Using the Leader Algorithm with Support Vector Machines for Large Data Sets

Enrique Romero*

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
eromero@lsi.upc.edu

Abstract. One of the main drawbacks of Support Vector Machines (SVM) is their high computational cost for large data sets. We propose the use of the Leader algorithm as a preprocessing procedure for SVM with large data sets, so that the obtained leaders are used as the training set for the SVM. The result is an algorithm where the Leader algorithm allows to construct a sample of the data set whose granularity level and computational cost are controlled by the threshold parameter. Despite its apparent simplicity, the proposed model obtains similar accuracies to *standard LIBSVM* with fewer number of support vectors and less execution times.

1 Introduction

Support Vector Machines (SVM) have been highly successful in several machine learning problems [25,19]. However, one of the main drawbacks of SVM is their computational complexity, leading to long training times for large data sets.

The optimization problem related to SVM is typically formulated as a quadratic programming problem. Given N training examples, standard quadratic programming solvers take $O(N^3)$ training time and $O(N^2)$ space to obtain a solution. Several approaches have been proposed for SVM to reduce time and space complexities. Chunking and decomposition methods [26,15] optimize the SVM with respect to subsets of the data. The Sequential Minimal Optimization algorithm [17] obtains the analytical solution of the subproblem with only two examples, and then heuristically choices the best pairs of parameters to optimize in a sequential process. Other incremental algorithms have also been described [4,7,9]. Another family of algorithms modify the objective function to apply efficient algorithms [6,13] or transform the problem to an equivalent one [23]. In general, these methods suffer from slow convergence when the number of support vectors is large [14].

A different approach aims to reduce the number of support vectors either directly or by reducing the size of the training set while keeping all the necessary information for the construction of a good model. Likelihood-based squashing is used in [16] to remove examples that contribute in a similar way to the likelihood of the SVM parameters. Active learning methods for SVM [18,22] try to sequentially add examples

* This work was supported in part by the Ministerio de Ciencia e Innovación (MICINN), under project TIN2009-13895-C02-01.

near the boundaries. Several sampling techniques are also based in similar idea [10,1]. The model proposed in [20] proposes a preprocessing algorithm that tries to select the examples near the decision boundaries by looking at the classes of their neighbors.

A third class of algorithms are based on clustering. The algorithm proposed in this paper also takes this approach. The intrinsic nature of SVM, which is a function only on the support vectors, makes clustering algorithms suitable for preprocessing and obtain a representative sample of the data set. By changing the parameters of the clustering algorithm, the number and shapes of the clusters change, leading to different granularity levels for the training set. In [27], the centroids of a hierarchical clustering tree are recursively selected to train a SVM at every step, where the examples near the boundaries are declustered. The k-means clustering algorithm is used in [21] to select the examples near the cluster boundaries as the input data for a SVM. A similar approach is taken in [3,28] where, starting from a clustering algorithm, the clusters are split or shrinked depending on a SVM trained with the centroids of the clusters. In [11] the learning problem of SVM is redefined assuming that the clusters have a Gaussian distribution in the feature space, and using a probability product kernel.

In this work we propose to use the Leader algorithm [8] as a preprocessing procedure for large data sets. The obtained leaders are used as the training set for SVM. In order to maintain the coherency between distances and inner products, the distance within the Leader algorithm is computed with the kernel function, which is equivalent to run the Leader algorithm in the feature space induced by the kernel. The proposed algorithm consists of two decoupled phases, allowing to control the execution times.

The advantages of using the Leader algorithm are threefold. First, it is a very fast clustering algorithm, compared to most common clustering algorithms such as k-means or hierarchical clustering methods. Second, all the areas of the input space with any example in the data set are represented by, at least, one leader. Finally, the leaders are always a subset of the original data set. Despite its apparent simplicity, the proposed model gives good experimental results on large data sets, obtaining similar accuracies to *standard LIBSVM* with fewer number of support vectors and less execution times.

2 Support Vector Machines

SVMs for classification can be described as follows [25]: the input vectors are mapped into a (usually high-dimensional) inner product space through some non-linear mapping ϕ, chosen *a priori*. In this space (the *feature space*), a maximal margin hyperplane is constructed. By using a (positive definite) kernel function $K(u, v)$ the mapping becames implicit, since the inner product defining the hyperplane can be evaluated as $\langle \phi(u), \phi(v) \rangle = K(u, v)$ for every two vectors $u, v \in \mathbb{R}^D$.

When the data set is not separable by a hyperplane (neither in the input space nor in the feature space), some tolerance to noise is allowed. Using Lagrangian and Kuhn-Tucker theory, the maximal margin hyperplane for a binary classification problem given by a data set X is a linear combination of simple functions depending on the data:

$$f(x) = b + \sum_{i=1}^{N} y_i \alpha_i K(x_i, x) \tag{1}$$

where the vector $(\alpha_i)_{i=1}^N$ is the (1-norm soft margin) solution of the following constrained optimization problem in the dual space:

$$\begin{array}{lll} \text{Maximize}_\alpha & \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i,j=1}^N y_i\alpha_i y_j\alpha_j K(x_i,x_j) & \\ \text{subject to} & \sum_{i=1}^N y_i\alpha_i = 0 & \text{(bias constraint)} \\ & 0 \leqslant \alpha_i \leqslant C & i = 1...N. \end{array} \qquad (2)$$

for a certain constant C. The parameter C allows to control the trade-off between the margin and the errors in the data set. By setting $C = \infty$, the hard margin hyperplane is obtained. The points x_i with $\alpha_i > 0$ (active constraints) are named *support vectors*. The most usual kernel functions $K(u, v)$ are polynomial, Gaussian-like or sigmoidal functions. It is worth noting that the kernel function depends on a certain parameter γ.

3 The Leader Algorithm as a Preprocessing Procedure for SMVs

3.1 The Leader Algorithm

Clustering algorithms divide data into groups (clusters) that are meaningful, useful, or both. Among the many clustering algorithms, the Leader algorithm [8] is one of the fastest ones, and it has been used in many successful applications (see [24], for example).

The Leader algorithm works with a distance or similarity measure and a predetermined threshold T. It constructs a partition of the data into clusters, assigning an example for each cluster (the leader), such that every example in a cluster is within a distance (or similarity) T of the leading example. The algorithm makes a single pass through the data set. For every example, it looks for the first cluster whose leader is close enough (or similar enough) to the current example with respect to the specified measure and threshold T. If such matching leader is found, then the current example is assigned to that cluster. Otherwise, the algorithm will add a new cluster whose leader is the current example. Several variations of the Leader algorithm have been described elsewhere (see [2], for example).

3.2 The Proposed Approach

To fix notation, consider the classification task given by a data set $X = \{(x_i, y_i)\}_{i=1}^N$, where each example $x_i \in \mathbb{R}^D$, $y_i \in \{-1, +1\}$. Let us define X^+ and X^- as the subsets of positive and negative examples, respectively.

The main idea of the proposed approach is to use the Leader algorithm as a preprocessing procedure to select a subset of the data (the leaders) for the subsequent training of the SVM. In order to lose as little information as possible (namely, possible support vectors), the Leader algorithm is applied independently to every class. In this way, examples of different classes (near the decision boundaries, for example) will always be represented by different leaders. Once the leaders of every class have been obtained, they are joined in a single data set that will be the input training set of a standard SVM algorithm. The proposed algorithm is summarized in Figure 1. We will call this scheme *Leader + SVM*. Extension to multiple-class problems is straightforward.

Given a data set X, a threshold T, a kernel function K and learning parameters for the SVM,

```
**Phase 1: Computing the leaders**
    $L^+ = \text{LeaderAlgorithmKernel}(X^+, T, K)$
    $L^- = \text{LeaderAlgorithmKernel}(X^-, T, K)$
    $Y = L^+ \cup L^-$

**Phase 2: Training the SVM**
    Model = $\text{TrainSVM}(Y, K, \text{learning parameters})$
```

The function $\text{LeaderAlgorithmKernel}(Z, T, K)$ runs the Leader algorithm on the data set Z with threshold T and distance $D(x, y) = \sqrt{K(x, x) - 2K(x, y) + K(y, y)}$

Fig. 1. Algorithm proposed for the *Leader + SVM* scheme

Note that, in order to maintain the coherency between distances within the Leader algorithm and inner products within the training of the SVM, the distance within the Leader algorithm is computed with the kernel function. This is equivalent to run the Leader algorithm in the feature space induced by the kernel K.

Different from other approaches (see [27,3,28], for example), the proposed algorithm consists of two decoupled phases. Therefore, the execution times can be roughly controlled by looking at the number of leaders obtained by the Leader algorithm (which, in turn, is controlled by the threshold T). The accuracy of the trained SVM will also depend on the leaders obtained in the first phase.

The advantages of using the Leader algorithm are threefold. First, it is a very fast algorithm, compared to most common clustering algorithms such as k-means or hierarchical clustering methods. Second, all the areas of the input space with any example in the data set are represented by, at least, one leader. This is a very important property when combined with SVM, since if there exist areas of the input space not covered by the clusters (represented by their centroids, for example), several potential support vectors could be lost. Finally, the leaders are always a subset of the original data set, so that there is no need to work with data subsets in the feature space as if they were an only point (the centroid of a cluster), as in [27,3,28].

In summary, the Leader algorithm allows to construct a sample of the data set whose granularity level and computational cost are controlled by the threshold T. Therefore, it is a suitable preprocessing procedure for SVM with large data sets. A similar approach has been presented in [12], with several important differences. First, the work in [12] is mostly focused on the comparison with other sub-sampling techniques. Second, the threshold T is fixed during the process, so that the computational cost is not controlled. Finally, it is only tested on small data sets (less than $1,000$ examples).

4 Experiments

We performed several experiments on benchmark data sets in order to validate the proposed model. For comparison, we also run a standard LIBSVM implementation [5].

Table 1. Description of the benchmark data sets, kernel and learning parameters. Column 'Frequencies' indicates the frequencies of the examples of every class in the training set.

Data Set	# Variables	# Ex.Train	# Ex.Test	Frequencies	γ	C
KDDCUP-99	127	4,898,431	311,029	0.801 / 0.199	0.1	10,000
Forest Cover	54	522,910	58,102	0.512 / 0.488	0.0001	10,000
Extended USPS	676	266,079	75,383	0.543 / 0.457	0.0078	10

4.1 Data Sets

Several benchmark data sets were used for the experiments: *KDDCUP-99*, *Forest Cover* and *Extended USPS*. These data sets are available at `http://www.cse.ust.hk/~ivor/cvm.html`. A brief description of these data sets is provided in Table 1.

4.2 Experimental Setting

Data Preprocessing. No preprocessing was applied to the data.

Kernel and kernel parameter. We used the Gaussian kernel $K(x, y) = e^{-\gamma\|x-y\|^2}$. In the Leader algorithm, the Gaussian function was normalized dividing by the number of input variables. The values of the γ parameter used for every data set are those of [14], and can be found in Table 1.

Threshold of the Leader algorithm. Different values of the threshold T were tested, ranging from 0.002 to 1.0.

Learning parameters. The values of the C parameter used for every data set are those of [14], and can be found in Table 1. The rest of parameters were used with their default values.

Software. For LIBSVM, we used the C++ implementation available at `http://www.cse.ust.hk/~ivor/cvm.html`. For the Leader algorithm, we used our own C implementation. Previous to every run, the examples in the data set were randomly shuffled.

Hardware. All the executions were run on an Intel Xeon CPU X3220 at 2.40GHz.

4.3 Results

Figure 2 shows the comparative results between *Leader + LIBSVM* and *standard LIBSVM* (i.e., trained with the whole training set) on the benchmark data sets studied as a function of the threshold value. Blue lines correspond to *Leader + LIBSVM* and red ones correspond to *standard LIBSVM*. We only show the results for the threshold values whose total execution time of *Leader + LIBSVM* (summing up the training times for the Leader algorithm and LIBSVM) was less than the training time of *standard LIBSVM*. Obviously, for threshold values near zero the computational cost of the *Leader + LIBSVM* scheme is larger than that of *standard LIBSVM*, since the leaders selected by the Leader algorithm are the whole data set. Table 2 shows a comparison between the best results obtained by the *Leader + LIBSVM* scheme and those of *standard LIBSVM*.

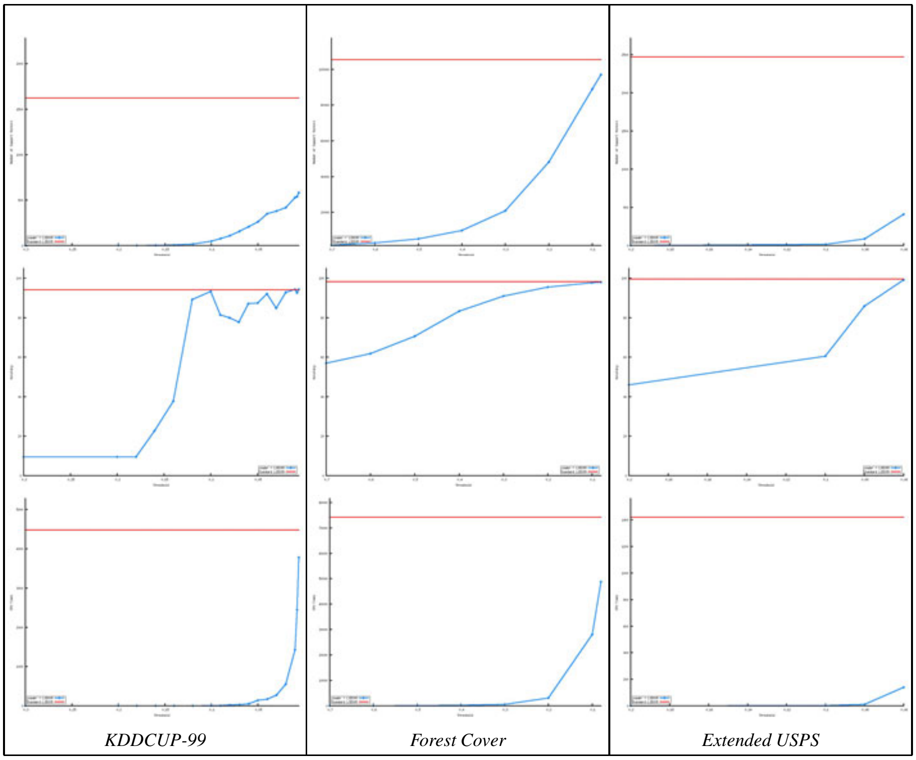

Fig. 2. Comparison between *Leader + LIBSVM* (blue) and *standard LIBSVM* (red) on the *KDDCUP-99* (left), *Forest Cover* (middle) and *Extended USPS* (right) data sets. Top row: number of support vectors. Middle row: test accuracies. Bottom row: execution times

The first thing to note from Figure 2 is that, except for the accuracies in the *KDDCUP-99* data set, the number of support vectors, the accuracies and the execution times increase as far a the threshold T decreases. This is as expected. More interestingly, the *Leader + LIBSVM* scheme is able to obtain similar accuracies to *standard LIBSVM* with fewer number of support vectors and less execution times. This is particularly remarkable for the *Extended USPS* and *KDDCUP-99* data sets (see Table 2).

Note that the behavior on the *Forest Cover* and *Extended USPS* data sets is extremely regular. The non-increasing behavior in the accuracies of the *KDDCUP-99* data set in Figure 2 is probably due to the highly unbalanced classes (see Table 1), that may need different granularity levels for the Leader algorithm. Anyway, very good accuracies can be obtained with very low execution times (see Table 2)

For the *KDDCUP-99* and *Extended USPS* data sets, most of the execution time of the *Leader + LIBSVM* scheme was spent in computing the leaders. For the *Forest Cover* data set, in contrast, the *LIBSVM* software took most of the time. This can be explained by looking at the number of support vectors of the obtained models, which is highly correlated with the number of leaders (see top row in Figure 2 and Table 2): when the number of examples is large and the number of leaders is small, *LIBSVM* may be faster

Table 2. Comparison between *standard LIBSVM* and the best results of the *Leader + LIBSVM* scheme. Columns 'Thr', '# Leads', '# SVs', 'Test' and 'Time' indicate the threshold, number of leaders, number of support vectors, test accuracy and execution times (in seconds), respectively.

	Standard LIBSVM			*Leader + LIBSVM*				
Data Set	# SVs	Test	Time	Thr	# Leads	# SVs	Test	Time
KDDCUP-99	1,624	94.20%	4,476.8	0.01	7,187	531	94.26%	1,431.1
				0.1	53	48	93.34%	6.2
Forest Cover	105,541	98.23%	74,135.9	0.1	176,102	88,916	97.75%	28,055.9
Extended USPS	2,468	99.53%	1,419.7	0.06	1,605	410	99.21%	139.5

than the Leader algorithm. Therefore, it is better to test high values of the threshold T first, since the number of leaders may be large for low values of T.

5 Conclusions and Future Work

This paper experimentally shows that, for large data sets, selecting a subset of the training set with the Leader algorithm may lead to an important decrease in the training and test times of SVM, without affecting the accuracies.

It could be worth modifying the Leader algorithm so that there were more leaders in the expected boundaries (for example, by comparing the distance to the current leader with the leaders of the other classes). If an example is suspected to be near a boundary, the threshold can be decreased. Similarly, a different threshold T can be used for every class, specially for data sets with highly unbalanced classes.

References

1. Balcázar, J.L., Dai, Y., Tanaka, J., Watanabe, O.: Provably Fast Training Algorithms for Support Vector Machines. Theory of Computing Systems 42(4), 568–595 (2008)
2. Barton, A.: Modelling Variability in the Leader Algorithm Family: A Testable Model and Implementation. Tech. Rep. NRC 47429, Institute for Information Technology, National Research Council Canada (2004)
3. Boley, D., Cao, D.: Training Support Vector Machine using Adaptive Clustering. In: International Conference on Data Mining, pp. 126–137 (2004)
4. Cauwenberghs, G., Poggio, T.: Incremental and Decremental Support Vector Machine Learning. In: Advances in Neural Information Processing Systems, vol. 12, pp. 409–415. MIT Press, Cambridge (2000)
5. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines (2002), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
6. Fung, G., Mangasarian, O.L.: Proximal Support Vector Machine Classifiers. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 77–86 (2001)
7. Fung, G., Mangasarian, O.L.: Incremental Support Vector Machine Classification. In: International Conference on Data Mining, pp. 247–260 (2002)
8. Hartigan, J.: Clustering Algorithms. John Wiley & Sons, Chichester (1975)
9. Keerthi, S.S., Chapelle, O., DeCoste, D.: Building Support Vector Machines with Reduced Classifier Complexity. Journal of Machine Learning Research 7, 1493–1515 (2006)

10. Lee, Y.J., Mangasarian, O.L.: RSVM: Reduced Support Vector Machines. In: International Conference on Data Mining (2004)
11. Li, B., Chi, M., Fan, J., Xue, X.: Support Cluster Machine. In: 24th International Conference on Machine Learning, pp. 505–512 (2007)
12. Li, D., Simke, S.: Training Set Compression by Incremental Clustering. Journal of Pattern Recognition Research 1, 56–64 (2011)
13. Mangasarian, O.L., Musicant, D.R.: Lagrangian Support Vector Machines. Journal of Machine Learning Research 1, 161–177 (2001)
14. Nguyen, D.D., Matsumoto, K., Takishima, Y., Hashimoto, K.: Condensed Vector Machines: Learning Fast Machine for Large Data. IEEE Transactions on Neural Networks 21(12), 1903–1914 (2010)
15. Osuna, E., Freund, R., Girosi, F.: Improved Training Algorithm for Support Vector Machines. In: IEEE Workshop on Neural Networks for Signal Processing, pp. 276–285 (1997)
16. Pavlov, D., Chudova, D., Smyth, P.: Towards Scalable Support Vector Machines using Squashing. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 295–299 (2000)
17. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
18. Schohn, G., Cohn, D.: Less is More: Active Learning with Support Vector Machines. In: 17th International Conference on Machine Learning, pp. 839–846 (2000)
19. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
20. Shin, H., Cho, S.: Fast Pattern Selection for Support Vector Classifiers. In: Fagerholm, J., Haataja, J., Järvinen, J., Lyly, M., Råback, P., Savolainen, V. (eds.) PARA 2002. LNCS, vol. 2367, Springer, Heidelberg (2002)
21. Sun, S.Y., Tseng, C.L., Chen, Y.H., Chuang, S.C., Fu, H.C.: Cluster-based Support Vector Machines in Text-independent Speaker Identification. In: International Joint Conference on Neural Networks, vol. 1, pp. 729–734 (2004)
22. Tong, S., Koller, D.: Support Vector Machine Active Learning with Applications to Text Classification. Journal of Machine Learning Research 2, 45–66 (2001)
23. Tsang, I.W.H., Kwok, J.T.Y., Zurada, J.A.: Generalized Core Vector Machines. IEEE Transactions on Neural Networks 17(5), 1126–1140 (2006)
24. Valdés, J.J., Barton, A.J.: Virtual Reality Visual Data Mining via Neural Networks Obtained from Multi-objective Evolutionary Optimization: Application to Geophysical Prospecting. In: International Joint Conference on Neural Networks, pp. 4862–4869 (2006)
25. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
26. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, NY (1998)
27. Yu, H., Yang, J., Han, J.: Classifying Large Data Sets using SVMs with Hierarchical Clusters. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 306–315 (2003)
28. Yuan, J., Li, J., Zhang, B.: Learning Concepts from Large Scale Imbalanced Data Sets using Support Cluster Machines. In: 14th Annual ACM International Conference on Multimedia, pp. 441–450 (2006)