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

Technical Report

On the Optimal Working Set Size in Serial and Parallel Support Vector Machine Learning with the Decomposition Algorithm

*Tatjana Eitrich, Bruno Lang**

FZJ-ZAM-IB-2006-11

August 2006
(last change: 10.8.2006)

Preprint: submitted for publication

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

On the Optimal Working Set Size in Serial and Parallel Support Vector Machine Learning with the Decomposition Algorithm

Tatjana Eitrich[1] and Bruno Lang[2]

[1] Central Institute for Applied Mathematics, Research Centre Jülich, Germany
t.eitrich@fz-juelich.de
[2] Applied Computer Science and Scientific Computing Group, Department of Mathematics, University of Wuppertal, Germany

Abstract. The support vector machine (SVM) is a well-established and accurate supervised learning method for the classification of data in various application fields. The statistical learning task – the so-called training – can be formulated as a quadratic optimization problem. During the last years the decomposition algorithm for solving this optimization problem became the most frequently used method for support vector machine learning and is the basis of many SVM implementations today. It is characterized by an internal parameter called working set size. Usually small working sets have been used. The increasing amount of data used for classification led to new parallel implementations of the decomposition method with efficient inner solvers. With these solvers larger working sets can be assigned. It was shown that for parallel training with the decomposition algorithm large working sets achieve good speedup values. However, the choice of the optimal working set size for parallel training is not clear. In this paper we show how the working set size influences the number of decomposition steps, the number of kernel function evaluations and the overall training time in serial and parallel computation.

1 Introduction

The support vector machine for classification and regression is a powerful machine learning method. Its popularity is mainly due to the applicability in various fields of data mining, such as text mining [1], biomedical research [2], and many more. SVM test accuracy is excellent and in many cases it outperforms other machine learning methods such as neural networks. SVM has its roots in the field of statistical learning which provides the reliable generalization theory [3]. Several properties that make this learning method successful are well-known, e.g. the kernel trick [4] for nonlinear classification and the sparse structure of the final classification function. In addition, SVM has an intuitive geometrical interpretation, and a global minimum can be located during the training phase.

Most current SVM implementations are based on the well known decomposition algorithm for solving the optimization problem of SVM training [5]. It

repeatedly selects a subset of the free variables and optimizes over these variables. Thus, decomposition provides a framework for handling large SVM training tasks, where the kernel matrix does not fit into the available memory. Its main advantage is the flexibility concerning the size of the subproblems – the working set size. All values larger than one and smaller or equal to the training set size are possible. The limitation for large working sets is due to memory requirements of the machine and the characteristics of the inner solver. A widely used decomposition method called SMO [6] uses the extreme case of only two free variables in each iteration. Other approaches use larger working sets. However, the optimal choice of the working set size is not clear, especially for large data sets. In this paper we will show, how the training time is influenced by the size of the subproblems for the inner solver.

Real world data sets are becoming increasingly large. The main drawback of current SVM models is their high computational complexity for large data sets [7]. Parallel processing is essential to provide the performance required by large-scale data mining tasks. Therefore the development of highly scalable parallel SVM algorithms is a new important topic of current SVM research. Recently parallelized decomposition algorithms have been proposed. For parallel computation large working set sizes are possible and lead good speedup values. However, it is not clear which influence the working set size has onto the overall training time for large data sets in serial and parallel mode. One goal of this paper is to show how the working set size controls the performance of SVM training in general. In addition the results of serial computation are broaden to parallel computations.

The paper is organized as follows. In Sect. 2 we review basics of binary SVM classification. We limit the discussion to the issues that are essential for understanding the following sections. In Sect. 3 we describe the scheme of the serial decomposition algorithm and explain the influence of the working set size. The parallelization of the support vector machine learning method is explained in Sect. 4. In Sect. 5 we show results of various tests using small and large data sets in serial and parallel mode. In view of overall learning time we discuss the issue of the optimal working set size for parallel SVM training.

2 Basic Concepts of the Support Vector Machine

Support vector learning means to determine functions that can be used to classify data points. Here, we discuss binary classification, but the SVM learning framework also works for multi-class and regression problems [8]. The supervised SVM learning method is based on so-called reference data of given input–output pairs (training data)

$$(\boldsymbol{x}^i, y_i) \in \mathbb{R}^n \times \{-1, 1\}, \quad i = 1, \ldots, l,$$

that are taken to find an optimal separating hyperplane [9]

$$f_{\mathrm{l}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = 0.$$

Using assumptions of statistical learning theory the desired classifier is then defined as

$$h(\boldsymbol{x}) = \begin{cases} +1, \text{ if } f_{\text{l}}(\boldsymbol{x}) \geq 0, \\ -1, \text{ if } f_{\text{l}}(\boldsymbol{x}) < 0, \end{cases}$$

with the linear decision function f_{l}, see Fig. 1.

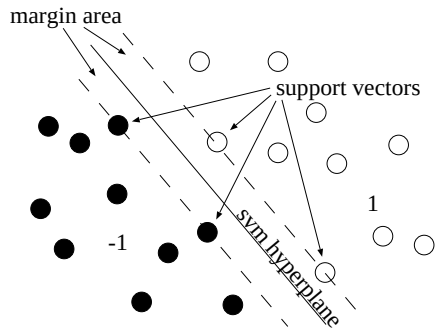

Fig. 1. Support vector machine classification function for given training data.

If the two classes are not linearly separable then f_{l} is replaced with a nonlinear decision function [10]

$$f_{\text{nl}}(\boldsymbol{x}) = \sum_{i=1}^{l} y_i \alpha_i K(\boldsymbol{x}^i, \boldsymbol{x}) + b,$$

where $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a (nonlinear) kernel function [4]. The classification parameters α_i can be obtained as the unique global solution of a suitable (dual) quadratic optimization problem [10]

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^l} \quad g(\boldsymbol{\alpha}) := \frac{1}{2} \boldsymbol{\alpha}^T H \boldsymbol{\alpha} - \sum_{i=1}^{l} \alpha_i \tag{1}$$

with $H \in \mathbb{R}^{l \times l}$, $H_{ij} = y_i K(\boldsymbol{x}^i, \boldsymbol{x}^j) y_j$ $(1 \leq i, j \leq l)$, constrained to

$$\boldsymbol{\alpha}^T \boldsymbol{y} = 0, \quad 0 \leq \alpha_i \leq C.$$

In the final solution only a part of the entries in $\boldsymbol{\alpha}$ are positive, whereas all others are zero. This is due to the Karush-Kuhn-Tucker conditions for convex optimization problems. In SVM theory the corresponding training points are called support vectors, see Fig. 1. The Hessian H is usually dense, and therefore the complexity of evaluating the objective function g in (1) scales quadratically

with the number l of training pairs, leading to very time-consuming computations. The parameter C controls the trade-off between the width of the classifier's margin and the number of weak and wrong classifications in the training set. This parameter has to be chosen by the user. Due to space limitations we omit a detailed introduction to the SVM theory. For readers who are not familiar with this topic we refer to the tutorial [11].

3 Decomposition and the Working Set Size

The high cost for solving (1) is due to the size and the density of the quadratic matrix H. Classical solvers for QP problems with simple constraints are the so called active set methods [12, 13], which in each iteration minimize the objective function over the active set (a subset of the constraints that are locally active), until a solution is reached. A variation of the active set approach for support vector machine training is the well known decomposition method described in [14]. It repeatedly splits the original optimization problem (1) into active and inactive parts. In each step, the active data points (or working set points) are used to define a new QP subproblem which is then handled by an inner solver. This method is very flexible since the user can choose the desired number of active points $\tilde{l} \leq l$ to control the size of the QP subproblems. Due to space limitations the decomposition algorithm cannot be discussed here in detail, we refer to [5, 15, 16] for a detailed description. To summarize, in each iteration of the decomposition algorithm the following five steps need to be processed:

1. Select a working set of $\tilde{l}$ "active" variables from the l free variables α_i.
2. Solve the quadratic subproblem of size $\tilde{l}$ that results from restricting the optimization in (1) to the active variables and fixing the remaining variables.
3. Update the global solution vector $\boldsymbol{\alpha}$.
4. Update the gradient of the overall problem.
5. Check a stopping criterion.

Implementation of a sophisticated working set selection scheme, an efficient subproblem solver and a fast but accurate gradient update are crucial for good overall performance of decomposition methods. Several approaches have been proposed and improved during the last decade. Promising suggestions are given in

- [17] – for the working set selection,
- [18, 19] – for fast inner solvers,
- [20, 21] – for sparse gradient updates.

As already mentioned, one important model parameter of the decomposition method is the working set size $\tilde{l}$. In is known, that for larger working sets the number of decomposition steps will decrease, but a single step may be more

expensive. In contrast for small working sets the quadratic subproblem may be solved very fast, but the number of steps will increase heavily. Choosing $\tilde{l} = 2$ results in the well-known Sequential Minimal Optimization (SMO) scheme [6]. SMO decomposition steps are fast, but this method suffers from a very large number of optimization steps even when using moderate problem sizes. Today, it is not clear which working set sizes are preferable. In this work we will analyze the overall behavior of this two competing effects. The training time not only depends on the working set selection scheme and the inner solver, but also on the data set and the characteristics of the machine used for running the software. However, as we will show in Sect. 5, the working set size has enourmous influence onto the training time of SVM training and needs to be optimized primarily. We will show that indeed a global minimum of the training time does exist and needs to be located, especially when using expensive parallel computing resources.

4 Parallel Support Vector Machine Decomposition

The SVM training, i.e. the solution of the quadratic program (1), suffers from large data sets [22]. Since data sets are becoming increasingly large in various fields of research, e.g. in text mining, parallel SVM training is essential to improve performance. The development of parallel SVM algorithms is a young and emerging field of research and, as we recently discussed in [23], the number of real parallel implementation is rare. In this work we target an already fruitful approach for serial and parallel SVM training with the decomposition method. In [20, 21] a parallelized MPI-based decomposition algorithm has been proposed. It is based on a variable projection method as inner solver [24, 25]. In [26] we have presented an implementation of parallel support vector machine decomposition training for shared memory systems based on this solver. The parallel SVM algorithm uses library and loop parallelism. Calls to the ESSLSMP library (Engineering Scientific Subroutine Library for Shared Memory Parallel Machines) [27] as well as OpenMP loop level parallelism lead to a scalable training method. In both approaches the main parallel parts of the decomposition method belong to the time consuming computations in each step [26], i.e.

1. the computation of the kernel matrix for the new subproblem,
2. expensive matrix-vector multiplications in the decomposition routine and the inner solver,
3. the gradient update for the overall optimization problem.

For details to the parallelization schemes we refer to [20] and [26]. Tests for both packages were run using large working sets, which led to promising speedup values, e.g. for $\hat{l} = 5000$ in [26] and 3600 in [20]. However, the improved serial algorithms have not been tested for smaller data sets or large data sets with small working set sizes. This aspect needs to be analyzed and will bring improvements for serial as well as parallel support vector machine training.

5 Experimental Results

We performed our tests on the Juelich Multi Processor (JUMP) [28] using our parallel SVM software which also provides a parallel validation loop [23]. JUMP is a distributed shared memory parallel computer consisting of 41 frames (nodes). Each node contains 32 IBM Power4+ processors running at 1.7 GHz, and 128 GB shared main memory. The 1312 processors have an aggregate peak performance of 8.9 TFlop/s. Our software is written using Fortran90 and the ESSLSMP library. For each thread we chose the following characteristics:

- 3.5 GB consumable memory,
- 3.0 GB data limit,
- 0.5 GB stack limit,
- 1h wall clock limit.

5.1 Description of the Data Sets

The so-called *australian* data set is available from [29]. It containes credit card applications with 14 attributes – 6 numerical and 8 categorical. The number of Instances is 690. The class distribution is 44.5% vs. 55.5%. The *fourclass* data set was introduced in [30]. It is artificial and is aimed at testing the performance of classifiers. 862 data points in a two-dimensional space have been assigned to four classes in the original data set. The classes are distributed irregularly and show isolated regions to complicate classification. The data set was transformed into a binary classification problem and is available from [31]. The *adult* data set is available unter [29]. The task is to predict whether a household has an income greater than $50000 [6]. Originally 14 attributes of a census form of a household were given. We use the data set in the discretized form with 123 binary features, which is available from [31] under the name $a9a$. 32561 training points are available. In this paper we use two versions of the data – the whole data set as well as a subset of 15000 points which we call *adultpart*.

	australian	*fourclass*	*adultpart*	*adult*
# features	14	2	123	123
# training points	690	862	15000	32561
# positive points	307	307	3562	7841
# negative points	383	555	11438	24720

Table 1. Characteristics of the data sets.

5.2 Influence of the Working Set Size for Serial Computation

In Tables 2 and 3 we show the number of decomposition steps D, the number of kernel function evaluations E, the number of support vectors sv as well as the

training time t (in seconds) assigning various working set sizes for the *australian* as well as the *fourclass* data set. For a better interpretation we show the plot of the training times (with some more values) in Fig. 2. Starting with 4 active

$\hat{l}$	4	6	10	20	50	80	100	120	150	200	350	500	600	650	690
$\#D$	7455	6193	4349	1882	543	90	31	14	9	7	5	3	4	3	1
$\#E$	19.9	24.2	28.8	25.5	18.9	5.1	2.2	1.2	0.9	0.9	1.1	1.4	1.4	1.6	0.7
$\#sv$	246	247	246	246	247	247	247	246	247	247	247	247	246	247	247
t	5.60	6.27	7.58	7.35	7.08	2.78	1.24	**0.94**	1.01	2.89	5.88	12.48	18.17	23.54	7.55

Table 2. Results for the *australian* data set.

$\hat{l}$	4	10	20	30	50	100	200	300	400	500	600	700	800	850	862
$\#D$	2260	640	41	25	15	13	5	4	4	5	3	4	3	2	1
$\#E$	7.76	5.33	0.52	0.39	0.34	0.42	0.39	0.54	0.81	1.43	1.22	2.11	2.04	1.55	0.83
$\#sv$	98	98	98	98	98	98	98	98	98	98	98	98	98	98	98
t	1.90	1.42	0.28	**0.26**	0.72	1.10	4.28	5.89	5.01	16.18	12.87	21.78	16.93	20.20	9.21

Table 3. Results for the *fourclass* data set.

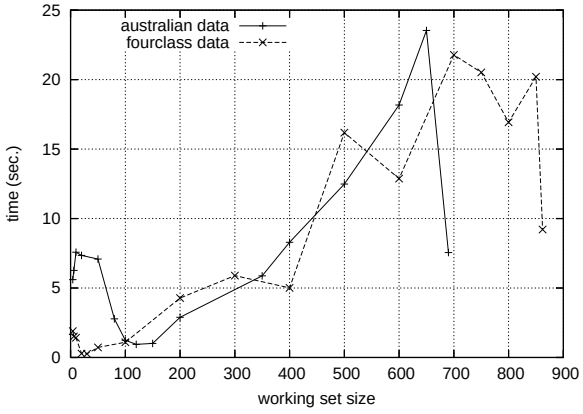

Fig. 2. Training times for different working set sizes (small data sets).

points we increased the working set size until the maximal value (the whole training set) was reached. The results show similar behavior. For small values

of $\hat{l}$ the number of decomposition steps is large and decreases with increasing $\hat{l}$. The number of kernel evaluations is not monotone in such a way. For both data sets there is an interval of working set sizes that lead to small numbers of kernel evaluations. For the australian data set this interval is approx. $[120, 350]$ and for the fourclass data set $[30, 200]$. Please note that for both data sets the largest possible value of $\hat{l}$ led to another minimum for the sum of kernel evaluations. This is due to the fact, that in this extreme case only a single decomposition step is required which saves a lot of overhead. This is an interesting result. However, the choice of $\hat{l} = l$ seems not to be useful. First, it does not always lead to a global minimum of kernel computations as for the fourclass data set and second the training times are not optimal. Although training times show local minima for $\hat{l} = l$ the optimal training times for both data sets are smaller. For the fourclass data set we can save a factor of 40 in training time. This behavior comes from the fact that we implemented a sparse gradient update, as it was introduced in [20]. Thus, each iteration of the decomposition method is extremely cheap for small data sets. It seems that together with a fast inner solver and sophisticated working set selection the small working set sizes still beat the bigger ones. Please note that no accuracy values are given. We did not run any tests, but used the whole data sets for training. Concerning accuracy of SVM and parameter fitting we refer to [32, 33].

$\hat{l}$	10	50	100	300	500	650	800	1000	1400	2000	2400	3000	3500	4000
$\#D$	6094	1465	563	109	47	32	28	23	18	13	12	10	9	7
$\#E$	810	1028	777	402	260	212	213	213	222	228	243	260	275	261
$\#sv$	5526	5549	5546	5546	5541	5547	5539	5544	5539	5543	5542	5558	5550	5555
t	310	375	286	156	109	**97**	108	122	187	316	443	639	925	1116

Table 4. Results for the *adultpart* data set.

$\hat{l}$	50	100	300	500	650	800	1200	1600	2000	2400	3000	3500
$\#D$	4472	1774	332	145	100	70	40	35	27	22	18	16
$\#E$	6803	5428	2832	1903	1591	1310	1013	1062	1042	992	1001	1031
$\#sv$	11779	11758	11768	11759	11755	11766	11766	11752	11771	11765	11751	11782
t	2477	1986	1063	874	681	616	**592**	793	962	1064	1477	2176

Table 5. Results for the *adult* data set.

In Tables 4 and 5 we show the number of decomposition steps D, the number of kernel function evaluations E, the number of support vectors sv as well as the training time t (in seconds) for the *adultpart* as well as the *adult* data set using different working set sizes. The corresponding plot with some more values is

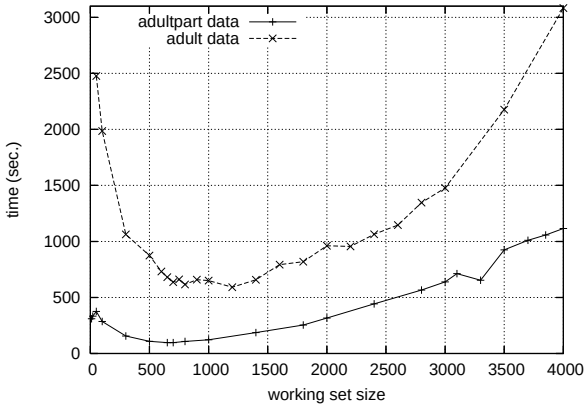

Fig. 3. Training times for different working set sizes (huge data sets).

shown in Fig. 3. Results for both data sets show similar behavior. The training time curve is convex and has a minimum at $\hat{l} = 650$ for the *adultpart* and $\hat{l} = 1200$ for the *adult* data set. The numbers of kernel evaluations are high for small data sets and decrease for increasing working set sizes. They reach a first minimum for the minimal training times, but, the they remain somehow stable while the training time increases dramatically. This is caused by the inner solver and gradient update costs. Working set sizes between 500 and 1400 led to acceptable training times and for the largest training set larger working set sizes are preferable.

5.3 Influence of the Working Set Size for Parallel Computation

The parallel decomposition scheme is based on parallel computations of the kernel matrix and parallel gradient update in each decomposition step as well as parallel matrix-vector multiplications in the inner solver. Usually applied to very large working sets [20, 34] we showed that smaller working set sizes lead to better results in the serial case. In this section we will broaden our analysis to parallel training and run the same tests for the largest data set (*adult*) to show the parallel behavior of the decomposition method for smaller working set sizes. In Table 6 some results for parallel runs using 1, 2 and 4 threads are given. With $t(\cdot)$ we denote the training time against the number of threads and with $s(\cdot)$ the corresponding speedup value, where the speedup is defined in the usual way as $s(n) = t(1)/t(n)$, where n is the number of threads. In Fig. 4 we show the corresponding plots for all our tests. The minimal training time for the parallel runs is again achieved with $\hat{l} = 1200$. The speedup values in this area are comparable to those with larger working sets. Thus, working set sizes around 1000 are preferable in this case since they lead to the best (smallest) training times in serial as well an parallel mode.

$\hat{l}$	50	100	300	500	650	800	1000	1200	1600	2000	2400	2800	3000	3500
$t(1)$	2477	1986	1063	874	681	616	651	**592**	793	962	1064	1346	1477	2176
$t(2)$	1141	1006	571	460	388	370	379	**347**	458	558	617	795	879	1224
$s(2)$	2.2	2.0	1.9	1.9	1.9	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.9
$t(4)$	659	610	318	276	234	201	200	**191**	250	318	356	457	486	713
$s(4)$	3.8	3.3	3.3	3.2	2.9	3.1	3.3	3.1	3.2	3.0	3.0	2.9	3.0	3.1

Table 6. Parallel training results for the *adult* data set with 1, 2 and 4 threads.

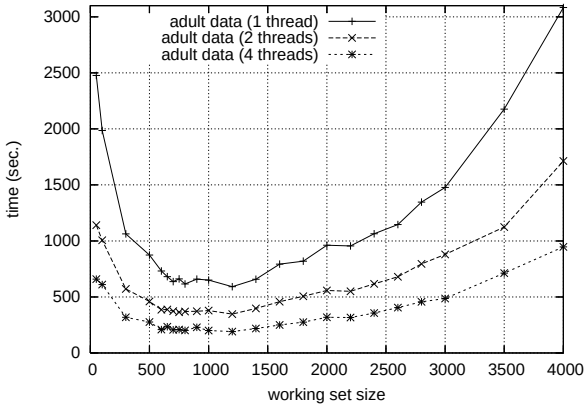

Fig. 4. Training times for different working set sizes with 1 and 2 threads (*adult* data set).

In parallel data mining the interest is in efficiently using the available resources. In our tests we observed acceptable speedup values for all working set sizes we had chosen. This is due to the fact that the parallel kernel matrix evaluation is perfectly scalable and the problem size for the parallel gradient update is not dependent on the working set size, so that the parallel scheme works fine. From our tests we conclude that the optimal working set size is indeed dependent on the data set size and increases for large data sets. Very large working sets lead to high training times in general. In our tests we observed global minima for the training time, that are smaller than we expected so far.

6 Summary and Future Work

We have analyzed the decomposition algorithm for SVM training with a fast inner solver. For several small and large data sets we have tested how the working set size influences the training time. For small data sets we observed enormous differences, whereas the variability was smaller for the large data sets. However, differences of one order of magnitude may occur easily if choosing a non-optimal

working set size. For small optimal working sets like in our results the attainable speedups for the parallel SVM training will be limited due to the fact that the efficiency of parallel matrix-vector multiplications decreases with increasing numbers of CPUs or threads. However, this is not a critical point in SVM learning. As expensive parameter tuning experiments need to be done, remaining CPUs can be assigned to either parallel cross validation schemes [23] or to parallel parameter optimization methods [35].

In the future we will continue with the first experiments presented in this paper using even larger data sets. In addition, we will work on methods that automatically compute the optimal working set size for parallel SVM learning depending on the data set, the characteristics of the machine used as well as the validation and/or parameter optimization scheme.

References

1. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Proceedings of ECML. Volume 1398 of Lecture Notes in Computer Science., Springer (1998) 137–142
2. Yu, H., Yang, J., Wang, W., Han, J.: Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In: CSB 2003, IEEE Computer Society (2003) 220–228
3. Vapnik, V.N.: Statistical learning theory. John Wiley & Sons, New York (1998)
4. Schölkopf, B.: The kernel trick for distances. In: Advances in Neural Information Processing Systems 13, MIT Press (2001) 301–307
5. Hsu, C.W., Lin, C.J.: A simple decomposition method for support vector machines. Machine Learning **46**(1-3) (2002) 291–314
6. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods — Support Vector Learning, MIT Press (1999) 185–208
7. Chen, N., Lu, W., Yang, J., Li, G.: Support vector machine in chemistry. World Scientific Pub Co Inc (2004)
8. Hsu, C., Lin, C.: A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks **13** (2002) 415–425
9. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
10. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press (2002)
11. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**(2) (1998) 121–167
12. Fletcher, R.: Practical methods of optimization, Vol II: constrained optimization. John Wiley & Sons (1981)
13. Leyffer, S.: The return of the active set method (2005) to appear in Oberwolfach Report.
14. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: IEEE Workshop on Neural Networks and Signal Processing. (1997)
15. Chang, C.C., Hsu, C.W., Lin, C.J.: The analysis of decomposition methods for support vector machines. IEEE Transactions on Neural Networks **11**(4) (2000) 1003–1008

16. Laskov, P.: Feasible direction decomposition algorithms for training support vector machines. Machine Learning **46**(1-3) (2002) 315–349
17. Serafini, T., Zanni, L.: On the working set selection in gradient projection-based decomposition techniques for support vector machines. Optimization Methods and Software **20**(4-5) (2005)
18. Ruggiero, V., Zanni, L.: An overview on projection-type methods for convex large-scale quadratic programs. Nonconvex Optimization and Its Applications **58** (2001) 269–300
19. Ruggiero, V., Zanni, L.: On the efficiency of splitting and projection methods for large strictly convex quadratic programs. Applied Optimization (1999) 401–413
20. Zanghirati, G., Zanni, L.: A parallel solver for large quadratic programs in training support vector machines. Parallel Computing **29**(4) (2003) 535–551
21. Serafini, T., Zanghirati, G., Zanni, L.: Parallel decomposition approaches for training support vector machines. In: Proceedings of ParCo 2003. (2004) 259–266
22. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: the cascade SVM. In: Advances in Neural Information Processing Systems 17. (2005) 521–528
23. Eitrich, T., Frings, W., Lang, B.: HyParSVM – a new hybrid parallel software for support vector machine learning on SMP clusters. In: Proceedings of EuroPar 2006, Dresden (to appear). (2006)
24. Zanghirati, G., Zanni, L.: Variable projection methods for large quadratic programs in training support vector machines. Preprint 339, Dept. of Math., University of Ferrara, Italy (2003)
25. Ruggiero, V., Zanni, L.: Variable projection methods for large convex quadratic programs. Recent Trends in Numerical Analysis (2000) 299–313
26. Eitrich, T., Lang, B.: Shared memory parallel support vector machine learning. Technical Report FZJ-ZAM-IB-2005-11, Research Centre Jülich (2005)
27. IBM: ESSL - Engineering and Scientific Subroutine Library for AIX version 4.1 (2006)
28. Detert, U.: Introduction to the JUMP architecture. (2004)
29. Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. (1998)
30. Ho, T.K., Kleinberg, E.M.: Building projectable classifiers of arbitrary complexity. In: Proc. of the 13th Int. Conf. on Pattern Recognition. (1996) 880–885
31. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001)
32. Eitrich, T., Lang, B.: On the advantages of weighted L_1-norm support vector learning for unbalanced binary classification problems. In: Proceedings of IEEE IS, London (to appear). (2006)
33. Eitrich, T., Lang, B.: Efficient optimization of support vector machine learning parameters for unbalanced datasets. Journal of Computational and Applied Mathematics (2005) in press.
34. Eitrich, T., Lang, B.: Efficient implementation of serial and parallel support vector machine training with a multi-parameter kernel for large-scale data mining. In: Proceedings of the 11. International Conference on Computer Science (ICCS), February 24-26, 2006, Prague, Czech Republic. (2006) 6–11
35. Eitrich, T., Lang, B.: Parallel tuning of support vector machine learning parameters for large and unbalanced data sets. In: Proceedings of CompLife, Konstanz. Volume 3695 of Lecture Notes in Computer Science., Springer (2005) 253–264