Scaling-up Support Vector Machines Using Boosting Algorithm

Dmitry Pavlov
Dept. of Info. and Comp. Science
Univ. of California at Irvine
Irvine, CA 92697, USA
pavlovd@ics.uci.edu

Jianchang Mao
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
mao@almaden.ibm.com

Byron Dom
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
dom@almaden.ibm.com

Abstract

In the recent years support vector machines (SVMs) have been successfully applied to solve a large number of classification problems. Training an SVM, usually posed as a quadratic programming (QP) problem, often becomes a challenging task for the large data sets due to the high memory requirements and slow convergence. We propose to apply boosting to Platt's Sequential Minimal Optimization (SMO) algorithm and to use resulting Boost-SMO method for speeding and scaling up the SVM training. Experiments on three commonly used benchmark data sets show that Boost-SMO achieves classification accuracy comparable to conventional SMO but is a factor of 3 to 10 faster. The speed-up could easily be orders of magnitude on the larger data sets.

Keywords: Support vector machines, boosting algorithm, machine learning.
Track: Pattern Recognition and Analysis.
Correspondence to: Dr. Jianchang Mao.

1. Introduction

SVMs have received a considerable attention in the recent years and many successful applications of SVMs have been described in the literature [10, 9, 4]. The objective of SVMs is to maximize the margin of separation between the classes. Larger margin ensures smaller Vapnik-Chervonenkis (VC) dimension, which yields a good generalization performance.

In spite of many desirable properties of SVMs, such as independence on the feature dimensionality, notion of support vectors, and high generalization performance, their training on large data sets is a challenging problem. Various training algorithms have been proposed to speed up the training, including chunking [9], Osuna's decomposition method [5], and Platt's Sequential Minimal Optimization (SMO) [6]. Although these algorithms have been proven to accelerate the training, they do not scale well with the size of the training data.

We propose to use *boosting* to solve the scaling problem. Boosting is a general technique for improving performance of any given classifier [3, 7]. Boosting can effectively convert a weak classifier (which does a little better than random guessing) into a strong classifier (which can achieve an arbitrarily low error rate given sufficient training data). Schapire et al. [3, 7] explain the effectiveness of the boosting algorithm in improving generalization performance based on the notion of a "margin" that can be interpreted as a measure of confidence in the prediction. They proved that larger margins translate into a superior upper bound on the generalization error and that boosting is particularly aggressive at reducing the margin [7].

It is generally believed that boosting is more effective on unstable classifiers such as decision trees and neural networks than on stable classifiers (such as linear classifiers) [2]. Linear SVMs with a dot-product kernel belong to the class of stable classifiers. Therefore, we do not expect boosting to help improve the classification accuracy. In this paper, we attempt to solve the scaling problem of SVMs, while preserving their classification accuracy.

2. Boost-SMO Algorithm

We apply boosting to Sequential Minimal Optimization (SMO) algorithm which is the most efficient state-of-the-art technique for training SVMs. Shortly, SMO decomposes the quadratic programming (QP) problem arising in SVM training into a sequence of minimal QP problems involving only two variables, and each of these problems is solved analytically. SMO heuristically selects a pair of variables for each problem and optimizes them. This procedure repeats until all the patterns satisfy the optimality conditions [6].

The main idea of boosting is to train a sequence of classifiers so that each subsequent classifier concentrates mostly on the errors made by the previous ones [7]. This is achieved by assigning a probability label to each training pattern and maintaining it over the whole training phase. The update rules for probability labels are fully specified by the boosting algorithm. The worse the performance of the previously built classifiers on a particular example is, the higher the probability it will get.

The basic boosting algorithm is described as follows.

1. Initialize $t = 1$, and labels $D_i^{(t)} = 1/n, i = 1, 2, \cdots, n$.
2. Train a weak classifier using distribution $D^{(t)} = \{D_i^{(t)} | i = 1, 2, \cdots, n\}$.
3. Get the weak hypothesis $h^{(t)} : X \to \{-1, +1\}$ by minimizing the error $\varepsilon^{(t)} = \sum_{i:h^{(t)}(x_i) \neq y_i} D_t(i)$
4. Choose $\alpha^{(t)} = \frac{1}{2} \ln \left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}} \right)$.
5. Update $D^{(t)}$:
$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha^{(t)} y_i h^{(t)}(\mathbf{x}_i))}{Z^{(t)}},$$
where $Z^{(t)}$ is a normalization factor.
6. Repeat steps 2-5 T times.

The final hypothesis is given by a *boosted* classifier which is a linear combination of individual locally optimal SVMs
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\mathbf{x}) \right).$$
Note that the final decision function obtained by boosting remains linear when linear SVMs are used as individual classifiers.

A direct way to use the probability distribution $D^{(t)}$ over examples in training SVMs is to create subsamples of data or the so-called boosting sets. The boosting set that will be used for training the classifier on the t^{th} iteration can be created by sampling examples from the original data set according to $D^{(t)}$. In our experiments the size of boosting sets was roughly equal to 2-4% of the original set size which allows for a very fast training of individual SVMs.

Each individual SVM is trained using regular SMO, hence it achieves maximum margin separability on the corresponding boosting set. But because of the limited amount of data used to train individual SVMs, their decision boundaries may be far from the global optimal solution. However, the ensemble effect of a sequence of SVMs (normally an order of 10-15) allows for a boosted classifier to have a high generalization performance. Since the boosting algorithm also has the effect of improving the margin [8], Boost-SMO is able to find a global solution which is comparable in terms of accuracy to that obtained by the standard SVM training algorithms.

3. Experiments

We compared performance of linear classifiers trained with the Boost-SMO and the Full-SMO (conventional SMO algorithm) on the following three data sets: the Reuters Data, the Microsoft Web Data and the UCI Adult Data.

For the Reuters Data we looked at the classes “acq” and “earn” that have the greatest number of positive examples.

The Microsoft Web Data is publicly available at the UCI KDD repository[1]. This data set reflects the Web pages of www.microsoft.com that each user visited during one week of February 1998. The classification task, as we pose it, is to predict whether a user will visit the most popular site S based on his/her visiting pattern of all other sites.

The Adult data set is available at UCI machine learning repository [1]. The task is to predict if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status and so forth. The parameters of the data sets are summarized in the Table 1.

In both algorithms we used dot-product caching with the maximum cache size of 6 million to avoid the usual bottleneck of SMO training - repeated kernel evaluations. After classifiers were trained we compared their performance on the provided test data. All experiments were run on the NT machine with Pentium II 450 MHz processor and 196 Mb of RAM.

Table 2 compares the Boost-SMO and the Full-SMO algorithms on the four data sets with respect to the precision/recall rates obtained by the algorithms and the training time. The analysis of the figures shows that Boost-SMO is able to reach the same rate of accuracy as the conventional, optimal Full-SMO algorithm. On the other hand, the training time with the Boost-SMO algorithm is 3 to 10 times less than in the Full-SMO.

[1]http://kdd.ics.uci.edu/databases/msweb/msweb.html

Table 1. Characteristics of the Data Sets

Parameter Name	Reuters Data, class “acq”	Reuters Data class “earn”	Web Data	Adult Data
Number of Attributes	11230	11230	294	14
Binary Attributes	no	no	yes	no
Training set	1650(+)	2877(+)	10057(+)	7508(+)
	7953(-)	6726(-)	21875(-)	22654(-)
Test set	719(+)	1087(+)	1561(+)	3700(+)
	2580(-)	2212(-)	3325(-)	11360(-)

Figure 1 shows how precision and recall rates of the Full-SMO and Boost-SMO change in the course of training on the Adult data set. The upper curves on the figure correspond to precision and the lower to recall. It is easy to see that Boost-SMO is much faster, yet is as accurate as the Full-SMO algorithm.

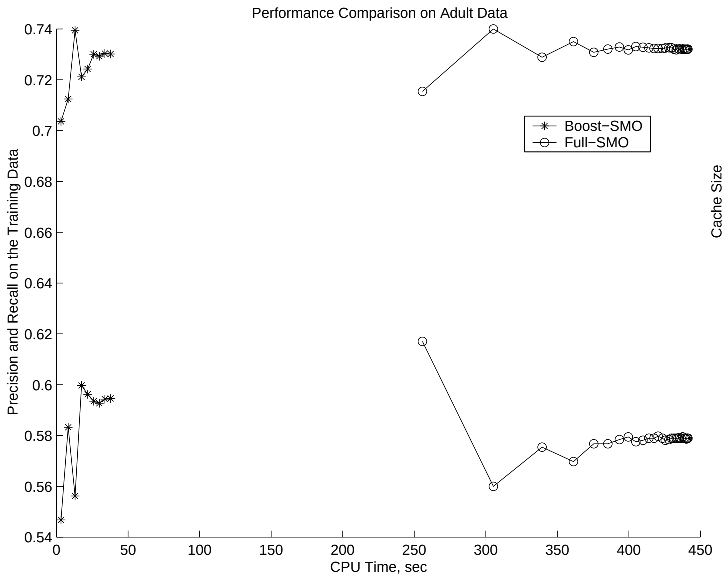

Figure 1. Precision and Recall Rates of the Full-SMO and Boost-SMO Versus CPU Time on UCI Adult Data

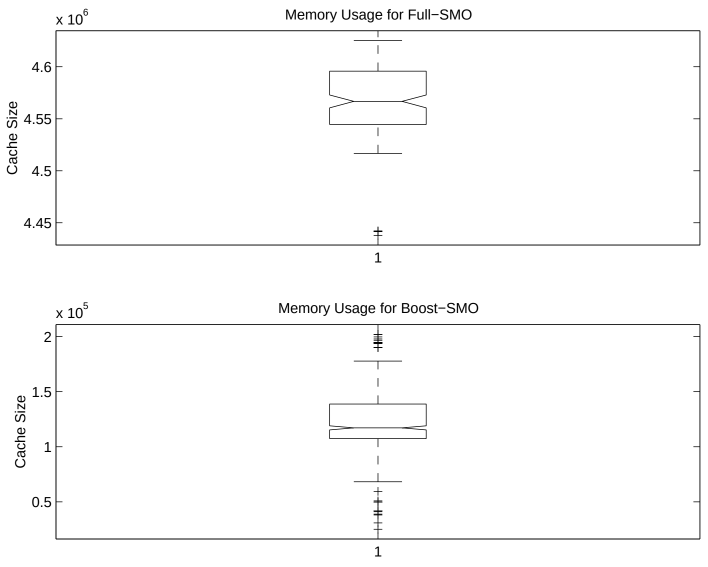

Figure 2. Box-plot of Cache Usage of the Full-SMO and Boost-SMO on UCI Adult Data

The box-plot of cache usage for Boost-SMO and Full-SMO during training shown in Figure 2 indicates that Boost-SMO uses much less memory than the Full-SMO. As the number of patterns in the training data increases, the difference in speed and memory requirements might be even more significant.

There are several important issues in the design of the boosted classifier. The first one is the choice of the boosting set size for training individual SVMs. This parameter depends on the size of physical memory of the computer used for training Boost-SMO. Typically, the larger the subset size is, the fewer number of boosting steps is needed. However, the large boosting set size may make the training of individual SVMs very slow especially if the cache size is frequently exceeded. We have done experiments with different boosting set sizes and found that accuracy performance of the Boost-SMO is not sensitive to the boosting set size for a large range from 5% to 90% of data. Table 3 shows the total number of errors (false-positives + false-negatives) versus the percentage of data used for training individual SVMs in Boost-SMO on the “acq” class of the Reuters data. It is easy to see that within a small random fluctuation the number of errors on the test data remains constant as the boosting set size increases. Compare the figures in the table with the number of error returned by the Full-SMO - 69. We see that in order to get an almost optimal performance in terms of accuracy it is sufficient to use only 2-4% of the data at each

Table 2. Performance Comparison of the Full-SMO and Boost-SMO

	Full-SMO			Boost-SMO		
Data set	Precision	Recall	CPU Time	Precision	Recall	CPU Time
Web	60.1	67.1	1546 sec.	60.2	66.9	576 sec.
Adult	73.2	57.9	443 sec.	72.9	59.1	44 sec.
Reuters, “acq”	92.6	97.7	119 sec.	96.7	93.2	27 sec.
Reuters, “earn”	98.6	98.1	94 sec.	99.0	97.7	26 sec.

Table 3. Number of Errors Versus the Percentage of Data Used to Train Individual SVMs in Boost-SMO

Percentage	1	2	5	10	20	30	40	50	60	70	80	90
# errors	231	72	73	75	74	73	75	73	70	71	74	76

step of Boost-SMO.

Another design issue is how to determine the number of boosting steps. In our experiments, boosting terminates when either of the following two conditions is met: (i) the prespecified maximum number of boosting steps is reached, or (ii) the error of the current SVM is ε-close to 0.5. Although this choice of the number of boosting steps leads to a reasonable performance, we have noticed that in certain cases smaller number of steps yields better generalization performance.

4. Conclusions

We have presented a method (Boost-SMO) for speeding and scaling up the SVM training using the boosting algorithm. Experiments on three well-known benchmark data sets show that not only Boost-SMO preserves the classification accuracy of the Full-SMO, but it is also significantly faster than the conventional algorithm. The memory requirements of Boost-SMO are almost independent of the original data set size, therefore it can be easily scaled up to large training data sets.

We also believe that the described method can be applied to non-linear SVMs and plan to test this in the nearest future.

An interesting open theoretical issue is finding conditions that could guarantee the optimality of the proposed algorithm.

Acknowledgment: We would like to thank Salil Prabhakar for many useful discussions.

References

[1] C. Blake and C. Merz. Uci repository of machine learning databases. Technical report, Department of Information and Computer Science, University of California, Irvine, CA. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[2] L. Breiman. Bias, variance, and arcing classifiers. Technical report, Statistics Department,University of California, Berkeley. April, 1996, Technical Report 460.

[3] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of the 13th Intl. Conf. on Machine Learning*, 1996.

[4] M. A. Hearst. Support vector machines. *IEEE Intelligent Systems*, pages 18–28, July/August 1998.

[5] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285. IEEE Press, New York, 1997.

[6] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

[7] R. E. Schapire. An introduction to boosting algorithm. In *Proc. of the sixteenth Intl. Joint Conf. on Artificial Intelligence*, 1999.

[8] R. E. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 1999.

[9] B. Scholkopf, C. Burges, and A. Smola (eds). *Advances in Kernel Methods – Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

[10] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.