Support Vector Machines with Clustering for Training with Very Large Datasets

Theodoros Evgeniou
Technology Management
INSEAD
Bd de Constance, Fontainebleau 77300, France
theodoros.evgeniou@insead.fr
Massimiliano Pontil
Department of Information
Engineering, University of Siena, Siena, Italy
Department of Mathematics, City University of Hong Kong, Hong Kong
mapontil@cityu.edu.hk

Abstract

We present a method for training Support Vector Machines (SVM) classifiers with very large datasets. We present a clustering algorithm that can be used to preprocess standard training data and show how SVM can be simply extended to deal with clustered data, that is effectively a set of weighted examples. The algorithm computes large clusters for points which are far from the decision boundary and small cluster size for points near the boundary. This implies that when SVMs are trained on the preprocessed cluster data set nearly the same decision boundary is found but the computational time decreases significantly. When the input dimensionality of the data is not large, for example of the order of ten, the clustering algorithm can significantly decrease the effective number of training examples, which is a useful feature for training SVM on large data sets. Preliminary experimental results indicate the benefit of our approach.

1 Introduction

The recent development of a new family of learning machines, namely Support Vector Machines (SVM) [2, 3, 13], whose training can be formulated as optimizing a quadratic programming (QP) problem with box constraints, has lead to a series of fast optimization methods for this type of QP problems [7, 6, 10]. Formulating the training of a learning machine as a standard optimization theory problem is a direction for developing fast training methods. A lot of work has been done to speed up the Support Vector Machines. A chunking algorithm is proposed by Vapnik [3]. This is an iterative method that at each iteration solves a small subproblem. The chunking algorithm uses the support vectors found in previous batches for use in next batches [3]. Advanced working set algorithms use only a subset of the variables as a working set and optimize the problem with respect to them while freezing the others [6]. The extreme case of the advance working set is to use only two variables in the working set as in Sequential Minimum optimization [7]. In a recent paper, the problem is formulated by using a random rectangular kernel sub-matrix instead of using a full square one [4]. Column generation techniques and Bender's Decomposition can be also applied to this problem [1, 11].

A lot of work has been done in the direction of speeding up (scaling up) data mining methods. Provost and Kolluri [9] give a recent review of various approaches, mostly focusing on learning methods for finding rules and for training decision trees. The paper categorizes the approaches into three groups: designing fast algorithms, partitioning the data, and using relational representations.

In this paper we describe a new approach to training SVM with very large datasets which is different from the three main approaches discussed in [9]. The approach is based on the characteristic of SVM that only training data near the separating boundary (for classification) are important. We therefore present a clustering method that yields only a few clusters away from the separating boundary, and many clusters near the boundary. The approach is similar to that proposed by [5]. This way the important information from the training data - namely that of the training data near the separating boundary - is preserved while at the same time the size of the training set is effectively decreased. Once clusters have been found, they are represented has a set of weighted examples which and the SVM can be simply extended to deal with

such a kind of data.

The paper is organized as follows: in section 2 we first define the notation and present the setup of the problem. Section 3 discusses the proposed method. In section 4 we present experiments comparing the proposed method to that of standard SVM training using all the initial training data. Finally section 5 is summary and conclusions.

An earlier version of this paper appeared in [8].

2 Background and Notation

We are given a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each point $\mathbf{x}_i$ belong to $\mathbb{R}^n$ and $y_i \in \{-1, 1\}$ is a label that identifies the class of point $\mathbf{x}_i$. Our goal is to determine a function

$$f(\mathbf{x}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b, \tag{1}$$

where $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x}))$ is a vector field from $\mathbb{R}^n$ into the feature space $\mathbb{R}^m$.

Statistical Learning Theory [13] establishes that in order to obtain a function with controllable generalization capability we need minimize the Structural Risk. By so doing we control the VC-dimension and hopefully produce a function with good generalization error. SVMs are a practical implementation of this idea. They are based on the following Quadratic Programming Problem [3]:

Problem **P1**

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum \xi_i \\ \text{subject to} & y_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad i = 1, 2, \ldots, N \\ & \boldsymbol{\xi} \geq 0. \end{array}$$

The solution $\bar{\mathbf{w}}$ of this problem is given by equation:

$$\bar{\mathbf{w}} = \sum_{i=1}^{N} \bar{\alpha}_i y_i \boldsymbol{\phi}(\mathbf{x}_i), \tag{2}$$

with $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N)$ the solution of the *Dual Problem*:

Problem **P2**

Maximize	$-\frac{1}{2}\boldsymbol{\alpha}^\top D\boldsymbol{\alpha} + \sum \alpha_i$
subject to	$\sum y_i\alpha_i = 0$
	$0 \leq \alpha_i \leq C, \qquad i = 1, 2, \ldots, N$

where both sums are for $i = 1, 2, \ldots, N$, and D is a $N \times N$ matrix such that

$$D_{ij} = y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j). \tag{3}$$

Combining Equations (1) and (2), the solution of Problem P1 is given by:

$$\sum_{i=1}^{N} \bar{\alpha}_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) + \bar{b}.$$

The points for which $\bar{\alpha}_i > 0$ are called Support Vectors (SVs). In many application they form a small subset of training points.

For certain choices of the mapping $\boldsymbol{\phi}(\mathbf{x})$ we can sum the dot product in feature space so that:

$$\boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \tag{4}$$

Observe that the spatial complexity of Problem **P2** is N^2, independent from the dimensionality of the feature space. This observation allow us to extend the method in feature spaces of infinite dimension. In practice, because of the memory requirement, Problem **P2** presents severe limitation on the size of the training set.

The approach that we suggest is based on clustering. The idea consists of substituting the training set with a smaller set of new weighted points:

$$\{(\mathbf{t}_1, y_1, n_1), \ldots, (\mathbf{t}_g, y_g, n_g)\},$$

so that each point $\mathbf{t}_i$ represents a cluster of n_i points in the training set, g is the number of clusters and $\sum_{i=1}^{g} n_i = N$. Problem **P1** can be adjusted to separate the clustering set as:

Problem **P3**

Minimize	$\frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum n_i \xi_i$
subject to	$y_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{t}_i) + b) \geq 1 - \xi_i \quad i = 1, 2, \ldots, K$
	$\boldsymbol{\xi} \geq 0.$

where we have modified the second term in the objective function with a weighed sum to take in account the number of points represented by each cluster. The Dual Problem becomes:

$$\begin{array}{ll} \text{Problem } \mathbf{P4} & \\ \text{Maximize} & -\frac{1}{2}\boldsymbol{\alpha}^\top D\boldsymbol{\alpha} + \sum \alpha_i \\ \text{subject to} & \sum y_i\alpha_i = 0 \\ & 0 \le \alpha_i \le n_i C. \qquad i = 1, 2, \ldots, N \end{array}$$

where now sums are for $i = 1, 2, \ldots, g$, and D is a $g \times g$ matrix such that

$$D_{ij} = y_i y_j K(\mathbf{t}_i, \mathbf{t}_j). \tag{5}$$

We note that the only difference with respect Problem **P2** is given by the new upper bound $n_i C$ for the variable α_i.

3 The Clustering Algorithm

In this section we introduce a clustering algorithm that can be used as a preprocessing step to train SVM on large databases.

The algorithm computes the clusters of the set of points in each class. We illustrate it in the case of class 1. First we initialize the set A_1 of clusters of class 1: $A_1 = \{(\mathbf{x}_i, 1) \mid (\mathbf{x}_i, y_i) \in S,\ y_i = 1\}$.

1. Set $\ell = 0$.
2. For each point $(\mathbf{x}_k, n_k) \in A$:
   (a) Compute the the nearest point[1] in $A_1 \setminus \{(x_k, n_k)\}$ to $(\mathbf{x}_k, n_k)$. Let $(\mathbf{x}_j, n_j)$ be this point and d their distance, $d = \|x_k - x_j\|$.
   (b) Compute the center of mass of the two previous points, $\mathbf{v} = \frac{n_k\mathbf{x}_k + n_j\mathbf{x}_j}{n_k + n_j}$.
   (c) Compute the distance D between $\mathbf{v}$ and the nearest training point in class -1.
   (d) If $\frac{d}{D} < \gamma$ delete the two previous points from A_1, add $(\mathbf{v}, n_k + n_j)$ to A_1, and set $\ell = \ell + 1$.

[1] We use the Euclidean distance.

3. If $\ell > 0$ goto step 1, otherwise stop.

After the algorithm has stop, A_1 is the set of clusters of class 1. The same procedure is then repeated to compute the set of clusters of class 2, A_2.

Notice that the algorithm tends to produce large clusters of points which are far away from the boundary between the two classes and small clusters of points near the boundary. Thus, we expect that the points candidate to be support vectors are not strongly affected by the clustering procedure, while the others are heavily reduced. The parameter γ controls the meaning of "near". If the dimensionality of the input space is not too big (say $n \approx 10$), we expect that the algorithm can considerably improve the training time of the SVM.

4 Experimental results

In this section we compare the performance of standard SVM with that of the SVM trained on the weighted examples computed by the cluster algorithm discussed above. We performed three set experiments in $\mathbb{R}^2$ with linear and non-linear kernels and one experiment in $\mathbb{R}^8$ with non-linear kernels. The experiments were performed on a DEC alpha 430MHz. The SVM is computed by solving the primal Problem **P1** (**P3** for clustered data). We used a software [12] based on Interior Point Methods [14].

Linear kernel: In the first example we randomly generated two sets of points inside two circles of same radius r and Bayes region of about $.03\pi r^2$. We then trained a linear SVM with and without the clustering preprocessing step. Table 1 shows the number of clusters and the performance of the algorithm reported in previous section on training sets of increasing size. The parameter γ was set to 2.5.
Table 2 is a comparison between the SVM obtained by training on the full set of 5000 points and the SVM trained on the clusters.

Non-linear kernel: In the second example the points $\mathbf{x} = (x_1, x_2)$ are randomly generated in a circle of radius r and the boundary between the two classes is given by $x_2 = rsin(\pi\frac{x_1}{r})$. We now work with polynomial kernel of third degree. Table 3 and 4 show the results of clustering and recognition.

N	K	Time
5000	470	8.5 sec
10000	849	39.7 sec
20000	1537	137.4 sec

Table 1: Experiment 1 - Size of the clustering set and user time of the clustering algorithm for training sets of increasing size.

C	Full Set		Cluster Set	
3	23.3 sec	2.8%	.73 sec	2.8%
10	22.2 sec	2.8%	.80 sec	2.7%
100	25.0 sec	2.8%	.89 sec	2.7%

Table 2: Experiment 1 - User time and error rate for the full training set (5000 points) and the cluster set for different values of the regularization parameter. The error rate is obtained on a test set of 5000 points.

The setting of the third experiment is as in experiment 2 with the addition that now we assign point $\mathbf{x}$ to the first class if $x_2 > rsin(\pi\frac{x_1}{r}) + \epsilon$, where ϵ is a random number $\in (-\frac{1}{10}r, \frac{1}{10}r)$. We again work with polynomial kernel of third degree.

The results are shown in Table 6. Table 5 is as Table 2 and 4, while Table 7 shows that for this particular example the choice of the parameter γ does not modify appreciably the true OSH.

To better understand how the dependence on the reduction rate with respect the parameter γ, we run the clustering algorithm for different values of γ.

5 Conclusions

We have presented a direction of developing methods for fast training using hierarchical type training. Experimental results indicate that this is a promising direction for research. A number of questions remains open. From

N	K	Time
5000	397	9.4 sec
10000	745	41.8 sec
20000	911	145 sec

Table 3: Experiment 2 - Size of the clustering set and user time of the clustering algorithm for training sets of increasing size.

C	Full Set		Cluster Set	
3	286 sec	0.48%	1.54sec	0.44%
10	316 sec	0.32%	1.54sec	0.30%
100	364 sec	0.14%	1.82 sec	0.16%

Table 4: Experiment 2 - User time and error rate of the SVM trained on the full training set (20000 points) and on the cluster set for different values of the regularization parameter C. The error rate was computed on a test set of 5000 points.

the theoretical point of view, it is an open question how close the solution found from each of the proposed methods is to the global optimal one that a single SVM using all the training data would have found. Appropriate measures of distance between the solutions need to be defined: for example one can probably use the margin and the number of training errors, or the value of the cost function of the QP, or the set of support vectors, as such measures. From the practical point of view, two important questions is first how to adapt the proposed methods to other learning techniques, and second how to design new hierarchical methods that are more efficient without loss in predictive performance. Finally, a theoretical framework for designing and studying this type of hierarchical training methods can be valuable towards the direction of developing new methods.

C	Full Set		Cluster Set	
3	327 sec	3.16%	4.51 sec	3.16%
10	385 sec	3.15%	4.95 sec	3.15%
100	356 sec	3.15%	4.82 sec	3.15%

Table 5: Experiment 3 - User time and error rate for the full training set of 20000 points and for the cluster set obtained with different values of the regularization parameter. The error rate is computed on a test set of 5000 points.

γ	K	Reduc. Rate
2.5	2518	87.4 %
1.5	1818	90.9 %
1.25	1582	92.1 %
1	1326	93.4 %

Table 6: Experiment 3 - Size of the clustering set and corresponding percentage of reduction for different values of the parameter γ, obtained from the original training set of 20000 points.

References

[1] Bennett, K., A. Demiriz and J. Shawe-Taylor: 2000, 'A Column Generation Algorithm for Boosting' In *Proc. of 17. International Conferance on Machine Learning*, p. 57-64, Morgan Kaufman, San Francisco

[2] Burges, C. J. C. 1998 'A tutorial on support vector machines for pattern recognition' In *Data Mining and Knowledge Discovery* 2(2):121-167.

[3] Cortes, C. and V. Vapnik: 1995, 'Support Vector Networks'. *Machine Learning* **20**, 1–25.

[4] Lee, Y. J. and O.L. Mangasarian: 2001 'RSVM: Reduced Support Vector Machines' In *First SIAM International Conference on Data Mining*

γ	Error Rate
2.5	3.15 %
1.5	3.16 %
1.25	3.14 %
1	3.17 %

Table 7: Experiment 3 - Error rate for different values of the parameter γ. In all cases the regularization parameter C was 10

[5] Musavi, M.T., Ahmed, W., Chan, H., Faris, K.B, Hummels, D.M. 1992. "On the Training of Radial Basis Function Classifiers," *Neural Network* **5**: 595–603.

[6] Osuna, E., R. Freund, and F. Girosi: 1997 'Improved Training Algorithm for Support Vector Machines' In Proceeding of IEEE Neural Networks and Signal Processing (NNSP'97)

[7] Platt, J.: 1999 'Fast training of support vector machines using sequential minimal optimization' In B. Schlkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods — Support Vector Learning, pages 185-208, Cambridge, MA, 1999. MIT Press.

[8] Pontil, M.: 1996. Support vector machines for clustered data. Technical Report, University of Genova, April 1996.

[9] Provost, F., and V. Kolluri: 1999 'A survey of methods for scaling up inductive algorithms' In *Machine Learning*, 1999, p. 1-42.

[10] Rifkin, R.: 2000 'SvmFu a Support Vector Machine Package' In http://five-percent-nation.mit.edu/PersonalPages/rif/SvmFu/index.html

[11] Trafalis T. and H. Ince: 2001, 'Benders Decomposition Technique for Support Vector Regression' *Working Paper, School of Industrial Engineering, University of Oklahoma, Norman, OK*

[12] Vanderbei, R.J. 1997. "LOQO User's Manual - Version 3.04." *Statistical and Operations Research* Tech. Rep. **SOR-97-07** Princeton University.

[13] Vapnik, V. N.: 1998, *Statistical Learning Theory.* New York: Wiley.

[14] Wright, M.H. 1992. "Interior Methods for Constrained Optimization" **Tech. Rep.** AT&T Bell Lab.