DECOMPOSITION METHODS FOR LINEAR SUPPORT VECTOR MACHINES

Kai-Min Chung, Wei-Chun Kao, Tony Sun, and Chih-Jen Lin

Department of Computer Science
National Taiwan University, Taipei 106, Taiwan
cjlin@csie.ntu.edu.tw

ABSTRACT

We explain that decomposition methods, in particular, SMO-type algorithms, are not suitable for linear SVMs with more data than attributes. To remedy this difficulty, we consider a recent result by Keerthi and Lin [7] that for an SVM which is not linearly separable, after C is large enough, the dual solutions are at similar faces. Motivated by this property, we show that alpha seeding is extremely useful for solving a sequence of linear SVMs. It largely reduces the number of decomposition iterations to the point that solving many linear SVMs requires less time than the original decomposition method for one single SVM. We also conduct comparisons with other methods which are efficient for linear SVMs, and demonstrate the effectiveness of the proposed approach for helping the model selection.

1. INTRODUCTION

Given training vectors $x_i \in R^n, i = 1, \ldots, l$, in two classes, and a vector $y \in R^l$ such that $y_i \in \{1, -1\}$, the standard SVM formulation [3] is as follows:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned} \tag{1}$$

If $\phi(x) = x$, (1) is the form of a linear SVM. On the other hand, if ϕ maps x to a higher dimensional space, we call (1) a non-linear SVM.

For a non-linear SVM, the number of variables depends on the size of w and can be very large (even infinite), so people solve the following dual form:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to} \quad & y^T \alpha = 0, \\ & 0 \leq \alpha_i \leq C, i = 1, \ldots, l, \end{aligned} \tag{2}$$

where Q is an $l \times l$ positive semi-definite matrix with $Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$, and e is the vector of all ones. Usually we call $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ the kernel function. (2) is solvable because its number of variables is the size of the training set, independent of the dimensionality of $\phi(x)$. The primal and dual relation shows $w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)$ so $\text{sgn}(w^T \phi(x) + b) = \text{sgn}(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b)$ is the decision function.

Unfortunately, for large training set, Q becomes a huge dense matrix that traditional optimization methods cannot be directly applied. Currently, some specially designed approaches such as decomposition methods [6] and finding the nearest points of two convex hulls [8] are major ways of solving (2).

On the other hand, for linear SVMs, if $n \ll l$, w is not a huge vector variable so (1) can be solved by many regular optimization methods. Currently, on a normal computer, people have been able to train a linear SVM with millions of data (e.g. [12]); but for a non-linear SVM with much fewer data, we already need more computational time as well as computer memory.

Therefore, it is natural to ask whether in an SVM software linear and non-linear SVMs should be treated differently and solved by two methods. It is also interesting to see how capable non-linear SVM methods (e.g. decomposition methods) are for linear SVMs.

Recently, in many situations, linear and non-linear SVMs are considered together. Some approaches [9, 11] approximate non-linear SVM by different problems which are in the form of linear SVM with $n \ll l$. In addition, for non-linear SVM model selection with Gaussian kernel, [7] proposed an efficient method which has to conduct linear SVM model selection first (i.e. linear SVMs with different C). Therefore, it is important to discuss optimization methods for linear and non-linear SVMs at the same time.

This paper is organized as follows. In Section 2, we show that existing decomposition methods are inefficient for training linear SVMs. Section 3 presents our new strategy of training linear SVM via decomposition methods, which is hundred or thousand times faster. The proposed method is compared with existing linear SVM methods in Section 4. We then, in Section 5, apply the new implementation to solve a sequence of linear SVMs required for the model selection method in [7]. Concluding remarks are in Section 6.

2. EXISTING DECOMPOSITION METHODS FOR LINEAR SVM WITH $n \ll l$

The decomposition method is an iterative procedure. In each step, the index set of variables are separated to two sets B and N, where B is the working set. Then in that iteration variables corresponding to N are fixed while a sub-problem on variables corresponding to B is minimized. If q is the size of the working set B, in each iteration, only q columns of the Hessian matrix Q are required. They can be calculated and stored in the computer memory when needed. Thus, unlike regular optimization methods which usually require the access of the whole Q, here, the memory problem is avoided. Clearly, decomposition methods are specially designed for nonlinear SVMs. In this section, we discuss issues when they are applied to solve linear SVMs.

2.1. Slow Convergence

Unlike popular optimization methods such as Newton or quasi-Newton which enjoy fast convergence, decomposition methods converges slowly as in each iteration only very few variables are

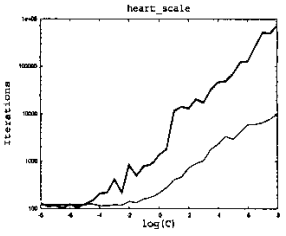

Fig. 1. Number of decomposition iterations for solving SVM with linear (the thick line) and RBF (the thin line) kernel

updated. We will show that the situation is even worse when solving linear SVMs.

It has been demonstrated (e.g. [5]) by experiments that if C is large and the Hessian matrix Q is not well-conditioned, decomposition methods converge very slowly. For linear SVMs, if $n \ll l$, then Q is a low-rank matrix which is not well-conditioned. When C is large, we can see the number of decomposition iterations dramatically increases. The situation is much worse than that for non-linear SVMs. In Figure 2.1, we demonstrate a simple example by using the problem heart from the statlog database. Each attribute is scaled to $[-1, 1]$. We use LIBSVM [1] to solve linear and nonlinear SVMs (RBF kernel, $e^{-\|x_i - x_j\|^2/(2\sigma^2)}$ with $1/(2\sigma^2) = 1/n$) with $C = 2^{-8}, 2^{-7.5}, \ldots, 2^8$ and present the number of iterations. Though two different optimization problems are solved (in particular, their Q_{ij}'s are in different ranges), Figure 2.1 clearly indicates the huge number of iterations for solving the linear SVM. In the following, we give some theoretical explanation about this difficulty.

For problems which are not linearly separable, [7] proved the following result:

Theorem 1 *There exists a finite value C^* and (w^*, b^*) such that $(w, b) = (w^*, b^*)$ solves (1) after $C \geq C^*$. In addition, $\sum_{i=1}^{l} \xi_i$ is a constant after $C \geq C^*$.*

Therefore, after $C \geq C^*$, $\alpha^T Q \alpha = w^T w$ becomes a constant. Since $\frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i = e^T \alpha - \frac{1}{2} \alpha^T Q \alpha$, after $C \geq C^*$, $\frac{1}{2} \alpha^T Q \alpha - e^T \alpha$ is a linear decreasing function of C.

It has been shown in [10] that, under some conditions, a commonly used decomposition method is linearly convergent. Therefore, using the zero vector as the initial point, the smaller the optimal value is, more decomposition iterations are needed.

Though linear SVM with $n < l$ does not satisfy the assumptions in [10], if it is still linearly convergent, when C is large, $\frac{1}{2} \alpha^T Q \alpha - e^T \alpha$ decreases linearly with C. This results in the difficulty that the number of decomposition iterations may increase linearly.

Furthermore, [10, Section 4] explains that the linear convergence rate of linear SVMs is relatively smaller than that of non-linear SVMs whose kernel matrices are well-conditioned. This indicates that decomposition methods is inherently not suitable for linear SVM due to its slow convergence.

2.2. Special Implementation for Linear SVMs

Though we have shown a disadvantage of using decomposition methods for linear SVMs, for practical implementations, there are special properties of linear SVMs which can speed up each iteration.

Most decomposition implementations maintain the gradient vector of the dual objective function during iterations. It is used for selecting the working set or checking the stopping condition. We usually calculate the gradient $Q\alpha - e$ by the following way: Suppose α^{k+1} and α^k are solutions of two consecutive decomposition iterations, $Q\alpha^{k+1} - e = Q\alpha^k - e + Q(\alpha^{k+1} - \alpha^k)$. Since from α^k to α^{k+1}, only q components are changed, $Q(\alpha^{k+1} - \alpha^k)$ involves with q columns of the matrix Q. For a non-linear SVM where Q is too large to be stored in the computer memory, the calculation of lq kernel elements becomes the main computational cost in each decomposition iteration. If each kernel evaluation requires $O(n)$ operations, $O(lnq)$ is the complexity of each iteration.

However, for linear SVM, $Q(\alpha^{k+1} - \alpha^k) = X^T(X(\alpha^{k+1} - \alpha^k))$, where $X = [y_1 x_1, \ldots, y_l x_l]$ is an n by l matrix. Thus, $X(\alpha^{k+1} - \alpha^k)$ involves $O(nq)$ operations and $X^T(X(\alpha^{k+1} - \alpha^k))$ needs $O(ln)$. Therefore, $O(lnq)$ operations are largely reduced to $O(ln)$. The first decomposition software which implements this for linear SVM is SVM^{light} [6]. Another implementation is BSVM [5].

However, for SMO [14] implementations where $q = 2$, the main cost of each iteration is only reduced by half. From this aspect, SMO type implementations are particular not suitable for linear SVM. In other words, using a larger q, the cost on updating the gradient is the same but the number of iterations is smaller. Thus, we should increase q until the cost of solving the sub-problem (usually $O(q^3)$) becomes a dominant part.

3. ALPHA SEEDING FOR LINEAR SVM

Results in [7] show the following properties of the dual linear SVM: There is C^* such that for all $C \geq C^*$, there are optimal solutions at the same face. In addition, for all $C \geq C^*$, the primal solution w is the same. The definition that two points are at the same face is as follows: Let α^1 be a feasible vector of (2) for $C = C_1$ and α^2 be a feasible vector of (2) for $C = C_2$. We say that α^1 and α^2 are on the same face if the following hold: (i) $\{i \mid 0 < \alpha_i^1 < C_1\} = \{i \mid 0 < \alpha_i^2 < C_2\}$; (ii) $\{i \mid \alpha_i^1 = C_1\} = \{i \mid \alpha_i^2 = C_2\}$; and, (iii) $\{i \mid \alpha_i^1 = 0\} = \{i \mid \alpha_i^2 = 0\}$. Therefore, a face means a partition of $\{1, \ldots, l\}$ to three sets where corresponding components of α are free, upper-bounded, and lower-bounded.

Based on these properties, we conjecture that for large C, optimal solutions are at similar faces. Therefore, if α^1 is an optimal solution at $C = C_1$, then $\alpha^1 C_2/C_1$ can be a very good initial point for solving (2) with $C = C_2$. This technique, called alpha seeding, was originally proposed for SVM model selection [4] where several (2) with different C have to be solved.

Earlier work which focus on nonlinear SVMs mainly use alpha seeding as a heuristic. Now for linear SVMs, $\alpha^1 C_2/C_1$ is at the same face as α^1 so most likely it is at a similar face of one optimal solution of $C = C_2$. This strongly supports its use as the initial solution.

Next, we conduct some comparisons between the proposed and the original implementations. Here, we consider two-class problems only. Some statistics of the data set used are in Table 2.

We train linear SVMs with $C = [2^{-8}, 2^{-7.5}, \ldots, 2^8]$. Table 1 presents the total number of iterations of training 33 linear SVMs using the alpha seeding approach. There are quite a few implementation considerations. Here, the code is modified from LIBSVM; details are in [2]. We then individually solve 33 problems and list the number of iterations (total, $C = 2^{7.5}$, and $C = 2^8$). For the new approach, we also list the approximate C^* for which linear SVM with $C \geq C^*$ have the same decision function. In addition, the constant $w^T w$ after $C \geq C^*$ is also given. For some problems

Problem	α-seeding			without α-seeding		
	#iter	C^*	$w^T w$	#total iter	$C = 2^{7.5}$	$C = 2^8$
heart	27231	$2^{3.5}$	5.712	2449067	507122	737734
australian	79162	$2^{2.5}$	2.071	20353966	3981265	5469092
diabetes	33264	$2^{6.5}$	16.69	1217926	274155	279062
german	277932	2^{10}	3.783	42673649	6778373	14641135
web	24044242	unstable	unstable	$\geq 10^8$	74717242	$\geq 10^8$
adult	3212093	unstable	unstable	$\geq 10^8$	56214289	84111627
ijcnn	590645	2^6	108.6	41440735	8860930	13927522

Table 1. Comparison of iterations (linear kernel); with and without alpha seeding.

Problem	l	n	α-seeding	without α-seeding
heart	270	13	43663	56792
australian	690	14	230983	323288
diabetes	768	8	101378	190047
german	1000	24	191509	260774
web	49749	300	633788	883319
adult	32561	123	2380265	4110663
ijcnn	49990	22	891563	1968396

Table 2. Comparison of iterations (RBF kernel); with and without alpha seeding.

(e.g. web and adult), $w^T w$ has not reached a constant until C is very large so we indicate them as "unstable" in Table 1.

It can be clearly seen that the alpha seeding approach performs so well to the point that its total number of iterations is much less than solving one single linear SVM with the original decomposition implementation. Therefore, even if we intend to solve one linear SVM with a particular C, using the proposed alpha seeding method starting from a smaller C may be more efficient.

Our experimental results indicate that the slow convergence of decomposition methods causes a huge number of iterations for changing some initial zero components to the upper bound C. On the other hand, the new approach starts from a small C where an optimal solution can be easily obtained. Then, as the next C is only slightly increased, the optimal solution face is not changed much. Hence, using the previous solution multiplied by the increase of C as initial points, the saving on the number of iterations is dramatic.

Furthermore, since we have solved linear SVMs with different C, model selection by cross validation is already done. To be more precise, we can randomly separate data to different folds first. If one fold is singled out as the validation set, we sequentially train the rest and predict the validation set using different C. This will be discussed in Section 5.

To demonstrate that alpha seeding is much more effective for linear than nonlinear SVMs, Table 2 presents the number of iterations using the RBF kernel $K(x_i, x_j) = e^{-\|x_i - x_j\|^2/(2\sigma^2)}$ with $1/2\sigma^2 = 1/n$. It can be clearly seen that the saving of iterations by using alpha seeding is marginal. In addition, comparing the "total iter." columns of both tables, we confirm again the slow convergence for linear SVMs if without alpha seeding.

To further justify the alpha seeding approach for linear SVMs, in [2] we prove the following theorem:

Theorem 2 *Assume that any two parallel hyperplanes in the feature space do not contain more than $n+1$ points of $\{x_i\}$ on them. We have*

1. *For any optimal solution of (2), it has no more than $n+1$ free components.*
2. *There is C^* such that after $C \geq C^*$, all optimal solutions of (2) share at least the same $l - n - 1$ upper and lower-bounded α variables.*

This result indicates that if $n \ll l$, starting from small C, most components of optimal solutions are at bounds. From any C_1 to C_2, even if all free components are different, two solutions share at least $l - 2(n+1)$ bounded variables. If C_2 is not far away from C_1, it is less likely that an upper (lower) component at C_1 becomes a lower (upper) bound at C_2. Thus, it is highly possible that the initial solution by alpha seeding has correctly identified at least $l - 2(n+1)$ components of an optimal solution.

Theorem 2 also helps to explain why web is the most difficult problem for results in Table 1. Its large number of attributes might lead to more free variables during iterations or at the final solution. Thus, alpha seeding is less effective.

Another result which supports the use of alpha seeding is the following theorem:

Theorem 3 *There are two vectors A, B, and a number C^* such that for any $C \geq C^*$, $AC + B$ is an optimal solution of (2).*

The proof is in [2]. This theorem extends the result in [7] which shows only that for any $\bar{C} > C^*$, there are optimal solutions α which form a linear function of C on the interval $[C^*, \bar{C}]$. Clearly $A_i \geq 0$ so we can consider the following three situations of vectors A and B:

1. $0 < A_i \leq 1$, **2.** $A_i = 0, B_i = 0$, **3.** $A_i = 0, B_i > 0$.

For the second case, $\alpha_i^1 \frac{C_2}{C_1} = A_i C_2 + B_i = 0$. For the first case, $A_i C \gg B_i$ after C is large enough. Thus, $\alpha_i^1 \frac{C_2}{C_1} = A_i C_2 + B_i \frac{C_2}{C_1} \approx A_i C_2 + B_i$. Using Theorem 2, there are few ($\leq n+1$) components satisfying the third case. This analysis also shows the effectiveness of using alpha seeding.

4. COMPARISON WITH OTHER APPROACHES

	Decomposition methods with α-seeding				Linear SVMs methods	
	$q = 2$ (LIBSVM)		$q = 30$ (BSVM)		ASVM	LSVM
problem	total iter	time	total iter	time	time	time
australian	79162	2.87	145	0.85	3.79	0.83
heart	27231	0.39	65	0.14	0.40	0.39
diabetes	33264	1.48	118	0.3	0.73	0.86
german	277932	21.43	144	0.65	2.80	3.50
ijcnn	590645	1119.99	1409	46.71	215.37	633.66
adult	3212093	1123.18	129440	153.52	896.56	4605.17
web	24044242	4154.53	1559664	1140.09	4277.12	44468.48

Table 3. Comparison of approaches for linear SVMs (time in second; q: size of the working set of decomposition methods.)

It is interesting to compare the proposed approach with other methods. In particular, there are approaches which are mainly suitable for linear SVMs. In this section, we consider Active SVM (ASVM) [12] and Lagrangian SVM (LSVM) [13].

LIBSVM, the software used in Section 3, implements an SMO-type decomposition method where in each iteration two variables are updated. For problems with more free variables at final solutions, many such updates (i.e. iterations) are needed. As now the number of free variables is generally less than $n+1$ and most bounded variables have been identified in the beginning using alpha seeding, we suspect that a decomposition method with a larger working set can benefit more from the alpha seeding. Thus, here we implement it in another decomposition software BSVM [5] which allows an arbitrary size of the working set.

However, BSVM, ASVM, and LSVM all solve slightly different formulations from (1). (Due to space limit, we discuss the difference of their implementations in [2].) Thus, it is very difficult to conduct a fair comparison. Our goal here is only to demonstrate

that decomposition methods, which are originally unsuitable for solving linear SVMs, can be competative with other linear-SVM methods, using the proposed implementation. Table 3 presents a comparison of the four implementations. We use the same benchmark problems as in Section 3. The computational experiments were done on a Pentium III-1000 with 1024MB RAM using the gcc compiler.

For each problem, linear SVMs with $C = 2^{-8}, 2^{-7.5}, \cdots, 2^{8}$ are solved. Except the total computational time, we also list the number of iterations of the two decomposition implementations. Clearly for problems with small n, for each C, BSVM takes very few iterations. The computational time is also less than that of LIBSVM. This is consistent with our earlier statement that for linear SVMs, SMO-type decomposition methods are less favorable than general decomposition methods with larger working sets.

Since alpha seeding is not applied to ASVM and LSVM, we admit that their computational time can be improved. Results here also serves as the first comparison between ASVM and LSVM. Clearly ASVM is faster. Moreover, due to the huge computational time, we set the maximal iterations of LSVM to be 1000. For problems adult and web, after C is large, the limit of iterations is reached before stopping conditions are satisfied.

5. EXPERIMENTS ON MODEL SELECTION

If the RBF kernel is used, [7] proposes the following model selection procedure for finding good C and σ^2:

1. Search for the best C of linear SVMs and call it $\tilde{C}$.
2. Fix $\tilde{C}$ from step 1 and search for the best (C, σ^2) satisfying $\log \sigma^2 = \log C - \log \tilde{C}$ using the RBF kernel.

That is, we solve a sequence of linear SVMs first and then a sequence of nonlinear SVMs with the RBF kernel. The advantage of this approach over an exhaust search of the parameter space is that only parameters on two lines are considered. If the original decomposition method is used for both linear and nonlinear SVMs here, due to the huge number of iterations, solving the linear SVMs becomes the bottleneck. Our goal in this section is to show that using the new approach for linear SVMs, the computational time spent on the linear part is no longer a problem.

Earlier in [7], due to the difficulty on solving linear SVMs, only small two-class problems are tested. Here we would like to evaluate this approach on large multi-class datasets. We consider four problems: dna, satimage, letter, and shuttle. Details of our settings are in [2].

We search for $\tilde{C}$ by five-fold cross validation on linear SVMs using uniformly spaced $\log_2 \tilde{C}$ value in $[-10, 10]$ (with grid space 0.5). Then the search of $(\log_2 C, \log_2 \sigma^2)$ is by considering points in $[-2, 12] \times [-10, 4]$ (with grid space 1) satisfying $\log \sigma^2 = \log C - \log \tilde{C}$. Thus, the number of SVMs solved in two steps may be different.

Table 4 presents experimental results. For each problem, we compare the test accuracy by a complete grid search and the model selection method in [7]. Their performance is very similar. However, the total model selection time of the new method is much shorter. We achieve this because using the alpha seeding, solving one linear SVM is as fast as solving a non-linear one. Otherwise, time for solving linear SVMs is a lot more so the proposed model selection method does not possess any advantage.

	Grid search		Method by Keerthi and Lin			
Problem	Time	Accuracy	Time	Linear(#SVMs)	RBF(#SVMs)	Accuracy
dna	15238	95.11	1286	1149(123)	137(23)	95.19
satimage	24505	92.2	4119	3879(615)	240(176)	91.3
letter	135881	97.72	22003	16802(13325)	5201(3983)	96.62
shuttle	243375	99.94	11783	9433(861)	2350(249)	99.85

Table 4. Comparison of different model selection methods (time in second; "Linear(#SVMs):" time for linear SVMs and the number of linear SVMs solved.)

6. CONCLUSION

In conclusion, we hope that based on this work, SVM software using decomposition methods can be suitable for all types of problems, no matter $n \ll l$ or $n \gg l$.

7. REFERENCES

[1] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] K.-M. Chung, W.-C. Kao, T. Sun, , and C.-J. Lin. Decomposition methods for linear support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2002.

[3] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

[4] D. DeCoste and K. Wagstaff. Alpha seeding for support vector machines. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, 2000.

[5] C.-W. Hsu and C.-J. Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291–314, 2002.

[6] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

[7] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel, 2002. `http://www.csie.ntu.edu.tw/~cjlin/papers/limit.ps.gz`.

[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, 2000.

[9] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining*, 2001.

[10] C.-J. Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2001.

[11] K.-M. Lin and C.-J. Lin. A study on reduced support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2002.

[12] O. L. Mangasarian and D. R. Musicant. Active set support vector machine classification. In *Advances in Neural Information Processing Systems*, pages 577–583, 2000.

[13] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.

[14] J. C. 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*, Cambridge, MA, 1998. MIT Press.