On Approximate Solutions to Support Vector Machines*

Dongwei Cao† Daniel Boley‡

Abstract

We propose to speed up the training process of support vector machines (SVM) by resorting to an approximate SVM, where a small number of representatives are extracted from the original training data set and used for training. Theoretical studies show that, in order for the approximate SVM to be similar to the exact SVM given by the original training data set, kernel k-means should be used to extract the representatives. As practical variations, we also propose two efficient implementations of the proposed algorithm, where approximations to kernel k-means are used. The proposed algorithms are compared against the standard training algorithm over real data sets.

1 Introduction

Support vector machines (SVM) [9, 27] have been successfully applied in a variety of domains, for example, [5, 7, 14]. One challenge in using SVM for large problems, which are common in data mining applications, is the expensive training process, where a huge quadratic optimization problem needs to be solved.

Many efforts have been made to design efficient training algorithms for SVM. One class of algorithms reduces the optimization problem to a series of small optimization problems, such as chunking and decomposition methods [5,15,16,21]. The noteworthy sequential minimal optimization (SMO) algorithm [23] uses only two training items in each small optimization problem. The SVMs given by these algorithms and other algorithms, such as [18], correspond to the entire training data set and are denoted as *exact SVM*s here.

Another class of algorithms accelerates the training process by giving up the exact SVM and resorting to an *approximate SVM* by, for example, low rank approximation [12], sampling [1,28], squashing [22], and other approaches [26].

In this paper, we propose to train an approximate SVM with a small number of representatives extracted from the original training data set, thus reducing the size of the optimization problem and speeding up the training process. Theoretical considerations developed in this paper indicate that, to minimize the difference between the approximate and exact SVMs, kernel k-means should be used to extract the representatives. Here, the difference between exact and approximate SVMs is measured by the difference between corresponding Gram matrices. It turns out that kernel k-means is also indicated when the measure is the difference between the weight vectors of corresponding hyperplanes, which is a more direct measure of the difference between two SVMs, though space limits precludes any discussion here. To achieve better efficiency, we also propose two efficient implementations that are approximations to the kernel k-means. Compared with similar strategies that are limited to a linear kernel [22,29], the proposed algorithms are applicable to both linear and non-linear kernels.

In the rest of this paper, section 2 briefly introduces SVM, section 3 describes the theory and algorithms for training approximate SVMs, section 4 shows experimental results, and section 5 concludes the paper.

2 Support Vector Machines

Given a training data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ of size n, where $\mathbf{x}_i \in \mathcal{X}$ is a vector of attributes and $y_i \in \{-1, 1\}$ is a label, for $i = 1, \cdots, n$, the SVM classifier h^* is a hyperplane classifier whose weight vector $\mathbf{w}^* = \sum_{i=1}^n \alpha_i^* y_i \phi(\mathbf{x}_i)$ [27]. The expansion coefficients $\boldsymbol{\alpha}^* = [\alpha_1^* \ \ldots \ \alpha_n^*]^T$ is the solution of the following quadratic optimization problem

$$\text{(2.1a)} \qquad \text{Maximize} : W(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{Y}^T \mathbf{G} \mathbf{Y} \boldsymbol{\alpha}$$

$$\text{(2.1b)} \qquad \text{Subject to} : 0 \le \alpha_i \le C, \ \forall i = 1, 2, ..., n$$

$$\text{(2.1c)} \qquad \boldsymbol{\alpha}^T \mathbf{y} = 0,$$

where $C > 0$ is the regularization coefficient, $\mathbf{1}$ is a vector of ones, $\mathbf{G}$ is the Gram matrix with entries $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $\boldsymbol{\alpha}$ is a vector of length n, and $\mathbf{Y}$ is an $n \times n$ diagonal matrix with diagonal entries $Y_{ii} = y_i$. Here, K is the so-called Mercer kernel satisfying $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, and ϕ is a

*This research was partially supported by NSF Grants IIS-0208621 and IIS-0534286. The authors would like to thank anonymous reviewers for their thoughtful comments.

†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. Email: dcao@cs.umn.edu

‡Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. Email: boley@cs.umn.edu

mapping from the data space $\mathcal{X}$ to a reproducing kernel Hilbert space $\mathbb{H}$ with dot product $\langle \cdot, \cdot \rangle$.

We denote the algorithm that solves optimization problem (2.1) with Gram matrix $\mathbf{G}$ as $\mathcal{A}_{\text{exact}}$, and call the resulting SVM h^* the *exact SVM*. The empirical time complexity of algorithm $\mathcal{A}_{\text{exact}}$ is $T(\mathcal{A}_{\text{exact}}) = \mathcal{O}(n^2)$ [6], although it could be better in practice [23].

3 Algorithms for Approximate SVM

A generic algorithm $\mathcal{A}_{\text{approx}}$ for training approximate SVMs is described in Algorithm 1, where it is assumed that each datum in $\mathcal{D}$ has exactly one representative, which has the same label as the original datum. The total number k of representatives is pre-specified and should depend on available computational resources. The empirical time complexity of $\mathcal{A}_{\text{approx}}$ is thus $\mathcal{O}(k^2)$ plus the time complexity of the underlying extraction algorithm $\mathcal{A}_{n \to k}$.

Algorithm 1 Algorithm $\mathcal{A}_{\text{approx}}$

Require: A training data set $\mathcal{D}$ of size n, kernel K, regularization coefficient C, number k of representatives, representative extraction algorithm $\mathcal{A}_{n \to k}$.

1: Extract k representatives from $\mathcal{D}$ using $\mathcal{A}_{n \to k}$.
2: Using kernel K and coefficient C, the approximate SVM $\bar{h}$ is trained on the k representatives.

The Gram matrix $\mathbf{G}$ in problem (2.1) can be rewritten as $\mathbf{G} = \mathbf{X}^T\mathbf{X}$, where $\mathbf{X} = [\phi(\mathbf{x}_1) \ \dots \ \phi(\mathbf{x}_n)]$. It can be shown that the approximate SVM $\bar{h}$ given by Algorithm 1 can also obtained by solving problem 2.1 by replacing $\mathbf{G}$ with $\widehat{\mathbf{G}} = \widehat{\mathbf{X}}^T\widehat{\mathbf{X}}$, where $\widehat{\mathbf{X}} = [\phi(\widehat{\mathbf{x}}_1) \ \dots \ \phi(\widehat{\mathbf{x}}_n)]$ and $(\widehat{\mathbf{x}}_i, \widehat{y}_i)$ is the representative of $(\mathbf{x}_i, y_i)$ with $\widehat{y}_i = y_i$ by assumption.

We claim that a good choice of algorithm $\mathcal{A}_{n \to k}$ should output such a set of representatives that minimizes the difference between two Gram matrices $\mathbf{G}$ and $\widehat{\mathbf{G}}$, which can be bounded above as follows

$$\begin{aligned} \left\|\mathbf{G} - \widehat{\mathbf{G}}\right\| &= \left\|\mathbf{X}^T\mathbf{X} - \widehat{\mathbf{X}}^T\widehat{\mathbf{X}}\right\| \\ &\le \left\|\mathbf{X} + \widehat{\mathbf{X}}\right\| \left\|\mathbf{X} - \widehat{\mathbf{X}}\right\| \\ &\le \left\|\mathbf{X} + \widehat{\mathbf{X}}\right\| \sqrt{\sum_{j=1}^{k} \sum_{(\mathbf{x},y)\in\mathcal{D}_j} \|\phi(\mathbf{x}) - \phi(\bar{\mathbf{x}}_j)\|^2}, \end{aligned} \tag{3.2}$$

where $\mathcal{D}_j$ is the set of data in $\mathcal{D}$ whose representative is the j-th representative $(\bar{\mathbf{x}}_j, \bar{y}_j)$.

By minimizing the term inside the square root in equation (3.2), a good set of representatives can be obtained (approximately) by applying kernel k-means with kernel K (see, e.g., [10, 13] for kernel k-means) to the data of class 1 and those of -1 separately and combining the resulting feature-space centroids. Assuming $\mathcal{D}'$ is a cluster of size n', its representative $\mathbf{x}'$ is given formally by

$$\phi(\mathbf{x}') = \frac{1}{n'} \sum_{\mathbf{x}\in\mathcal{D}'} \phi(\mathbf{x}). \tag{3.3}$$

This quantity is well defined even if $\mathbf{x}'$ does not exist, and inner products involving $\phi(\mathbf{x}')$ can be computed without knowing its value explicitly using K.

Let k^+ and k^- be the number of representatives for the data of class 1 and -1 respectively, we try $k-1$ combinations of k^+ and k^- satisfying $k^+ + k^- = k$ and choose the one that minimizes the term inside the square root of equation (3.2). For each trial, the time complexity of kernel k-means is $\mathcal{O}(n^2)$ [10]. Thus, the time complexity of Algorithm 1 with above strategy for representative extraction is $\mathcal{O}(n^2k + k^2 + l^2)$, where l is the size of the largest cluster and the term l^2 reflects the cost of evaluating the kernel between the representatives of two clusters, which has also been observed in [10].

The above implementation of Algorithm 1 is efficient when one wants to train multiple SVMs with the same training data set $\mathcal{D}$ and the same kernel K, but different values of the regularization coefficient C. In these situations, the time complexity of above implementation is $\mathcal{O}(n^2k + k^2 + l^2)$ for the first SVM and $\mathcal{O}(k^2)$ for the rest, since kernel k-means does not depend on the choice of C and needs to run once and, because k is usually small, the Gram matrix can be cached. This compares favorably with training algorithms for the exact SVM where the time complexity is always $\mathcal{O}(n^2)$.

However, the above implementation is not efficient when one wants to train a single SVM or multiple SVMs with different kernels. Thus, we provide next two simplified implementations of Algorithm 1 whose time complexity is better than $\mathcal{O}(n^2)$.

First, instead of trying all $k-1$ combinations of k^+ and k^- for a pre-specified k, we choose $k^+ = \text{round}\left(\sqrt{n^+}\right)$ and $k^- = \text{round}\left(\sqrt{n^-}\right)$, where n^+ (n^-) be the number of data in class 1 (-1) and round$(\cdot)$ is the rounding operation. The total number k of representatives becomes $k^+ + k^-$. This heuristic was shown to be effective in [4]. Second, we replace the expensive kernel k-means with less expensive data space clustering algorithms. Due to its computational efficiency, Principal Direction Divisive Partition (PDDP) [3] is used here, whose time complexity is $\mathcal{O}(n \log k)$ [20]. Other algorithms such as standard k-means can also be used.

With these simplifications in mind, we have following two implementations of Algorithm 1 which differ on the definition of a cluster's representative.

- **Algorithm $\mathcal{A}^{\phi}_{\text{approx}}$**: The representative of a clus-

ter is defined as the feature space center (3.3). This strategy can be seen as an approximation to kernel k-means, where there is no further iterations after clustering initialization (by PDDP) and centroid computation. The resulting implementation of Algorithm 1 is denoted as $\mathcal{A}^{\phi}_{\text{approx}}$ and its time complexity is $\mathcal{O}(n \log k + k^2 + l^2)$.

- **Algorithm $\mathcal{A}^{\mathbf{P}-\phi}_{\mathbf{approx}}$**: For a cluster $\mathcal{D}'$ of size n', we define its representative $\mathbf{x}'$ as the feature-space pseudo-center, i.e.,

$$\mathbf{x}' = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{D}'} \left\| \phi(\mathbf{x}) - \frac{1}{n'} \sum_{\mathbf{x}'' \in \mathcal{D}'} \phi(\mathbf{x}'') \right\|. \tag{3.4}$$

  Using an explicit datum $\mathbf{x}' \in \mathcal{D}$ as a representative substantially reduces the cost of kernel evaluations that involve it, compared to using (3.3). This pseudo-center can also be seen as a crude approximation to the pre-image defined in [24, 25], where "argmin" is taken over the entire data space $\mathcal{X}$ instead of just $\mathcal{D}'$. The resulting implementation of Algorithm 1 is denoted as $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$ and its time complexity is $\mathcal{O}(n \log k + k^2)$. Thus, this algorithm is faster than algorithm $\mathcal{A}^{\phi}_{\text{approx}}$.

Finally, as a baseline algorithm, the representatives are chosen by randomly selecting elements from $\mathcal{D}$. This method is in the class of sampling-based SVM training algorithms. Here, we *arbitrarily* partition the training data set into clusters and defines the representative of a cluster by *arbitrarily* drawing a datum from this cluster. The resulting approximate SVM training method is denoted as $\mathcal{A}^{\mathrm{Rnd}}_{\text{approx}}$ and its time complexity is $\mathcal{O}(k^2)$.

4 Experiments

In this section, we compare the proposed algorithms $\mathcal{A}^{\phi}_{\text{approx}}$ and $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$, which give approximate SVMs h^{ϕ} and $h^{\mathrm{P}-\phi}$ respectively, against the standard training algorithm SMO [8, 11, 23] and the baseline algorithm $\mathcal{A}^{\mathrm{Rnd}}_{\text{approx}}$, which give the exact SVM h^* and an approximate SVM h^{Rnd} respectively.

We use four binary classification data sets in our experiments. The data sets "Adult" and "Web" come from UCI data mining repository [2], the data set "MNIST$^{\mathbf{b}}$" is constructed from the MNIST handwritten digits recognition data set [19] by treating the data representing digits $1, 2, 3, 4, 5$ as class 1 and the data representing digits $6, 7, 8, 9, 0$ as class -1, and the data set "Yahoo" concerns the prediction of customer's behavior and is extracted from a data set [17] of 1 million samples.

We implement Algorithm 1 based on LibSVM [8, 11], an efficient implementation of SMO [23]. The clustering needed by algorithms $\mathcal{A}^{\phi}_{\text{approx}}$ and $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$ is performed by a Matlab® implementation of PDDP [3]. The exact SVM h^* is given by LibSVM [8, 11]. All experiments were run on a PC running Windows 2000 Server® with one Pentium IV 2.8 GHz CPU and 1 GB RAM. The kernel cache is 128MB and the KKT tolerance is 10^{-3}.

To demonstrate that the proposed strategy can apply to non-linear kernels, which is an advantage over other algorithms such as those in [22, 29], we use Gaussian kernel in all experiments. Furthermore, since our goal is not to show the superior performance of SVM compared to other non-SVM methods, a comprehensive parameter tuning was not performed, thus the error rates reported here may be worse than those reported elsewhere.

The results are summarized in Table 1. Taking the Adult data set as an example, we observe the following for all data sets. (i) The algorithm $\mathcal{A}^{\phi}_{\text{approx}}$ speeds up the training process significantly with slight loss of accuracy. The reason for the extended test time of h^{ϕ} is that h^{ϕ} now has "support clusters" instead of "support vectors", and the number of data in all support clusters could be larger than the number of support vectors of h^*. (ii) The algorithm $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$ speeds up both training and test dramatically, but the resulting SVM $h^{\mathrm{P}-\phi}$ has a rather high error rate, fairly close to that of h^{Rnd}. (iii) The algorithm $\mathcal{A}^{\mathrm{Rnd}}_{\text{approx}}$ has not only the smallest training time but also the largest error rate.

Figure 1 compares the scaling behavior of algorithms $\mathcal{A}^{\phi}_{\text{approx}}$, $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$, and SMO [8, 23] over the data sets "Web" and "Adult", where a nested sequence of training data sets of increasing size is created [23]. It can be seen that, in terms of scaling behavior, algorithm $\mathcal{A}^{\mathrm{P}-\phi}_{\text{approx}}$ is better than algorithm $\mathcal{A}^{\phi}_{\text{approx}}$, both of which are better than SMO.

In summary, the algorithm A^{ϕ}_{approx} has the best trade-off in terms of training time, test time, and classification accuracy.

5 Conclusion and Future Work

Aiming at speeding up the training process of SVM, we studied a kind of approximate algorithms where the number of training data is reduced by extracting and using only a small number of representatives of the original training data set. We show that, to minimize the difference between the exact and approximate SVMs, the kernel k-means should be used to extract the representatives. We also proposed two efficient implementations and compared them against SMO [23] and a random-sampling based algorithm over real data sets, showing that the algorithm $\mathcal{A}^{\phi}_{\text{approx}}$ has the best trade-off between time complexity and accuracy.

Table 1: Comparison of exact SVM training algorithm SMO [8, 23] and the approximate SVM training algorithms $\mathcal{A}^{\phi}_{\text{approx}}$, $\mathcal{A}^{\text{P}-\phi}_{\text{approx}}$, $\mathcal{A}^{\text{Rnd}}_{\text{approx}}$ in terms of corresponding SVMs. $T(A_{n\to k})$ is the time used to extract the representatives. T(SVM) is the time used to solve the optimization problem (2.1). T(Total) = $T(A_{n\to k})$ + T(SVM). "Test Time" is the time used to classify all N_{test} test data. Every entry in the row of h^{Rnd} takes the form of "mean ± standard deviation" over 10 independent runs. For each data set, we also show the number of features d, the number of training data N_{train}, the number of test data N_{test}, the regularization coefficient C, and the parameter σ in the Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sigma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$.

Data	SVMs	Training Time (sec.)			Test Time (sec.)	Error Rate (%)
		$T(A_{n\to k})$	T(SVM)	T(Total)		
Adult ($d = 123$,	Exact SVM h^*	NA	1877	**1877**	246	4.3
$N_{\text{train}} = 32561$,	Approx. SVM h^{ϕ}	8.8	237	**245.8**	277	6.1
$N_{\text{test}} = 29589$,	Approx. SVM $h^{\text{P}-\phi}$	9.8	< 0.05	**9.8**	2	23.3
$C = 5, \sigma = 1.0$.)	Approx. SVM h^{Rnd}	0.7 ± 0.48	0.1 ± 0.32	$\mathbf{0.8 \pm 0.42}$	2.1 ± 0.57	23.9 ± 0.17
Web ($d = 300$,	Exact SVM h^*	NA	2908	**2908**	442	0.2
$N_{\text{train}} = 49749$,	Approx. SVM h^{ϕ}	13.4	474	**487.4**	559	1.3
$N_{\text{test}} = 38994$,	Approx. SVM $h^{\text{P}-\phi}$	18.4	< 0.05	**18.4**	4	2.6
$C = 5, \sigma = 1.0$.)	Approx. SVM h^{Rnd}	2.1 ± 0.01	< 0.05	$\mathbf{2.1 \pm 0.01}$	2.6 ± 1.07	2.7 ± 0.03
MNIST[b] ($d = 784$,	Exact SVM h^*	NA	6718	**6718**	298	1.2
$N_{\text{train}} = 60000$,	Approx. SVM h^{ϕ}	99	2827	**2926**	605	4.6
$N_{\text{test}} = 10000$,	Approx. SVM $h^{\text{P}-\phi}$	122	< 0.05	**122**	7	7.3
$C = 5, \sigma = 0.04$.)	Approx. SVM h^{Rnd}	2.7 ± 0.52	0.2 ± 0.42	$\mathbf{2.9 \pm 0.79}$	7.8 ± 1.93	9.2 ± 0.73
Yahoo ($d = 80$,	Exact SVM h^*	NA	18437	**18437**	6055	16.2
$N_{\text{train}} = 100000$,	Approx. SVM h^{ϕ}	31.3	1921	**1952.3**	4677	19.9
$N_{\text{test}} = 200000$,	Approx. SVM $h^{\text{P}-\phi}$	38.3	1	**39.3**	23	19.8
$C = 1, \sigma = 0.01$.)	Approx. SVM h^{Rnd}	7.5 ± 0.52	0.1 ± 0.32	$\mathbf{7.6 \pm 0.53}$	25.5 ± 2.17	20.3 ± 0.89

Future research directions will focus on (i) developing a PAC-style bound on the generalization performance of h^{ϕ} given by $\mathcal{A}^{\phi}_{\text{approx}}$, and (ii) designing algorithms that can efficiently compute an approximation to the feature-space centroid that is better than the feature-space pseudo-center, thus giving an algorithm having the speed of $\mathcal{A}^{\text{P}-\phi}_{\text{approx}}$ and the accuracy of $\mathcal{A}^{\phi}_{\text{approx}}$.

References

[1] D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In S. B. Thomas, G. Dietterich, and Z. Ghahramani, editors, *NIPS 14*, pages 335–342, 2002.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[3] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

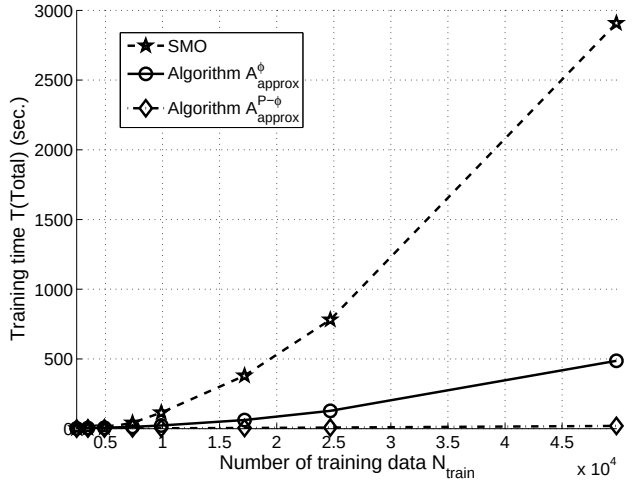
[4] D. L. Boley and D. Cao. Training support vector machine using adaptive clustering. In *SIAMDM*, pages 126–137, Lake Buena Vista, FL, USA, April 22-24 2004. SIAM.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

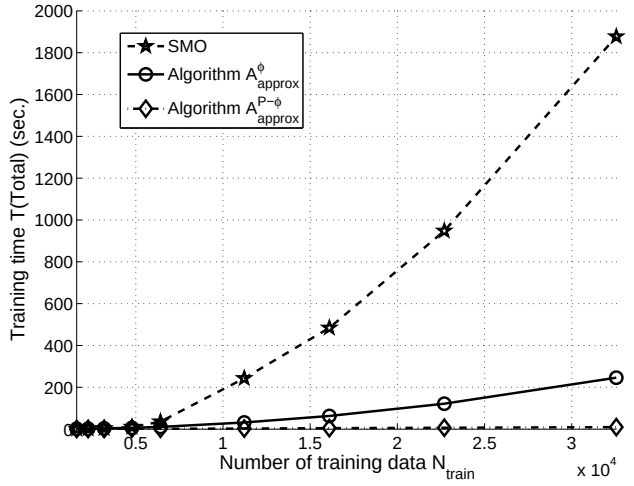
[6] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[7] D. Cao, O. T. Masoud, D. L. Boley, and N. Papanikolopoulos. Online motion classification using support vector machines. In *IEEE ICRA*, 2004.

[8] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines, 2001.



(a) Data set "Web"



(b) Data set "Adult"

Figure 1: Scaling behavior of SMO [8, 23], $\mathcal{A}^{\phi}_{\text{approx}}$, and $\mathcal{A}^{\text{P}-\phi}_{\text{approx}}$. See Table 1 for the definition of T(Total).

[9] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[10] I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *KDD*, 2004.

[11] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training svm. Technical report, National Taiwan University, April 2005.

[12] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, 2001.

[13] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002.

[14] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, 1998.

[15] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, 1999.

[16] L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 147–167. MIT Press, 1999.

[17] S. S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *JMLR*, 6:341–361, 2005.

[18] 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), January 2000.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[20] D. Littau and D. Boley. Using low-memory representations to cluster very large data sets. In *Third SIAMDM*, pages 341–345, May 2003.

[21] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In A. Island, editor, *IEEE Neural Networks for Signal Processing VII Workshop*, pages 276–285, 1997.

[22] D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *KDD*, pages 295–299, 2000.

[23] 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. MIT Press, 1999.

[24] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-basd methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[25] B. Schölkopf and A. J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[26] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *JMLR*, 6:363–392, 2005.

[27] V. N. Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.

[28] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Diettrich, and V. Tresp, editors, *NIPS 13*, pages 682–688. MIT Press, 2001.

[29] H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. In *KDD*, pages 306–315, 2003.