Two-stage Incremental Working Set Selection for Support Vector Training

Dung Duc NGUYEN † Kazunori MATSUMOTO † Kazuo HASHIMOTO ‡

Yasuhiro TAKISHIMA † Masahiro TERABE ‡

† KDDI R&D Laboratories 2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502 Japan

‡ Graduate School of Information Science, Tohoku University, 6-6-11 Aobaku, Sendai, Miyagi, 980-8579, Japan

E-mail: † {dd-nguyen, matsu, takisima}@kddilabs.jp, ‡ {kh, terabe}@aiet.ecei.tohoku.ac.jp

Abstract We introduce iSVM - an incremental algorithm that achieves high speed in training support vector machines (SVMs) on large datasets. In the common decomposition framework, iSVM starts with a minimum working set (WS), and then iteratively selects one training example to update the WS in each optimization loop. iSVM employs a two-stage strategy in processing the training data. In the first stage, the most prominent vector among randomly selected data is added to the WS. This stage results in an approximate SVM solution. The second stage uses temporal solutions to scan through the whole training data once again to find the remaining support vectors (SVs). We show that iSVM is especially efficient for training SVMs on applications where data size is much larger than number of SVs. On the KDD-CUP 1999 dataset with nearly five millions training examples, iSVM takes less than one hour to train an SVM with 94% testing accuracy, compared to seven hours with LibSVM – one of the state-of-the-art SVM implementations. We also provide analysis and experimental comparisons between iSVM and the related algorithms.

Keyword Support Vector Machine, Optimization, Decomposition Method, Sequential Minimal Optimization

1. Introduction

In recent years support vector machine (SVM) [1] has been successfully applied in various machine learning applications. However, scalability still remains one of biggest challenges for SVM in particular and kernel-based methods in general. It is due to the fact that training an SVM requires solving a quadratic programming (QP) problem in which, for the worst case, the complexity becomes $O(l^3)$ for time and $O(l^2)$ for memory requirement, where l is the number of training examples [3].

There have been number of approaches to scalability problem of SVM training. Among them decomposition is the most widely implemented method in various SVM software and libraries, e.g. LibSVM [4], SVM^{light} [9], CoreSVM [5, 6], HeroSVM [14], and SimpleSVM [12]. The main idea of decomposition algorithms is to divide training data into two sets: an active working set (WS) whose coefficients can be updated, and an inactive set whose coefficients are temporally fixed [7, 8]. The extreme case of this decomposition approach is the Sequential Minimal Optimization (SMO) algorithm [10] that does optimization on a set of only two examples. For each optimization loop, SMO scans through the whole training data to find a good pair of vectors, and then updates coefficients of the two selected vectors analytically. In order to find a good pair in the next iteration, it is required to update the violation of optimality criteria of all training vectors. It is very expensive for applications where the number of updates (training data) is huge.

In this paper, we introduce a two-stage incremental WS selection method in training SVMs. The proposed algorithm starts from finding an initial SVM solution on a minimum WS (two vectors from opposite classes in a two-class classification task). It then iteratively selects one training vector to update the WS. The selection of new vectors is divided into two stages. In the first stage, only the most prominent vector among a fixed number of sampling data is added to the WS while all others remain in the training data. From the second stage, all training examples are checked once more. For both stages, each optimality violated vector is used to update the WS and find a new SVM solution. We show that with this two-stage WS selection strategy the proposed iSVM has a linear time complexity in number of training examples and cubic in number of SVs. Experiments on large benchmark datasets show that iSVM is very fast when working on applications where number of SVs is much smaller than number of training data. On the KDD-CUP 1999 network

intrusion detection dataset with nearly five millions training examples iSVM takes less than one hour to train a SVM with 94% testing accuracy. With LibSVM [4] – one of the state-of-the-art SMO implementations, it takes seven hours.

The rest part of this paper is organized as follows. In section II we briefly describe the QP problem in SVM training and decomposition method for solving the QP. We introduce iSVM and its complexity analysis in section III. Experiments for evaluating iSVM and comparison with other algorithms are reported in section IV. Section V is for conclusion.

2. SVM and Decomposition Algorithms

2.1. Support Vector Training

In support vector learning [1, 2], we are given a set of training examples $x_i \in R^d$ with labels $y_i \in \{-1,+1\}$, $i=1,...,l$. The main task of training an SVM is to solve the following optimization problem:

$$\min_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^{l} \alpha_i \qquad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$0 \le \alpha_i \le C, i = 1,...,l$$

where $K(x_i,x_j)$ is a kernel function calculating dot product between two vector x_i and x_j in some feature space; C is a parameter penalizing each "noisy" example in the given training data. The optimal coefficients $\{\alpha_i \mid i=1,\ldots,l\}$ will form a decision function:

$$y = \operatorname{sgn}\left(f(x) = \sum_{\alpha_i \neq 0} y_i \alpha_i K(x_i, x) + b \right) \qquad (2)$$

Problem (1) involves l variables α_i and l^2 parameters $K_{ij} = K(x_i,x_j)$, $i, j = 1,...,l$. The l^2 number of parameters K_{ij} quickly exceeds memory capacity of a normal computer when the number of training examples gets larger than, say, 100,000. This over demanding in memory requirement causes the main difficulty in training SVMs.

2.2. SMO and Decomposition Algorithms

The main idea of decomposition algorithms, e.g. [7, 8, 9, 10], is to break down the QP problem (1) of size l into a series of much smaller QPs, and iteratively perform optimization on these sub-problems. In each iteration decomposition algorithms divide l training examples into two categories: a set of active vectors W that corresponding coefficients α_i can be updated and a set of inactive vectors that corresponding coefficients are temporally fixed. Active vectors are updated by some optimization method to minimize objective function L on W. After that fixed vectors are checked and used for updating the working set W. Optimization loop will stop when all optimality conditions are satisfied. The extreme case of the decomposition method is the Sequential Minimal Optimization (SMO) algorithm [10] that optimizes a set of only two vectors. The power of SMO resides in the fact that updating scheme could be done analytically. Call x_i and x_j be chosen vectors, then the best new values of α_i and α_j in terms of reducing best the objective function L in (1) are (ignoring the box constraint $0 \le \alpha_i \le C$, $i=1,...,l$):

$$\alpha_i^{new} = \alpha_i^{old} + \frac{(E_j - E_i)}{\kappa}$$
$$\alpha_j^{new} = y_j\left(const - y_i \alpha_i^{new}\right) \qquad (3)$$

where $const = y_i\alpha_i^{old} + y_j\alpha_j^{old}$, $\kappa = K_{ii} + K_{jj} - 2K_{ij}$, and $E_k = f(x_k) - y_k$. This updating scheme leads to the reduction of objective function L an amount of

$$\Delta L_{ij} = -\frac{(E_i - E_j)^2}{2\kappa} \qquad (4)$$

Based on this reduction rate different heuristics have been proposed to select the best pair of vectors (x_i, x_j) with a reasonable cost [10, 4].

The analytical solution property of SMO makes it become a core optimizer for many SVM implementations. In **Table 1** we describe main procedures in an SMO implementation.

Table 1: The SMO algorithm

Input: Training data $T = \{x_k \mid k=1,...,l\}$

0. Initialize a feasible solution
1. Set iteration $t = 0$
2. **While** *StoppingCondition* is not satisfied
3. Select a pair of vectors (x_i, x_j)
4. Update (α_i, α_j) analytically
5. Update violation state E_k, $k=1,...,l$
6. Set $t = t + 1$
7. **Endwhile**

Output: Coefficients $\boldsymbol{\alpha} = \{\alpha_k \mid k=1,...,l\}$

3. iSVM - an Incremental SVM Training Algorithm

The most expensive procedure of the SMO is updating violation of optimality criteria E_k of all training vectors in *step 5*. This calculation is used to select the best pair of vectors in the next iteration; and it is required for every training example. *Step 5* becomes very expensive when the number of training data is huge. Moreover, as only SVs (training vectors x_i with corresponding coefficients $\alpha_i > 0$) will contribute to form the final decision function (2), *step 5* of SMO can be very inefficient when many training examples are not SVs. To improve the efficiency shrinking technique [9] can be applied to remove non-support vectors. However, there has been no way to determine whether a training example is a SV or not from the beginning.

In this section we introduce an incremental strategy for selecting SVs. The main idea is using temporal SVM solutions to determine good candidates of SVs and optimization process is performed only on a small set of selected candidates. The following subsections describe in detailed the main steps of our proposed algorithm iSVM in **Table 2**.

Table 2: iSVM algorithm

Input: Training data $T = \{x_k \mid k=1,\ldots,l\}$

0. Select the first working set W_0. Find the first temporal solution S_0. Set $T = T - W_0$
1. Set iteration $t = 0$
2. **While** *StoppingCondition* is not satisfied
3. Select a one vector x_t in T
4. Update $W_{t+1} = W_t \cup \{x_t\}$, $T = T - \{x_t\}$
5. Find new solution S_{t+1} on W_{t+1}
6. Set $t = t + 1$
7. **Endwhile**

Output: Coefficients $\boldsymbol{\alpha} = \{\alpha_k \mid k=1,\ldots,|W_t|\}\}$

3.1. Initialization

As SMO is used for optimization on the selected working set W_t in *step 5*, a minimum set of two vectors are selected to build the first working set W_0. For a two-class classification application, this is simply selecting any two training vectors from two opposite classes. Compared with previously proposed methods, this initialization step is simpler. In [11] the authors suggested to select randomly a set of p training instances, where p is a training parameter. In [4], [12], two closest vectors were recommended to form the first working set (for the best reduction rate in (4)). However, our primary experiments indicated that the result of iSVM does not depend much on this initialization scheme.

3.2. Updating the Working Set

There have been different schemes to update the WS. In fact, this is a distinctive step for each incremental algorithm. Different updating strategies will produce different results in terms of convergence speed and final solution. In iSVM we propose a two-stage process for expanding and updating the WS. In the first stage, iSVM tries to find a good approximation of SVM solution as quickly as possible. It then scans through the whole training data once again to examine all vectors one-by-one. Potential SVs will be used to update the WS and find a new and better SVM solution.

3.2.1. Re-sampling selection

In support vector learning, if we know in advance which training example will be SV, we can remove all non support vectors without changing the optimal solution. Effective removal strategies will grant an efficient algorithm. We do this by a two-stage data processing procedure.

In the first stage, *step 3* of iSVM examines only a small number training examples and selects the most prominent vector to add to the WS. The selection is based on violation of optimality criteria of a training vector with respect to a temporal solution found in previous iteration. At iteration t, x_t is selected based on the following criterion:

$$x_t = \arg\max_{x_k} f_t(x_k) - y_k \tag{5}$$

where

$$f_t(x) = \sum_{x_i \in W_t} y_i \alpha_i K(x_i, x) + b_t \tag{6}$$

is the temporal solution S_t at iteration t. The selection heuristic (5) is exactly the maximal violating heuristic that has been used by SMO [10] and other early decomposition implementations. The difference is that SMO updates all E_k, $k=1,\ldots,l$, and then scans through all of them to select the best pair. iSVM uses temporal solutions to examine a fixed number of training examples. Difference in complexity will be analyzed in more detailed in subsection *D*.

In the first stage, only the selected vector is removed from T, all other vectors remain in the training data. They

will be re-examined in the next iterations or in the second phase. There are two reasons for the re-examination. Firstly, only the most prominent vector is added to the WS, not every optimality violated vector. The second reason is that temporal solutions are not close enough to the optimal solution (as only a small number of vectors are examined). Removing training examples from very beginning might mistakenly remove the true SVs.

The first stage will finish when we are sure at some extend that S_t is a good approximation. In iSVM, we use the following heuristics for starting the second phase:

i) None of N randomly selected vectors violates optimality criteria, *or*

ii) Size of W_t is bigger than a predefined number (1,000 in our experiments), *or*

iii) All training vectors are examined once in average

The first condition has been used in various situations including kernel matrix approximation [13] and CoreSVM [5, 6]. This heuristic is based on the fact that with a sample size of 59, we still can catch one among 5% most violating vectors [13]. The second and third conditions mean that the temporal solution will be considered stable and close enough to the optimal solution when a big number of training examples are examined. After a good approximation has been achieved we can switch to the second phase to remove non-support vectors without much affect to the final solution.

3.2.2. Final Scanning

Based on the assumption that phase one produces a good approximate solution, phase two examines all training examples remaining in T one-by-one. If vector x_t violates optimality criteria with respect to temporal solution S_t then it is immediately used to update W_t to form a new solution. Otherwise, it is removed from training data T. The algorithm will stop when all training examples in T are examined.

3.3. Optimization

The SMO is used to minimize the objective function L on the selected set of vectors W_t. It is very efficient because W_t is much smaller than the whole training data. Moreover, SMO can start optimization on W_t from S_{t-1} - the optimal solution on W_{t-1} in previous iteration. The difference is only about one newly added vector. This makes SMO converge after only a small number of iterations.

3.4. Complexity Analysis

In this section we analyze the computational complexity of the proposed iSVM algorithm. At each iteration t, *step 3* takes time $O(|W_t|)$ to examine one training example. In the first phase, each training example is examined at most once (when the stopping condition iii) is applied). The second phase scans training data once more. Thus, each training example is examined at most twice. Totally *step 3* takes time $O(|W_t|l)$. Theoretically solving a QP problem of size $|W_t|$ in *step 3* takes time $O(|W_t|^3)$. However, solution S_{t-1} at iteration $t-1$ is used as an initial point, then *step 5* requires only time $O(|W_t|^2)$ per iteration (in fact it can be done in $O(|W_t|)$ by an efficient updating procedure[12, 17]). Totally *step 5* takes time $O(|W_t|^3)$. From the second phase the shrinking technique is applied (a non support vector in W_t is replaced by a new vector found by *step 3*), then size of the final working set approximates the number of final support vectors. In total, the time complexity of iSVM is $O(|W_t|^3 + |W_t|l)$, which is linear with number of training examples l and cubic with number of support vectors.

Time complexity of the SMO algorithm described in **Table 1** is $O(\tau l)$ where τ is the number of SMO iterations. In iSVM, SMO is used to solve the QP on a small set of selective training examples. From the complexity analysis above we can see that iSVM has advantage over the traditional SMO implementation when number of final SVs is much smaller than data size.

4. Related Work

iSVM belongs to the decomposition family of SVM training algorithms. In this section we discuss properties of iSVM and its relation to previously proposed methods.

In the initialization step, iSVM selects any two vectors from opposite classes (for a two-class classification task). Based on calculation (2), a closer pair of vectors will produce a better reduction in objective function L. However, our primary experiments indicate that final solutions are not affected much by this initialization scheme which has been used by CoreSVM [5, 6] or SimpleSVM [12]. Other initialization strategies include selecting randomly a set of p vectors [11], or using all training data from the beginning [4].

For updating the working set, several algorithms share the same way of adding only one vector to the WS in each optimization loop, e.g. SimpleSVM [12], CoreSVM [5, 6]. Different selection criteria have been proposed, including optimality violation [12], [5, 6], [4], probabilistic estimation [11]. iSVM differs from others in its two-stage

strategy. In the first stage, temporal solutions are not good enough to justify which training example is surely a support vector or not, so iSVM re-examines them in the second phase. Note that the working set grows from a minimum size, thus the first phase goes very fast because size of the working set $|W_t|$ is small and only the best among N examined vectors is added to the working set. It is not clearly described in [5, 6] and [12] that training examples are re-examined or not. If they are removed from T too early from beginning then it is with high probability that many good training examples (or SVs) might be removed. In contrast, if they are remaining in T all the time and re-examined many times then the computation is not efficient.

Comparing with other approximation methods, iSVM uses a rather simple stopping condition. In our point of view, CoreSVM uses a looser condition: none of N sampled training data violates the optimality condition with respect to temporal solution at iteration t^{th}. In the experiment section we will show that using this stopping condition will lead to trained SVMs with smaller number of SVs, faster training time, but bigger variation in predictive performance.

5. Experiment

In this section we describe our experiments to evaluate iSVM and comparisons with other SVM training algorithms. We select four datasets from different domains: web page categorization from UCI machine learning repository ("Web"), text-decoding used in IJCNN 2001 conference competition ("IJCNN"), extended USPS hand written digit recognition data for discriminating '0' and '1' ("zero-one"), and KDD-CUP 1999 network intrusion detection datasets used in the KDD 1999 conference competition ("KDD-CUP99"). All of the datasets summarized in **Table 3** have nearly or more than 50,000 training examples. All of our experiments were conducted on a PC Windows machine with a 3GHz CPU and 2GB RAM memory.

Table 3: Datasets used in experiment

Dataset	Dimension	# Training	# Testing
Web	300	49,749	14,951
IJCNN	22	49,990	91,701
USPS zero-one	676	266,079	75,383
KDD-CUP 1999	127	4,989,431	311,029

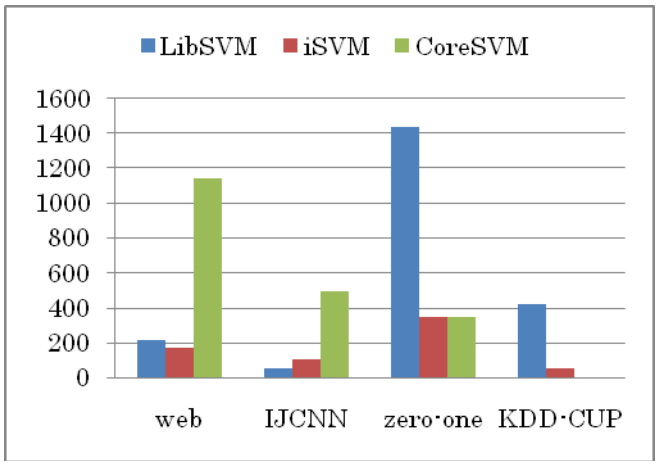

Figure 1: *Training time comparison among LibSVM, iSVM, and CoreSVM on selected datasets. Time unit is in minute for the KDD-CUP data, in second for other datasets.*

5.1. How Fast the iSVM Is

In the first experiment, we compare training performance of iSVM with LibSVM [4] – one of the best SMO implementation, CoreSVM [5, 6] – a recent proposed algorithm that has achieved a remarkable performance on the KDD-CUP 1999 dataset. Comparison criteria include training time, number of support vector, testing accuracy. Parameters were chosen for achieving good accuracy on testing data: Gaussian kernel $K(x,y) = exp(-\gamma||x-y||^2)$ with $\gamma = 0.1$, $C = 10$ for "Web", $\gamma = 1$, $C = 10$ for "IJCNN", $\gamma = 0.003$, $C = 10$ for "USPS zero-one", $\gamma = 0.1$, $C = 1{,}000{,}000$ for "KDD-CUP". As both iSVM and CoreSVM [5] use probabilistic trick to speedup training process in *step 3* training results are random variables. We conduct this experiment ten times and estimate statistics of these variables (for CoreSVM we randomly mix the original data ten times and run training program on these mixed data).

From experiment data reported in **Table 4** and **Figure 1** we can see that iSVM runs much faster than LibSVM on the "zero-one" and "KDD-CUP99" where the number of SVs is much smaller than number of training data (0.45% and 0.01% correspondingly). This shows the advantage of the two-stage incremental WS selection strategy. On the KDD-CUP 1999 data Core SVM has an incredible training time: two second in average for training on a nearly five millions training examples. In the next experiment we investigate and analyze more to show that iSVM with an early stopping condition can also achieve this training time performance. However the price is a big variation of trained machines.

Table 4: Experiment results of LibSVM, iSVM, and CoreSVM. For both iSVM and CoreSVM a probabilistic trick is used for speeding up, training time, number of SVs, and accuracy on testing data are average numbers (and standard variations for accuracy) of a ten-time experiment.

Dataset	iSVM				LibSVM			CoreSVM		
	# SV	*# SV (%)*	*Accuracy (%)*	*Time (s)*	*# SV*	*Accuracy (%)*	*Time (s)*	*# SV*	*Accuracy (%)*	*Time (s)*
Web	3,289	6.61	99.26±0.02	101	4,660	99.32	214	8,828	99.36±0.00	1140
IJCNN	2,810	5.62	98.91±0.04	170	3,154	98.99	55	5,326	99.07±0.01	495
zero-one	1,200	0.45	99.54±0.00	344	1,598	99.54	1432	1,639	99.52±0.02	343
KDD-CUP99	735	0.01	94.07±0.90	2919	1,132	92.48	25,200	43	90.66±4.16	2

5.2. iSVM With Early Stopping Conditions

In the second experiment we try iSVM with different stopping conditions on the KDD-CUP 1999 dataset. The first one is right after the first stage finished, or when there is no vector out of 59 randomly selected training data violates temporal optimality conditions. Other stopping conditions are based on total number of examined examples (by the second stage): 10%, 20%, 40% and 80% of the whole data. This experiment is also conducted ten times to have estimation of training times and testing accuracies. As we can see in **Figure 2**, with an early stopping condition iSVM runs faster but produces SVMs with higher variation in predictive accuracy). Especially, the first stage finishes after a very small number of iterations (32 in average). It means that trained machines are formed by a maximum of only 1900 training examples, corresponding to 0.04% of the whole data. This number explains why iSVM takes only 2.8 seconds to train on the KDD-CUP 1999 data with nearly five million records.

In our point of view the same phenomenon happens for CoreSVM. The big variation of the trained machine indicates that i) is a too loose stopping condition for the KDD-CUP 1999 data case. In **Figure 2** we draw variations in predictive accuracy and average training time of iSVM and CoreSVM for a comparison.

Note that there has been different approaches tackling the KDD-CUP 1999 problem. In **Table 5** we report performances produced by different methods, including data random sampling, active SVM learning [15], clustering-based [16], CoreSVM [5, 6], LibSVM [4], and iSVM.

6. Conclusion

We have introduced a new incremental algorithm for training SVMs. iSVM differs from other methods in its two-stage strategy to process training data. The first phase aims at finding a good approximate SVM solution as quickly as possible. The second phase uses temporal solutions to find out the remaining SVs. Analysis and experimental result indicate that iSVM has advantage over conventional SMO implementation on applications where number of training examples is much larger than number of SVs. Training SVMs with large number of SVs is our research issue in the future.

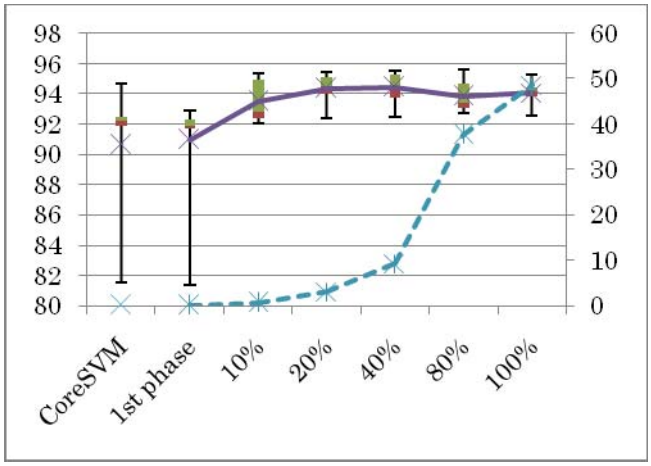

Figure 2: *Predictive accuracy in percentage (box and whisker plot) and training time in minute (dot-line plot) of iSVM with different stopping conditions on the KDD-CUP 1999 dataset. Performance of the CoreSVM on the left-side shows indicates that CoreSVM and the first phase of iSVM runs very fast (several seconds on a nearly five millions training data) but trained machines have big variation in predictive performance. Note that the testing data consists of 60,593 attacks and 250,436 normal connections, or the trivial classifier (assigning all connections to normal) can achieve 80.52% testing accuracy.*

Table 5: Performance of iSVM and other methods on the KDD-CUP 1999 dataset. Training time of random sampling and cluster-based methods include preprocessing time (sampling and clustering) and SVM training on sampled data or on clusters. Performances of LibSVM and iSVM are our experiment results. Others are taken from [5] for a reference.

Method	Training time (s)	Testing errors
Random sampling (5%)	16,351	25,587
Active learning	94,192	21,634
Cluster-based	4,752	20,938
LibSVM	25,200	23,389
CoreSVM	1	19,513
iSVM	2,919	18,444

Reference

[1] C. Cortes and V. Vapnik, “Support vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.

[2] C. Cristianini and J. Shawe-Taylor, “An Introduction to Support Vector Machines”, Cambridge University Press, 2000.

[3] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121-167, 1998.

[4] R. Fan, P. Chen, and C. Lin, “Working Set Selection Using Second Order Information for Training Support Vector Machines". J. Mach. Learn. Res. 6, 1889-1918, 2005.

[5] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, “Core vector machines: Fast SVM training on very large datasets," in J. Mach. Learn. Res., vol. 6, pp. 363-392, 2005.

[6] I. W. Tsang, A. Kocsor, and J. T. Kwok. “Simpler core vector machines with enclosing balls” Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML), pp.911-918, Corvallis, Oregon, USA, June 2007.

[7] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: an application to face detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[8] E. Osuna, R. Freund, and F. Girosi, “An improved training algorithm for support vector machines," in Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop, N. M. J. Principe, L. Gile and E. Wilson, Eds., New York, pp. 276-285, 1997.

[9] T. Joachims, “Making large-scale support vector machine learning practical," in Advances in Kernel Methods: Support Vector Machines, A. S. B. Scholkopf, C. Burges, Ed., MIT Press, Cambridge, MA, 1998.

[10] J. Platt, “Fast training of support vector machines using sequential minimal optimization," in Advances in Kernel Methods - Support Vector Learning, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds., Cambridge, MA: MIT Press, 1999.

[11] P. Mitra, C.A. Murthy, S. K. Pal, "Data Condensation in Large Databases by Incremental Learning with Support Vector Machines," icpr, p. 2708, 15th International Conference on Pattern Recognition (ICPR'00) - Volume 2, 2000.

[12] S. Vishwanathan, A. Smola, and M. Murty, "SimpleSVM", Proceedings of the Twentieth International Conference on Machine Learning (pp. 760–767), 2003.

[13] A. Smola and B. Scholkopf, "Sparse greedy matrix approximation for machine learning", in Proceedings of the Seventeenth International Conference on Machine Learning, pages 911–918, Stanford, CA, USA, June 2000.

[14] J. X. Dong, A. Krzyzak, and C. Y. Suen, "Fast SVM Training Algorithm with Decomposition on Very Large Datasets," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 4, pp. 603--618, April 2005.

[15] H. Yu, J. Yang, and J. Han. “Classifying large data sets using SVM with hierarchical clusters”. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and *Data Mining*, pages 306–315, Washington DC, USA, 2003.

[16] D. Boley and D. Cao. “Training support vector machine using adaptive clustering”. In Proceedings of the SIAM International Conference on Data Mining, pages 126–137, Lake Buena Vista, FL, USA, April 2004.

[17] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning", In T. Leen, T. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems 13, Cambridge, MA, 2001. MIT Press.