FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

Analysis of Support Vector Machine Training Costs for Large and Unbalanced Data from Pharmaceutical Industry

*Tatjana Eitrich, Bruno Lang**

FZJ-ZAM-IB-2005-07

June 2005

(last change: 28.6.2005)

Preprint: submitted for publication

(*) Applied Computer Science and Scientific Computing,
Department of Mathematics, University of Wuppertal, Germany

Analysis of Support Vector Machine Training Costs for Large and Unbalanced Data from Pharmaceutical Industry

T. Eitrich
John von Neumann Institute for Computing,
Central Institute for Applied Mathematics, Research Centre Jülich, Germany,
t.eitrich@fz-juelich.de,
http://www.fz-juelich.de/zam

B. Lang
Applied Computer Science and Scientific Computing,
Department of Mathematics, University of Wuppertal, Germany,
lang@math.uni-wuppertal.de,
http://www.math.uni-wuppertal.de

Abstract

Modern classification methods are able to analyze large, complex and sometimes also unbalanced datasets. It is important not only to produce good results but also to control their time effort. High computational costs lead to the exclusion of data mining methods even in case of good accuracy. Our paper deals with support vector machines in view of the consumption of CPU time. We study their learning behavior for unbalanced datasets with increasing size. We also examine the question whether it is necessary and practicable to parallelize this method.

Keywords: *support vector machines, training time, unbalanced data, parallelization.*

1 Introduction

All branches of industry work with difficult datasets often generated by experiments, measurements or surveys. An important goal of data analysis is the detection of certain hidden structures in datasets. One characteristic part of it is classification. There are several well known classification approaches, for example decision trees, that are effective and well-established data mining algorithms for nonlinear learning. They are well accepted, especially because of their understandability. Parallelization techniques for decision trees were introduced in [14].

This paper deals with one of the latest methods in machine learning. Support vector machines (SVMs) were developed by Vladimir Vapnik and achieved enormous popularity during the past ten years. In the meantime SVMs own a permanent position in the field of machine learning and they even form a recent field of research.

Sometimes large datasets are extremely unbalanced where in general the smaller class is the interesting one. Such datasets occur e.g. in tumor detection. Unbalanced classes can cause problems for classification algorithms in general, especially when there is noise in the data and the boundary between the large negative class and the small positive class is difficult to find [3].

A lot of publications deal with results obtained with support vector machines. In general one tries to outperform other classification methods, for example artificial neural networks. But how can we assess increasing classification accuracy on a test set, if we have to accept longer time to receive it? Therefore one important task is to investigate the changes in running time of support vector machines for different data sizes and parameter values. One aim of this paper is to use an own implementation of an efficient SVM algorithm to train and test on large datasets. Some results will show that the application of SVMs for large datasets is not always easy and parallelization can make sense. So far it is common in practice to avoid this problem. Either the dataset or the number of parameter values for validation are reduced [10, 6]. The results will show that both methods can lead to suboptimal outputs and even to increasing training time when applying for unbalanced data. For this reason some facts about the parallelization of support vector machines will be discussed at the end of this paper.

Section 2 introduces the method of support vector machines. Note that only necessary details will be mentioned. Section 3 presents and interprets some test results for real world unbalanced data from industry. Ideas for parallelizing SVMs will be discussed in Sect. 4. Finally Sect. 5 contains some conclusions.

2 Support Vector Machines for Classification

Support vector machines can be used to implement supervised learning [1], which means to learn input-output dependencies under usage of a certain amount of data pairs. The idea behind this procedure is, that an unknown functional correlation between input and output can approximately be identified in a finite random sample of representative data pairs. Let $n \in \mathbb{N}$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Then the pair (x, y) is called

input-output pair.

Throughout this paper binary classification will be studied, that means $y \in \{-1, 1\}$. Support vector machines are also suitable for multiple classification und for regression [1]. To generate a classification function that will assign any testing point $x \in \mathbb{R}^n$ to one class, supervised learning requires a finite number of input-output pairs (training pairs) $(x^i, y^i)_{i=1,...,l}$ where $l \in \mathbb{N}$ is called the size of the training data.

2.1 Classification Function and Optimization

Let x be a vector with unknown class label. The linear learning approach used by SVMs considers functions of the form

$$f_{\text{linear}}(x) = \sum_{k=1}^{n} w_k x_k + b \quad (w \in \mathbb{R}^n,\ b \in \mathbb{R}) \ . \tag{1}$$

The so far unknown parameters w and b characterize a classification function that can only form linear combinations of the n input values. Of course it is essential to find a way to get a nonlinear classification function. For this reason we look for a transformation ϕ that maps the input data to a feature space F with dimension N. N as well as ϕ are unknown but they yield to

$$f_{\text{nonlinear}}(x) = \sum_{k=1}^{N} w_k \phi_k(x) + b \quad (w \in F,\ b \in \mathbb{R}) \, . \tag{2}$$

Statistical learning theory [15] proves that one has to solve the optimization problem

$$\min_{w \in F, b \in \mathbb{R}} \langle w, w \rangle_F \tag{3}$$

under the constraints

$$y^i \left(\langle w, \phi(x^i) \rangle_F + b \right) \geq 1 \quad (i = 1, ..., l) \tag{4}$$

to get a reasonable classification function. (3) is the separable case, for the nonseparable see e.g. [13] where the parameter C is explained. C is one of the parameters for the tests in Sect. 3.

The resulting Lagrange function has the form

$$L(w, b, \alpha) = \frac{1}{2} \langle w, w \rangle_F \tag{5}$$

By reason of duality theory [13] one can solve the following problem instead of (3):

$$\max_{\alpha \in \mathbb{R}^l} W(\alpha) \tag{6}$$

under the constraints

$$\sum_{i=1}^{l} y^i \alpha_i = 0 \quad \forall i = 1, ..., l, \tag{7}$$

$$\alpha_i \geq 0 \quad \forall i = 1, ..., l, \tag{8}$$

where W has the form

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y^i y^j \alpha_i \alpha_j \langle \phi(x^i), \phi(x^j) \rangle_F . \tag{9}$$

Note that for every $i \in \{1, ..., l\}$ the corresponding value α_i is either zero or strictly positive. Training pairs with strictly positive values for α_i are called support vectors and have influence on the resulting dual classification function

$$f_{\text{dual}}(x) = \sum_{i=1}^{l} y^i \alpha_i \langle \phi(x^i), \phi(x) \rangle_F + b. \tag{10}$$

The advantage of (6) is that it has no local solutions [13]. In addition statistical learning theory provides error bounds for the estimation of classification risks [15]. One can see that the length of the vector α corresponds to the number of used training pairs $(x^i, y^i)_{i=1,...,l}$. By now it is clear where the problem occurs when training SVMs on large datasets.

2.2 Kernel Trick

Optimization of α by maximization of (9) under the constraints (7) and (8) requires the choice of a feature mapping ϕ that is supposed to transform the input data in a way that allows for separation of the training pairs with a linear function. Support vector machines avoid this problem by use of kernels. This means, the dot products $\langle \phi(x^i), \phi(x^j) \rangle_F$ in (9), that cannot be evaluated without knowledge of ϕ, are replaced by the function values $k(x^i, x^j)$ of a specific kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. These values generate the kernel matrix K where $K_{ij} = k(x^i, x^j)$.

The usage of a kernel makes it possible to work solely with the original data format. The difficulty lies in the choice of a kernel as the function k has to meet some conditions [1]. On the other hand it has to be suitable for the special dataset. In general the solution to this dilemma is a standard kernel [1]. For our tests the Gaussian kernel

$$K(x^i, x^j) = \exp\left(- \sum_{k=1}^{n} \frac{(x_k^i - x_k^j)^2}{2\sigma^2} \right) \tag{11}$$

was chosen. Note that the kernel parameter $\sigma > 0$ is the second parameter used in all tests.

3 Experimental Evaluation

For all tests we used the iterative nearest point algorithm [8], that was first introduced in 1999. This algorithm implements a support vector machine for the 2-norm soft margin approach [1]. We also tested the sequential minimal optimization algorithm (SMO [12]), that implements 1-norm soft margin [1], but it consumed more training time on every dataset and so the results are omitted. Our Fortran90 code was compiled with *ifort*[1]. The tests were run on an Intel Pentium 4 processor with 2,4 GHz and a cache size of 512 KB.

The underlying data was obtained from pharmaceutical industry and is aimed at *QSAR* modeling. The complete dataset consists of 40000 data pairs, 20 attributes and 2 classes of active and nonactive points. Only 38 of the pairs belong to class 1. Thus the dataset is extremely unbalanced. To our knowledge such data is not typical for publications dealing with SVM classification. For our tests the number of attributes has been reduced from 20 down to 5 with a step size of 5 attributes, the number of data pairs has been reduced from 40000 to 200 with a step size of 4000 and some smaller steps at the end, but all 38 pairs in class 1 have been saved. We always used 50% of a set for training. The other pairs were used for the test. Please note that the aim was not to produce optimal classification results, because this would lead to different parameters C and σ for all sizes of the training matrix $A = (x^1, x^2, ..., x^l) \in \mathbb{R}^{n,l}$.

3.1 Number of Input-Output Pairs

In principle the training time of support vector machines is quadratic in l [16, 17]. Furthermore it is known that the fraction of support vectors is small [13], but [7] shows that sometimes nearly all training points can become support vectors. As we do not store the kernel matrix the nearest point algorithm as well as other SVM algorithms (e.g. SMO) have to evaluate the kernel function everytime they need a kernel value. For this reasons the following test results contain running time as well as the absolute and relative frequency of support vectors and the number of kernel evaluations during training.

Table 1 shows that training time t is a not a quadratic function of l. The functional dependency can approximatively be formulated as

$$\log(t) \approx 1,5 \cdot \log(l \cdot 10^{-3}) - 1,4. \qquad (12)$$

In our example the number of kernel evaluations ke is also proportional to the number of training pairs, we propose

$$\log(ke \cdot 10^{-3}) \approx 1,25 \cdot \log(l \cdot 10^{-3}) + 0,59. \qquad (13)$$

[1] Intel Fortran Compiler for Linux.

Table 1: Influence of l with constant $C = 10$, $\sigma = 2$ and 10 attributes

l	100	1000	2000	4000	6000	8000	10000	12000
time[2]	0,02	0,35	0,95	2,40	3,62	5,38	7,29	9,72
support vectors[3]	66	174	208	249	271	300	328	347
% support vectors[4]	66,0%	17,4%	10,4%	6,2%	4,5%	3,8%	3,3%	2,9%
kernel evaluations[5]	0.124	2.035	5.103	11.668	16.344	22.547	29.803	37.304

14000	16000	18000	20000
12,32	18,80	22,27	27,37
369	374	389	414
2,6%	2,3%	2,2%	2,1%
44.652	57.864	73.420	85.006

Figure 1 shows a plot of our test cases. Please note that the number of data points includes training as well as test points.

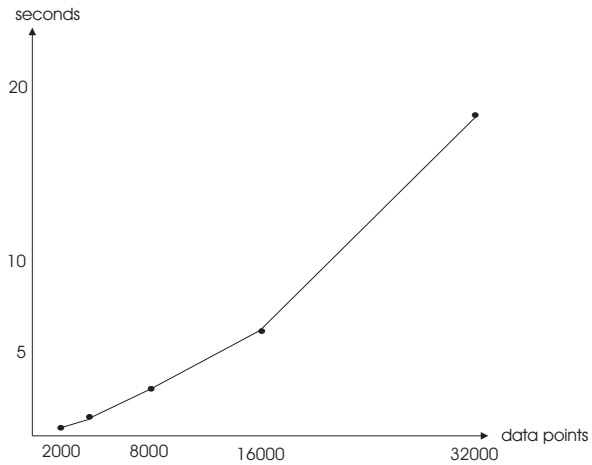

Figure 1: SVM running time as a function of the number of data points

3.2 Number of Attributes

Training time of support vector machines is said to be linear in n [16, 17]. Table 2 shows a reverse character for the unbalanced dataset. This fact is important

[2] time for complete training and testing in seconds

[3] absolute frequency of support vectors in training data

[4] ratio of support vectors and training points

[5] sum of all kernel evaluations during training divided by 10^3

Table 2: Influence of n for constant $C = 10$, $\sigma = 2$ and 4000 training points

n	5	10	15	20
time	7, 39	2, 42	1, 60	1, 39
support vectors	672	249	226	226
% support vectors	16, 8%	6, 2%	5, 7%	5, 7%
kernel evaluations	45.843	11.668	7.093	5.977

since often one tries to reduce the number of attributes before starting support vector training. For the tests presented here this technique would result in more support vectors and kernel evaluations. We reduced the number of attributes in a way, that always preserved the most important attributes. The tests with $n = 5$ and $n = 10$ have the same accuracy, but training times differ by a factor of 3 plus the time spent for searching the important 5 attributes out of 10. By now it is clear that feature selection can also have negative effects and it is necessary to examine pros and cons of data reduction algorithms.

Figure 2 shows a plot of our test cases.

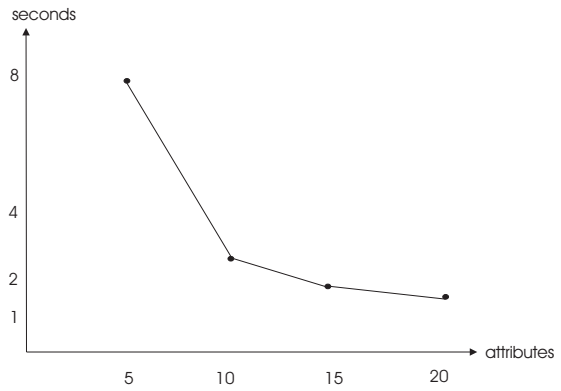

Figure 2: SVM running time as a function of the number of attributes

3.3 Parameter C

C is an important parameter for the application of support vector machines. It is recommended to choose small values for C as in general larger values take longer training time. The following results, see Table 3, show that smaller values for C sometimes result in longer training time as well as more support vectors and kernel evaluations. Note that $C = 100$, that led to minimal training time and minimal number of kernel evaluations, is the optimal value in view of accuracy. Possibly these characteristics can help to produce good classification results. In general it makes no sense to avoid large values for C as we are always interested in high accuracy.

Table 3: Influence of C for constant $\sigma = 2$ and a 4000×10 training matrix

C	0.1	1
time	4, 86	2, 63
support vectors	1407	529
% support vectors	35, 2%	13, 2%
kernel evaluations	30.660	13.720

10	100	1000
2, 39	1, 99	3, 22
249	128	90
6, 2%	3, 2%	2, 3%
11.668	6.829	9.091

3.4 Implications

The tests showed that for our dataset running time is directly proportional to l but not to n. In addition there are important dependencies between accuracy, training time, fraction of support vectors and the number of kernel evaluations, see e.g. Table 3, where the training for $C = 100$ with the smallest number of kernel evaluations led to the highest accuracy. In this paper we don't show accuracy valules, since non-optimal parameter values have been used for comparision of training time. For the performance of SVM's in real life unbalanced data we refer to our work [9]. We have to mention that often good parameter values led to the smallest training times and thus we conclude, training time can be seen as key-figure for SVM model selection, too. We also conclude that reduction of the data matrix in one or both dimensions and changes in the value for C do not always lead to dramatical changes of time in a certain direction. At first they result in different mathematical models. The benignity of such models influences running time. For practice it is important to understand that feature selection and data splitting can cause suboptimal classification functions with nearly the same training time.

4 Parallelization

By now it is clear that for reason of accuracy it is best to use the entire information a dataset provides to the user. It goes without question that training and testing time can be a problem for very large datasets. Parameter optimization is very time consuming, too. Usually the optimal parameter values are unknown.

Assuming one has to train a support vector machine with Gaussian kernel and two parameters for C (C^+ and C^-) [11]. (A value for C^+ that is greater than the value for C^- supports the small positive class.) If 10 values for σ, 5 for C^+ and also 5 values for C^- are used to perform tenfold cross validation,

the task is to train the SVM 2500 times on 90 percent of the training data and test it on the remainder respectively.

The example above points out the importance of self-acting and parallel parameter search to ensure that chosen parameters are close to the optimal values. This type of parallelization is trivial but in addition to the idea of grid-search [6] it makes sense not to test all intended combinations. At first the intervals should be large, after some parallel training steps they should get smaller. This procedure guarantees maximum usage of resources and allows e.g. to perform a multiple classification task using the one against one approach [5] with acceptable running time and optimal results. Other parallel parameter optimization methods can be used, too [4].

But there is another approach. For very large datasets it can make sense to parallelize training as well as testing. For the test stage it means again farming because every processor uses the same classification function for its test points independently. In contrast to farming tasks implementation of a parallel SVM training is difficult.

The objective function of the optimization problem is

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j Q_{ij}, \qquad (14)$$

with

$$Q_{ij} = y^i y^j K_{ij} \quad (1 \leq i, j \leq l). \qquad (15)$$

The Gram matrix $Q \in \mathbb{R}^{l,l}$ is symmetric [1] and can be transformed to a block diagonal matrix, that splits the problem into independend tasks [2] and allows parallel handling of the optimization process.

In general the transformation is not easy, because it is necessary to generate a sensible number of blocks with nearly the same size to operate at full capacity. But there is another problem concerning (7). The constraint cannot be divided trivially into subproblems. In [2] this problem is not mentioned.

Another promising parallelization technique presented in [18] uses parallel handling of time consuming matrix operations to speed up SVM training of decomposition methods, nevertheless the are still some bottlenecks to consider.

5 Conclusion and Outlook

Support vector learning is very sensitive regarding the quality of the training data and the parameter values, especially for unbalanced data. For real life data with a very small amount of positive points it is essential to use all training points, if possible. Pruning of training data can lead to suboptimal decision functions that are not able to classify unseen points, especially the positives. For this reason it is of great interest to study parallelization techniques for SVMs. These may include parallel parameter search, parallel validation steps and parallel training. We showed that the farming approach for parameter optimization and for classification of test points is easy, but the implementation of a parallel training algorithm is complicated and depends on the characteristics of the Gram matrix. Our future work will concentrate on SVM's with different stages of parallelism specialized for large and unbalanced data from real world applications with a certain amount of noise.

Acknowledgements

We would like to acknowledge the support by Research Centre Jülich and the Grünenthal Group.

References

[1] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[2] J. Dong, A. Krzyzakak, and C. Suen. A fast parallel optimization for training support vector machine. In P. Perner and A. Rosenfeld, editors, *Proceedings of 3rd International Conference on Machine Learning and Data Mining*, pages 96–105, 2003.

[3] Tatjana Eitrich and Bruno Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. Preprint BUW-SC 2005/2, University of Wuppertal, May 2005.

[4] Tatjana Eitrich and Bruno Lang. Parallel tuning of support vector machine learning parameters for large and unbalanced data sets. Preprint BUW-SC 2005/7, University of Wuppertal, May 2005.

[5] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[6] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification, 2003.

[7] S. S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. *IEEE Transactions on Neural Networks*, 13:1225–1229, 2002.

[8] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest

point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, January 2000.

[9] Achim Kless and Tatjana Eitrich. Cytochrome p450 classification of drugs with support vector machines implementing the nearest point algorithm. In Jesús A. López, Emilio Benfenati, and Werner Dubitzky, editors, *Knowledge Exploration in Life Science Informatics, International Symposium,KELSI 2004, Milan, Italy, November 25-26, 2004, Proceedings*, volume 3303 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2004.

[10] K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.

[11] David R. Musicant, Vipin Kumar, and Aysel Ozgur. Optimizing F-measure with support vector machines. In Ingrid Russell and Susan M. Haller, editors, *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference, May 12-14, 2003, St. Augustine, Florida, USA*, pages 356–360. AAAI Press, 2003.

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

[13] Bernhard Schölkopf and Alexander J. Smola. *Learning With Kernels*. MIT Press, Cambridge, MA, 2002.

[14] A. Srivastava, E. Han, V. Kumar, and V. Singh. Parallel formulation of decision-tree classification algorithms. *Data Mining and Knowledge Discovery*, 3(3):237–261, 1999.

[15] Vladimir N. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

[16] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 306–315. ACM, 2003.

[17] Hwanjo Yu, Jiong Yang, Wei Wang, and Jiawei Han. Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In *2nd IEEE Computer Society Bioinformatics Conference (CSB 2003), 11-14 August 2003, Stanford, CA, USA*, pages 220–228. IEEE Computer Society, 2003.

[18] Gaetano Zanghirati and Luca Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing*, 29(4):535–551, 2003.