An Effective Support Vector Machines (SVMs) Performance Using Hierarchical Clustering*

Mamoun Awad, Latifur Khan, Farokh Bastani, and I-Ling Yen
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688
Email: [maa013600, lkhan, bastani, ilyen]@utdallas.edu

Abstract

The training time for SVMs to compute the maximal marginal hyper-plane is at least $O(N^2)$ with the data set size N, which makes it non-favorable for large data sets. This paper presents a study for enhancing the training time of SVMs, specifically when dealing with large data sets, using hierarchical clustering analysis. We use the Dynamically Growing Self-Organizing Tree (DGSOT) Algorithm for clustering because it has proved to overcome the drawbacks of traditional hierarchical clustering algorithms. Clustering analysis helps find the boundary points, which are the most qualified data points to train SVMs, between two classes. We present a new approach of combination of SVMs and DGSOT, which starts with an initial training set and expands it gradually using the clustering structure produced by the DGSOT algorithm. We compare our approach with the Rocchio Bundling technique in terms of accuracy loss and training time gain using two benchmark real data sets.

1. Introduction

Support Vector Machines (SVMs) technique is one of the most powerful classification techniques that was successfully applied to many real world problems [3]. SVMs are based on the idea of mapping data points to a high dimensional feature space where a separating hyper-plane can be found. This mapping can be carried on by applying the kernel trick which implicitly transforms the input space into another high dimensional feature space. The hyper-plane is computed by maximizing the distance of the closest patterns, i.e., margin maximization, avoiding the problem of over-fitting. The separating hyper-plane is found using a quadric programming routine which is computationally very expensive. Furthermore, this routine depends on the data set size, taking impractical time when dealing with huge data sets.

Many applications, such as Data Mining and Bio-Informatics, require the processing of huge data sets. The training time of SVMs is a serious obstacle for these kinds of data sets. According to [4], it would take years to train SVMs on a data set of size one million records.

Many proposals have been submitted to enhance SVMs to increase its training performance with large data sets. Techniques include: Random Selection [5, 6], clustering analysis [4, 13], Bagging [9], and Rocchio Bundling [10]. Random Sampling and Rocchio Bundling could over-simplify the training set, hence losing the benefits of using SVMs; especially if the probability distribution of training and testing data is different [4]. In bagging technique, the testing process becomes very expensive [10]. Also, clustering analysis adds expensive computations in building the hierarchical structure.

This paper proposes a new approach for enhancing the training process of SVMs when dealing with large data sets. It is based on the combination of SVMs and clustering analysis. The idea is to approximate the *support vectors* (see section 2 for details) by applying clustering analysis on the fly and without adding a lot of computations in building the hierarchical structure.

Traditionally, clustering algorithms can be classified into two main types, namely, hierarchical clustering and partitioning clustering. Partitioning, also called flat clustering, directly seeks a partition of the data which optimizes a predefined numerical measure. In partitioning clustering, the number of clusters is predefined, and determining the optimal number of clusters may involve more computational cost than clustering itself. Furthermore, a priori knowledge may be necessary for initialization and the avoidance of local minima. Hierarchical clustering, on the other hand, returns a series of nested partitions. The inner

* This study was supported by the National Science Foundation grant NGS-0103709.

Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)
1082-3409/04 $20.00 © 2004 **IEEE**

partitions are smaller and closer to each others (in one outer partition). Hence, we favor hierarchical clustering over flat clustering because we can explore smaller partitions and relate them to an outer one.

In the course of our approaches, we do not use the original data set to train SVMs; instead, we use the tree node references generated by the clustering algorithm. Since the size of the hierarchical tree is significantly less than the size of the original data set, the training process will be very fast. We also improve the accuracy of the classifier by applying a de-clustering process (adding new training examples, which are the children of the support vectors). By applying training and de-clustering repeatedly, the training process becomes fast and the training set grows gradually to improve accuracy. From now on when we say de-clustering we mean that the expansion phase of the clustering algorithm is invoked.

2. Support Vector Machines Overview

Support Vector Machines (SVMs) are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory. This learning strategy, introduced by Vapnik and co-workers, is a very powerful and **popular in recent areas for various application domains [2, 3]**. SVMs are based on the idea of hyper-plane classifier, or linear separability. Suppose we have N training data points $\{(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots (x_N, y_N)\}$, where $x_i \in R^d$ and $y_i \in \{+1,-1\}$. We would like to find a linear separating hyper-plane classifier as in Equation 2.1. Furthermore, we want this hyper-plane to have the maximum separating margin with respect to the two classes. This problem can be formalized as in Equations 2.2 and 2.3. Instead, we can use the Wolfe dual formalization as in Equation 2.4 (see [7] for more details).

$$f(x) = sign(w \cdot \chi - b) \quad (2.1)$$

$$\text{Minimize}_{(w,b)} 1/2 w^T \cdot w \quad (2.2)$$

$$\text{Subject to } y_i.(w \cdot x_i - b) \geq 1 \quad (2.3)$$

$$\text{Maximize } L(w,b,\alpha) \equiv 1/2 w^T w - \sum_i^N \alpha_i y_i (w \cdot x_i - b) + \sum_i^N \alpha_i \quad (2.4)$$

$$f(x) = sign(w.x + b) = sign(\sum_i^N \alpha_i y_i (x_i.x) - b) \quad (2.5)$$

This is a convex quadratic programming problem (in w, b) in a convex set. We can classify a new object x using Equation 2.5. Note that the training vectors x_i occur only in the form of dot product; there is a Lagrangian multiplier α for each training point, which reflects the importance of the data point. When the maximal margin hyper-plane is found, only points that lie closest to the hyper-plane will have $\alpha > 0$ and these points are called *support vectors*. All other points will have $\alpha = 0$. This means that the representation of the hypothesis/classifier is given only by those points that lie closest to the hyper-plane and they are the most important data points that serve as ***support vectors***.

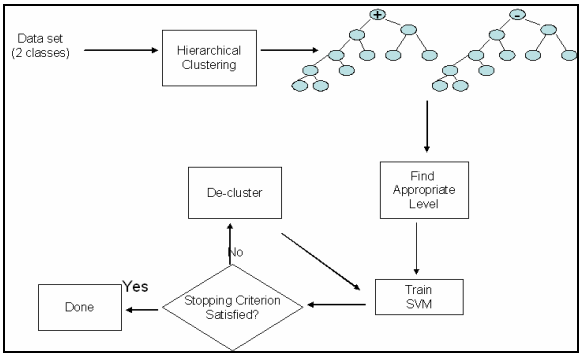

Figure 1 The flowchart of the main algorithm.

3. Our Approach

Our approach for enhancing the training process of SVMs is to find those support vectors or approximate them in advance. We use the hierarchical structure produced by the DGSOT algorithm [8] to approximate support vectors. Then we train SVMs on a small training set, which is the clusters' references of the top levels of the trees, hence the training process is very fast. After computing the margin, we can determine whether a node is a support vector node or not (see Section 2 for details). Support vector nodes are de-clustered by adding their children nodes to the training set. We repeat this process until a stopping criterion holds true. Therefore, clustering and SVMs are mingled with each other. There are several ways we can set the **stopping criteria**. For example, we can stop at a certain level of the tree, or upon reaching a certain number of nodes in the trees, or when a certain accuracy level is attained. In our implementation, we adopt the second strategy so that we can compare it with the Rocchio Bundling algorithm [10] which reduces the data set into a specific size. In the following subsections, we present our approaches for support vectors approximation.

3.1 Two Clustering Trees Based on SVMs (TCT_SVM)

Figure 1 outlines the steps of this approach for enhancing the training process of SVMs. First, we apply the DGSOT algorithm to build an initial hierarchical tree for each class. Basically, we want to start with a reasonable number of nodes in each tree. Second, we train SVMs on the tree nodes. Third, we de-cluster *support vectors* by including their children in the training data set. Notice that the number of nodes is relatively small in the top levels of the tree making the training process very fast. Also, based on a stopping criterion, we can decide whether to stop the algorithm or continue before completely building (or expanding) the hierarchical trees.

One subtlety of using clustering analysis is that each epoch might take long time to converge, which overcomes any benefit of improving SVMs. However, our strategy here is that we do not wait unit DGOST finishes. Instead, after each epoch/iteration of the DGSOT algorithm which takes a small amount of time, we train the SVMs on the generated nodes. After each training process, we control the growth of the hierarchical tree because non-support vector nodes will be stopped from growing. We have updated the DGSOT algorithm to accommodate that. Figure 2 shows the growth of one of the hierarchical trees during this approach. The bold nodes represent the support vector nodes. Notice that nodes 1, 2, 3, 5, 6, and 9 were allowed to expand because they are support vector nodes, while nodes 4, 8, 7, and 10 were stopped from growing because they are not support vector nodes. The de-cluster step is very important to increase the number of points in the data set to obtain a more accurate classifier. Figure 3 shows an illustrative example of applying de-clustering. The initial data set and the new data set are shown after de-clustering support vector nodes (bold nodes). Bold nodes represent non support vector references (such as +4 and +7).

3.2 Two Clustering Trees with extra Distance measure SVMs (TCTD-SVM)

The TCTD-DVM approach is a variation of the TCT-SVM. We add an extra step before de-clustering. Specifically, we measure the distance between nodes in the training set. Since the distance between nodes lying in the boundary area is most probably the least, then we can exclude those nodes having distance more than the average distance. For example, Figure 4 shows two hierarchical trees representing two classes. After training, we determine that nodes (+1, +2, +3, -1, -2, -3) are support vector nodes. We measure the Euclidean distance between every two nodes, one is a positive node and the other is a negative node in the training set. For example, we measure the distance between [+1, -2], [+1,-2], [+1,-3], [+2,-1], [+2, -2], [+2, -3],…. At the end, we will find that nodes +2, +1, +3, -1, and -2 are the closest to each other; furthermore nodes +2 and -3 are distant from every other node in the other tree. Hence, we remove nodes +2 and -3 from the training set or, at least, we do not de-cluster them because they will not make any contribution to the margin computation. Since the distance varies even between support vectors, we can choose those nodes (one from the positive tree and one from the negative tree) with distance less than the average distance between nodes from both trees.

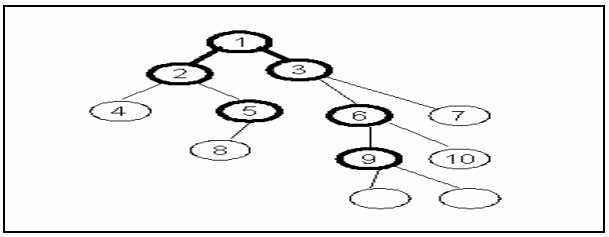

Figure 2 controlling the DGSOT growth.

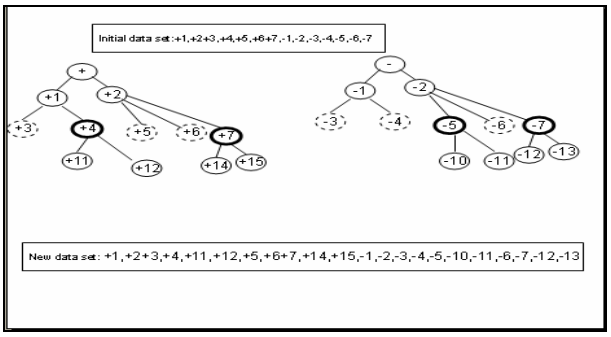

Figure 3 illustrative de-clustering example: The new data set after applying de-clustering.

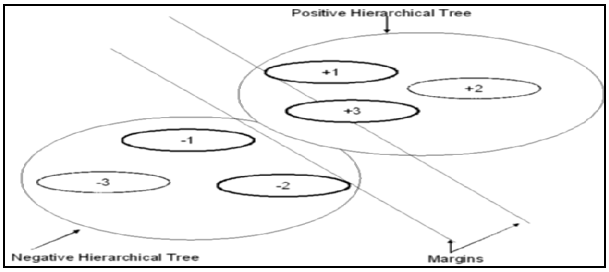

Figure 4 support vectors from both trees most probably closest.

3.3 One Clustering Tree based SVM (OCT-SVM)

In the OCT-SVM approach, we choose the most qualified nodes in a hierarchical tree to train SVMs based on a single tree for the whole data set. Specifically, we study the relationship between the data

points in each cluster to find out whether that cluster is a support vector cluster or not. Basically, we apply clustering analysis to the whole data set and generate one hierarchical tree. We are interested in those clusters having data points from different classes assigned to them, because those clusters most probably lie in the marginal area between the two classes. We call those clusters *heterogeneous* clusters. But the question is how to determine whether a heterogeneous node is positive or negative? We need this because training examples bear classes. Here, we simply use a majority voting mechanism to determine the class of the node. Specifically, if the number of positive data points of a node is greater than the number of negative data points of that node, the node will be declared to be a positive class. We break the tie arbitrarily. Furthermore, to approximate *support vectors*, when we de-cluster a node we only use closest children to the margin and discard the rest.

This approach differs from the previous two as follows. First, we only construct only hierarchical tree. Second, we always start from the top level of the tree, and repeatedly train and de-cluster support vector nodes. Since the top levels of the tree span more area in the space, most probably we find that they are heterogeneous. After we find those heterogeneous nodes in an initial level, we de-cluster those nodes. Third, we use majority voting to determine the label of a cluster. Furthermore, we only include those heterogeneous children in the training set and discard the rest.

Figure 5 shows one hierarchical clustering tree spanning the data set. The clusters depicted in big circles are the nodes of one of the top levels, say l level of the tree. Notice that clusters lying in the margin area are heterogeneous, e.g. A and B, while clusters lying far from the margin area are homogeneous, e.g., E and C. In OCT-SVM, we discard clusters C and E because they are not heterogeneous. Meanwhile, we train SVMs on clusters A, B, and D.

4 Experimental Results

In this section, we present our experimental results of applying our approaches and compare them with other techniques such as random selection and Rocchio Bundling [10]. We use the LIBSVM for SVSM [11] implementation and use the ν-SVM with linear kernel. In our experiments we set ν very low ($\nu = 0.01$ or $\nu = 0.1$). We have evaluated the performance of our different approaches on several standard classification data sets. Because of the space limitations we are only reporting the results of two.

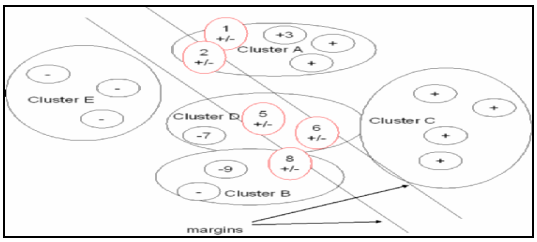

Figure 5 heterogeneous clusters in the hierarchical tree.

The first data set is the Waveform data set. The Waveform data set has 5000 waves; each belongs to one of three classes. There are 40 attributes, all of which include noise [12]. The second data set is the Adult data set. This data set has over 32000 data points and two classes. Each data point composed of 14 features (8 symbolic features and 6 continuous features) [12]. For both data sets, we created one independent "zero-one" (predicate) feature for each value of the symbolic features such that "one" indicates the existence of the value.

For each data set, we use cross-validation (n-fold = 20) to train SVMs and obtain the training time and generalization accuracy. Table 1 shows the training time and testing accuracy of all techniques on the Wave dataset. For the sake of comparison, we measure the training time and the generalization accuracy when the size of the training set reaches 4% (first two columns) and 12% (last two columns) of the size of the original dataset. The first row shows the training results by applying the cross-validation techniques (n-folds = 20).

Table 2 shows the same comparison results but with the Adult dataset. Notice that the size of the hierarchical tree (training data) in our techniques (TCT-SVM, TCTD-SVM, and OCT-SVM) does not reach the 12%; because the algorithms prune many nodes by not growing them and converges with small hierarchical tree. As expected, the training time of all techniques is very good (notice that we added the pre-processing time to the training time of SVMs in all techniques). However, the accuracy varies from one approach to another and from one data set to another.

The general conclusion is that the TCT-SVM is the best stable approach in terms of accuracy and training time. The distance measure in the TCTD-SVM approach was not very effective, and sometimes some accuracy was lost. In the OCT-SVM approach, the heterogeneity and majority vote techniques are not quite affective to approximate support vector machines. Furthermore, they might add extra overhead to compute them. The Random Sampling and Rocchio Bundling techniques suffers from the same problem of TCTD-SVM in reducing the dataset, and their performance vary from one dataset to another.

Table 1 Training time and accuracy results for the Wave dataset and using 4% and 12% of the total size of the data set.

	Training Time in seconds (4%)	Testing Accuracy (4%)	Training Time in seconds (12%)	Testing Accuracy (12%)
SVMs	19.2	100%		
Random Selection	0.2	82%	1.6	85%
Rocchio Bundling	0.2	54%	1.5	55%
TCT-SVM	0.2	84%	0.22	83%
TCTD-SVM	0.95	73%	N/A	N/A
OCT-SVM	0.5	82%	0.9	82%

Table 2 Training time and accuracy results for the Adult dataset and using 4% and 12% of the total size of the data set.

	Training Time in seconds (4%)	Testing Accuracy (4%)	Training Time in seconds (12%)	Testing Accuracy (12%)
SVMs	8000	76%		
Random Selection	66	71%	200	69%
Rocchio Bundling	80	75%	82	75%
TCT-SVM	82	74%	N/A	N/A
TCTD-SVM	85	69%	N/A	N/A
OCT-SVM	81	72%	N/A	N/A

5 References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag; 1999, ISBN: 0387987800.

[2] S. Terrence, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data", *Bioinformatics Vol. 16 no.* 10 2000, pages 906-914.

[3] Y. LeCun, L.D. Jackel, L. Bottou, A. Brunot, C. Cortes, J.S. Deker, H. Drucker, I. Guyon, U.A. Muller, E. Sackinger, P. Siard, and V. Vapnik, "Comparison of Learning Algorithms for Handwritten Recognition", *in F. Fogelman and P. Gallinari, editors, International Conference on Artificial Neural Networks*, pages 53-60, Paris, 1995 EC2 & Cie.

[4] Hwanjo Yu, Jiong Yang, and Jiawei Han, "Classifying Large Data sets Using SVMs with Hierarchical Clusters", *in Proc. of the 9th ACM SIGKDD 2003*, August 24-27, 2003, Washington, DC, USA.

[5] J. L. Balcazar, Y. Dai and O. Watanabe, "Provably Fast Training Algorithms for Support Vector Machines", *in Proc. of the 1st IEEE International Conference on Data mining,*, IEEE Computer Society (2001) pp.43-50.

[6] D. K. Agarwal, "Shrinkage estimator generalizations of proximal support vector machines", *in Proc. of the 8th ACM SIGKDD international conference of knowledge Discovery and data mining,* Edmonton, Canada, 2002.

[7] N. Cristianini and J. Shawe-Taylor, *Introduction to Support Vector Machines*, Cambridge University Press 2000 ISBN: 0 521 78019 5.

[8] F. Luo, L. Khan, F. Bastani, I-Ling Yen, and J. Zhou, "A Dynamical Growing Self-Organizing Tree (DGSOT) for Hierarchical Clustering Gene Expression Profiles", to appear *in the Bioinformatics Journal, Oxford University Press*, UK.

[9] G. Valentini and T.G. Dietterich, "Low Bias Bagged Support Vector Machines", *in Proc. of the 20th International Conference on Machine Learning ICML 2003*, Washington D.C. USA, pp. 752-759.

[10] L. Shih, Y D.M Rennie, Y. Chang, and D.R. Karger, "Text Bundling : Statistics-based Data Reduction", *in Proc. of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.

[11] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001, http://www.csie.ntu.edu.tw/~cjlin/libsvm

[12] C.L. Blake and C.J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California at Irvine, Department of Information and Computer Science.

[13] B. Daniael and D. Cao, "Training Support Vector Machines Using Adaptive Clustering", *in Proc. of SIAM International Conference on Data Mining 2004*, Lake Buena Vista, FL, USA.