A Probabilistic Active Support Vector Learning Algorithm

Pabitra Mitra, *Student Member*, *IEEE*, C.A. Murthy, and Sankar K. Pal, *Fellow*, *IEEE*

Abstract—The paper describes a probabilistic active learning strategy for support vector machine (SVM) design in large data applications. The learning strategy is motivated by the statistical query model. While most existing methods of active SVM learning query for points based on their proximity to the current separating hyperplane, the proposed method queries for a set of points according to a distribution as determined by the current separating hyperplane and a newly defined concept of an adaptive confidence factor. This enables the algorithm to have more robust and efficient learning capabilities. The confidence factor is estimated from local information using the k nearest neighbor principle. The effectiveness of the method is demonstrated on real-life data sets both in terms of generalization performance, query complexity, and training time.

Index Terms—Data mining, learning theory, query learning, incremental learning, statistical query model, classification.

1 Introduction

THE support vector machine (SVM) [1] has been successful as a high-performance classifier in several domains including pattern recognition, data mining, and bioinformatics. It has strong theoretical foundations and good generalization capability. A limitation of the SVM design algorithm, particularly for large data sets, is the need to solve a quadratic programming (QP) problem involving a dense $n \times n$ matrix, where n is the number of points in the data set. Since QP routines have high complexity, SVM design requires huge memory and computational time for large data applications. Several approaches exist for circumventing the above shortcomings. These include simpler optimization criterion for SVM design, e.g., the linear SVM and the kernel adatron, specialized QP algorithms like the cojugate gradient method, decomposition techniques which break down the large QP problem into a series of smaller QP subproblems, the sequential minimal optimization (SMO) algorithm and its various extensions, Nystrom approximations [2], and greedy Bayesian methods [3]. Many of these approaches are discussed in [4]. A simple method to solve the SVM QP problem has been described by Vapnik, which is known as "chunking" [5]. The chunking algorithm uses the fact that the solution of the SVM problem remains the same if one removes the points that correspond to zero Lagrange multipliers of the QP problem (the non-SV points). The large QP problem can thus be broken down into a series of smaller QP problems whose ultimate goal is to identify all of the nonzero Lagrange multipliers (SVs) while discarding the zero Lagrange multipliers (non-SVs). At every step, chunking solves a QP problem that consists of the nonzero Lagrange multiplier points from the previous step, and a chunk of p other points. At the final step, the entire set of nonzero Lagrange multipliers has been identified, thereby solving the large QP problem. Several variations of the chunking algorithm exist depending upon the method of forming the chunks [6]. Chunking greatly reduces the training time compared to batch learning of SVMs. However, it may not handle large-scale training problems due to slow convergence of the chunking steps when p new points are chosen randomly.

• *The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 108, India.*
E-mail: {pabitra_r, murthy, sankar}@isical.ac.in.

Manuscript received 15 Apr. 2002; revised 15 Jan. 2003; accepted 14 July 2003.
Recommended for acceptance by C. Schmid.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 116349.

Recently, active learning has become a popular paradigm for reducing the sample complexity of large-scale learning tasks [7], [8]. It is also useful in situations where unlabeled data is plentiful but labeling is expensive. In active learning, instead of learning from "random samples," the learner has the ability to select its own training data. This is done iteratively and the output of a step is used to select the examples for the next step. In the context of support vector machine, active learning can be used to speed up chunking algorithms. In [9], a query learning strategy for large margin classifiers is presented which iteratively requests the label of the data point closest to the current separating hyperplane. This accelerates the learning drastically compared to random sampling. An active learning strategy based on version space splitting is presented in [10]. The algorithm attempts to select the points that split the current version space into two halves having equal volumes at each step, as they are likely to be the actual support vectors. Three heuristics for approximating the above criterion are described, the simplest among them selects the point closest to the current hyperplane as in [9]. A greedy optimal strategy for active SV learning is described in [11]. Here, logistic regression is used to compute the class probabilities, which is further used to estimate the expected error after adding an example. The example that minimizes this error is selected as a candidate SV. Note that the method was developed only for querying single point, but the result reported in [11] used batches of different sizes in addition to single point.

Although most of these active learning strategies query only for a single point at each step, several studies have noted that the gain in computational time can be obtained by querying multiple instances at a time. This motivates the formulation of active learning strategies which query for multiple points. Error driven methods for incremental support vector learning with multiple points are described in [12]. In [12], a chunk of p new points having a fixed ratio of correctly classified and misclassified points are used to update the current SV set. However, no guideline is provided for choosing the above ratio. Another major limitation of all the above strategies is that they are essentially greedy methods where the selection of a new point is influenced only by the current hypothesis (separating hyperplane) available. The greedy margin-based methods are weak because focusing purely on the boundary points produces a kind of nonrobustness, with the algorithm never asking itself whether a large number of examples far from the current boundary do, in fact, have the correct implied labels. In the above setup, learning may be severely hampered in two situations: A "bad" example is queried which drastically worsens the current hypothesis and the current hypothesis itself is far from the optimal hypothesis (e.g., in the initial phase of learning). As a result, the examples queried are less likely to be the actual support vectors.

The present paper describes an active support vector learning algorithm which is a probabilistic generalization of purely margin based methods. The methodology is motivated by the model of *learning from statistical queries* [13] which captures the natural notion of learning algorithms that construct a hypothesis based on statistical properties of large samples rather than the idiosyncrasies of a particular example. A similar probabilistic active learning strategy is presented in [14]. The present algorithm involves estimating the likelihood that a new example belongs to the actual support vector set and selecting a set of p new points according to the above likelihood, which are then used along with the current SVs to obtain the new SVs. The likelihood of an example being an SV is estimated using a combination of two factors: The margin of the particular example with respect to the current hyperplane and the degree of confidence that the current set of SVs provides the actual SVs. The degree of confidence is quantified by a measure that is based on the local properties of each of the current support vectors and is computed using the nearest neighbor estimates.

The aforesaid strategy for active support vector learning has several advantages. It allows for querying multiple instances and, hence, is computationally more efficient than those that are querying for a single example at a time. It not only queries for the

error points or points close to the separating hyperplane but also a number of other points which are far from the separating hyperplane and also correctly classified ones. Thus, even if a current hypothesis is erroneous, there is scope for it being corrected owing to the later points. If only error points were selected, the hypothesis might actually become worse. The ratio of selected points lying close to the separating hyperplane (and misclassified points) to those far from the hyperplane is decided by the confidence factor, which varies adaptively with iteration. If the current SV set is close to the optimal one, the algorithm focuses only on the low margin points and ignores the redundant points that lie far from the hyperplane. On the other hand, if the confidence factor is low (say, in the initial learning phase), it explores a higher number of interior points. Thus, the trade off between efficiency and robustness of performance is adequately handled in this framework. This results in a reduction in the total number of labeled points queried by the algorithm in addition to speed up in training, thereby making the algorithm suitable for applications where labeled data is scarce.

Experiments are performed on four real-life classification problems. The size of the data ranges from 684 to 495, 141, dimension from 9 to 294. Our algorithm is found to provide superior performance and faster convergence compared to several related algorithms for incremental and active SV learning.

2 Support Vector Machine

Support vector machines are a general class of learning architecture inspired from statistical learning theory that performs *structural risk minimization* on a nested set structure of separating hyperplanes [1]. Given training data, the SVM training algorithm obtains the optimal separating hyperplane in terms of generalization error. Though SVMs may also be used for regression and multiclass classification, in this article, we concentrate only on two-class classification problem.

Algorithm: Suppose we are given a set of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \mathbf{x} \in R^N, y_i \in \{-1, +1\}$. We consider decision functions of the form $sgn((\mathbf{w} \cdot \mathbf{x}) + b)$, where $(\mathbf{w} \cdot \mathbf{x})$ represents the inner product of $\mathbf{w}$ and $\mathbf{x}$. We would like to find a decision function $f_{\mathbf{w},b}$ with the properties

$$y_i((\mathbf{w} \cdot x_i) + b) \geq 1, \quad i = 1, \ldots, l. \tag{1}$$

In many practical situations, a separating hyperplane does not exist. To allow for possibilities of violating (1), slack variables are introduced like

$$\xi_i \geq 0, \quad i = 1, \ldots, l \tag{2}$$

to get

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{3}$$

The support vector approach for minimizing the generalization error consists of the following:

$$\textbf{Minimize :} \qquad \Phi(\mathbf{w}, \xi) = (\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^{l} \xi_i, \tag{4}$$

subject to constraints (2) and (3).

It can be shown that minimizing the first term in (4) amounts to minimizing a bound on the VC-dimension and minimizing the second term corresponds to minimizing the misclassification error [1]. The above minimization problem can be posed as a constrained quadratic programming (QP) problem.

The solution gives rise to a decision function of the form:

$$f(\mathbf{x}) = sgn\left[\sum_{i=1}^{l} y_i \alpha_i (\mathbf{x} \cdot \mathbf{x_i}) + b\right].$$

Only a small fraction of the α_i coefficients are nonzero. The corresponding pairs of $\mathbf{x_i}$ entries are known as *support vectors* and they fully define the decision function.

3 Probabilistic Active Support Vector Learning Algorithm

In the context of support vector machines, the target of the learning algorithm is to learn the set of support vectors. This is done by incrementally training an SVM on a set of examples consisting of the previous SVs and a new set of points. In the proposed algorithm, the new set of points, instead of being randomly generated, is generated according to a probability $\mathbf{Pr}_{\chi(\mathbf{x},f(\mathbf{x}))}$. $\chi(\mathbf{x}, f(\mathbf{x}))$ denotes the event that example $\mathbf{x}$ is an SV. $f(\mathbf{x})$ is the optimal separating hyperplane. The methodology is motivated by the statistical query model of learning [13], where the oracle, instead of providing actual class labels, provides an (approximate) answer to the statistical query "what is the probability that an example belongs to a particular class?"

We define the probability $\mathbf{Pr}_{\chi(\mathbf{x},f(\mathbf{x}))}$ as follows: Let $< \mathbf{w}, b >$ be the current separating hyperplane available to the learner.

$$\begin{aligned} P_{\chi(\mathbf{x},f(\mathbf{x}))} &= c \quad \text{if } y(\mathbf{w} \cdot \mathbf{x} + b) \leq 1 \\ &= 1 - c \quad \text{otherwise.} \end{aligned} \tag{5}$$

Here, c is a *confidence parameter* which denotes how close the current hyperplane $< \mathbf{w}, b >$ is to the optimal one. y is the label of $\mathbf{x}$.

The significance of $P_{\chi(\mathbf{x},f(\mathbf{x}))}$ is as follows: If c is high, which signifies that the current hyperplane is close to the optimal one, points having margin value less than unity are highly likely to be the actual SVs. Hence, the probability $P_{\chi(\mathbf{x},f(\mathbf{x}))}$ returned to the corresponding query is set to a high value c. When the value c is low, the probability of selecting a point lying within the margin decreases and a high probability value $(1 - c)$ is then assigned to a point having high margin. Let us now describe a method for estimating the confidence factor c.

3.1 Estimating the Confidence Factor for an SV Set

Let the current set of support vectors be denoted by $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_l\}$. Also, consider a test set $T = \{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_m\}$ and an integer k (say, $k = \sqrt{l}$). For every $\mathbf{s}_i \in S$, compute the set of k nearest points in T. Among the k nearest neighbors, let k_i^+ and k_i^- number of points have labels $+1$ and -1, respectively. The confidence factor c is then defined as

$$c = \frac{2}{lk} \sum_{i=1}^{l} \min(k_i^+, k_i^-). \tag{6}$$

Note that the maximum value of the confidence factor c is unity when $k_i^+ = k_i^- \ \forall i = 1, \ldots, l$, and the minimum value is zero when $\min(k_i^+, k_i^-) = 0 \ \forall i = 1, \ldots, l$. The first case implies that all the support vectors lie near the class boundaries and the set $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_l\}$ is close to the actual support vector set. The second case, on the other hand, denotes that set S consists only of interior points and is far from the actual support vector set. Thus, the confidence factor c measures the degree of closeness of S to the actual support vector set. The higher the value of c is, the closer the current SV set is to the actual SV set.

3.2 Algorithm

The active support vector learning algorithm, which uses the probability $\mathbf{Pr}_{\chi(\mathbf{x},f(\mathbf{x}))}$, estimated above, is presented below.

Let $A = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ denote the entire training set used for SVM design. $SV(B)$ denotes the set of support vectors of the set B obtained using the methodology described in Section 2. $S_t = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_l\}$ is the support vector set obtained after tth iteration, and $< \mathbf{w}_t, b_t >$ is the corresponding separating hyperplane. $Q_t = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p\}$ is the set of p points actively queried for at step t.

TABLE 1
Comparison of Performance of SVM Design Algorithms

Data	Algorithm	a_{test}(%) Mean	SD	$\mathcal{D}$	n_{query}	t_{cpu} (sec)
Cancer	BatchSVM	96.32	0.22	-	-	1291
	IncrSVM	86.10	0.72	10.92	0.83	302
	QuerySVM	96.21	0.27	9.91	0.52	262
	StatQSVM	96.43	0.25	7.82	0.41	171
	SMO	96.41	0.23	-	-	91
Twonorm	BatchSVM	97.46	0.72	-	-	8.01×10^4
	IncrSVM	92.01	1.10	12.70	0.24	770
	QuerySVM	93.04	1.15	12.75	0.07	410
	StatQSVM	96.01	1.52	12.01	0.02	390
	SMO	97.02	0.81	-	-	82
Covertype	IncrSVM	57.90	0.74	-	0.04	4.70×10^4
	QuerySVM	65.77	0.72	-	0.008	3.20×10^4
	StatQSVM	74.83	0.77	-	0.004	2.01×10^4
	SMO	74.22	0.41	-	-	0.82×10^4
Microsoft Web	IncrSVM	52.10	0.22	-	0.10	2.54×10^4
	QuerySVM	52.77	0.78	-	0.04	1.97×10^4
	StatQSVM	63.83	0.41	-	0.01	0.02×10^4
	SMO	65.43	0.17	-	-	0.22×10^4

c is the confidence factor obtained using (6). The learning steps involved are given below:

Initialize: Randomly select an initial starting set Q_0 of p instances from the training set A. Set $t = 0$ and $S_0 = SV(Q_0)$. Let the parameters of the corresponding hyperplane be $< \mathbf{w}_0, b_0 >$.
While *Stopping Criterion* is not satisfied:
  $Q_t = \emptyset$.
  While $\text{Cardinality}(Q_t) \leq p$:
    Randomly select an instance $\mathbf{x} \in A$.
    Let y be the label of $\mathbf{x}$.
    If $y(\mathbf{w}_t \cdot \mathbf{x} + b) \leq 1$:
      Select $\mathbf{x}$ with probability c. Set $Q_t = Q_t \cup \mathbf{x}$.
    Else:
      Select $\mathbf{x}$ with probability $1 - c$. Set $Q_t = Q_t \cup \mathbf{x}$.
    End If
  End While
  $S_t = SV(S_t \cup Q_t)$.
  $t = t + 1$.
End While

The set S_T, where T is the iteration at which the algorithm terminates, contains the final SV set.

Stopping Criterion: Among the p points actively queried at each step t, let p_{nm} points have margin greater than unity ($y(\mathbf{w}_t \cdot \mathbf{x} + b) > 1$). Learning is stopped if the quantity $\frac{c \cdot p_{nm}}{p}$ exceeds a threshold Th (say, $= 0.9$).

The stopping criterion may be interpreted as follows: A high value of the quantity $\frac{p_{nm}}{p}$ implies that the query set contains a small number of points with margin less than unity. No further gain can be thus achieved by the learning process. The value of p_{nm} may also be large when the value of c is low in the initial phase of learning. However, if both c and p_{nm} have high values, the current SV set is close to the actual one (i.e., a good classifier is obtained) and also the margin band is empty (i.e., the learning process is saturated); hence, the learning may be terminated.

4 Experimental Results and Comparison

Organization of the experimental results is as follows: First, the characteristics of the four data sets used are discussed briefly. Next, the performance of the proposed algorithm in terms of generalization capability, training time, and some related quantities, is compared with two other incremental support vector learning algorithms as well as the batch SVM. Linear SVMs are used in all the cases. The effectiveness of the confidence factor c, used for active querying, is then studied.

4.1 Data Sets

Four public domain data sets are used, two of which are large and two relatively smaller. All the data sets have two overlapping classes. Their characteristics are described below. The data sets are available in the UCI machine learning and KDD repositories [15].

Wisconsin Cancer: The popular Wisconsin breast cancer data set contains 9 features, 684 instances, and 2 classes.

Twonorm: This is an artificial data set, having dimension 20, 2 classes, and 20,000 points. Each class is drawn from a multivariate normal distribution with unit covariance matrix. Class 1 has mean $(a, a, \ldots, a)$ and class 2 has mean $(-a, -a, \ldots, -a)$. $a = \frac{2}{\sqrt{20}}$.

Forest Cover Type: This is a GIS data set representing the forest cover type of a region. There are 54 attributes, out of which we select 10 numeric valued attributes. The original data contains

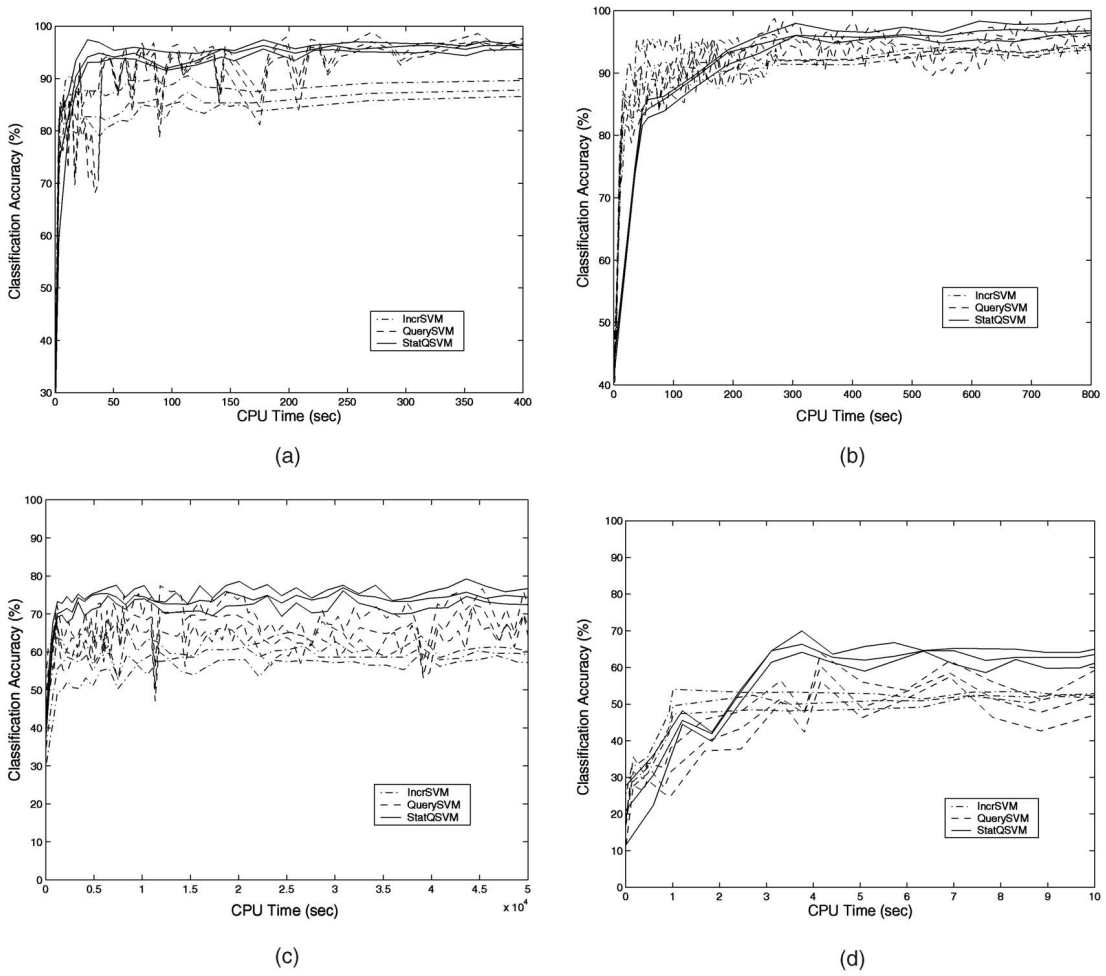

Fig. 1. Variation of a_{test} (maximum, minimum, and average over 10 runs) with CPU time for (a) Cancer, (b) Twonorm, (c) Forest cover type, and (d) Microsoft Web data.

581,012 instances and eight classes, out of which only 495,141 points, belonging to classes 1 and 2, are considered here.

Microsoft Web Data: There are 36,818 examples with 294 binary attributes. The task is to predict whether a user visits a particular site.

4.2 Classification Accuracy and Training Time

The algorithm for active SV learning with statistical queries (StatQSVM) is compared with two other techniques for incremental SV learning as well as the actual batch SVM algorithm. Only for the Forest Cover type data set, batch SVM could not be obtained due to its large size. The sequential minimal optimization (SMO) algorithm [16] is also compared for all the data sets. The following incremental algorithms are considered: 1) incremental SV learning with random chunk selection [17]. (Denoted by IncrSVM in Table 1.) 2) SV learning by querying the point closest to the current separating hyperplane [9]. (Denoted by QuerySVM in Table 1.) This is also the "simple margin" strategy in [10]. Comparison is made on the basis of the following quantities: Results are presented in Table 1.

1. Classification accuracy on test set (a_{test}). The test set has size 10 percent of that of the entire data set, and contains points that do not belong to the (90 percent) training set. Means and standard deviations (SDs) over 10 independent runs are reported.
2. Closeness of the SV set. We measure closeness of the SV set ($\tilde{S}$), obtained by an algorithm, with the corresponding actual one (S). These are measured by the distance $\mathcal{D}$ defined as follows [18]:

$$\mathcal{D} = \frac{1}{n_{\tilde{S}}} \sum_{x \in \tilde{S}} \delta(x, S) + \frac{1}{n_S} \sum_{y \in S} \delta(y, \tilde{S}) + Dist(\tilde{S}, S), \quad (7)$$

where

$$\delta(x, S) = \min_{y \in S} d(x, y), \quad \delta(y, \tilde{S}) = \min_{x \in \tilde{S}} d(x, y),$$

and $Dist(\tilde{S}, S) = \max\{\max_{x \in \tilde{S}} \delta(x, S), \max_{y \in S} \delta(y, \tilde{S})\}$. $n_{\tilde{S}}$ and n_S are the number of points in $\tilde{S}$ and S, respectively. $d(x, y)$ is the usual Euclidean distance between points x and y. The distance measure has been used for quantifying the errors of set approximation algorithms [18], and is related to the ϵ cover of a set.

3. Fraction of training samples queried (n_{query}) by the algorithms.
4. CPU time (t_{cpu}) on a Sun UltraSparc 350MHz workstation.

It is observed from the results shown in Table 1 that all three incremental learning algorithms require several orders less training time as compared to batch SVM design, while providing comparable classification accuracies. Among them, the proposed one achieves the highest or second highest classification score in the least time and number of queries for all the data sets. The superiority becomes more apparent for the Forest Cover type data set, where it significantly outperforms both QuerySVM and

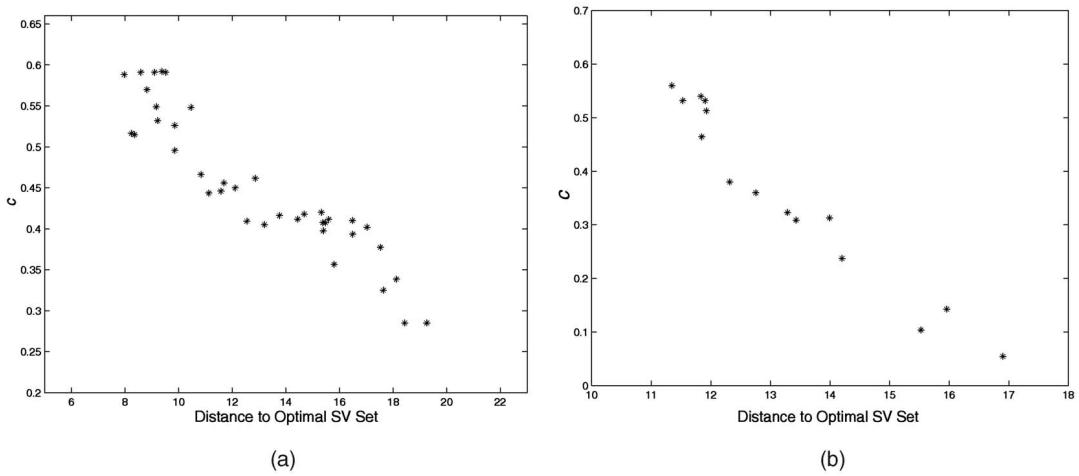

Fig. 2. Variation of confidence factor c and distance $\mathcal{D}$ for (a) Cancer and (b) Twonorm data.

IncrSVM. The QuerySVM algorithm performs better than IncrSVM for Cancer, Twonorm, and the Forest Covertype data sets.

It can be seen from the values of n_{query} in Table 1 that the total number labeled points queried by StatQSVM is the least among all the methods including QuerySVM. This is in spite of the fact that StatQSVM needs the label of the randomly chosen points even if they wind up not being used for training, as opposed to QuerySVM, which just takes the point closest to the hyperplane (and so does not require knowing its label until one decides to actually train on it). The overall reduction in n_{query} for StatQSVM is probably achieved by its efficient handling of the exploration-exploitation trade off in active learning.

The SMO algorithm requires substantially less time compared to the incremental ones. However, SMO is not suitable to applications where labeled data is scarce. Also, SMO may be used along with the incremental algorithms for further reduction in design time.

The nature of convergence of the classification accuracy on test set a_{test} is shown in Fig. 1 for all the data sets. It is observed that the convergence curve for the proposed algorithm dominates those of QuerySVM and IncrSVM. Since the IncrSVM algorithm selects the chunks randomly, the corresponding curve is smooth and almost monotonic, although its convergence rate is much slower compared to the other two algorithms. On the other hand, the QuerySVM algorithm selects only the point closest to the current separating hyperplane and achieves a high classification accuracy in a few iterations. However, its convergence curve is oscillatory and the classification accuracy falls significantly after certain iterations. This is expected as querying for points close to the current separating hyperplane may often result in gain in performance if the current hyperplane is close to the optimal one. While querying for interior points reduces the risk of performance degradation, it also achieves poor convergence rate. Our strategy for active support vector learning with statistical queries selects a combination of low margin and interior points and, hence, maintains a fast convergence rate without oscillatory performance degradation.

In a part of the experiment, the margin distribution of the samples was studied as a measure of generalization performance of the SVM. The distribution in which a larger number of examples have high positive margin values leads to a better generalization performance. It was observed that, although the proposed active learning algorithm terminated before all the actual SVs were identified, the SVM obtained by it produced a better margin distribution than the batch SVM designed using the entire data set. This strengthens the observation of [11] and [9] that active learning, along with early stopping, improves the generalization performance.

4.3 Effectiveness of the Confidence Factor c

Fig. 2 shows the variation of the confidence factor c for the SV sets with distance $\mathcal{D}$. It is observed that, for all the data sets, c is linearly correlated with $\mathcal{D}$. As the current SV set converges closer to the optimal one, the value of $\mathcal{D}$ decreases and the value of confidence factor c increases. Hence, c is an effective measure of the closeness of the SV set with the actual one.

5 Conclusions and Discussion

A method for probabilistic active SVM learning is presented. Existing algorithms for incremental SV learning either query for points close to the current separating hyperplane or select random chunks consisting mostly of interior points. Both these strategies represent extreme cases; the former one is fast but unstable, while the later one is robust but slowly converging. The former strategy is useful in the final phase of learning, while the later one is more suitable in the initial phase. The proposed active learning algorithm uses an adaptive confidence factor to handle the above trade off. It is more robust than purely margin based methods and potentially faster than random chunk selection because it can, to some extent, avoid calculating margins for nonsupport vector examples. The superiority of our algorithm is experimentally demonstrated for some real life data sets in terms of both training time and number of queries. The strength of the proposed StatQSVM algorithm lies in the reduction of the number of labeled points queried, rather than just speed up in training. This makes it suitable for environments where labeled data is scarce.

The selection probability (P_χ, (5)) used for active learning is a two level function of the margin ($y(\mathbf{w}\cdot\mathbf{x}+b)$) of a point $\mathbf{x}$. Continuous functions of the margin of $\mathbf{x}$ may also be used. Also, the confidence factor c may be estimated using a kernel-based relative class likelihood for more general kernel structures. Logistic framework and probabilistic methods [14] may also be employed for estimating the confidence factor.

Acknowledgments

Financial support from the Council of Scientific and Industrial Research (CSIR), New Delhi, through Grant No. 22/346/02-EMR II, is gratefully acknowledged.

References

[1] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[2] C.K.I. Williams and M. Seeger, "Using the Nystrom Method to Speed Up Kernel Machines," *Proc. Advances in Neural Information Processing System*, vol. 14, 2001.

[3] M.E. Tipping and A. Faul, "Fast Marginal Likelihood Maximization for Sparse Bayesian Models," *Proc. Int'l Workshop AI and Statistics,* 2003.
[4] B. Scholkopf, *Advances in Kernel Methods—Support Vector Learning,* C.J.C. Burges and A.J. Smola, eds. MIT Press, 1998.
[5] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* vol. 2, no. 2, pp. 1-47, 1998.
[6] L. Kaufman, "Solving the Quadratic Programming Problem Arising in Support Vector Classification," *Advances in Kernel Methods—Support Vector Learning,* B. Scholkopf, C.J.C. Burges, and A. J. Smola, eds. pp. 147-168, MIT Press, 1998.
[7] D. Cohn, Z. Ghahramani, and M. Jordan, "Active Learning with Statistical Models," *J. AI Research,* vol. 4, pp. 129-145, 1996.
[8] D. MacKay, "Information Based Objective Function for Active Data Selection," *Neural Computation,* vol. 4, no. 4, pp. 590-604 1992.
[9] C. Campbell, N. Cristianini, and A. Smola, "Query Learning with Large Margin Classifiers," *Proc. 17th Int'l Conf. Machine Learning,* pp. 111-118, 2000.
[10] S. Tong and D. Koller, "Support Vector Machine Active Learning with Application to Text Classification," *J. Machine Learning Research,* vol. 2, pp. 45-66, 2001.
[11] G. Schohn and D. Cohn, "Less Is More: Active Learning with Support Vector Machines," *Proc. 17th Int'l Conf. Machine Learning,* pp. 839-846, 2000.
[12] P. Mitra, C.A. Murthy, and S.K. Pal, "Data Condensation in Large Data Bases by Incremental Learning with Support Vector Machines," *Proc. 15th Int'l Conf. Pattern Recognition,* pp. 712-715, 2000.
[13] M.J. Kearns, "Efficient Noise-Tolerant Learning from Statistical Queries," *Proc. 25th ACM Symp. Theory of Computing,* pp. 392-401, 1993.
[14] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian Process Regression: Active Data Selection and Test Point Rejection," *Proc. Int'l Joint Conf. Neural Networks,* vol. 3, pp. 241-246, 2000.
[15] C. Blake and C. Merz, *UCI Repository of Machine Learning Databases,* http://www.ics.uci.edu/mlearn/MLRepository.html, Dept. of Information and Computer Sciences, Univ. of California, Irvine, 1998.
[16] J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimisation," *Advances in Kernel Methods—Support Vector Learning,* B. Scholkopf, C.J.C. Burges, and A.J. Smola, eds. pp. 185-208, MIT Press, 1998.
[17] N.A. Sayeed, H. Liu, and K.K. Sung, "A Study of Support Vectors on Model Independent Example Selection," *Proc. First Int'l Conf. Knowledge Discovery and Data Mining,* pp. 272-276, 1999.
[18] D.P. Mandal, C.A. Murthy, and S.K. Pal, "Determining the Shape of a Pattern Class from Sampled Points in R^2," *Int'l J. General Systems,* vol. 20, no. 4, pp. 307-339, 1992.