Two-Stage SVM Classification for Large Data Sets via Randomly Reducing and Recovering Training Data

Xiaoou Li, Jair Cervantes, and Wen Yu

Abstract—Despite of good theoretic foundations and high classification accuracy of support vector machine (SVM), normal SVM is not suitable for classification of large data sets, because the training complexity of SVM is very high. This paper presents a novel two stages SVM classification approach for large data sets by randomly selecting training data. The first stage SVM classification gets a sketch of support vector distribution. Then the neighbors of these support vectors in original data set are used as training data for the second stage SVM classification. Experimental results demonstrate that our approach have good classification accuracy while the training is significantly faster than other SVM classifiers.

I. INTRODUCTION
There are a number of 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 these techniques, support vector machine (SVM) is one of the best-known technique for its computational complexity. Many researchers have tried to find possible methods to apply SVM classification for large data set. Generally, these methods can be divided into two types: 1) modify SVM so that it could be applied on large data sets, and 2) reduce a large data set to some smaller sets so that the normal SVM could be applied.

For the first type, a standard projected conjugate gradient (PCG) chunking algorithm can scale somewhere between linear and cubic in the training set size [7][13]. Sequential Minimal Optimization (SMO) is a fast method to train SVM [19][6]. Training SVM requires the solution of QP optimization problem, SMO breaks this large QP problem into a series of smallest possible QP problems, it is faster than PCG chunking. [9] 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 subproblems. A recursive and computational superior mechanism referred as adaptive recursive partitioning was proposed in [14], 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 [10]. Neural networks technique can also be applied for SVM to simplify the training process [12].

For the second type, clustering has been proved to be an effective method to reduce data set size. For examples, hierarchical clustering [24] [1], k-means cluster [3] and parallel clustering [6]. Clustering based methods can reduce the computations burden of SVM, but they are very complex for large data set. Random selection is to select data in such way that the learning is maximized by the data. However, it could over-simplify the training data set and lose the benefit of SVM. Rocchio bundling is a statistics-based data reduction method [21]. Another approach is to apply the Bayesian committee machine to train SVM on large data sets [22] where the data is divided into m subsets of the same size, and m models are derived from the individual sets. But it has higher error rate then normal SVM and the sparse property does not hold.

The goal of clustering is to separate a 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 have been developed, e.g., squared error-based k-means [2], fuzzy C-means [18], kernel-base clustering [11]. For these clustering, the optimal number of clusters should be predefined which involves more computational cost than clustering itself [23].

However, clustering algorithms are generally time-consuming. In fact, the majority of the clustering calculation on the whole large data set is not useful for finding few support vectors. So, it is possible to find a way to reduce the data set without making calculation on all data, and the trade-off is the classification precision. Random selection may be a such method.

In this paper, we propose a new random selection technique to reduce data set in place of clustering. Firstly, we select a predefined number of data from the original data set using our random selection method, and these selected data are used in the first stage SVM. Secondly, the obtained support vectors of the first stage SVM are used to select
The resulting classifier is

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

where \( b \) is determined by Kuhn-Tucker conditions,

\[
\begin{align*}
\frac{\partial L}{\partial w} &= 0, \quad w = \sum_{k=1}^{n} \alpha_k y_k \varphi(x_k) \\
\frac{\partial L}{\partial \alpha_k} &= 0, \quad \sum_{k=1}^{n} \alpha_k y_k = 0 \\
\frac{\partial L}{\partial b} &= 0, \quad c - \alpha_k - \nu_k = 0 (\alpha_k - c \geq 0) \\
&\alpha_k \{ y_k [w^T \varphi(x_k) + b] - 1 + \xi_k \} = 0
\end{align*}
\]

In primal problem (1) the size of \( w \) is fixed which is independent of the number of data points. In dual problem (2) the solution vector \( \alpha \) grows with the number of data points \( n \). For high dimensional input space it is better to solve the dual problem, while for large data sets it might be advantageous to solve the primal problem.

Sequential minimal optimization (SMO) breaks the large QP problem into a series of smallest possible QP problems [19]. 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 [13]. A requirement in (3) is \( \sum_{k=1}^{n} \alpha_k y_k = 0 \), 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 \( \alpha_i \) and \( \alpha_j \) 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 [19].

III. RANDOM SELECTION ALGORITHM

Many efforts have been made on producing “random” sequences of integers. In recent years, algorithms have been proposed to measure how random a sequence is. However, most generators were subjected to rigorous statistical testing where factors including speed, setup time, length of the compiled code, machine independence, range of the set of applications, simplicity and readability, should be considered. It is impossible to obtain a perfect uniform random number generator.

In this paper, we propose a random selection algorithm to select a small percentage of data from original large data set, so that SVM algorithm can be applied directly. We assume that \((X,Y)\) be the training patterns set, where \(X = \{x_1, x_2, \ldots, x_n\}\) is the input data set, \( x_i \) can be represented by a vector of \( p \) dimension, i.e., \( x_i = (x_{i1}, \ldots, x_{ip})^T \), \( Y = \{y_1, y_2, \ldots, y_n\} \) is the label set, label \( y_i \in \{-1, 1\} \). Our objective is to select a sample set \( C = \{c_1, c_2, \ldots, c_l\} \) \( \ll n \) from \( X \). Note that \( l \) need to be predefined and should be
an integer that is suitable to be training data size for normal SVM algorithms.

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

\[
X^+ = [x^+(1), \ldots, x^+(q)]
\]

\[
X^- = [x^-(1), \ldots, x^-(m)]
\]

\[
Y^+ = [y^+(1), \ldots, y^+(q)] = [1, \ldots, 1]
\]

\[
Y^- = [y^-(1), \ldots, y^-(m)] = [-1, \ldots, -1]
\]

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

Secondly, we select sample data by sampling the subsets \( X^+ \) and \( X^- \) independently. Here we use swapping method during selection process, see Fig. 1, and 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 \( x^+(m) \) (or \( x^-(m) \)), i.e., \( \text{Swap}(c_1, x^+(m)) \) (or \( \text{Swap}(c_1, x^-(m)) \)). The second sample \( c_2 \) is selected from the remaining data \( \{x^+(1), \ldots, x^+(q-1)\} \), then it is exchanged with the last data \( x^+(q-1) \) (or \( x^-(m-1) \)), i.e., \( \text{Swap}(c_2, x^+(q-1)) \) (or \( \text{Swap}(c_2, x^-(m-1)) \)), and so on, until the required number of data for \( X^+ \) (or \( X^- \)) are selected. Generally, we may select \( l/2 \) sample 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 sample for each label according to their distribution on original data or as wanted. The proportion may be defined in various different ways which are out of the range of our investigation in this paper.

This swapping method can be also regarded as generating the first \( l \) entries in a random permutation. Above random selection can be formed as the following algorithm.

For \( i := n \) to \( n - l + 1 \) Do
Generate a uniform \([0, 1]\) random variate \( X \).

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

\[
X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\}
\]

\[
y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p
\]

IV. TWO-STAGE SVM CLASSIFICATION
Let \((X, Y)\) be the training patterns set,
\[ X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\} \]
\[ y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p \]

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. By the sparse property of SVM, the data which are not support vectors will not contribute the optimal hyperplane. The input data sets which are far away from the decision hyperplane should be eliminated, meanwhile the data sets which are possibly support vectors should be used.

Our two stages SVM Classification can be summarized into 4 steps which is shown in Fig. 2: 1) data selection via random selection technique proposed in Section III, 2) the first stage SVM classification, 3) data recovery, 4) the second stage SVM classification. The following subsections will give a detailed explanation on each step.

A. Selecting training data
We denote the data set of selected samples as \( C \). We may divide \( C \) into two subsets, one is positive labeled, and the other is negative labeled, i.e.,

\[
C^+ = \{ UC_i | y = +1 \} \text{ positive labeled data} \\
C^- = \{ UC_i | y = -1 \} \text{ negative labeled data}
\]

Fig. 3 (a) shows the original data, where the red circles represent positive labeled data, and the green squares represent negative labeled data. After random selection, the selected sample data are shown in Fig. 3 (b). Then, \( C^+ \) consists of those data which are represented by red circles in Fig. 3 (b), and \( C^- \) consists of those data which are represented by green squares in 3 (b). Only these selected data will be used as training data in the first stage SVM classification.
B. The first stage SVM classification

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

$$\sum_{k \in V_1} y_k \alpha_{1,k}^* K(x_k, x) + b_1^* = 0$$  \hspace{1cm} (5)$$

where $V_1$ is the index set of the support vectors in the first stage. Here, the training data set is $C^+ \cup C^-$ which has been obtained by random selection in the last sub section. Fig. 3(b) and 4(a) illustrate the training data set and the support vectors obtained in the first stage SVM classification, respectively.

C. Data recovery

Note that, the original data set is reduced significantly by random selection. In the first stage SVM classification, the training data set 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, a refined process is necessary. However, at least it gives us a reference on which data can be eliminated, the data far from the hyperplane can be removed for further classification stage. On the other hand, we like to make a classification using useful data as many as possible. But, some useful data hasn’t been selected during the random selection stage. So, a natural idea is to recover those data which are near to the support vectors, and use the recovered data to make classification again.

We propose to recover the data by using the support vectors as centers and circling their neighbors in original data set with certain radius, Fig. 4 illustrates the recovery process. The data in circles in Fig. 4 (b) are the recovered data. Of course, how many data will be recovered depends on the radius selection. We intend to recover a data set size near to the training data set size suitable for the SVM algorithm we are using. An extreme case is that all original data are recovered if the radius is large enough.

In this paper we use the margin of the first stage SVM to determine the radius of circles which are used to recover original data for the second stage SVM.

$$B_i(c_i, r), \text{ } c_i \in V, \text{ } \gamma_i = \frac{K}{\|u^*\|_2^2} = \frac{K}{\sum_{k \in V} y_i \alpha_{1,k}^* x_i}$$

And the recovered data set is $\bigcup_{c_i \in V} \{B_i(c_i, r)\}$.

D. 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

$$\sum_{k \in V_2} y_k \alpha_{2,k}^* K(x_k, x) + b_2^* = 0$$  \hspace{1cm} (6)$$

where $V_2$ is the index set of the support vectors in the second stage. Generally, the hyperplane (5) is close to the hyperplane (6).

Note that, the original data set is reduced significantly by random selection and the first stage SVM. The recovered data should consist of the most useful data from the SVM optimization viewpoint, because the data far away from the decision hyperplane have been eliminated. On the other hand, almost all original data near the hyperplane (5) are included by our data recovery technique, the hyperplane (6) should be an approximation of the hyperplane of an normal SMO with all original training data set. Fig. 5 illustrates the training data and hyperplane of the second stage SVM classification.
Fig. 5. The second stage SVM

E. Example

We use the IJCNN 2001 data set [20] as an example to compare our approach with other SVM classification methods including LIBSVM [5], SMO [19] and Simple SVM [8]. All the experiments were done with a PC Pentium-IV 1.8GHz host computer.

Example 1: IJCNN 2001 The data set is available at [20] and [5]. 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.

In our experiments, the RBF kernel is

\[
    f(x, z) = \exp \left( \frac{-(x - z)^T(x - z)}{2\sigma_k^2} \right)
\]

we choose \( \sigma_k = r/5 \).

The comparison results are shown in Table 1, where the notations are explained as follows:

"#" 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;

"#MC" is the number of clusters used in the experiment;

"TrD1" is the training data size of all mixed labeled clusters;

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

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

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

Table 1 shows our experiment results on different data size. For example, in the experiment on 49990 data points we sectioned it into 2000 clusters including one mixed cluster. In the first stage classification, we got 2200 training data after data selection including 2000 cluster centers and 201 data in a mixed labeled cluster. Then 185 support vectors were obtained. Following the data recovery process, 2370 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, 430 support vectors were obtained. The total training time is 78.08 seconds, and the accuracy reaches 97.7%.

Table 2 shows the comparison results on training time and accuracy between our classifier 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 with our approach, the training time of LIBSVM is almost 10 times of ours, on the other hand, the accuracies are almost the same. This experiment implies that our approach can reach the same accuracy as the other algorithm can in a very short training time when the data set is large.

V. Conclusion

A random selection method and a two stage SVM classification approach for large data sets are proposed in this work. Comparing with other SVM classifiers, our classifier via randomly reducing and recovering training data has the following characteristics:

1) The proposed approach is very practical for large data set classification. It can be used as a primitive classifier since it can be as fast as possible (depending on the
accuracy requirement) no matter how large is the data set size. Generally, SVM training time depends on the training data size. Our approach can overcome the QP problem using the proposed random selection method, so our total training data size is much smaller than that other SVM approaches, although we need twice classifications.

2) The training data size is reduced dramatically after random selection in our classification, however, the classification accuracy doesn’t decrease. This thanks to the data recovery process, since the recovered data used in the second stage SVM are all important samples, and thus we guarantee the improvement of accuracy. In our previous works [3][4][15], some experiments had been done on showing the accuracy difference between the first and second stage SVM of our approach.

3) Random selection is much faster than other clustering based data selection because it does not partition data. But, it restricts that the original data set should be relatively uniform. Although we select same number of data with each label, it is still possible that data are when data labels are not distributed relatively uniform. In this case, we need to consider the class distribution which is our on going work.

REFERENCES


