Multiclass Core Vector Machine

S. Asharaf ASHARAF@CSA.IISC.ERNET.IN
M. Narasimha Murty MNM@CSA.IISC.ERNET.IN
S. K. Shevade SHIRISH@CSA.IISC.ERNET.IN
Computer Science and Automation, Indian Institute of Science, Bangalore, India - 560012

Abstract

Even though several techniques have been proposed in the literature for achieving multiclass classification using Support Vector Machine(SVM), the scalability aspect of these approaches to handle large data sets still needs much of exploration. Core Vector Machine(CVM) is a technique for scaling up a two class SVM to handle large data sets. In this paper we propose a Multiclass Core Vector Machine(MCVM). Here we formulate the multiclass SVM problem as a Quadratic Programming(QP) problem defining an SVM with vector valued output. This QP problem is then solved using the CVM technique to achieve scalability to handle large data sets. Experiments done with several large synthetic and real world data sets show that the proposed MCVM technique gives good generalization performance as that of SVM at a much lesser computational expense. Further, it is observed that MCVM scales well with the size of the data set.

1. Introduction

Ever since Vapnik's influential work in Statistical Learning theory(Vapnik, 1998), kernel based methods have gained profound interest amidst the researchers in machine learning and exploratory data analysis. In these methods, kernel functions are used to compute the inner product of data vectors in an implicitly defined kernel induced feature space. By choosing a suitable kernel function any machine learning algorithm that requires only the inner product between data vectors can be transformed into a kernel based method and this technique is known as kernel trick. One of the most celebrated kernel based methods is the two class Support Vector Machine(SVM) for classification problem. SVMs are hyperplane classifiers defined in a kernel induced feature space. They achieve optimal separation of patterns by margin maximization. The popularity of SVM is mainly attributed to its firm theoretical foundation, good generalization performance in several domains, the geometric interpretability and ease of application in different fields.

Several attempts have been made to equip the SVM to handle multiclass classification problems. There are mainly two types of approaches for multiclass SVM. The first approach is to construct and combine several binary SVMs to obtain a multiclass classifier. Some of the techniques that fall in this category are one-against-all, one-against-one and DAGSVM methods(Hsu & Lin, 2002). This approach is computationally expensive due to the large number of binary SVMs that are to be trained for handling each binary sub problem. The second approach is to formulate and solve an optimization problem handling the multiclass scenario(Hsu & Lin, 2002). This optimization problem directly takes care of all the data points and all the available classes. Some representatives from this category are a) the multiclass SVM by Szedmak and Shawe-Taylor (2005), b) multiclass SVM by Weston and Watkins (1999) and multi-prototype multiclass SVM(Aiolli & Sperduti, 2005). The optimization problems involved in these methods are fairly complicated and hence are computationally expensive. Even though the decomposition techniques like the one proposed by Hsu and Lin (2002) and Soman et al. (2007) help to reduce the complexity of the optimization problem, they are still expensive for use in applications involving large data sets.

Several real world applications typically deal with massive collection of data and hence the main issue in using SVM here is that of scalability. The classification problem using SVMs is usually formulated as a Quadratic Programing(QP) problem. The exist-

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

ing solution strategies for this problem have an associated time and space complexity that is (at least) quadratic in the number of data points. This makes the SVM very expensive to use even on data sets having a few thousands of elements. Core Vector Machine(CVM)(Tsang et al., 2005) is a suitable technique for scaling up a two class SVM to handle large data sets. In CVM, the quadratic optimization problem involved in SVM is formulated as an equivalent Minimum Enclosing Ball(MEB) problem. The MEB problem is then solved by using a faster approximation algorithm introduced by Bădoiu and Clarkson (2002).

In this paper we propose the Multiclass Core Vector Machine(MCVM). Here we formulate the multiclass classification problem as a quadratic programming problem. When the kernel function satisfies certain property, it is shown to be equivalent to an MEB learning problem. This problem is then solved using the MEB approximation algorithm used in CVM.

The rest of the paper is organized as follows. Section 2 deals with Core Vector Machines. In Section 3, the proposed Multiclass SVM formulation is discussed. The MCVM is introduced in Section 4. The experimental results are given in Section 5 and Section 6 deals with conclusions.

2. Core Vector Machines

Core Vector Machine applies kernel methods to data intensive applications involving large data sets. In CVM, the quadratic optimization problem involved in SVM is formulated as an equivalent MEB problem. It is then solved using a fast approximate MEB finding algorithm employing the concept of core sets(Tsang et al., 2005).

Given a set of data points $S = \{x_i\}_{i=1}^m$ where $x_i \in R^d$ for some integer $d > 0$, the minimum enclosing ball of S (denoted as $MEB(S)$) is the smallest ball that contains all the points in S. Let k be a kernel function with the associated feature map ϕ. Then $k(x_i, x_j) = \ < \phi(x_i), \phi(x_j) >$. Here $< \cdot, \cdot >$ denotes the inner product. Now the primal problem for the MEB in the kernel induced feature space to find the MEB $B(a, R)$ with center a and radius R can be stated as

$$\min_{R,a} \quad R^2$$

$$\text{s.t.} \quad \|\phi(x_i) - a\|^2 \leq R^2 \qquad \forall i \tag{1}$$

The corresponding Wolfe Dual(Fletcher, 2000) is

$$\min_{\alpha} \quad \sum_{i,j=1}^{m} \alpha_i \alpha_j k(x_i, x_j) - \sum_{i=1}^{m} \alpha_i k(x_i, x_i)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i = 1 \qquad \alpha_i \geq 0 \quad \forall i \tag{2}$$

where α_is are the Lagrange multipliers.

Now consider a situation where

$$k(x, x) = \kappa, \text{ a constant}$$

This is true for kernels like Gaussian given by $k(x_i, x_j) = e^{-q\|x_i - x_j\|^2}$. Here $\| \cdot \|$ represents the L_2 norm and q is a user given parameter.

The dot product kernel like polynomial kernel given by $k(x_i, x_j) = (< x_i, x_j > +1)^\lambda$ with normalized inputs x_i and x_j also satisfies the above condition. Here λ is a non-negative integer.

The Wolfe Dual of the MEB problem can now be written as

$$\min_{\alpha} \quad \sum_{i,j=1}^{m} \alpha_i \alpha_j k(x_i, x_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i = 1 \qquad \alpha_i \geq 0 \quad \forall i \tag{3}$$

When the kernel function satisfies $k(x, x) = \kappa$, any QP of the above form can be regarded as an MEB problem.

2.1. A Two Class SVM Problem as an MEB Problem

Given a training data set $S = \{(x_i, y_i)\}_{i=1}^m$ where $x_i \in R^d$ and $y_i \in \{+1, -1\}$, the primal for the two class SVM problem can be written(Tsang et al., 2005) as

$$\min_{w,b,\rho,\xi_i} \quad \|w\|^2 + b^2 - 2\rho + C \sum_{i,j=1}^{m} \xi_i^2$$

$$\text{s.t.} \quad y_i(w'\phi(x_i) + b) \geq \rho - \xi_i \quad \forall i \tag{4}$$

The Wolfe Dual is

$$\min_{\alpha} \quad \sum_{i,j=1}^{m} \alpha_i \alpha_j \left(y_i y_j k(x_i, x_j) + y_i y_j + \frac{\delta_{ij}}{C} \right)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i = 1 \qquad \alpha_i \geq 0 \quad \forall i \tag{5}$$

Here δ_{ij} is the Kronecker delta function.

To simplify the notation let us denote the pair (x_i, y_i) as z_i. Now the training data set can be denoted as $S = \{z_i\}_{i=1}^m$. The above equation can now be rewritten as

$$\min_{\alpha} \quad \sum_{i,j=1}^{m} \alpha_i \alpha_j \tilde{k}(z_i, z_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i = 1 \quad \alpha_i \geq 0 \quad \forall i \tag{6}$$

where

$$\tilde{k}(z_i, z_j) = y_i y_j k(x_i, x_j) + y_i y_j + \frac{\delta_{ij}}{C}.$$

When $k(x, x) = \kappa$ is satisfied this transformed kernel function $\tilde{k}$ satisfies the condition

$$\tilde{k}(z, z) = \kappa_1, \text{ some constant.}$$

Hence the above mentioned problem is an MEB problem. We now describe the algorithm to find approximate MEB.

2.2. Approximate MEB Finding Algorithm

The traditional algorithms for finding exact MEB are not efficient for $d > 30$(Tsang et al., 2005) and hence the CVM method adopts a faster approximation algorithm introduced by Bădoiu and Clarkson (2002). It returns a solution within a multiplicative factor of $(1+\epsilon)$ to the optimal value, where ϵ is a small positive number.

The $(1 + \epsilon)$ approximation of the MEB problem is obtained by solving the problem on a subset of the data set called *Core Set*. Let $B_S(a, R)$ be the exact MEB with center a and radius R for the data set S and $B_Q(\tilde{a}, \tilde{R})$ be the MEB with center $\tilde{a}$ and radius $\tilde{R}$ found by solving the MEB problem on a subset of S called Core Set(Q). Given an $\epsilon > 0$, a ball $B_Q(\tilde{a}, (1 + \epsilon)\tilde{R})$ is an $(1 + \epsilon)$-approximation of $MEB(S) = B_S(a, R)$ if $S \subset B_Q(\tilde{a}, (1 + \epsilon)\tilde{R})$ and $\tilde{R} \leq R$.

Formally, a subset $Q \subseteq S$ is a core set of S if an expansion by a factor $(1+\epsilon)$ of its MEB contains S(i.e. $S \subset B_Q(\tilde{a}, (1 + \epsilon)\tilde{R})$) as shown in Figure 1.

The approximate MEB finding algorithm uses a simple iterative scheme: At the t^{th} iteration, the current estimate $B_Q(\tilde{a}_t, \tilde{R}_t)$ is expanded incrementally by including that data point in S that is farthest from the center $\tilde{a}_t$ and falls outside the $(1 + \epsilon)$-ball $B_Q(\tilde{a}_t, (1 + \epsilon)\tilde{R}_t)$. The computation to find the farthest point becomes very expensive when the number of data points in S is very large. Hence to speed up the process CVM uses a probabilistic method. Here a random sample S' having 59 points is taken from the points in S (Smola & Schölkopf, 2000). Then the point in S' that is farthest from the center $\tilde{a}_t$ is taken as the approximate farthest point from S. The iterative strategy to include the farthest point in the MEB is repeated until all the points in S are covered by $B_Q(\tilde{a}_t, (1 + \epsilon)\tilde{R}_t)$. The set of all such points that got added forms the core set of the data set.

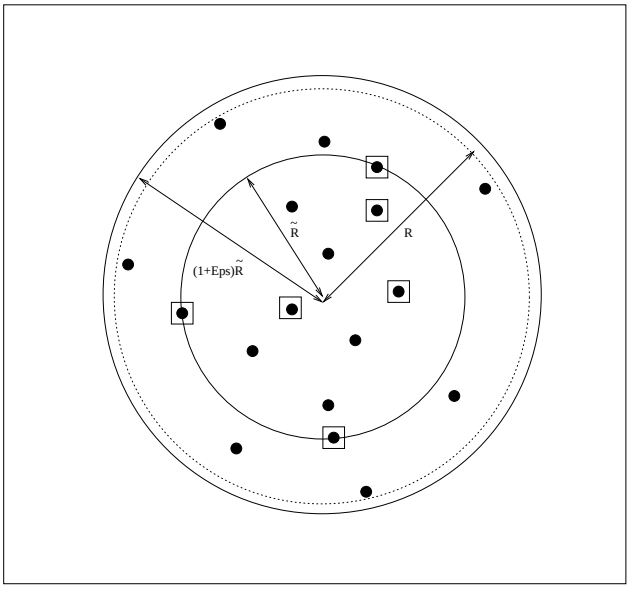

Figure 1. The dotted circle is the exact MEB of the entire data. The inner circle is the exact MEB of Core Set(the set of points enclosed in squares) and its $(1+\epsilon)$ expansion(the outer circle) covers all points.

3. Formulation of Multiclass SVM

The Multiclass SVM(MSVM) is formulated here as an SVM with vector output. This idea comes from a simple reinterpretation of the normal vector of the separating hyperplane(Szedmak & Shawe-Taylor, 2005). This vector can be viewed as a projection operator of the feature vectors into a one dimensional subspace. An extension of the range of this projection into multidimensional subspace gives the solution for vector labelled training of SVM.

Let the training data set be $S = \{(x_i, y_i)\}_{i=1}^m$ where $x_i \in R^d$, $y_i \in R^T$ for some integers $d, T > 0$. i.e. we have m training points whose labels are vector valued.

For a given training task having T classes, these label vectors are chosen out of the finite set of vectors $\{y_1, y_2, ...y_T\}$. Now we can define the primal for the learning problem as

$$\min_{W,b,\rho,\xi_i} \quad trace(W^TW) + \|b\|^2 - 2\rho + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_1^2$$
$$\text{s.t.} \quad y_i^T(W\phi(x_i) + b) \geq \rho - \xi \tag{7}$$

The corresponding Wolfe Dual is

$$\min_{\alpha} \quad \sum_{i,j=1}^{m}\alpha_i\alpha_j(< y_i, y_j > k(x_i, x_j) + < y_i, y_j > +\delta_{ij}\nu m)$$
$$\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i = 1 \qquad \alpha_i \geq 0 \quad \forall i \tag{8}$$

From the Karush-Kuhn-Tucker(KKT)(Fletcher, 2000) conditions on equation(7) we get

$$W = \sum_{i=1}^{m}\alpha_i y_i \phi(x_i)^T \qquad\qquad b = \sum_{i=1}^{m}\alpha_i y_i$$

So the decision function predicting one of the labels from 1...T for any test pattern x_j can be expressed as

$$\arg\max_{t=1...T} < y_t, (W\phi(x_j) + b) >$$
$$= \arg\max_{t=1...T}\left(\sum_{i=1}^{m}(\alpha_i < y_i, y_t > (k(x_i, x_j) + 1))\right) \tag{9}$$

Now the question that arises is about choosing the label vectors. Let $y_i(t)$ denote the t^{th} element of the label vector y_i corresponding to the pattern x_i. One of the convenient ways is to choose it as

$$y_i(t) = \begin{cases} \sqrt{\dfrac{(T-1)}{T}} & \text{if item } i \text{ belongs to category } t \\ \sqrt{\dfrac{1}{T(T-1)}} & \text{otherwise} \end{cases}$$

The inner product between the vectors will then be

$$< y_i, y_j >= \begin{cases} 1 & \text{if } i \text{ and } j \text{ is of same class} \\ \dfrac{(3T-4)}{T(T-1)} & \text{otherwise} \end{cases}$$

It may be observed that this kind of an assignment is suitable for any $T > 2$.

4. Multiclass CVM

4.1. Viewing MSVM as an MEB Problem

To simplify the notation let us denote the pair (x_i, y_i) as z_i. Now the training data set can be denoted as $S = \{z_i\}_{i=1}^m$.

To view the MSVM formulation discussed in the above section as an MEB problem let us rewrite the dual of MSVM problem given by equation(8) as

$$\min_{\alpha} \quad \sum_{i,j=1}^{m}\alpha_i\alpha_j\tilde{k}(z_i, z_j)$$
$$\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i = 1 \qquad \alpha_i \geq 0 \quad \forall i \tag{10}$$

where $\tilde{k}$ is the modified kernel function given by

$$\tilde{k}(z_i, z_j) = (< y_i, y_j > k(x_i, x_j) + < y_i, y_j > +\delta_{ij}\nu m) \tag{11}$$

When $k(x, x) = \kappa$ is satisfied this transformed kernel function $\tilde{k}$ also satisfies the condition

$$\tilde{k}(z, z) = \kappa_1, \text{ some constant.}$$

Hence the above mentioned problem is an MEB problem. We can now use the approximate MEB finding algorithm discussed in Section 2.2 to solve it.

4.2. Solving the MEB Problem

Once the Multiclass SVM problem is formulated as an MEB problem, we get a modified kernel function $\tilde{k}$ with an associated mapping function $\tilde{\phi}$. This MEB problem is then solved using the approximation algorithm introduced by Bădoiu and Clarkson (2002). The idea here is to incrementally expand the ball by including the point that is farthest from the center of the MEB obtained from the previous iteration.

The distance $G(z_i)$ of a point z_i from the center a of the MEB is given by

$$G^2(z_i) = \|\phi(z_i) - a\|^2$$
$$= \tilde{k}(z_i, z_i) - 2\sum_{j=1}^{n}\alpha_j\tilde{k}(z_j, z_i) + \sum_{j,k=1}^{n}\alpha_j\alpha_k\tilde{k}(z_j, z_k) \tag{12}$$

where n is the number of points in the core set from which the MEB was found in the previous iteration.

The above equation is obtained using the expression

$$a = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

given by the KKT conditions on the Lagrangian for equation (1). Now the radius R of the MEB computed at the current iteration can be given as

$$R = G(z_i) \text{ where } 0 < \alpha_i < C \tag{13}$$

The approximate MEB learning algorithm starts with a core set containing one point(randomly chosen) from each of the competing classes. The MEB is then learned in the incremental manner as explained above.

As in CVM, here also we employ the probabilistic speed up(Smola & Schölkopf, 2000) method to reduce the computational effort. A random sample of 59 points is taken from the training set and the point from this sample that is farthest from the center of the MEB obtained from the previous iteration is taken as the point to be added to the core set in current iteration.

4.3. MCVM Algorithm

To write the algorithm formally let us denote

- the data set by S
- core set at iteration i by Q_i
- center of the MEB obtained for core set at iteration i by $\tilde{a}_i$
- radius of the MEB obtained for core set at iteration i by $\tilde{R}_i$
- MEB found for core set Q as $B_Q(\tilde{a}, \tilde{R})$

The MCVM algorithm employs three user defined parameters. They are:

- The parameter q of the Gaussian kernel function.
- The ϵ used by the approximate MEB finding algorithm used in MCVM.
- The ν parameter value of MSVM implemented by MCVM.

Now the algorithm can be given as

1. Initialize the core set Q_0 with one point from each of the classes in S.
2. Train the MEB with the current core set Q_0.
3. Terminate if there is no point z such that $\tilde{\phi}(z)$ falls outside the $(1+\epsilon)$-ball $B_Q(\tilde{a}_i, (1+\epsilon)\tilde{R}_i)$.
4. Find z such that $\tilde{\phi}(z)$ is farthest from $\tilde{a}_i$. Set $i = i+1, \quad Q_{i+1} = Q \cup \{z\}$.
5. Find the new MEB $B_Q(\tilde{a}_{i+1}, \tilde{R}_{i+1})$ from Q_{i+1} and go to step 3.

The low training time requirements of MCVM makes it amenable to use the standard parameter tuning techniques like cross validation to determine the parameters q, ϵ and ν.

4.4. Time and Space Complexities of MCVM

Let m be the number of points in the training set. First, let us consider the case when the probabilistic speed up is not used in MCVM. In Bădoiu and Clarkson (2002), it is proved that the approximate MEB finding algorithm converges in at most $\frac{2}{\epsilon}$ iterations. In other words, the total number of iterations τ is of $O(\frac{1}{\epsilon})$.

We start the approximate MEB finding algorithm with a core set having one randomly chosen data point from each of the classes. So, if T is the number of classes, we have $\mid Q_0 \mid = T$. Here $\mid \cdot \mid$ denotes the cardinality. Since only one data point is added at each iteration we get $\mid Q_i \mid = i + T$. Since the total number of iterations is of $O(\frac{1}{\epsilon})$, the size of the final core set is also of $O(\frac{1}{\epsilon})$. In practice it is observed that the size of the core set is much smaller than this worst case theoretical upper bound(Kumar et al., 2003). To get an upper bound on the computational expense, we assume that the QP problem to find the MEB for a core set is of $O(n^3)$ complexity, where n is the size of the core set (we do it even though $O(n^2)$ algorithms such as the one discussed in (Schölkopf et al., 2001) exist in the literature).

Now we can see that core set initialization takes $O(1)$ time(since we do a random initialization) and the distance computations in steps 3 and 4 take $O((i+T)^2 + im) = O(i^2 + im)$ time. Finding the MEB in step 5 takes $O((i+T)^3) = O(i^3)$ time, and the other operations take constant time. Hence the i^{th} iteration takes a total of $O(im + i^3)$ time. Now the total time taken by τ iterations is

$$\sum_{i=1}^{\tau} O(im + i^3) = O(\tau^2 m + \tau^4) = O\left(\frac{m}{\epsilon^2} + \frac{1}{\epsilon^4}\right)$$

which is linear in m for a given ϵ.

We know that only the core vectors are involved in approximate MEB finding QP problem. So at any iteration i, the space requirement is of $O(\mid Q_i \mid^2)$. Since the number of iterations τ is of $O(\frac{1}{\epsilon})$ and $\mid Q_i \mid = i+T$, the space complexity for the whole procedure is $O(\frac{1}{\epsilon^2})$, and hence is independent of m for a given ϵ.

Now let us consider the case when the probabilistic speed up is used. Here also the core set initialization takes $O(1)$ time. But the distance computations in steps 3 and 4 take only $O((i+T)^2) = O(i^2)$ time (since there is no scan of the entire training data to find the farthest point outside the current MEB). Time for the other operations remain the same. So the i^{th} iteration takes $O(i^3)$ time. As the probabilistic speed up method may not find the farthest point in each iteration(Tsang et al., 2005), the number of iterations τ may be larger than $\frac{2}{\epsilon}$ though it can still be bounded by $\frac{1}{\epsilon^2}$(Bădoiu & Clarkson, 2002). Hence, the time complexity of the entire algorithm is

$$\sum_{i=1}^{\tau} O(i^3) = O(\tau^4) = O\left(\frac{1}{\epsilon^8}\right)$$

thus for given ϵ, it is independent of m.

As discussed above, the space complexity depends on the size of the core set which in turn depends on the number of iterations. In probabilistic speed up method, the number of iterations is bounded by $\frac{1}{\epsilon^2}$ and hence space complexity is of $O(\frac{1}{\epsilon^4})$.

When ϵ decreases, the MCVM solution becomes closer to the exact optimal solution, but at the expense of higher time and space complexities. Such a trade off between efficiency and approximation quality is typical of all approximation schemes(Tsang et al., 2005). For handling large data sets, an algorithm with the best asymptotic efficiency(both time and space) is normally used. Since MCVM is designed for handling large data sets, the asymptotic efficiency(both time and space) of the method is analyzed using the O-notation. But for smaller problems, the MCVM method may be out performed by algorithms that are not as efficient as MCVM asymptotically.

It may also be noted that the proposed MCVM technique is much better than the other direct multiclass SVM formulations because they have a prohibitive $O(m^3)$ time complexity (note that SVM using SMO algorithm is only of $O(m^2)$ time complexity) for solving the associated QP problem.

Since both CVM and MCVM techniques solve an identical QP problem using the same approximate MEB finding(Bădoiu & Clarkson, 2002) algorithm, the convergence proof given by Tsang et al. (2005) holds good for MCVM technique also.

5. Experimental Results

Experiments are done with two synthetic data sets and two real world data sets. We compared the results obtained with one-against-one SVM(OvO-SVM)(Chang & Lin, 2001), one-against-one CVM(OvO-CVM)(Tsang et al., 2005) and the proposed MCVM. The one-against-one SVM is chosen for comparison because it is reported to have given the best performance among the one-against-all, one-against-one and DAGSVM approaches(Hsu & Lin, 2002). The LIBSVM Version 2.8 (Chang & Lin, 2001) implementation is used for the experiments with SVM. In all the experiments we used Gaussian kernel function. All the experiments were done on a Intel Xeon(TM) 3.06GHz machine with 2GB RAM.

5.1. Synthetic Data Sets

To evaluate the performance of SVM and the proposed MCVM technique we generated two synthetic data sets viz; Synth 1 and Synth 2. These data sets are obtained from 10 multivariate(5 dimensional) normal distributions. Each of these distributions is given a different label and hence we get a 10 class problem. Synth 1 has 100000 points in the training set and 25000 points in the test set. In case of Synth 2 there are 1000000 points in the training set and 200000 points in the test set. The parameters used in the experiments and the results obtained are shown in Table 1 and Table 2 respectively.

5.2. Real World Data Sets

The real world data sets used are: SensIT Vehicle (acoustic) data set from the LIBSVM page available at *"http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass"* and Intrusion Detection data set from the UCI KDD archive available at *"http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html"*. The parameters used in the experiments are given in Table 1.

5.2.1. SensIT Vehicle (acoustic)

Acoustic(Acst) data set pertains to a 3 class problem with 78823 patterns in the training set and 19705 patterns in the test set. Each pattern here is described using 50 numerical features. The results obtained for this data set are given in Table 3.

Data	#TR	#TST	SVM	CVM			MCVM		
			q	q	ϵ	C	q	ϵ	ν
Synth 1	100000	25000	0.2	0.12	10e-6	0.05	0.075	1.5e-3	9e-3
Synth 2	1000000	200000	0.2	0.05	10e-6	500	0.1	7e-4	9e-3
Acoustic	78823	19705	0.02	3.5	1e-6	1e4	7.5	2e-3	9e-3
Intrusion	4898431	311029	0.026	1e-3	1e-6	1e6	10	5e-3	5e-3

Table 1. Details of data sets and parameter values used in experiments. The abbreviations used are: number of training points(#TR), number of points in the test data(#TST).

Data	#SV			TT			GP			CT		
	SVM	CVM	MCVM	SVM	CVM	MCVM	SVM	CVM	MCVM	SVM	CVM	MCVM
Synth 1	2338	2722	291	296	130	10	99.548	98.8	98.912	14	11	4
Synth 2	9326	2908	322	18276	23935	29	99.587	95.531	98.814	414	148	35

Table 2. Results for the experiments with Synthetic data sets for SVM, CVM and MCVM. The abbreviations used are: one-against-one SVM(SVM), one-against-one CVM(CVM), Multiclass Core Vector Machine(MCVM), number of support vectors(#SVs), training time(TT) in seconds, generalization performance(GP) on test data in percentage and classification time(CT) for test data in seconds.

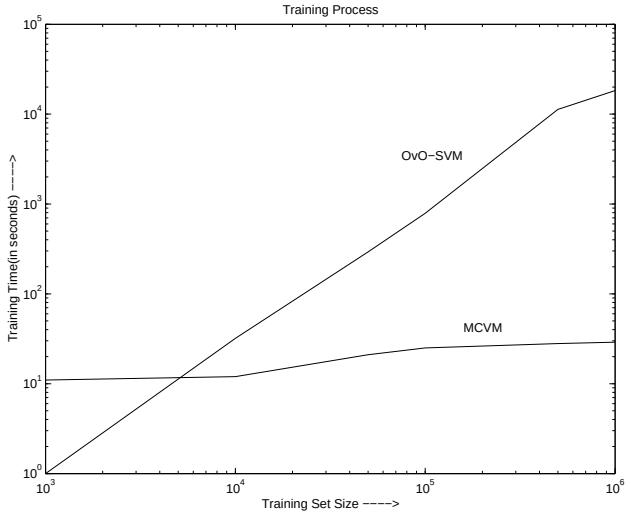

Figure 2. A plot of Training Time(in seconds, in log scale) of OvO-SVM and MCVM against the Training set size(in log scale) for Synthetic Data set.

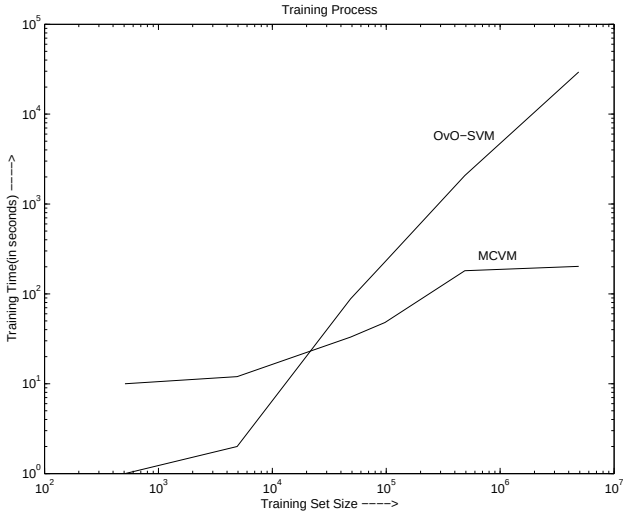

Figure 3. A plot of Training Time(in seconds, in log scale) of OvO-SVM and MCVM against the Training set size(in log scale) for Intrusion Detection Data Set.

5.2.2. Intrusion Detection Data Set

The Intrusion Detection(Intr) data set has 4898431 training patterns and 311029 test set patterns. We have considered the original 5 class problem. Only the 38 numerical features available were considered and they were normalized to have values between zero and one by dividing them by their maximum values. The results obtained for this data set are given in Table 3.

From the empirical results it is observed that MCVM technique gives comparable generalization performance(overall and class wise generalization) as that of one-against-one SVM at a very less computational expense. Further, the results show that MCVM gives better generalization performance than the one-against-one CVM at a lesser computational expense.

To study the scalability of the proposed MCVM technique we have done experiments with the synthetic data set(Synth 2) and the Intrusion Detection Data Set. In these experiments we sampled the training set keeping the class distributions. Training samples of different sizes are obtained and the corresponding

Data	#SV			TT			GP			CT		
	SVM	CVM	MCVM	SVM	CVM	MCVM	SVM	CVM	MCVM	SVM	CVM	MCVM
Acst	51022	13186	1300	5468	4984	4196	70.109	62.233	68.876	718	113	32
Intr	21714	154	187	29486	220	202	91.652	87.023	91.671	1891	24	66

Table 3. Results for the experiments with Real world data sets for SVM, CVM and MCVM. The abbreviations used are: one-against-one SVM(SVM), one-against-one CVM(CVM), Multiclass Core Vector Machine(MCVM), number of support vectors(#SVs), training time(TT) in seconds, generalization performance(GP) on test data in percentage and classification time(CT) for test data in seconds.

training time taken by OvO-SVM and MCVM algorithms are recorded (OvO-CVM is not included here because its generalization performance is not comparable). A plot of the increase in Training Time(in seconds, in log scale) against the increase in Training set size(in log scale) for synthetic and intrusion detection data sets are shown in Figures 2 and 3 respectively. It can be observed that the increase in training time for MCVM is comparatively lesser when compared to that of OvO-SVM with the increase in the training set size. The plot corresponding to MCVM clearly illustrates that the proposed MCVM technique scales well with the size of the data set.

6. Conclusions

A scalable kernel based multiclass classifier namely Multiclass Core Vector Machine is proposed in this paper. The proposed method achieves comparable generalization performance as that of one-against-one SVM at a very less computational expense. Further, MCVM gives better generalization performance than the one-against-one CVM at a lesser computational expense. Some properties of the proposed method are **a**) The method scales well with the size of the data set and **b**) Incremental training is possible for the method and hence can be used for online learning.

References

Aiolli, F., & Sperduti, A. (2005). Multiclass Classification with Multi-Prototype Support Vector Machines. *Journal of Machine Learning Research*, *6*, 817–850.

Bădoiu, M., & Clarkson, K. L. (2002). Optimal core sets for balls. *In DIMACS workshop on Computational Geometry.*

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: A library for Support Vector Machines. http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

Fletcher, R. (2000). *Practical methods of optimization, 2nd ed.* New York: Wiley-Interscience.

Hsu, C.-W., & Lin, C.-J. (2002). A Comparison of Multiclass Support Vector Machines. *IEEE Transactions on Neral Networks*, *13*, 415–425.

Kumar, P., Mitchell, J. S. B., & Yildirim, E. A. (2003). Approaximate minimum enclosing balls in high dimensions using core sets. *ACM Journal of Experimental Algorithms*, *8*, Article No. 1.1.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1472.

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 911–918). Stanford, CA,USA.

Soman, K. P., Loganathan, R., Vijaya, M. S., Ajay, V., & Shivsubramani, K. (2007). Fast Single-shot Multiclass Proximal Support Vector Machines and Perceptrons. *Proceedings of the International Conference on Computing: Theory and Applications (ICCTA)* (pp. 294–298). Kolkata, India.

Szedmak, S., & Shawe-Taylor, J. (2005). Multiclass learning at one-class complexity. *Technical Report No: 1508, School of Electronics and Computer Science, Southampton, UK.*

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core Vector Machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, *6*, 363–392.

Vapnik, V. N. (1998). *Statistical learning theory.* John Wiley and Sons.

Weston, J., & Watkins, C. (1999). Support Vector Machines for Multi-class Pattern Recognition. *Proceedings of the Seventh European Symposium On Artificial Neural Networks* (pp. 219–224). Brussels.