International Journal of Computational Intelligence Systems, Vol. 7, No. 2 (April 2014), 197-212

Fast Support Vector Machine Classification for Large Data Sets

Xiaoou Li
CINVESTAV-IPN, Departamento de Computacion,
Mexico City, Mexico

Wen Yu
CINVESTAV-IPN, Departamento de Control Automatico,
Mexico City, Mexico

Received 14 June 2011
Accepted 12 December 2011

Abstract

Normal support vector machine (SVM) algorithms are not suitable for classification of large data sets because of high training complexity. This paper introduces a novel two-stage SVM classification approach for large data sets. Fast clustering techniques are introduced to select the training data from the original data set for the first stage SVM, and a de-clustering technique is then proposed to recover the training data for the second stage SVM. The proposed two-stage SVM classifier has distinctive advantages on dealing with huge data sets such as those in bioinformatics. Finally, we apply the proposed method on several benchmark problems. Experimental results demonstrate that our approach has good classification accuracy while the training is significantly faster than other SVM classifiers.

Keywords: support vector machine, classification, large data sets

1. Introduction

With the development of software and hardware, enormous quantities of high dimensional data are being stored in databases continuously. Semi-automatic data classification methods are needed to analyze and understand huge amounts of data. A classification model can serve as an explanatory tool to distinguish between objects of different classes [34]. A classification process includes two phases which are training phase and testing phase. In the training phase, a training set is used to decide how the parameters ought to be weighted and combined in order to separate various classes of objects. The learning attempts to discover an optimal representation of a data set with known class memberships. In the test phase, the weights determined at the training phase are applied to a set of objects with unknown class labels (test set) to determine their classes. Some practical classification methods involve a heuristic approach intending to find a "good-enough" solution to the optimization problem.

There are several standard classification techniques in literature, such as simple rule based and nearest neighbor classifiers, Bayesian classifiers, artificial neural networks, decision tree, support vector machine (SVM), ensemble methods, etc. Among those, neural networks are one of the most widely used technique [25]. As a universal approximator, neural network classifier has a very expressive hypothesis space. However, neural network training is a time consuming process, and the classification accuracy cannot be always guaranteed since they are quite sensitive to the presence of noise in the training data. Decision trees have also been used for classification problems. Decision trees technique is faster at the training phase than neural networks. But, they are not flexible at modeling parameter space [30]. A simple classifier may

be the nearest-neighbor approach [15]. Nearest neighbor methods have the advantage that they are easy to implement. But, this technique is still too slow if the input data sets have many examples. On the other hand, they are very sensitive to the presence of irrelevant parameters.

Among these techniques, SVM is one of the best-known techniques for its optimization solution [10]. Recently, some new SVM classifications are proposed. A geometric approach to SVM classification is given by Mavroforakis et al [22]. Fuzzy neural network SVM classifier is studied by Lin [21]. Despite of its good theoretic foundations and generalization performance, SVM is not suitable for classification of large data sets since it needs to solve the quadratic programming (QP) in order to find a separation hyper-plane, which causes an intensive computational complexity. Many researchers have tried to find possible methods to apply SVM classification for large data sets. Generally, these methods can be divided into two types; Modify SVM classifier so that it could deal with large data sets within an acceptable time, and Reduce a large data set to smaller one so that a normal SVM could be applied.

Chunking is the first decomposition method used, a standard projected conjugate gradient (PCG) chunking algorithm can scale somewhere between linear and cubic in the training set [9][17]. Sequential Minimal Optimization (SMO) is a fast method to train SVM [27][8]. SMO breaks the large QP problem into a series of smallest possible QP problems; it is faster than PCG chunking. Dong et al introduced a parallel optimization step where block diagonal matrices are used to approximate the original kernel matrix so that SVM classification can be split into hundreds of sub-problems [1]. A recursive and computational superior mechanism referred as adaptive recursive partitioning was proposed in [19], where the data is recursively subdivided into smaller subsets. Genetic programming is able to deal with large data sets that do not fit in main memory [12]. Neural networks technique can also be applied for SVM to simplify the training process [16].

Clustering is an effective tool to reduce data set size. For examples, hierarchical clustering [39][1], k-means cluster [28] and parallel clustering [33]. Clustering based methods can reduce the computations burden of SVM, but they are very complex for large data set. Rocchio bundling is a statistics-based data reduction method [35]. Another approach is to apply the Bayesian committee machine to train SVM on large data sets [32] where the data is divided into m subsets of the same size, and m models are derived from the individual sets. But it has more error rate than normal SVM and the sparse property does not hold. Random selection is to select data in such way that the learning is maximized by the data <cite>Schohn:00</cite>. However, it could over-simplify the training data set and lose the benefit of SVM.

In this paper, we propose a two-stage SVM classification method by using reduced data set in each stage.

Clustering is an important technique for fast retrieval of relevant information from databases. The goal of clustering is to separate finite unlabeled items into a finite and discrete set of "natural" hidden data structures, such that items in the same cluster are more similar to each other and those in different clusters tend to be dissimilar, according to certain measure of similarity or proximity. A large number of clustering methods has been developed, e.g., squared error-based k-means [2], fuzzy C-means [26], kernel-base clustering [14]. However, for these clustering methods, the optimal number of clusters should be predefined which involves more computational cost than clustering itself [38]. In this paper, we select the cluster centers and data of mix-labeled clusters as training data for the first stage SVM, we believe these data are the most useful and representatives in a large data set for finding support vectors. However, our objective is to select representatives data from a large data set for training SVM, we do not take care of the optimal number of clusters. So we need to find clustering techniques in which this computation cost can be eliminated. Two fast data selection techniques, MEB and random selection, are introduced for this purpose in the next section.

However, are the selected data representatives enough? Note that, data of the clusters near the hyper-plane are not used totally for training SVM we have only selected the cluster centers and mixed clusters. So, this may affect the classification precision, i.e., the obtained decision hyper-plane may not be precise enough. However, at least it gives us a reference on data distribution. To compensate this disadvantage, we train SVM again using the data near the hyper-plane obtained by the last SVM (we call it the first stage SVM), it is the second stage SVM.

By the sparse property of SVM, the data which are not support vectors will not contribute the optimal hyper-plane. The input data sets which are far away from the decision hyper-plane should be eliminated, meanwhile the data sets which are possibly support vectors should be used. According to above analysis, we make the following modification on the training data set of the first stage SVM.

1). Remove the data far from the hyper-plane from the training data set because they will not contribute to find the support vectors,

2). Keep the data of the mix-labeled clusters since they are more likely support vectors.

3). Additionally, we add the data of the clusters whose centers are support vectors of the first stage SVM.

In general, our approach consists of four steps which are shown in Figure 1: 1) data selection, 2) the first stage SVM classification, 3) de-clustering, 4) the second stage SVM classification.

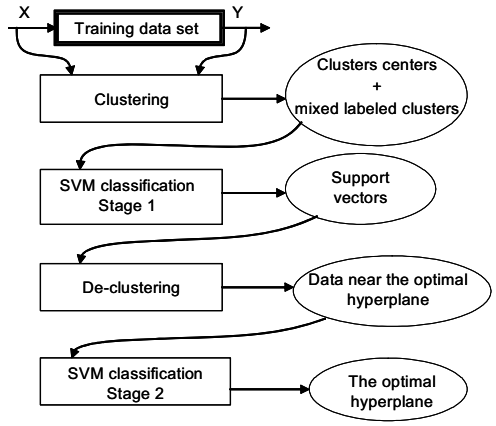

Figure 1. Two-stage SVM classification

We firstly obtain clusters by using a clustering algorithm and the first stage SVM is applied on the data which are cluster centers or in the mixed clusters. Then, we remove the clusters whose centers are not support vectors from the original data set. For the remaining clusters, we apply the de-clustering technique, and use the data in these clusters to train the second stage SVM. The experimental results show that the accuracy obtained by our approach is very close to classic SVM methods, while the training time is significantly shorter.

The rest of the paper is organized as follows. Section II introduces two new clustering methods. Section III explains the two-stage SVM classification. Performance and complexity analysis are given in Section IV. Section V shows our experiment results on four well-known data sets, comparisons with other SVM based algorithms are made. Conclusion and discussion are given in Section VI.

2. Related work

To use clustering techniques before computing the classifier is an interesting strategy for large data sets problems. Although many methods for solving the optimization problem of SVM by clustering methods are available, we list here only some interesting techniques which can be used to train SVMs on a large data set, such as CB-SVM [39], CB-SOCP [31], CT-SVM [18], and RS-MCS [7].

CB-SVM [39] applies a hierarchical micro-clustering that scans the entire data set only once, the proposed method scales well for large data sets and gives accuracies comparable to other SVM implementations. However CB-SVM the hierarchical micro-clustering employed is hight dependent of the dimension of the input data set and may not perform as well with high dimensional data sets.

CB-SOCP [31] presents a formulation for large data sets. Their authors assume that the class conditional densities of mixture data points can be modeled using mixture models. The algorithm uses BIRCH in order to estimate a second order statistics of the components. The proposed method is scalable for large data sets and the accuracies obtained using CB-SOCP are comparable to other SVM implementations. However CB-SOCP algorithm sampled the input data and is clear that if we sampling the input data we could hurt the training process of SVM, especially when the probability distribution of training and testing data were different [39].

CT-SVM [18] applies reduction techniques by clustering analysis to find relevant support vectors in order to speed up the training process, the algorithm builds a hierarchical clustering tree for each class in the data set iteratively in several epochs. SVM is trained on the nodes of each tree. The support vectors of the classifier are used as prior knowledge which govern the tree growth. Only support vectors are allowed to grow, and non support vectors are stopped. This method is scalable for large data sets and the accuracies obtained are comparable to other SVM implementations. However the algorithm is susceptive to noisy and incomplete data sets.

RS-MCS [7] use mirror point pairs and a multiple classifier system to reduce the training time of a support vector machines. The authors developed a approach by K Means Clustering in order to selecting and combining a given number of member classifiers which takes into account accuracy and efficiency. The accuracies obtained using this algorithm is comparable to other SVM implementations. However this method works only with small and medium size data sets.

The main contribution of this paper is the demonstration that the speed of SVM can be increased by reducing drastically the input data set to SVM (which is crucial in large data sets). This has been done using two stages of SVM and by demonstrating its power in both speed and accuracy.

The first main advantage of this mechanism, when compared to other SVM implementations, is that this algorithm obtain support vectors from a first stage of SVM and use them in order to obtain all the data points near support vectors. The great advantage of this step is to remove most nonsupport vectors quickly and collect training sets for the next step.

3. Clustering algorithms

Clustering is an unsupervised classification method, it divides a data set into subsets (clusters), so that the patterns belonging to any one of the clusters are similar and the patterns of the different clusters are as dissimilar as possible. A large number of clustering methods have been developed, e.g., squared error-based k-means [2], fuzzy C-means [26], kernel-base clustering [14]. However, clustering algorithms are generally time consuming for large data sets. Clustering is not the final objective of this paper; we only use the cluster centers that are support vectors and data points enclosed on them as representative data set from original large data set, so that SVM could be applied. Therefore, the precision of clustering algorithm does not affect too much on our classification. What we need is a fast clustering algorithm which can give us a scheme of cluster distribution. In this section, we propose two new fast clustering algorithms, one is the minimum enclosing ball (MEB) clustering, the other is a random selection technique. We use them to partition the training data set and select data according to the partition result to train the first stage SVM. Both methods do not need to calculate the optimal number of clusters, so, they can be done in a short time.

3.1. *Fuzzy C means clustering*

Consider a finite set of elements $X = \{x_1, x_2, \ldots, x_n\}$ as being elements of the $p-$ dimensional Euclidian space R^p, that is, $x_i \in R^p$ The problem is to perform a partition of this data set into l fuzzy sets. The criterion is usually to optimize an objective function. The final result of fuzzy clustering can be expressed by a partition matrix U such that $U = [u_{ij}], \quad i = 1 \ldots n, \quad j = 1 \ldots l, \quad u_{ij}$ is a numeric value in $[0,1]$. There are two constraints on the value of u_{ij}. First, a total membership of the element $x_i \in X$ in all classes is equal to 1, second every constructed cluster is non-empty. That is,

$$\begin{aligned} &\textstyle\sum_{j=1}^{l} u_{ij} = 1, \ \text{for all } i = 1,2,\ldots,n \\ &0 < \textstyle\sum_{i=1}^{n} u_{ij} < n, \ \text{for all } j = 1,2,\ldots l \end{aligned} \tag{1}$$

A general form of the objective function is $J(u_{ij}, \mathbf{v}_k) = \sum_{j=1}^{l} \sum_{i=1}^{n} \sum_{k=1}^{l} g[w(x_j), u_{ij}]$ $d(x_i, v_k)$, where $w(x_j)$ is the a prior weight for each x_j, $d(x_i, v_k)$ is the degree of dissimilarity between the data x_i and the supplemental element v_k, which can be considered as the central vector of the k th cluster. The degree of dissimilarity is defined as a measure that satisfies two axioms: 1) $d(x_i, v_k) \geq 0$, 2) $d(x_i, v_k) = d(x_k, v_i)$. So the fuzzy clustering can be formulated as an optimization problem:

$$\begin{aligned} &\min J(u_{ij}, v_k), \\ &\text{Subject}: \textstyle\sum_{j=1}^{l} u_{ij} = 1, \ 0 < \sum_{i=1}^{n} u_{ij} < n \end{aligned} \tag{2}$$

where $i,k=1,2\ldots l,\quad i=1,2\ldots n.$ The objective function of the fuzzy C-means (FCM) is

$$J(u_{ij},v_k)=\sum_{j=1}^{l}\sum_{i=1}^{n}u_{ij}^{m}\left\|x_i,v_j\right\|^2 \tag{3}$$

where u_{ij}^{m} denotes the membership grade of x_i in the cluster A_k , v_j denotes the center of A_k, and $\left\|x_i,v_j\right\|$ is the distance of x_i to center v_j, x_i and v_j are $p-$ dimension vectors, $m>1,$ is called an exponential weight which influences the degree of fuzziness of the membership function, m influences the degree of fuzziness of the membership function. Note that the total membership of the element x_i in all classes is 1 , *i.e.*, $\sum_{k=1}^{l}u_{ki}=1,\quad i=1\cdots n.$ To solve the minimization problem (2) with respect to the objective function (3), we can fix u_{ij} and v_i and apply the conditions (1), that is

$$\begin{aligned} v_j &= \frac{1}{\sum_{i=1}^{n}(u_{ij})^m}\sum_{i=1}^{n}(u_{ij})^m x_i \\ u_{ij} &= \frac{\left(1/\left\|x_i,v_i\right\|^2\right)^{1/m-1}}{\sum_{k=1}^{c}\left(1/\left\|x_i-v_k\right\|^2\right)^{1/m-1}} \end{aligned} \tag{4}$$

3.2. Minimum enclosing ball (MEB) clustering

The MEB problem can be dated back as early as in 1857, when Sylvester first investigated the smallest radius disk enclosing m points on the plane. It has found applications in diverse areas such as computer graphics (e.g., for collision detection, visibility culling), machine learning (e.g., similarity search), facility locations problems, shape fitting problems. In this paper, by $(1+\varepsilon)$ -approximation of MEB we propose MEB clustering approach. The running time of the new algorithm is linearly or near linearly dependent on the number of points and the dimension. MEB clustering proposed in this paper uses the concept of core-sets in [20][3]. It is defined as follows.

Definition 1. *The ball with center* c *and radius* r *is denoted as* $B(c,r).$

Definition 2. *Given a set of points* $S=\{x_1,\ldots x_m\}$ *with* $x_i\in R^p$ *the minimum enclosing ball (MEB) of* S *is the smallest ball that contains all balls and also all points in* $S,$ *it is denoted as* $MEB(S).$

Because it is very complicated to find the optimal ball $MEB(S),$ we use an approximation method defined as follow.

Definition3. $(1+\varepsilon)-$ *approximation of* $MEB(S)$ *is denoted as a ball* $B(c,(1+\varepsilon)r),\quad \varepsilon>0$ *with* $r\geq r_{MEB(S)}$ *and* $S\subset B(c,(1+\varepsilon)r).$

Definition 4. *A set of points* S *is a core-set of* S *if* $MEB(S)=B(c,r)$ *and* $S\subset B(c,(1+\varepsilon)r).$

For clustering problem, there should be many balls in the data set $S.$ So the definition of $(1+\varepsilon)-$ approximation of $MEB(S)$ is modified as

Definition 5. *In clustering,* $(1+\varepsilon)-$ *approximation of MEB(* S *) is denoted as a set of* k *balls* B_i $(i=1\cdots k)$ *containing* S *with the same radius, i.e.,* $S\subset B_1\cup B_2\cup\ldots\cup B_k$.

In other words, given $\varepsilon>0,$ a subset $S,$ is said to be a $(1+\varepsilon)-$ approximation of S for clustering if $MEB(S)=\cup_{i=1}^{k}B_i(c_i,(1+\varepsilon)r_i)$ and $S\subset\cup_{i=1}^{k}B_i(c_i,(1+\varepsilon)r_i),$ *i.e.,* S is a $(1+\varepsilon)-$ approximation with an expansion factor $(1+\varepsilon)$.

In order to include include all recovered data, a possible way is to select the radius r as the maxima margin,

$$r=\frac{2}{\left\|w^*\right\|} \tag{5}$$

How to choose the user-defined r is a trade-off problem. If r is too small, there will be many groups at the end, their centers will be applied for the first-stage SVM. The data reduction is not good. Conversely, if r is too large, many objects that are not very similar may end up in the same cluster, some information will be lost.

In this practice, we use the following equation

$$r_k = (k - 1 + \text{rand})\frac{c_{\max} - c_{\min}}{l} \qquad (6)$$

where l is the number of the balls, n_V is the number of the support vector in the first stage, rand is a random number in $(0,1)$, $c_{\max} = \max(c_k)$, $k = 1 \cdots n_V$, $c_{\min} = \min(c_k)$. In order to simplify the algorithm, we use the same r for all balls

$$r = \frac{\sqrt{\sum_{i=1}^{n_V}(c_{\max} - c_{\min})^2}}{2l} \qquad (7)$$

Now we consider a finite set of elements $X = \{x_1, x_2, \ldots, x_n\}$ in $p-$ dimensional Euclidian space $x_i = (x_{i1}, \ldots, x_{ip})^T \in R^p$. At first we randomly select the ball centers in the data set such that they can cover all range of the data. In most cases, there is no obvious way to select the optimal number of balls (clusters), l. An estimate of l can be obtained from the data using cross-validation, for example, the *v-fold cross-validation* algorithm [4] can automatically determine the number of clusters in the data, but this procces can be very cost and determining the optimal number of custers may involve a conputational cost very high. For this algorithm, we firstly guess the number of clusters l as 0.1% of the data number n. If the accuracy is bad, we increase 0.05% the number of clusters.

MEB clustering is a process to partition the data set into l balls with a minimum radius that include all data. Now consider a simple MEB clustering case when the number of balls (or clusters) is three, i.e., $l = 3$, see Figure 2. After randomly selecting three centers, we need to check if the three balls with radius r determined by (5). If all data has been included in the balls, we end the clustering process, and the three balls are the three clusters. If not, we enlarge the radius to $(1+\varepsilon)r$, and repeat to check again if all data are included into the new balls with new radius. In Figure 2, A1, B1 and C1 are not included in the three small balls Ball_A, Ball_B and Ball_C. then we enlarge the radius to $(1+\varepsilon)r$, and the new balls in dashed lines have already included them. But, A2, B2 and C2 are still outside of the new balls, so we need to enlarge ε again, and so on, until all data in X are inside the balls, i.e., $X \subset \cup_{i=1}^{l} B(c_i, (1+\varepsilon)r_i)$.

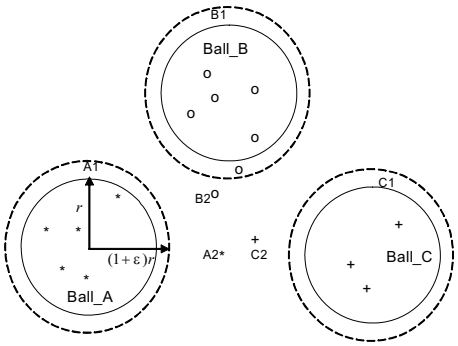

Figure 2. MEB clustering when the number of clusters is three.

The MEB clustering algorithm of sectioning l balls is as follows:

Step 1: Use the random sampling method (6) to generate l *ball centers* $C = \{c_1, \ldots, c_l\}$, *select the ball radius* r *as (5), and set* $\varepsilon = 0$.

Step 2: For point x_i, *calculate its distance to each ball center*

$$\varphi_k(x_i) = \|x_i - c_k\|^2, \quad k = 1, \cdots, l, \quad , i = 1, \cdots, n$$

The minimum distance is $\overline{\varphi}(x_i) = \min_k[\varphi_k(x_i)]$, suppose the minimum value occurs at center c_m.

- *If* $\overline{\varphi}(x_i) \le r$, *then put point* x_i *into the ball* $B(c_m, r)$. *Goto Step 3.*
- *If* $\overline{\varphi}(x_i) > r$, *then we increase the radius to* $(1+\varepsilon)r$, *where* $\varepsilon = \varepsilon + \frac{r}{\Delta}$, Δ *is the increasing step. Repeat increasing* ε *until* $\overline{\varphi}(x_i) \le (1+\varepsilon)r$. *Then put point* x_i *into the ball* $B(c_m, (1+\varepsilon)r)$. *Set* $r = (1+\varepsilon)r$ *and goto Step 3.*

Step 3: Set $i = i + 1$,

- *if* $i > n$, then *goto Step 4;*
- *otherwise, goto Step 2.*

Step 4: Complete the clustering and output the obtained clusters $B[c_k, (1+\varepsilon)r], k = 1, \cdots, l$.

3.3. Random selection clustering

Is impossible to obtain a perfect uniform random number generator. Fortunately, the restriction on the randomness of selected data for our first stage SVM classification is not very strict, since we have an

additional step (de-clustering) to compensate the information lost of this step.

We propose a new random selection algorithm to select training data for the first stage SVM. Assume that (X,Y) be the training patterns set, where $X=\{x_1,x_2,...,x_n\}$ is the input data set, x_i can be represented by a vector of p dimension, i.e., $x_i=(x_{i1}\dots x_{ip})^T$, $Y=\{y_1,y_2,...,y_n\}$ is the label set, label $y_i\in\{-1,1\}$. Our objective is to select a sample set $C=(c_1,c_2,\cdots,c_l)$ from X . Note that l has a similar meaning as number of cluster centers in MEB clustering, it needs to be predefined and suitable to be training data size for normal SVM algorithms.

Firstly, we divide the input data X into two groups according to their label Y . Suppose the number of positive labeled data is q and the number of negative labeled data is m , where $n=q+m$, we define the positive and negative labeled input data in array form as X^+ and X^- , and their corresponding label Y^+ and Y^- respectively. i.e.,

$$X^+=[x^+(1),\dots,x^+(q)]$$
$$X^-=[x^-(1),\dots,x^-(m)]$$
$$Y^+=[y^+(1),\dots,y^+(q)]=[1,\dots,1]$$
$$Y^-=[y^+(1),\dots,y^+(m)]=[-1,\dots,-1]$$

Thus, the original input data set is the union of X^+ and X^- , i.e., $X=X^+\cup X^-$.

Secondly, we select data by sampling the subsets X^+ and X^- independently. Here we use swapping method during selection process, see Figure 3. The selection is done in the following way: the first sample data c_1 is chosen from X^+ (or X^-) uniformly and randomly, then it is exchanged with the last data $x^+(q)$ (or $x^-(m)$) , i.e., $Swap(c_1,x^+(q))$ (or $Swap(c_1,x^-(m))$). The second sample c_2 is selected from the remaining data $\{x^+(1),\cdots,x^+(q-1)\}$, then it is exchanged with the second last data $x^+(q-1)$ (or $x^-(m-1)$), , i.e., $Swap(c_2,x^+(q-1))$ (or $Swap(c_2,x^-(m-1))$), and so on, until the required number of data for X^+ (or X^-) is selected. Generally, we may select $l/2$ samples from each labeled original data sets, i.e., $l/2$ samples from X^+ and $l/2$ samples from X^- if the labels are distributed evenly. If not, we may predefine a proportion on the samples for each label according to their distribution on original data or experience.

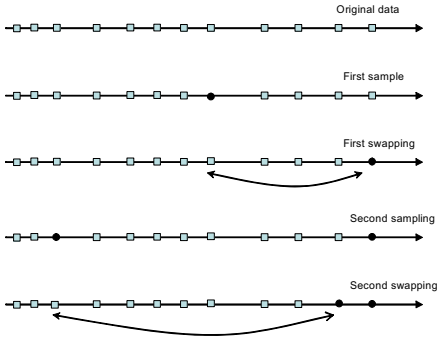

Figure 3. Random selection clustering

This swapping method can be also regarded as generating the first l entries in a random permutation. Above random selection can be realized by the following algorithm.

For $i:=n$ to $n-l+1$ Do
Generate a uniform $[0,1]$ random variate X.
$s\leftarrow\lceil iX\rceil$
Swap $(x[s],\ x[i])$
Return $x[n-l+1],\dots,x[n]$

4. Two-stage SVM classifier

In our approach, the classification task is divided into two stages. The first stage SVM classification obtains an approximated hyperplane, then the data near by the hyperplane of the first stage SVM will be de-clustered, and be used as training data for the second stage SVM, so that a more precise classification can be obtained. Our classification can be summarized as 4 steps which are shown in Figure 1. The following subsections will give a detailed explanation on each step.

4.1. *Lists of items*

After clustering, the obtained clusters can be classified into three types:

- clusters with only positive labeled data, denoted by Ω^+, i.e., $\Omega^+ = \{\cup\Omega_i \mid y = +1\}$;
- clusters with only negative labeled data, denoted by Ω^-, i.e., $\Omega^- = \{\cup\Omega_i \mid y = -1\}$;
- clusters with both positive and negative labeled data (or mix-labeled), denoted by Ω_m, i.e., $\Omega_m = \{\cup\Omega_i \mid y = \pm 1\}$.

Figure 3 (a) illustrates the clusters after MEB, where the clusters with only red points are positive labeled (Ω^+), the clusters with green points are negative labeled (Ω^-), and clusters A and B are mix-labeled (Ω_m). We select not only the centers of the clusters but also all the data of mix-labeled clusters as training data in the first SVM classification stage. If we denote the set of the centers of the clusters in Ω^+ and Ω^- by C^+ and C^- respectively, i.e.,

$C^+ = \{\cup C_i \mid y = +1\}$ positive labeled centers

$C^- = \{\cup C_i \mid y = -1\}$ negative labeled centers

then the selected data set which will be used in the first stage SVM classification is the union of C^+, C^- and Ω_m, i.e., $C^+ \cup C^- \cup \Omega_m$. In Figure 4 (b), the red centers belong to C^+, and the green centers belong to C^-. It is clear that the data in Figure 4 (b) are all cluster centers except the data in mix-labeled clusters A and B. Figure 5 (a) and (b) show the similar results of random selection. Note that the training data set via random selection is composed of only selected data without center and mix-labeled cluster identification process.

In fact, data selection in our two-stage classification can be based on any clustering result regardless of clustering algorithm. However, we only use MEB and random selection.

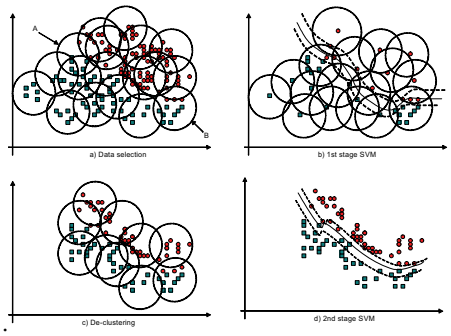

Figure 4. Two-stage classification via MEB clustering

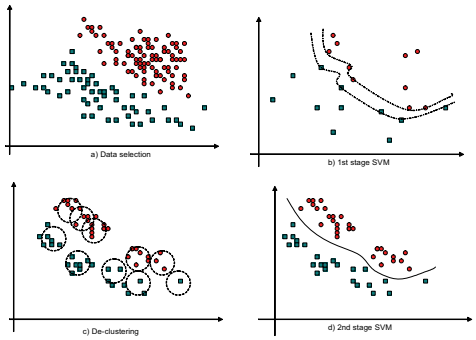

Figure 5. Two-stage classification via random selection

4.2. The first stage SVM classification

Let (X, Y) be the training patterns set,

$$X = \{x_1, \cdots, x_n\}, \quad Y = \{y_1, \cdots, y_n\}$$
$$y_i = \pm 1, \quad x_i = \left(x_{i1}, \ldots, x_{ip}\right)^T \in R^p$$

The training task of SVM classification is to find the optimal hyperplane from the input X and the output Y, which maximize the margin between the classes. i.e., training SVM yields to find an optimal hyperplane or to solve the following quadratic programming problem (primal problem),

$$\min_{w,b} J(w) = \tfrac{1}{2} w^T w + c \sum_{k=1}^{n} \xi_k \qquad (8)$$
$$\text{subject} : \; y_k \left[w^T \varphi(x_k) + b\right] \geq 1 - \xi_k$$

where ξ_k is slack variables to tolerate mis-classifications $\xi_k > 0, \quad k = 1 \cdots n, \quad c$ ⑩ 0 , w_k is the distance from x_k to the hyperplane $\left[w^T \varphi(x_k) + b\right] = 0$, $\varphi(x_k)$ is a nonlinear function. The kernel which satisfies the Mercer condition [10] is $K(x_k, x_i) = \varphi(x_k)^T \varphi(x_i)$. (8) is equivalent to the

following quadratic programming problem which is a dual problem with the Lagrangian multipliers $\alpha_k \geq 0$,

$$\max_{\alpha} J(\alpha) = -\tfrac{1}{2}\sum_{k,j=1}^{n} y_k y_j K(x_k, x_j)\alpha_k\alpha_j + \sum_{k=1}^{n}\alpha_k \qquad (9)$$
$$\text{subject}: \sum_{k=1}^{n}\alpha_k y_k = 0,\ 0 \leq \alpha_k \leq c$$

Many solutions of (9) are zero, *i.e.*, $\alpha_k = 0$, so the solution vector is sparse, the sum is taken only over the non-zero α_k. The x_i which corresponds to nonzero α_i is called a support vector. Let V be the index set of support vectors, then the optimal hyperplane is

$$\sum_{k\in V}\alpha_k y_k K(x_k, x_j) + b = 0 \qquad (10)$$

The resulting classifier is

$$y(x) = sign\left[\sum_{k\in V}\alpha_k y_k K(x_k, x) + b\right]$$

where b is determined by Kuhn-Tucker conditions.

Sequential minimal optimization (SMO) breaks the large QP problem into a series of smallest possible QP problems [27]. These small QP problems can be solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The memory required by SMO is linear in the training set size, which allows SMO to handle very large training sets [17]. A requirement in (9) is $\sum_{i=1}^{l}\alpha_i y_i = 0,$ it is enforced throughout the iterations and implies that the smallest number of multipliers can be optimized at each step is two. At each step SMO chooses two elements α_i and α_j to jointly optimize, it finds the optimal values for these two parameters while all others are fixed. The choice of the two points is determined by a heuristic algorithm, the optimization of the two multipliers is performed analytically. Experimentally the performance of SMO is very good, despite needing more iterations to converge. Each iteration uses few operations such that the algorithm exhibits an overall speedup. Besides convergence time, SMO has other important features, such as, it does not need to store the kernel matrix in memory, and it is fairly easy to implement [27].

In the first stage classification, we use SVM with SMO algorithm to get the decision hyperplane

$$\sum_{k\in V_1} y_k\alpha^*_{1,k}K(x_k, x) + b^*_1 = 0$$

with training data set $C^+ \cup C^- \cup \Omega_m$ obtained in section data selection, V_1 is the support vector set. Figure 4 (b) and Figure 5 (b) show the results of the first stage SVM classification via MEB and random selection respectively.

4.3. De-clustering

Note that, the original data set is reduced significantly after data selection, and the training data set in the first stage SVM classification is only a small percentage of the original data. This may affect the classification precision, i.e., the obtained decision hyperplane cannot be precise enough. However, at least it gives us a reference on selecting training data for the second stage SVM. On the other hand, we like to make a classification using as many useful data as possible. But, some useful data has not been selected during the data selection stage. So, a natural idea is to recover those data which are near to the support vectors, and use the recovered data to train the second stage SVM.

By the sparse property of SVM, the data which are not support vectors do not contribute to the optimal hyperplane. The input data which are far away from the decision hyperplane can be eliminated, meanwhile the data sets which are possibly support vectors should be used. Therefore, we propose to recover the data into training data set by including the data in the clusters whose centers are support vectors of the first stage SVM, we call this process *de-clustering*. Thus, more original data near the hyperplane can be found through the de-clustering. Figure 4 (c) illustrates the de-clustering process via MEB and random selection.

Note that the data selected by random selection don´t belong to clusters, so we need to give a technique to make clusters for them. Here we propose to draw circles with certain radius by taking select data as centers, see Figure f5 (c). It is clear that how many data will be recovered depends on the radius selection and an extreme case is that all original data are recovered if the radius is large enough. We intend to recover a data set size near to the training data set size suitable for the SVM algorithm we are using. In this paper, we use the maxima margin of the first-stage SVM as the radius

$$r = \frac{2}{\|w^*\|}$$

where $w^* = \sum_{k \in V_1} \alpha_k^* y_k x_k$, α_k^* is the solution of (9). The recovered data set is $\cup_{c_i} \{A(c_i, r)\}$, where $i \in V_1$ and V_1 is the support vector set of the first stage classification.

The de-clustering process not only overcomes the drawback that only small part of the original data near the support vectors are trained, but also helps to improve the accuracy since data near by the hyperplane are used efficiently in the second stage SVM.

4.4. The second stage SVM classification

Taking the recovered data as new training data set, we use again SVM classification with SMO algorithm to get the final decision hyperplane

$$\sum_{k \in V_2} y_k \alpha_{2,k}^* K(x_k, x) + b_2^* = 0 \qquad (12)$$

where V_2 is the index set of the support vectors in the second stage classification. Generally, the hyperplane (10) is close to the hyperplane (12).

In the second stage SVM, the training data set is $\cup_{c_i} \{\Omega_i\} \cup \Omega_m$, where $\cup_{c_i} \{\Omega_i\}$ consists of the data of the clusters whose centers are support vectors of the first stage classification, $i \in V_1$, and Ω_m is the data of mix-labeled clusters. For random selection clustering case, Ω_m is empty.

Figure 4 (d) and Figure 5 (d) illustrate the second stage classification results via MEB and random selection respectively. One can observe that the two hyperplanes in Figure 4 (d) and Figure 4 (b) (also Figure 5 (d) and Figure 5 (b)) are different but similar.

5. Performance analysis

In this Section, we show the space and time complexities. Is clear that without a decomposition method, is almost impossible for normal SVM to obtain the optimal hyperplane when the training data size n is huge. Is very difficult to analyze the complexity of SVM algorithm precisely. This operation involves multiplication of matrices of size n, which has complexity $O(n^{2.3})$ and $O(n^{2.83})$ at worst. In the following, we assume that a QP implementation of each stage of SVM takes $O(n^3)$ time and $O(n^2)$ space for n input data.

5.1. Memory space

In the clustering stage, the input data of p dimensions are loaded into the memory. The data type is float, so the data size is 4 bytes. If we use normal SVM classification, the memory size for the input data should be $4(n \times p)^2$ at worst, on the other hand the clustering data is $4(n \times p)$, Is clear that, in modern SVM implementations, it is not needed that the entire kernel matrix be put into the memory simultaneously.

In the first stage SVM classification, the training data space is $4[(l+m) \times p]^2$, where l is the number of the clusters, m is the number of the elements in the mixed clusters.

In the second stage SVM classification, the training data size is $4\left[\left(\sum_{i=1}^{l} n_i + m\right) \times p\right]^2$, where n_i is the number of the elements in the clusters whose centers are support vectors.

The total storage space of our two stage classification via FCM and MEB clustering is

$$4(n \times p) + 4p^2 \left[\left(\sum_{i=1}^{l} n_i + m \right)^2 + (l+m)^2 \right]. \qquad (13)$$

which consists of the storage of the clustering algorithm and both stages SVM.

The total storage space of our approach via random selection clustering (where $m = 0$) is

$$4(n \times p) + 4p^2 \left[\left(\sum_{i=1}^{l} n_i \right)^2 + l^2 \right] \qquad (14)$$

When n is large (large data sets), n_i, m and $l << n$, the memory spaces by (13) and (14) of our approach is much smaller than $4(n \times p)^2$ which is needed by a normal SVM classification

5.2. Algorithm complexity

The complexity of this algorithm can be approximated as follows. The complexity of the clustering of FCM is $O(ncp)$ where n is the number of input data, c is the number of clusters and p is the dimension of the data points. The approximate complexity of the two SVM training is $O\left[(l+m)^3\right] + O\left[\left(\sum_{i=1}^{l} n_i + m\right)^3\right]$ The total complexity of two stage classification via FCM is

$$O(ncp) + O\left[(l+m)^3\right] + O\left[\left(\sum_{i=1}^{l} n_i + m\right)^3\right] \qquad (15)$$

where l is the total number of cluster, n_i is the number of the elements in the i th clusters whose centers are support vectors, m is the number of the elements in the mixed labeled clusters. The choice of l is very important to obtain fast convergence. When n is large, the cost for each iteration will be high, and a smaller l needs more iterations, hence, and will converge more slowly.

The complexity of the MEB clustering is $O(\frac{1}{\varepsilon})$ as proved. The total complexity of two stage classification via MEB is

$$O(np^2)+O\left[(l+m)^3\right]+O\left[\left(\sum_{i=1}^{l} n_i+m\right)^3\right] \qquad (16)$$

Obviously, (25) is much smaller than the complexity of a normal SVM $O(n^3)$.

5.3. Training time

The training time of the approach proposed in this paper includes two parts: clustering algorithm and two SVMs. The training time of FCM is

$$T_f = C_n^2 \times c_f = \frac{n\times(n-1)}{2} c_f$$

where c_f is the cost of the evaluating the fuzzy distance, n is the number of the training data. The training time of MEB is

$$T_e = \pi \times l \times n \times c_e$$

where π is the times of $(1+\varepsilon)-$ approximation, l is number of clusters, c_e is the cost of the evaluating the Euclidian distance.

The training time of SVM can be calculated easily as follows. We assume that the major computational cost comes from multiplication operators (SMO) without considering the cost of the other operators such as memory access. The growing rate of the probability of support vectors is assumed to be constant.

Let $n_m(t)$ be the number of non-support vector points at time t. The probability of the number of support vectors at time t is $F(t)$ which satisfies

$$F(t) = \frac{l+m-n_m(t)}{l+m}$$
$$n_m(t) = (l+m)\left[1-F(t)\right]$$

where l is the number of the clusters centers, m is the number of the elements in the mixed labeled clusters. The growth rate of the number of support vectors (or decreasing rate of the number of non-support vectors) is

$$h(t) = -\frac{d\left[n_m(t)\right]}{dt}\frac{1}{n_m(t)} = \frac{\dot{F}(t)}{(l+m)\left[1-F(t)\right]}$$

Since the growth rate is constant $h(t)=\lambda,$ the solution of the following ODE

$$\dot{F}(t) = -\lambda(l+m)F(t)+\lambda(l+m)$$

with $F(0)=0$ is

$$F(t) = 1-e^{-\lambda(l+m)t}$$

The support vector number of the first stage SVM at time t is $n_{sv1}(t),$ it satisfies

$$n_{sv1}(t) = (l+m)F(t) = (l+m)(1-e^{-\lambda(l+m)t}) \quad (17)$$

Here $\lambda>0.$ It is monotonically increasing. The model (17) can be regarded as a growing model by the reliability theory [13].

The support vector number of the second stage SVM at time t is $n_{sv2}(t),$ it satisfies

$$n_{sv2}(t) = (l_1+m)(1-e^{-\lambda(l_1+m)t}),\ \ \lambda>0$$

where $l_1 = \sum_{i=1}^{l} n_i + m$

We define the final support vector number in each cluster in the first stage SVM is $h_i,\ \ i=1\cdots l.$ From (17) we know $h_i = (l+m)(1-e^{-\lambda(l+m)t}),$ so

$$t_i = \frac{1}{\lambda(l+m)}\ln\left(\frac{l+m}{l+m-h_i}\right)\ \ i=1\cdots l$$

We define c_1 as the cost of each multiplication operation for SMO. For each interactive step, the main cost is $4(l+m)c_1.$ The cost of the optimization in the first stage classification is

$$T_{op}^{(1)} = \sum_{i=1}^{l} 4(l+m)c_1 \frac{1}{\lambda(l+m)} \ln\left(\frac{l+m}{l+m-h_i}\right)$$
$$= \sum_{i=1}^{l} \frac{4}{\lambda} c_1 \ln\left(1 + \frac{h_i}{l+m-h_i}\right)$$
$$\le \sum_{i=1}^{l} \frac{4c_1}{\lambda} \frac{h_i}{l+m-h_i} \le \frac{4c_1}{\lambda}(l+m-1)$$

In the second stage classification, it is

$$T_{op}^{(2)} = \sum_{i=1}^{l} 4(l_1+m)c_1 \frac{1}{\lambda(l_1+m)} \ln\left(\frac{l_1+m}{l_1+m-h_i}\right)$$
$$\le \frac{4c_1}{\lambda} \sum_{i=1}^{l} \frac{h_i}{l_1+m-h_i} \le \frac{4c_1}{\lambda}(l_1+m-1)$$

Another cost of computing is the calculation of kernels. We define c_2 be the cost of evaluating each element of K. In the first stage is

$$T_{\text{ker}}^{(1)} = (l+m)c_2, \quad T_{\text{ker}}^{(2)} = (l_1+m)c_2$$

The total time for the three approaches (fuzzy C-means (FCM), minimum enclosing ball (MEB) and random selection (RAN)) is

$$\text{FCM}: T_1 \le \frac{n\times(n-1)}{2} c_f + (l+l_1+2m-2)\left[\frac{4}{\lambda}c_1 + c_2\right]$$
$$\text{MEB}: T_2 \le \pi \times l \times n \times c_e + (l+l_1+2m-2)\left[\frac{4}{\lambda}c_1 + c_2\right]$$
$$\text{RAN}: T_3 \le (l+l_1+2m-2)\left[\frac{4}{\lambda}c_1 + c_2\right]$$

the training time of a classic SVM is

$$T \le (n-1)\left[\frac{4}{\lambda}c_1 + c_2\right]$$

For a large data set, $c_1 >> c_f$, $c_1 >> c_e$, $n >> (l+l_1+2m)$, so, $T >> T_i$, $i = 1,2,3$, the training time of SVM is much longer than that of our approach.

6. Experimental results

We use three examples to compare our algorithms with some other SVM classification methods. In order to clarify our basic idea, we consider a very simple case of classification and clustering in Example 1, and compare our two-stage SVM via MEB clustering (abbreviated as *MEB two-stage*) with two classic SVMs, SMO [27] and simple SVM [10]. Example 2 is a benchmark data set which was proposed in IJCNN 2001 [29][6]. With this data set, we compare our two-stage SVM via random selection (abbreviated as *RS two-stage*) with LIBSVM [6], simple SVM and our two-stage SVM via MEB clustering.

Example 3 is a RNA sequence data set. SVM has been proposed to detect RNA sequences [24] [36]. However, long training time is needed generally, so, it is impossible to repeat a normal SVM classification on the updated data set in an acceptable time when new data are included into the data set frequently or continuously. In Example 3, our two-stage SVM classifier is used to show how to deal with the training speed problem without loss of accuracy when data set is very large. Additionally, comparison are made between our two-stage SVM via MEB clustering and three classic SVMs, LIBSVM, SMO and simple SVM.

Example 1. We generate a set of data randomly in the range of $(0,40)$. The data set has two dimensions $X_i = [x_{i,1}, x_{i,2}]$. The output is decided as follows:

$$y_i = \begin{cases} +1 & \text{if } WX_i + b > th \\ -1 & \text{otherwise} \end{cases} \quad (18)$$

where $W = [1.2, 2.3]^T$, $b = 10$, $th = 95$. In this way, the data set is linearly separable. We generate $500,000$ data randomly whose range and radius are the same as in [39]. The RBF kernel is chosen as

$$f(x,z) = \exp\left(-\frac{(x-z)^T(x-z)}{2r_{ck}^2}\right) \quad (19)$$

we choose $r_{ck} = r/5$. Experiments were done using MEB two-stage (SMO+MEB), Comparisons were made with SMO [27] and simple SVM [10]. Figure 6 shows the results on "running time" vs. "training data size". We can see that, SMO has less training time and higher accuracy than our approach for small data set (less than 10^4 data). But, for large data set (more than 10^4 data), the training time is dramatically increased in other SVMs while ours only increases a little. Figure 7 shows the results on testing accuracy vs. training data size. Although the classification accuracy cannot be improved significantly when data size is very large, it does not get worse, and the testing accuracy is still acceptable.

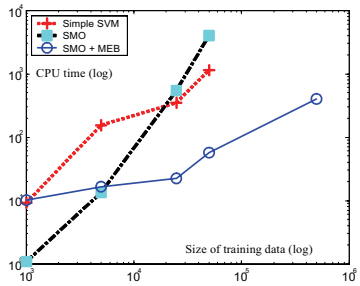

Figure 6. Example 1: running time vs training data size

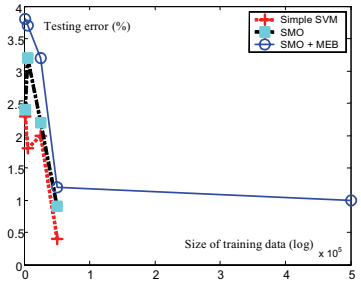

Figure 7. Example 1: testing accuracy vs training data size

Example 2.(IJCNN 2001) The data set is available at [29] and [6]. There are 49990 training data points and 91701 testing data points, each record has 22 attributes. The sizes of the training data we used are $1,000$, $5,000$, $12,500$, $25,000$, $37,500$ and $49,990$.

The following notations are used in the following tables.

"#" is the data size;

"t" is the training time of the whole classification which includes the time of clustering, the first stage SVM training, de-clustering and the second stage SVM training;

"Acc" is the accuracy;

" l " is the number of clusters used in the experiment;

"#MC" is the number of data in all mixed labeled clusters;

"TrD1" is the training data size of the first stage SVM classification;

"SV1" is the number of support vectors obtained in the first stage SVM;

"TrD2" is the training data size of the second stage SVM classification;

"SV2" is the number of support vectors obtained in the second stage SVM.

Table 1. Two-stage SVM classification results on IJCNN 2001 data set

IJCNN data set						
MEB two-stage						
#10^3	t	Acc	#MC	SV1	TrD2	SV2
1	22	93	39	125	199	51
5	31	95	84	128	467	115
12	38	97	105	105	733	160
25	59	97	147	143	1342	228
37	196	97	179	254	1399	267
50	462	98	201	748	1728	295

IJCNN data set						
RS two-stage						
#10^3	t	Acc	l	SV1	TrD2	SV2
1	4.53	92.8	350	86	473	175
5	8.73	95.1	400	109	541	180
12	13.31	97.3	450	127	673	193
25	25.98	97.4	500	118	712	287
37	45.30	97.7	1000	122	1693	358
50	78.08	98.0	2000	185	2370	430

Table 1. Two-stage SVM classification results on IJCNN 2001 data set

Table 1 shows our experiment results on different data size with MEB two-stage and RS two-stage. For example, in the experiment on 49990 data points we sectioned it into 2000 clusters using MEB clustering and random selection. In the first stage classification of MEB two-stage, we got 2200 training data after data selection where cluster centers and data in mixed labeled clusters were selected, and 748 support vectors were obtained. Following the de-clustering technique, 1728 data were recovered as training data for the second stage SVM, which included the cluster centers which are support vectors and clusters with mixed labels. In the second stage SVM, 295 support vectors were obtained. From Table 1, we can also see that MEB two-stage has a little better accuracy than RS two-stage, while its training time is longer than that of RS two-stage.

IJCNN data set						
	MEB two-stage		RS two-stage		SMO	
#	t	Acc	t	Acc	t	Acc
1000	22.34	93.1	4.53	92.8	1.672	96.1
5000	31.28	95.7	8.73	95.1	89.87	97.2
12500	38.41	97.5	13.31	97.3	—	—
25000	59.18	97.5	25.98	97.4	—	—
37500	196.6	97.7	45.30	97.7	—	—
49990	462.4	98.2	78.08	98.0	—	—

IJCNN data set			
Simple SVM		LIBSVM	
t	Acc	t	Acc
2.1	94.3	0.5	93.1
4.6	96.3	7.5	96.3
182.6	96.1	47.6	98.2
3823	97.2	177.8	98.6
10872	97.7	394.1	98.8
20491	98.2	730.7	98.8

Table 2. Training time and accuracy of different algorithms on IJCNN 2001 data set

Table 2 shows the comparison results on training time and accuracy between our two-stage classification and some other SVM algorithms including SMO, simple SVM and LIBSVM. For example, to classify 1000 data, LIBSVM is the fastest, and SMO has the best accuracy, our two approaches are not better than them, although the time and accuracy are still acceptable. However, to classify 49990 data, Simple SVM and SMO have no better accuracy than the others, but their training time is tremendous longer. Comparing to our two approaches, LIBSVM takes almost double training time of MEB two-stage, and almost 10 times of the time of RS two-stage, although it has the same accuracy as ours. This experiment implies that our approach has great advantage on large data sets since it can reach the same accuracy as the other algorithm can in a very short training time.

Example 3 (**RNA Data set)** The RNA data set is available at *http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1570369#top* from Supplementary Material (additional file 7). The data set consists of 23605 data points, each record has 8 attributes with continuous values between 0 to 1. The data set contains 3919 ncRNAs and 19686 negative sequences. We used sizes $2,500$, $10,000$ and $23,605$ in our experiments.

Experiments were done using MEB two-stage, RS two-stage, SMO, LIBSVM and simple SVM. Table 3 shows our experiment results on different data size with MEB two-stage and RS two-stage. Table 4 shows the comparisons between our approach and other algorithms. Similarly to the conclusion of Example 2, our approach can reach the same accuracy as the others within a much shorter training time, while the data size is large.

RNA sequence data set						
MEB two-stage						
#	t	Acc	*l*	SV1	RD	SV2
23605	174.5	88.4	1500	278	1307	416

RNA sequence data set						
RS two-stage						
#	t	Acc	*l*	SV1	RD	SV2
23605	65.7	88.3	1500	257	1275	381

Table 3. Two-stage SVM classification results on RNA sequence data set

RNA sequence data set				
	MEB two-stage		RS two-stage	
#	t	Acc	t	Acc
2500	15.56	87.3	11.21	87.1
10000	69.26	88.2	30.22	87.8
23605	174.5	88.4	65.7	88.3

RNA sequence data set					
LIBSVM		SMO		Simple SVM	
t	Acc	t	Acc	t	Acc
3.06	87.4	4.20	87.7	561.3	88.1
48.38	88.2	1122.5	89.6	—	—
298.3	88.6	—	—	—	—

Table 4. Training time and accuracy comparison between different algorithms on RNA sequence data set.

7. Conclusions and discussions

In this paper, we have presented a classification method which have applied reduction techniques in order to speed up the training time of SVM. In order to solve the trade-off problem between SVM classification accuracy and training time for large data sets, a two-stage SVM classification approach is proposed. To reduce SVM training time for large data sets, two fast clustering methods are introduced to select training data. From our experiments, we conclude our approach as following:

1. Our two stage classification approach is convenient for large data sets. But not good for small data sets since

data reduction may affect a lot on the accuracy when the data set size is small.

2. Generally our approach can have almost the same accuracy as other SVM classifiers when data set is large, while its training time is super short.
3. .Random selection is faster than MEB and other clustering based data selection because it does not partition data, but it restricts that the original data set should be relatively uniform.
4. Two stage SVM classifier via MEB clustering may be the best method for general use comparing with other SVMs including two stage SVM via random selection.

The accuracy decrease is caused by the lost of support vectors, several possible solutions may avoid it.

1. Increasing cluster number may increase the training data for the first stage SVM classification, so more support vectors may be obtained.
2. The relations between the clustering and the support vectors play an important role for classification accuracy. It may be solved from the point of data density.
3. .Since clustering is unsupervised, some useful information (support vectors) of original data set may be lost. New clustering approaches which use label information may improve the accuracy. For example, the random selection of this paper is carried on in the two classes (labels ±1) independently. This kind of method may be extended furthermore to more general semi-supervied clustering.

References

1. M. Awad L. Khan, F.Bastani and I. L.Yen , An Effective support vector machine(SVMs) Performance Using Hierarchical Clustering, Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04),Pages: 663- 667, 2004
2. G. Babu and M. Murty, A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm, Pattern Recognit. Lett., vol. 14, no. 10, pp. 763--769, 1993.
3. M.Badoiu, S.Har-Peled and P.Indyk. Approximate clustering via core-sets. Proceedings of the 34th Symposium on Theory of Computing, 2002.
4. P.Burman, A Comparative Study of Ordinary Cross-Validation, v-Fold Cross-Validation and the Repeated Learning-Testing Methods,Biometrika, Vol. 76, No. 3, pp. 503-514, 1989
5. J. Cervantes, X.Li and W.Yu, Support Vector Machine Classification Based on Fuzzy Clustering for Large Data Sets, MICAI 2006: Advances in Artificial Intelligence, Srpinger-Verlgag, Lecture Notes in Computer Science (LNCS), vol. 4293, 572-582, 2006
6. C-C.Chang and C-J.Lin, LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.
7. Chen J.-H. and Chen Ch.-S., Reducing SVM classification time using multiple mirror classifiers, Systems, Man, and Cybernetics, Part B, IEEE Transactions on, April 2004, Vol: 34, Issue: 2, pp: 1173-1183
8. P-H.Chen, R-E.Fan and C-J.Lin, A Study on SMO-Type Decomposition Methods for Support Vector Machines, IEEE Trans. Neural Networks, Vol.17, No.4, 893-908, 2006.
9. R.Collobert and S.Bengio, SVMTorch: Support vector machines for large regression problems, Journal of Machine Learning Research, Vol.1, 143-160, 2001.
10. N.Cristianini, J.Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods , Cambridge University Press, 2000.
11. J-X.Dong,A.Krzyzak,and C.Y.Suen, Fast SVM Training Algorithm with Decomposition on Very Large Data Sets, IEEE Trans. Pattern Analysis and Machine Intelligence, vol.27, no.4, pp.603-618, 2005
12. G.Folino,C.Pizzuti and G.Spezzano, GP Ensembles for Large-Scale Data Classification, IEEE Trans. Evol. Comput., vol.10, no.5, pp.604--616, 2006.
13. B.V.Gnedenko, Y.K.Belyayev, and A.D.Solovyev, Mathematical Methods of Reliability Theory, NewYork: Academic Press, 1969
14. M. Girolami, Mercer kernel based clustering in feature space, IEEE Trans. Neural Networks, vol. 13, no. 3, pp. 780--784, May 2002.
15. T.Hastie and R.Tibshirani, Discriminant adaptive nearest neighbor classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 18, no 6, 607-616, 1996
16. G.B. Huang, K.Z. Mao, C.K. Siew and D.-S. Huang, Fast Modular Network Implementation for Support Vector Machines", IEEE Trans. on Neural Networks, 2006

17. T.Joachims, Making large-scale support vector machine learning practice, Advances in Kernel Methods: Support Vector Machine. MIT Press, Cambridge, MA 1998.
18. Khan L., Awad M and Thuraisingham B., A new intrusion detection system using support vector machines and hierarchical clustering, The VLDB Journal, Vol. 16, No. 4, pp 507-521, 2007.
19. S-W.Kim and B.J.Oommen, Enhancing Prototype Reduction Schemes with Recursion: A Method Applicable for "Large" Data Sets, IEEE Trans. Syst., Man, Cybern. B, Vol.34, No.3, 1184-1397, 2004.
20. P. Kumar, J. S. B. Mitchell, and A. Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. ACM Journal of Experimental Algorithmics, 8, January 2003.
21. C-T.Lin,C-M.Yeh,S-F.Liang,J-F.Chung and N.Kumar, Support-Vector-Based Fuzzy Neural Network for Pattern Classification, IEEE Trans. Fuzzy Syst., Vol.14, No.1, 31-41, 2006.
22. M.E.Mavroforakis and S.Theodoridis, A Geometric Approach to Support Vector Machine(SVM) Classification, IEEE Trans. Neural Networks, Vol.17, No.3, 671-682, 2006.
23. O. L. Mangasarian and D. R. Musicant,Successive overrelaxation for support vector machines, IEEE Trans. on Neural Networks, vol.~10, pp.~1032-1037, 1999.
24. W.S. Noble, S. Kuehn, R.Thurman, M. Yu and J.Stamatoyannopoulos, Predicting the in vivo signature of human gene regulatory sequences, by . Bioinformatics, Vol.21, No.1, i338- 343, 2005.
25. A.Nurnberger, W.Pedrycz , R.Kruse, Data mining tasks and methods: Classification: neural network approaches, Oxford University Press, Inc., New York, NY, 2002
26. N. Pal and J. Bezdek, On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Syst., vol. 3, no. 3, pp. 370--379, Aug. 1995.
27. J.Platt, Fast Training of support vector machine using sequential minimal optimization, Advances in Kernel Methods: support vector machine, MIT Press, Cambridge, MA, 1998.
28. C.Pizzuti and D.Talia, P-Auto Class: Scalable Parallel Clustering for Mining Large Data Sets, IEEE Trans. Knowledge and Data Eng., vol.15, no.3, pp.629-641, 2003.
29. D.Prokhorov. IJCNN 2001 neural network competition, Ford Research Laboratory, http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf, 2001.
30. R Safavian, D Landgrebe , A survey of decision tree classifier methodology, IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no3, pp. 660-674, 1991.
31. SAKETHA NATH J., BHATTACHARYYA C., MURTY M. N., Clustering based large margin classification : A scalable approach using SOCP formulation, Proceedings of the Twelfth ACM SIGKDD international conference on knowledge discovery and data mining, August 20-23, 2006, Philadelphia, PA, USA
32. G.Schohn and D.Cohn , Less is more : Active Learning with support vector machines, Proc. 17th Int. Conf. Machine Learning, Stanford, CA, 2000.
33. L.Shih, D.M.Rennie, Y.Chang and D.R.Karger, Text Bundling: Statistics-based Data Reduction, Proc of the Twentieth Int. Conf. on Machine Learning (ICML-2003), Washington DC, 2003.
34. P.Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, Pearson Addison Wesley, May, 2005
35. V.Tresp, A Bayesian Committee Machine, Neural Computation, vol.12, no.11, pp.2719-2741, 2000.
36. Uzilov AV, Keegan JM, Mathews DH, Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change, BMC Bioinformatics, Vol.173, No.7, 2006
37. Y.S. Xia and J. Wang, A One-layer Recurrent Neural Network for Support Vector Machine Learning, IEEE Transactions on Systems, Man and Cybernetics - Part B, pp. 1261-1269, 2004.
38. R.Xu and D.WunschII, Survey of Clustering Algorithms, IEEE Trans. Neural Networks, Vol.16, No.3, 645-678, 2005.
39. H.Yu , J.Yang and J.Han , Classifying Large Data Sets Using SVMs with Hierarchical Clusters, Proc. of the 9th ACM SIGKDD 2003, Washington, DC, USA, 2003.