.755%.

For specified data sets, calculating the area under the ROC curve assesses the classification performance. Here sensitivity or detections is plotted against (1-specificity) or false alarms. For a perfect classifier, the area under the ROC must approach unity. Fig. 2(a) and 2(b) demonstrates ROC curve for MLP NN based classifier on test data sets.

Results of ROC analysis of the MLP NN Classifier for different data partition are listed in Table IV. It is seen from Table III and IV, that the MLP classifier gives an excellent performance on the training dataset, which is not desirable. This is because; it might simply indicate a tendency of subtle memorization while learning. However, this performance degrades considerably on the testing exemplars. In view of these facts, it may be inferred that the chosen configuration of the network may not be capable to operate as a reasonable classifier. Its performance is not consistently good showing somewhat dependency on specific data partition chosen for training the MLP NN model.

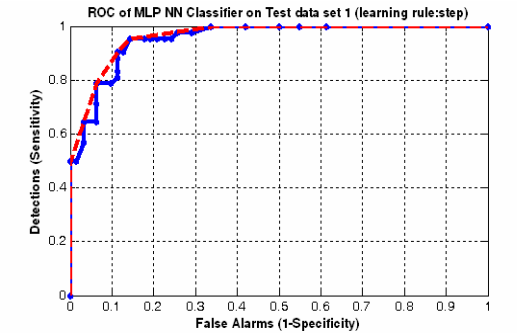

Fig. 2 (a) ROC curve for MLP based classifier on test data set. (Set 1: Normal tagging)

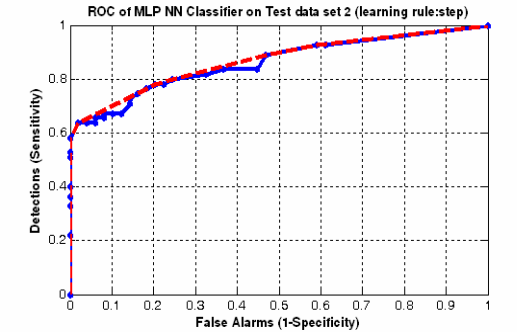

Fig. 2 (b) ROC curve for MLP based classifier on test data set. (Set 2: Reverse tagging)

TABLE IV
RESULTS OF ROC ANALYSIS OF MLP NN CLASSIFIER

Data set	ROC Analysis of MLP NN Classifier	
	Area under an ROC curve	Area under an ROC curve
Testing Set 1	0.953533	0.962558
Testing Set 2	0.861224	0.869388

V. DESIGN OF A SVM-BASED CLASSIFIER

SVM are based on statistical learning theory and have the aim of determining the location of decision boundaries that produce the optimal separation of classes. In the case of a two-class pattern recognition problem in which the classes are linearly separable the SVM selects from among the infinite number of linear decision boundaries the one that minimises the generalisation error. Thus, the selected decision boundary will be one that leaves the greatest margin between the two classes, where margin is defined as the sum of the distances to the hyperplane from the closest points of the two classes). This problem of maximising the margin can be solved using standard Quadratic Programming (QP) optimisation techniques. The data points that are closest to the hyperplane are used to measure the margin; hence these data points are termed 'support vectors'. Consequently, the number of support vectors is small. If the two classes are not linearly separable, the SVM tries to find the hyperplane that maximises the margin while, at the same time, minimising a quantity proportional to the number of misclassification errors. The trade-off between margin and misclassification error is controlled by a user-defined constant. SVM can also be extended to handle non-linear decision surfaces. Boser et al. [11] propose a method of projecting the input data onto a high-dimensional feature space using kernel functions and formulating a linear classification problem in that feature space. Further, more detailed discussion of the computational aspects of SVM can be found in [12]. The proposed SVM is implemented using the kernel Adatron algorithm and uses the idea of large margin classifiers for training. This decouples the capacity of the classifier from the input space and at the same time provides good generalization. This is an ideal combination for classification.

A. Topology of the SVM machine with RBF kernels

Support vector machines (SVMs) are a radically different type of classifier that have attracted a great deal of attention lately due to the novelty of the concepts that they bring to pattern recognition, their strong mathematical foundation, and their excellent results in practical problems. Two of the motivating concepts behind SVMs are, namely, the idea that transforming the data into a high-dimensional space makes linear discriminant functions practical and the idea of large margin classifiers for training. By mapping the input to a sufficiently large feature space, patterns become linearly separable, so a simple perceptron in feature space can do the classification. The first step in a SVM is transforming the data

into a high-dimensional space. This is done using a Radial Basis Function (RBF) network that places a gaussian at each data sample. The RBF network can be considered a kernel classifier that depicted in Fig. 3, where we can easily see that it is an RBF, but where each Gaussian is centered at each sample and the weights are the multipliers α_i.

In fact, the RBF places Gaussian kernels over the data and linearly weights their outputs to create the system output. It conforms exactly to the notion of the kernel machine. When used as an SVM, the RBF network places a Gaussian in each data sample such that the feature space becomes as large as the number of samples.

Assume we have a set of data samples

$$S = \left\{(\mathbf{x}, d_1), ..., (\mathbf{x_N}, d_N)\right\} \qquad d_i \in \{-1\ ,\ 1\} \tag{1}$$

What we want is to find the hyperplane $y=\mathbf{w.x+b}$ with the smallest norm of coefficients $\|\mathbf{w}\|^2$ (the largest margin). To find this hyperplane, we can solve the following quadratic programming problem: minimize the functional

$$\phi(\mathbf{w}) = \frac{1}{2}(\mathbf{w.w}) \tag{2}$$

under the constraint of inequality

$$d_i[(\mathbf{x_i.w}) + b] \geq 1 \qquad i = 1, 2, ..., N \tag{3}$$

where the operation is an inner product. The solution to this optimization is given by the saddle points of the Lagrangian:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}(\mathbf{w.w}) - \sum_{i=1}^{N} \alpha\{[\mathbf{x.w}) + b]d_i - 1\} \tag{4}$$

By using the dual formulation, we can rewrite Eq.4 as

$$J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j (\mathbf{x_i}.\mathbf{x_j}) \tag{5}$$
$$\text{subject to} \qquad \alpha_i \geq 0, \quad \forall\ i \in \{1, ..., N\}$$

under the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0.$ the solution is a set of α^*. We can show that only some of the samples will correspond to

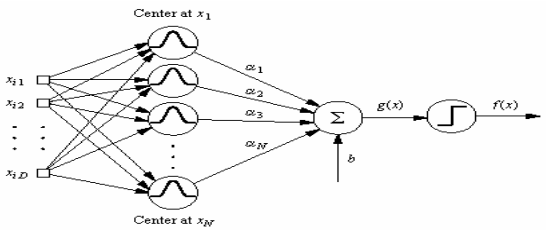

Fig. 3 Topology of the SVM machine with RBF kernels

Lagrangian multipliers different from zero and will be called the support vectors. They are the ones that control the positioning of the optimal hyperplane. The large margin classifier thus is specified by

$$f(x) = \text{sgn}\left(\sum_{\substack{\text{support} \\ \text{vectors}}} d_i \alpha_i^* (\mathbf{x_i}.\mathbf{x}) - b^*\right) \tag{6}$$

One of the characteristics of the SVM is that the user has no control over the number of support vectors (i.e. the size of the final machine). During training, all the RBFs are used, but once the SVM is trained, the RBF should be trimmed, discarding the RBFs that are not support vectors. The number of support vectors depends on the data, which makes sense but is not always useful since we never know the size of the model. The expressions we arrived at are exactly the same as the one for the Adatron algorithm, except that Vapnik suggests a quadratic programming solution, while the Adatron is an "on-line" solution, easily implemented in neural network software. Like any on-line algorithm, the Adatron requires control of learning rate and suffers from the problem of misadjustment and stopping criterion. We can expect that training SVMs with large data sets demands a lot from computer resources (memory or computation).

The learning algorithm is based on the Adatron algorithm extended to the RBF network. The Adatron algorithm can be easily extended to the RBF network by substituting the inner product of patterns in the input space by the kernel function, leading to the following quadratic optimization problem:

$$J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j G(x_i - x_j, 2\sigma^2)$$
$$\text{subject to} \qquad \sum_{i=1}^{N} d_i \alpha_i = 0 \quad \alpha_i \geq 0, \quad \forall\ i \in \{1, ..., N\} \tag{7}$$

We can then define

$$g(x_i) = d_i\left(\sum_{j=1}^{N} d_j \alpha_j G(x_i - x_j, 2\sigma^2) + b\right) \quad \text{and} \quad M = \min_i\ g(x_i) \tag{8}$$

and choose a common starting multiplier (e.g. α_i, α_j =0.1), learning rate η, and a small threshold (e.g., t = 0.01). While M>t, we choose a pattern $\mathbf{x_i}$ and calculate an update $\Delta\alpha_i = \eta[1-g(x_i)]$ and perform the update

$$\begin{cases} \alpha_i(n+1) = \alpha_i(n) + \Delta\alpha_i(n), & b(n+1) = b(n) + di\Delta\alpha_i \quad \text{if} \quad \alpha_i(n) + \Delta\alpha_i \succ 0 \\ \alpha_i(n+1) = \alpha_i(n), & b(n+1) = b(n) \quad \text{if} \quad \alpha_i(n) + \Delta\alpha_i \le 0 \end{cases} \tag{9}$$

After adaptation only some of the α_i are different from zero (called the support vectors). They correspond to the samples that are closest to the boundary between classes. This algorithm is called the kernel Adatron and can adapt an RBF to have an optimal margin. This algorithm can be considered the "on-line" version of the quadratic optimization approach utilized for SVMs, and it can find the same solutions as Vapnik's original algorithm for SVMs. Notice that it is easy to implement the kernel Adatron algorithm since $g(x_i)$ can be computed locally to each multiplier, provided that the desired response is available in the input file. In fact, the expression for $g(x_i)$ resembles the multiplication of an error with an activation, so it can be included in the framework of neural network learning. The Adatron algorithm essentially prunes the RBF network so that its output for testing is given by

$$f(x) = \operatorname{sgn}\left(\sum_{\substack{i \in \text{support} \\ \text{vectors}}} d_i \alpha_i G(x - x_i, 2\sigma^2) - b \right) \tag{10}$$

B. *Computer Simulation:*

In this paper, the SVM classifier is implemented using the kernel Adatron algorithm and it is trained for both the data sets for 1000 epochs. For Pentium 4, 1.7 GHz, 768 MB RAM Computer with optimal Step size = 0.01, training of SVM took about 27.77 (average value) epochs per second. Variation of Mean Square Error on training data set 1 with respect to epochs is as shown in Fig. 4.

It is observed that the training is not affected by different random weight initializations. Different initial conditions are

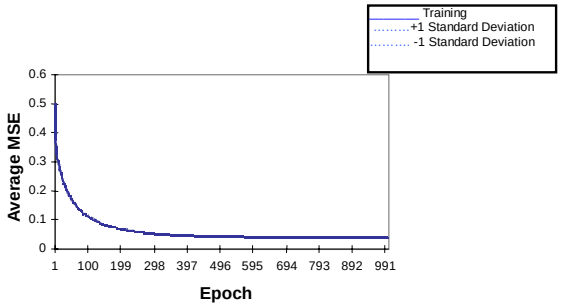

Fig. 4 Average MSE versus number of Epoch

tried to make sure that one is really converging to the absolute minimum. Therefore, the network is run at least three times with different weight-initializations with the specified training epochs to gauge performance. Even though SVM is run three times with different random weight initializations, identical results are obtained for a specific data set. This proves consistency and robustness of the model. Table V displays the various important performance measures of SVM classifier on different data sets with respect to “Rock” as well as “Metal” instances.

It is noticed that the classifier recognizes 40 “Rock” instances as “Rock” ones and two “Rock” instances as “Metal” ones entailing a classification accuracy of 95.238 % for “Rock” instances. However, it classifies 7 “Metal” instance as “Rock” and 55 “Metal” instances as “Metal” that result into a classification accuracy of 88.70967 % over “Metal” instances. In order to confirm whether the proposed model is really consistently capable of near optimum classification, different data partition as in Set 2 (reverse tagging order) is used to train the classifier. For the testing dataset, the classification accuracy for “Rock” samples is seen as 67.272727, whereas for the “Metal” samples, the classification accuracy is observed as 95.91836735. The ROC curves of the SVM classifier on different test data set are sketched in Fig. 5(a) and 5(b).

From the graph it is seen that the area under the ROC curve on different test data set 1 and set 2 are computed as 0.981183 and 0.939518 respectively, which is indicative of reasonable

TABLE V
PERFORMANCE MEASURES OF SVM CLASSIFIER ON DIFFERENT DATA SETS

S. N.	Data set	Performance measures of SVM Classifier							
		“Rock” instances				“Metal” instances			
		% Correct	MSE	NMSE	r	% Correct	MSE	NMSE	r
1	Testing Set 1	95.238	0.0828976	0.3443246	0.83928	88.70967	0.08362867	0.3473608	0.8390735
2	Training Set 1	100	0.0131840	0.0529122	0.987175	100	0.01331346	0.05343168	0.9871467
3	Testing Set 2	67.272727	0.13755835	0.552070914	0.744452149	95.91836735	0.139111408	0.558303893	0.744005351
4	Training Set 2	100	0.014012203	0.058201226	0.990985809	100	0.014190286	0.058940911	0.990941784

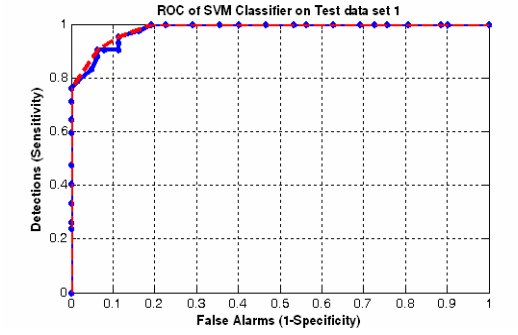

Fig. 5 (a) ROC curve for SVM based classifier on test data set. (Set 1: Normal tagging)

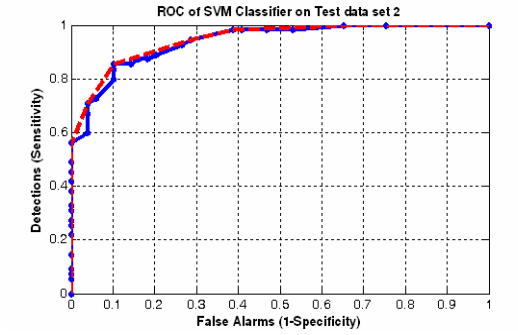

Fig. 5 (b) ROC curve for SVM based classifier on test data set. (Set 2: Reverse tagging)

classification.

Results of ROC analysis of the SVM classifier for different data partition are listed in Table VI. It is seen from table V and VI, that the SVM classifier provides an excellent performance on both the data sets. In addition, this performance is seen to be consistently good.

VI. Conclusion

As a classifier, the best single hidden layered MLP NN is not seen to perform reasonably. When it is evaluated on the training instances, it works as an almost perfect classifier. Here, the area under the ROC curve is found as 1.0 and the average classification accuracy of 100 %. However, it is revealed that this good performance on the training data set has not been maintained on the test data set. The performance of this MLP classifier on the test data degrades slightly and average classification accuracy = 90.3609 % and the area under the ROC curve = 0.953533. Therefore, it could be inferred that such a performance may indicate a subtle memorization of the neural network without adequate learning. In order to ensure learning and generalization, different data partition is employed with reverse tagging order for training of the network. Results show that the performance drastically degrades as now average classification accuracy drops to 77.5138635 % and the area under the ROC curve deviates away from 1.0 (Now it is 0.861224). The MLP

TABLE VI
Results of ROC Analysis of SVM Classifier

Data set	ROC Analysis of SVM Classifier	
	Area under an ROC curve	Area under an ROC curve
Testing Set 1	0.981183	0.983871
Testing Set 2	0.939518	0.947310

classifier is not seen to provide consistency in the classification accuracy. These findings are confirmed to satisfaction after repeating the simulation experiments a number of times on different data partitions.

Results show that SVM classifier worked as an optimal classifier for the given task. For the testing dataset, the classification accuracy for “Rock” samples is seen as 95.238, whereas for the “Metal” samples, the classification accuracy is observed as 88.70967(an average of 91.973835 % correct classifications). The area under the ROC curve for testing dataset is computed as 0.981183, which is indeed very close to unity. It is worthwhile to notice that the SVM has consistently performed elegantly as a near-optimal classifier even after repeating the simulation experiments a number of times with different initial weights. It is seen that the SVM based classifier satisfies almost all the essential qualities and tests of a near-perfect (near-optimal) classifier up to the end-user’s expectations. More importantly, its performance is seen to be consistently good. For the classification of sonar signals, the decision boundaries formed by the SVM classifier are seen to be more accurate than those formed by MLP classifier and the discriminating ability of SVM is remarkable in separating data as well as possible into classes.

The performance of MLP classifiers is seen to get affected by the training algorithm, input exemplars, and parameter initialization. However, the proposed SVM with kernel Adatron algorithm is observed to be relatively insensitive to the parameter initialization. It consistently performs well as a reasonable classifier with acceptable and reliable performance measures.

References

[1] R. P. Lippmann. “An introduction to computing with neural nets,” IEEE ASSP Magazine, pp. 4-22, April 1987.
[2] H.L. Roitblat, W.W.L. Au, P.E. Nachtigall, R. Shizumura and G. Moons, “Sonar recognition of targets embedded in sediments,” Neural Networks, Vol. 8, No. 7/8, pp. 1263-1273, 1995.
[3] J.A. Simmons, P.A. Saillant, J.M. Wotton, T. Haresign, M.J. Ferragamo and C.F. Moss, “Composition of biosonar images for target recognition by echolocating bats,” Neural Networks, Vol. 8, No. 7/8, pp. 1239-1261, 1995.
[4] W.W.L. Au, “Comparision of sonar discrimination- dolphin and artificial neural network,” The Journal of the Acoustical Society of America, Vol. 95, No. 5, Part 1, pp. 2728-2735, May 1994.
[5] E. Alpaydin, “Multiple networks for function learning,” Proceedings of IEEE International Conference on Neural Networks, San Francisco, pp. 9-14, March 1993.
[6] W.W.L. Au, L.N. Andersen, A.R. Rasmussen, H.L. Roitblat and P.E. Nachtigall, “Neural Network modeling of a dolphin’s sonar discrimination capabilities,” The Journal of the Acoustical Society of America, Vol. 98, No. 1, pp. 43-50, July 1995.

[7] R. P. Gorman and T. J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75-89, 1988.
[8] R. P. Gorman and T. J. Sejnowski, "Learned classification of sonar targets using a massively parallel network," IEEE Transactions on Acoustic, Speech and Signal Processing, Vol. 36, No. 7, pp. 1135-1140, July 1998.
[9] Vapnik, V. N., The Nature of Statistical Learning Theory. New York: Springer-Verlag 1995.
[10] Hornik K. M., Stinchcombe M., and White H. (1989). "Multilayer Feedforward Networks Are Universal Approximators," Neural Networks, vol.2 no. (5), pp. 359-66.
[11] Boser, H., Guyon, I. M., & Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. In Haussler, D., Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory 144-152. Pittsburgh, PA: ACM Press.
[12] Cristianini, N. and Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and other Kernel-based Learning Methods. Cambridge, UK: Cambridge University Press.
[13] S. Haykin, 1994. Neural Networks: A Comprehensive Foundation, McMillan, New York.
[14] T. Friess, N. Cristianini and C. Campbell, "The Kernel-Adatron: A fast and simple learning procedure for support vector machines," Proceedings of the 15th International Conference in Machine Learning, pp. 188-196, 1998.
[15] Alim, O.A.; Hashem, H.F.; "Automatic recognition of the sonar signals using neural network," Proceedings of International Conference on Information, Communications and Signal Processing, ICICS1997, 9-12 Sept. 1997, vol.2, pp.740 - 744.
[16] Freiss T., Support Vector Neural Networks: The Kernel Adatron with Bias and Soft Margin, Uniiversity of Sheffield Technical Report, 1998.
[17] Andersen, L.N.; Au, W.; Larsen, J.; Hansen, L.K.; "Sonar discrimination of cylinders from different angles using neural networks," Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '99, Volume 2, 15-19 March 1999, vol.2 pp.1121 - 1124.
[18] Jose C. Principe, Neil R. Euliano, and W. Curt Lefebvre, Neural and Adaptive Systems: Fundamentals Through Simulations, John Wiley & Sons, Inc., 2000.
[19] Ming Hsuan Yang and Narendra Ahuja, "A geometric approach to train support vector machines," Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, CVPR 2000, Hilton Head Island, Vol. 1, pp. 430-437, June 2000.
[20] Dai H.K., Jing Peng, and Heisterkamp Douglas R. "LDA/SVM driven nearest neighbour classification," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Vol. 1, pp. 158-163, 2001.
[21] Ward, M.K.; Stevenson, M.; "Sonar signal detection and classification using artificial neural networks," Electrical and Computer Engineering, 2000, Canadian Conference on 7-10 March, 2000, Volume 2, pp.717 – 721.

Suresh S. Salankar is a Assistant Professor in the Department of Electronics and Telecommunication at B. D. College of Engineering, Sevagram, India. His research interests is in the design and evaluation of learning algorithms for pattern reccognition applications. This includes, in particular, neural network classifiers, support vector machines classifiers, and classifier combing strategies. He is a member of the Institution of Engineers (India) and Indian Society for Technical Education.

Balasaheb M. Patre was born in Basmathnagar, India in 1965. He received B. E. and M. E. degree in Instrumentation and Control Engineering in 1986 and 1990 respectively from Marathwada University, Aurangabad and subsequently doctorate degree (Ph. D.) in Systems and Control Engineering from IIT, Bombay in 1998. He has published sixty papers in the National/International conferences/journals. He has presented his research work at Cambridge University, UK and in Germany. He is a life member ISTE and Instrument Society of India, member of IETE, IEEE, and IET. He is reviewer for several International Journals. His area of interest includes robust control, VSS, interval arithmetic applications in robust control, intelligent control etc. Presently he is working as Professor of Instrumentation Engineering at SGGS Institute of Engineering and Technology, Nanded, India.