Fast Support Vector Machine Classification of Very Large Datasets

Janis Fehr[1], Karina Zapién Arreola[2] and Hans Burkhardt[1]

[1] University of Freiburg, Chair of Pattern Recognition and Image Processing
79110 Freiburg, Germany
fehr@informatik.uni-freiburg.de
[2] INSA de Rouen, LITIS
76801 St Etienne du Rouvray, France

Abstract. In many classification applications, Support Vector Machines (SVMs) have proven to be highly performing and easy to handle classifiers with very good generalization abilities. However, one drawback of the SVM is its rather high classification complexity which scales linearly with the number of Support Vectors (SVs). This is due to the fact that for the classification of one sample, the kernel function has to be evaluated for all SVs. To speed up classification, different approaches have been published, most which of try to reduce the number of SVs. In our work, which is especially suitable for very large datasets, we follow a different approach: as we showed in [12], it is effectively possible to approximate large SVM problems by decomposing the original problem into linear subproblems, where each subproblem can be evaluated in $\Omega(1)$. This approach is especially successful, when the assumption holds that a large classification problem can be split into mainly easy and only a few hard subproblems. On standard benchmark datasets, this approach achieved great speedups while suffering only sightly in terms of classification accuracy and generalization ability. In this contribution, we extend the methods introduced in [12] using not only linear, but also non-linear subproblems for the decomposition of the original problem which further increases the classification performance with only a little loss in terms of speed. An implementation of our method is available in [13]. Due to page limitations, we had to move some of theoretic details (e.g. proofs) and extensive experimental results to a technical report [14].

1 Introduction

In terms of classification-speed, SVMs [1] are still outperformed by many standard classifiers when it comes to the classification of large problems. For a non-linear kernel function k, the classification function can be written as in Eq. (1). Thus, the classification complexity lies in $\Omega(n)$ for a problem with n SVs. However, for linear problems, the classification function has the form of Eq. (2), allowing classification in $\Omega(1)$ by calculating the dot product

with the normal vector $\mathbf{w}$ of the hyperplane. In addition, the SVM has the problem that the complexity of a SVM model always scales with the most difficult samples, forcing an increase in Support Vectors. However, we observed that many large scale problems can easily be divided in a large set of rather simple subproblems and only a few difficult ones. Following this assumption, we propose a classification method based on a tree whose nodes consist mostly of linear SVMs (Fig.(1)).

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i k\left(\mathbf{x_i}, \mathbf{x}\right) + b\right) \tag{1}$$

$$f(\mathbf{x}) = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \tag{2}$$

This paper is structured as follows: first we give a brief overview of related work. Section 2 describes our initial linear algorithm in detail including a discussion of the zero solution problem. In section 3, we introduce a non-linear extension to our initial algorithm, followed by Experiments in section 4.

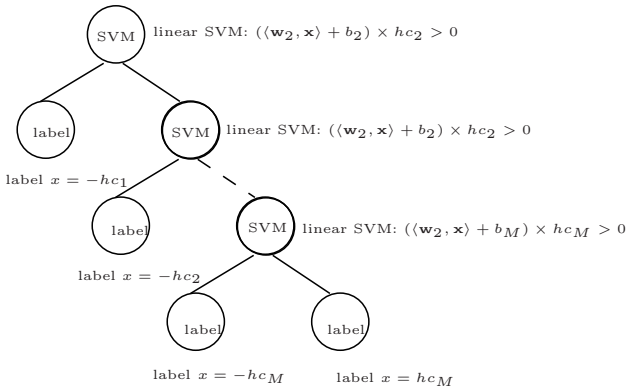

Fig. 1. Decision tree with linear SVM

1.1 Related work

Recent work on SVM classification speedup mainly focused on the reduction of the decision problem: A method called *RSVM* (Reduced Support Vector Machines) was proposed by Lee [2], it preselects a subset of training samples as SVs and solves a smaller Quadratic Programming problem. Lei and Govindaraju [3] introduced a reduction of the feature space using principal component analysis and Recursive Feature Elimination. Burges and Schölkopf [4] proposed a method to approximate $\mathbf{w}$ by a list of vectors associated with

coefficients α_i. All these methods yield good speedup, but are fairly complex and computationally expensive. Our approach, on the other hand, was endorsed by the work of Bennett et al. [5] who experimentally proved that inducing a large margin in decision trees with linear decision functions improved the generalization ability.

2 Linear SVM trees

The algorithm is described for binary problems, an extension to multiple-class problems can be realized with different techniques like one vs. one or one vs. rest [6] [14].
At each node i of the tree, a hyperplane is found that correctly classifies all samples in one class (this class will be called the "hard"' class, denoted hc_i). Then, all correctly classified samples of the other class (the "soft" class) are removed from the problem, Fig. (2). The decision of which class is to be

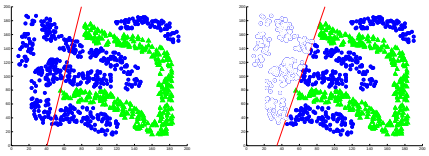

Fig. 2. Problem fourclass [8]. Left: hyperplane for the first node. Right: Problem after first node ("hard" class = triangles).

assigned "hard" is taken in a greedy manner for every node [14]. The algorithm terminates when the remaining samples all belong to the same class. Fig.(3) shows a training sequence. We will further extend this algorithm, but first we give a formalization for the basic approach.
Problem Statement. Given a two class problem with $m = m_1 + m_{-1}$ samples $\mathbf{x}_i \in \mathbb{R}^n$ with labels y_i, $i \in \mathscr{C}$ and $\mathscr{C} = \{1, ..., m\}$. Without loss of generality we define a ***Class 1 (Positive Class)*** $\mathscr{C}_1 = \{1, ..., m_1\}$, $y_i = 1$ for all $i \in \mathscr{C}_1$, with a global penalization value D_1 and individual penalization values $C_i = D_1$ for all $i \in \mathscr{C}_1$ as well as an analog ***Class -1 (Negative Class)*** $\mathscr{C}_{-1} = \{m_1 + 1, ..., m_1 + m_{-1}\}$, $y_i = -1$ for all $i \in \mathscr{C}_{-1}$, with a global penalization value D_{-1} and individual penalization values $C_i = D_{-1}$ for all $i \in \mathscr{C}_{-1}$.

2.1 Zero vector as solution

In order to train a SVM using the previous definitions, taking one class to be "hard" in a training step, e.g. $\mathscr{C}_{-1}$ is the "hard" class, one could simply set $D_{-1} \to \infty$ and $D_1 << D_{-1}$ in the primal SVM optimization problem:

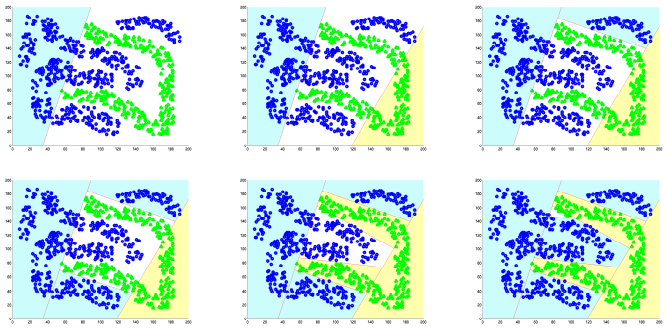

Fig. 3. Sequence (left to right) of hyperplanes for nodes 1-6 of the tree.

$$\underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} \quad \tau(\mathbf{w},\boldsymbol{\xi}) = \tfrac{1}{2}\|\mathbf{w}\|^2 + \textstyle\sum_{i=1}^{m} C_i\xi_i, \tag{3}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i, \ i = 1, .., m, \tag{4}$$

$$\xi_i \geq 0, \ i = 1, .., m. \tag{5}$$

Unfortunately, in some cases the optimization process converges to a trivial solution: the zero vector. We used the convex hull interpretation of SVMs [5], in order to determine under which circumstances the trivial solution is occurring and proved the following theorems [14]:

Theorem 1: If the convex hull of the "hard" class $\mathscr{C}_1$ intersects the convex hull of the "soft" class $\mathscr{C}_{-1}$, then $\mathbf{w} = \mathbf{0}$ is a feasible point for the primal Problem (4) if $D_{-1} \geq \max_{i\in\mathscr{C}_1}\{\lambda_i\} \cdot D_1$, where λ_i are such that

$$\mathbf{p} = \sum_{i\in\mathscr{C}_1} \lambda_i \mathbf{x}_i,$$

is a convex combination for a point $\mathbf{p}$ that belongs to both convex hulls.

Theorem 2: If the center of gravity $\mathbf{s}_{-1}$ of class $\mathscr{C}_{-1}$ is inside the convex hull of class $\mathscr{C}_1$, then it can be written as

$$\mathbf{s}_{-1} = \sum_{i\in\mathscr{C}_1} \lambda_i \mathbf{x}_i \quad \text{and} \quad \mathbf{s}_{-1} = \sum_{j\in\mathscr{C}_{-1}} \frac{1}{m_{-1}} \mathbf{x}_j$$

with $\lambda_i \geq 0$ for all $i \in \mathscr{C}_1$ and $\sum_{i\in\mathscr{C}_1} \lambda_i = 1$. If additionally, $D_1 \geq \lambda_{\max} D_{-1} m_{-1}$, where $\lambda_{\max} = \max_{i\in\mathscr{C}_1}\{\lambda_i\}$, then $\mathbf{w} = \mathbf{0}$ is a feasible point for the primal Problem.

Please refer to [14] for detailed proofs of both theorems.

2.2 H1-SVM problem pormulation

To avoid the zero vector, we proposed a modification of the original SVM optimization problem, which is taking advantage of the previous theorems: the H1-SVM (H1 for one hard class).

H1-SVM Primal Problem

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 - \textstyle\sum_{i\in\mathscr{C}_{\bar{k}}} y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \tag{6}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 \text{ for all } i \in \mathscr{C}_k, \tag{7}$$

where $k = 1$ and $\bar{k} = -1$, or $k = -1$ and $\bar{k} = 1$.
This new formulation constraints Eq. (7) to classify all samples in the class $\mathscr{C}_k$ perfectly, forcing a "hard" convex hull (H1) for $\mathscr{C}_k$. The number of misclassification on the other class $\mathscr{C}_{\bar{k}}$ is added to the objective function, hence the solution is a trade-off between a maximal margin and a minimum number of misclassifications in the "soft" class $\mathscr{C}_{\bar{k}}$.

H1-SVM Dual Formulation

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \tag{8}$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq C_i,\ i \in \mathscr{C}_k, \tag{9}$$

$$\alpha_j = 1,\ j \in \mathscr{C}_{\bar{k}}, \tag{10}$$

$$\textstyle\sum_{i=1}^m \alpha_i y_i = 0, \tag{11}$$

where $k = 1$ and $\bar{k} = -1$, or $k = -1$ and $\bar{k} = 1$.
This problem can be solved in a similar way as the original SVM Problem using the SMO algorithm [8][14], and adding some modifications to force $\alpha_i = 1\ \forall i \in \mathscr{C}_{\bar{k}}$.

Theorem 3: For the H1-SVM the zero solution can only occur if $|\mathscr{C}_k| \geq (n-1)$ and there exists a linear combination of the sample vectors in the "hard" class $\mathbf{x}_i \in \mathscr{C}_k$ and the sum of the sample vectors in the "soft" class, $\sum_{i\in\mathscr{C}_{\bar{k}}} \mathbf{x}_i$.
Proof: Without loss of generality, let the "hard" class be class $\mathscr{C}_1$. Then,

$$\begin{aligned}\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \quad &= \quad \sum_{i\in\mathscr{C}_1} \alpha_i \mathbf{x}_i - \sum_{i\in\mathscr{C}_{-1}} \alpha_i \mathbf{x}_i \\ &= \sum_{i\in\mathscr{C}_1} \alpha_i \mathbf{x}_i - \sum_{i\in\mathscr{C}_{-1}} \mathbf{x}_i. \end{aligned} \tag{12}$$

If we define $\mathbf{z}_i = \sum_{i\in\mathscr{C}_{-1}} \mathbf{x}_i$ and $|\mathscr{C}_1| \geq (n-1) = dim(\mathbf{z}_i) - 1$, there exist $\{\alpha_i\}, i \in \mathscr{C}_1, \alpha_i \neq 0$ such that

$$\mathbf{w} = \sum_{i\in\mathscr{C}_1} \alpha_i \mathbf{x}_i - \mathbf{z}_i \quad = \mathbf{0}.$$

The usual threshold calculation ([9] and [8]) can no longer be used to define the hyperplane, please refer to [14] for details on the threshold computation. The basic algorithm can be improved with some heuristics for greedy "hard"-class determination and tree pruning, shown in [14].

3 Non-linear extension

In order to classify a sample, one simply runs it down the SVM-tree. When using only linear nodes, we already obtained good results [12], but we also observed that first of all, most errors occur in the last node, and second, that over all only a few samples will reach the last node during the classification procedure. This motivated us to add a non-linear node (e.g. using RBF kernels) to the end of the tree. Training of this extended SVM-tree is analogous

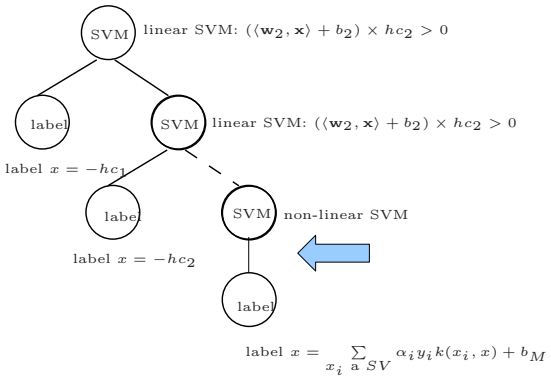

Fig. 4. SVM tree with non-linear extesion

to the original case. First a pure linear tree is build. Then we use a heuristic (trade-off between average classification depth and accuracy) to move the final, non-linear node from the last node up the tree. It is very important to notice, that to avoid overfitting, the final non-linear SVM has to be trained on the entire initial training set, and not only on the samples remaining after the last linear node. Otherwise the final node is very likely to suffer from strong overfitting. Of cause, then the final model will have many SVs, but since only a few samples will reach the final node, our experiments indicate that the average classification depth will be hardly affected.

4 Experiments

In order to show the validity and classification accuracy of our algorithm we performed a series of experiments on standard benchmark data sets. These

experiments were conducted[3] e.g. on *Faces* [10] (9172 training samples, 4262 test samples, 576 features) and *USPS* [11] (18063 training samples, 7291 test samples, 256 features) as well as on several other data sets. More and detailed experiments can be found in [14]. The data was split into training and test sets and normalized to minimum and maximum feature values (Min-Max) or standard deviation (Std-Dev).

Faces (Min-Max)	RBF Kernel	H1-SVM	H1-SVM *Gr-Heu*	RBF/H1	RBF/H1 *Gr-Heu*
Nr. SVs or Hyperplanes	2206	4	4	551.5	551.5
Training Time	14:55.23	10:55.70	14:21.99	1.37	1.04
Classification Time	03:13.60	00:14.73	00:14.63	***13.14***	***13.23***
Classif. Accuracy %	95.78 %	91.01 %	91.01 %	1.05	1.05

USPS (Min-Max)	RBF Kernel	H1-SVM	H1-SVM *Gr-Heu*	RBF/H1	RBF/H1 *Gr-Heu*
Nr. SVs or Hyperplanes	3597	49	49	73.41	73.41
Training Time	00:44.74	00:22.70	02:09.58	1.97	0.35
Classification Time	01:58.59	00:19.99	00:20.07	***5.93***	***5.91***
Classif. Accuracy %	95.82 %	93.76 %	93.76 %	1.02	1.02

Comparisons to related work are difficult, since most publications ([5], [2]) used datasets with less than 1000 samples, where the training and testing time are negligible. In order to test the performance and speedup on very large datasets, we used our own *Cell Nuclei Database*[14] with 3372 training samples, 32 features each, and about **16 million** test samples:

	RBF-Kernel	linear tree H1-SVM	non-linear tree H1-SVM
training time	≈1s	≈3s	≈5s
Nr. SVs or Hyperplanes	980	86	86
average classification depth	-	7.3	8.6
classifiaction time accuracy	≈1.5h 97.69%	≈2 min 95.43%	≈2 min 97.5%

5 Conclusion

We have presented a new method for fast SVM classification. Compared to non-linear SVM and speedup methods our experiments showed a very competitive speedup while achieving reasonable classification results (loosing only marginal when we apply the non-linear extension compared to non-linear

[3] These experiments were run on a computer with a P4, 2.8 GHz and 1G in Ram.

methods). Especially if our initial assumption holds , that large problems can be split in mainly easy and only a few hard problems, our algorithm achieves very good results. The advantage of this approach clearly lies in its simplicity since no parameter has to be tuned.

References

[1] V. VAPNIK, The Nature of Statistical Learning Theory, New York: Springer Verlag, 1995

[2] Y. LEE and O. MANGASARIAN, RSVM: Reduced Support Vector Machines, Proceedings of the first SIAM International Conference onData Mining, 2001 SIAM International Conference, Chicago, Philadelphia, 2001

[3] H. LEI and V. GOVINDARAJU, Speeding Up Multi-class SVM Evaluation by PCA andFeature Selection, Proceedings of the Workshop on Feature Selection for DataMining: Interfacing Machine Learning and Statistics, 2005 SIAM Workshop

[4] C. BURGES and B. SCHOELKOPF, Improving Speed and Accuracy of Support Vector Learning Machines, Advances in Neural Information Processing Systems9, MIT Press, MA, pp 375-381, 1997

[5] K. P. BENNETT and E. J. BREDENSTEINER, Duality and Geometry in SVM Classifiers, Proc. 17th International Conf. on Machine Learning, pp 57-64, 2000

[6] C. HSU and C. LIN, A Comparison of Methods for Multi-Class Support Vector Machines, Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[7] T. K. HO AND E. M. KLEINBERG, Building projectable classifiers of arbitrary complexity, Proceedings of the 13th International Conference onPattern Recognition, pp 880-885, Vienna, Austria, 1996

[8] B. SCHOELKOPF and A. SMOLA, Learning with Kernels, The MIT Press, Cambridge,MA, USA, 2002

[9] S. KEERTHI and S. SHEVADE and C. Bhattacharyya and K. Murthy, Improvements to Platt's SMO Algorithm for SVM Classifier Design, Technical report, Dept. ofCSA, Banglore, India, 1999

[10] P. CARBONETTO, Face data base, http://www.cs.ubc.ca/ pcarbo/, University of British Columbia Computer Science Deptartment

[11] J. J. HULL, A database for handwritten text recognition research, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 16, No 5, pp 550-554, 1994

[12] K. ZAPIEN, J. FEHR and H. BURKHARDT, Fast Support Vector Machine Classification using linear SVMs, in Proceedings: ICPR, pp. 366- 369 ,Hong Kong 2006.

[13] O. RONNEBERGER and et al, SVM template library, University of Freiburg, Department of Computer Science, Chair of Pattern Recognition and Image Processing, http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/index.en.html

[14] K. ZAPIEN, J. FEHR and H. BURKHARDT, Fast Support Vector Machine Classification of very large Datasets, Technical Report 2/2007, University of Freiburg, Department of Computer Science, Chair of Pattern Recognition and Image Processing. http://lmb.informatik.uni-freiburg.de/people/fehr/svm_tree.pdf