Journal of Machine Learning Research 6 (2005) 1579–1619 Submitted 3/05; Published 9/05

Fast Kernel Classifiers with Online and Active Learning

Antoine Bordes ANTOINE.BORDES@BDE.ESPCI.FR
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA, and
Ecole Supérieure de Physique et Chimie Industrielles
10 rue Vauquelin
75231 Paris CEDEX 05, France

Seyda Ertekin SEYDA@PSU.EDU
The Pennsylvania State University
University Park, PA 16802, USA

Jason Weston JASONW@NEC-LABS.COM
Léon Bottou LEON@BOTTOU.ORG
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA

Editor: Nello Cristianini

Abstract

Very high dimensional learning systems become theoretically possible when training examples are abundant. The computing cost then becomes the limiting factor. Any efficient learning algorithm should at least take a brief look at each example. But should all examples be given equal attention?

This contribution proposes an empirical answer. We first present an online SVM algorithm based on this premise. LASVM yields competitive misclassification rates after a single pass over the training examples, outspeeding state-of-the-art SVM solvers. Then we show how active example selection can yield faster training, higher accuracies, and simpler models, using only a fraction of the training example labels.

1. Introduction

Electronic computers have vastly enhanced our ability to compute complicated statistical models. Both theory and practice have adapted to take into account the essential compromise between the number of examples and the model capacity (Vapnik, 1998). Cheap, pervasive and networked computers are now enhancing our ability to collect observations to an even greater extent. Data sizes outgrow computer speed. During the last decade, processors became 100 times faster, hard disks became 1000 times bigger.

Very high dimensional learning systems become theoretically possible when training examples are abundant. The computing cost then becomes the limiting factor. Any efficient learning algorithm should at least pay a brief look at each example. But should all training examples be given equal attention?

©2005 Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou.

This contribution proposes an empirical answer:

- Section 2 presents kernel classifiers such as Support Vector Machines (SVM). Kernel classifiers are convenient for our purposes because they clearly express their internal states in terms of subsets of the training examples.
- Section 3 proposes a novel online algorithm, LASVM, which converges to the SVM solution. Experimental evidence on diverse data sets indicates that it reliably reaches competitive accuracies after performing a single pass over the training set. It uses less memory and trains significantly faster than state-of-the-art SVM solvers.
- Section 4 investigates two criteria to select informative training examples at each iteration instead of sequentially processing all examples. Empirical evidence shows that selecting informative examples without making use of the class labels can drastically reduce the training time and produce much more compact classifiers with equivalent or superior accuracy.
- Section 5 discusses the above results and formulates theoretical questions. The simplest question involves the convergence of these algorithms and is addressed by the appendix. Other questions of greater importance remain open.

2. Kernel Classifiers

Early linear classifiers associate classes $y = \pm 1$ to patterns x by first transforming the patterns into feature vectors $\Phi(x)$ and taking the sign of a linear discriminant function:

$$\hat{y}(x) = w' \Phi(x) + b. \tag{1}$$

The parameters w and b are determined by running some learning algorithm on a set of training examples $(x_1, y_1) \cdots (x_n, y_n)$. The feature function Φ is usually hand chosen for each particular problem (Nilsson, 1965).

Aizerman et al. (1964) transform such linear classifiers by leveraging two theorems of the *Reproducing Kernel* theory (Aronszajn, 1950).

The *Representation Theorem* states that many Φ-machine learning algorithms produce parameter vectors w that can be expressed as a linear combinations of the training patterns:

$$w = \sum_{i=1}^{n} \alpha_i \Phi(x_i).$$

The linear discriminant function (1) can then be written as a *kernel expansion*

$$\hat{y}(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b, \tag{2}$$

where the *kernel* function $K(x,y)$ represents the dot products $\Phi(x)'\Phi(y)$ in feature space. This expression is most useful when a large fraction of the coefficients α_i are zero. Examples such that $\alpha_i \neq 0$ are then called *Support Vectors*.

Mercer's Theorem precisely states which kernel functions correspond to a dot product for some feature space. Kernel classifiers deal with the kernel function $K(x,y)$ without explicitly using the

corresponding feature function $\Phi(x)$. For instance, the well known *RBF* kernel $K(x,y) = e^{-\gamma\|x-y\|^2}$ defines an implicit feature space of infinite dimension.

Kernel classifiers handle such large feature spaces with the comparatively modest computational costs of the kernel function. On the other hand, kernel classifiers must control the decision funcion complexity in order to avoid overfitting the training data in such large feature spaces. This can be achieved by keeping the number of support vectors as low as possible (Littlestone and Warmuth, 1986) or by searching decision boundaries that separate the examples with the largest margin (Vapnik and Lerner, 1963; Vapnik, 1998).

2.1 Support Vector Machines

Support Vector Machines were defined by three incremental steps. First, Vapnik and Lerner (1963) propose to construct the *Optimal Hyperplane*, that is, the linear classifier that separates the training examples with the widest margin. Then, Guyon, Boser, and Vapnik (1993) propose to construct the Optimal Hyperplane in the feature space induced by a kernel function. Finally, Cortes and Vapnik (1995) show that noisy problems are best addressed by allowing some examples to violate the margin condition.

Support Vector Machines minimize the following objective function in feature space:

$$\min_{w,b} \|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{with} \quad \begin{cases} \forall\, i & y_i\hat{y}(x_i) \geq 1-\xi_i \\ \forall\, i & \xi_i \geq 0. \end{cases} \tag{3}$$

For very large values of the hyper-parameter C, this expression minimizes $\|w\|^2$ under the constraint that all training examples are correctly classified with a margin $y_i\hat{y}(x_i)$ greater than 1. Smaller values of C relax this constraint and produce markedly better results on noisy problems (Cortes and Vapnik, 1995).

In practice this is achieved by solving the dual of this convex optimization problem. The coefficients α_i of the SVM kernel expansion (2) are found by defining the dual objective function

$$W(\alpha) = \sum_i \alpha_i y_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(x_i,x_j) \tag{4}$$

and solving the SVM *Quadratic Programming* (QP) problem:

$$\max_{\alpha} W(\alpha) \quad \text{with} \quad \begin{cases} \sum_i \alpha_i = 0 \\ A_i \leq \alpha_i \leq B_i \\ \quad A_i = \min(0, Cy_i) \\ \quad B_i = \max(0, Cy_i). \end{cases} \tag{5}$$

The above formulation slightly deviates from the standard formulation (Cortes and Vapnik, 1995) because it makes the α_i coefficients positive when $y_i = +1$ and negative when $y_i = -1$.

SVMs have been very successful and are very widely used because they reliably deliver state-of-the-art classifiers with minimal tweaking.

Computational Cost of SVMs There are two intuitive lower bounds on the computational cost of any algorithm able to solve the SVM QP problem for arbitrary matrices $K_{ij} = K(x_i,x_j)$.

1. Suppose that an oracle reveals whether $\alpha_i = 0$ or $\alpha_i = \pm C$ for all $i = 1 \ldots n$. Computing the remaining $0 < |\alpha_i| < C$ amounts to inverting a matrix of size $R \times R$ where R is the number of support vectors such that $0 < |\alpha_i| < C$. This typically requires a number of operations proportional to R^3.

2. Simply verifying that a vector α is a solution of the SVM QP problem involves computing the gradients of $W(\alpha)$ and checking the Karush-Kuhn-Tucker optimality conditions (Vapnik, 1998). With n examples and S support vectors, this requires a number of operations proportional to nS.

Few support vectors reach the upper bound C when it gets large. The cost is then dominated by the $R^3 \approx S^3$. Otherwise the term nS is usually larger. The final number of support vectors therefore is the critical component of the computational cost of the SVM QP problem.

Assume that increasingly large sets of training examples are drawn from an unknown distribution $P(x,y)$. Let $\mathcal{B}$ be the error rate achieved by the best decision function (1) for that distribution. When $\mathcal{B} > 0$, Steinwart (2004) shows that the number of support vectors is asymptotically equivalent to $2n\mathcal{B}$. Therefore, regardless of the exact algorithm used, the asymptotical computational cost of solving the SVM QP problem grows at least like n^2 when C is small and n^3 when C gets large. Empirical evidence shows that modern SVM solvers (Chang and Lin, 2001-2004; Collobert and Bengio, 2001) come close to these scaling laws.

Practice however is dominated by the constant factors. When the number of examples grows, the kernel matrix $K_{ij} = K(x_i, x_j)$ becomes very large and cannot be stored in memory. Kernel values must be computed on the fly or retrieved from a cache of often accessed values. When the cost of computing each kernel value is relatively high, the kernel cache hit rate becomes a major component of the cost of solving the SVM QP problem (Joachims, 1999). Larger problems must be addressed by using algorithms that access kernel values with very consistent patterns.

Section 3 proposes an Online SVM algorithm that accesses kernel values very consistently. Because it computes the SVM optimum, this algorithm cannot improve on the n^2 lower bound. Because it is an online algorithm, early stopping strategies might give approximate solutions in much shorter times. Section 4 suggests that this can be achieved by carefully choosing which examples are processed at each iteration.

Before introducing the new Online SVM, let us briefly describe other existing online kernel methods, beginning with the kernel Perceptron.

2.2 Kernel Perceptrons

The earliest kernel classifiers (Aizerman et al., 1964) were derived from the Perceptron algorithm (Rosenblatt, 1958). The decision function (2) is represented by maintaining the set S of the indices i of the support vectors. The bias parameter b remains zero.

Kernel Perceptron
1) $\mathcal{S} \leftarrow \emptyset, \quad b \leftarrow 0.$
2) Pick a random example (x_t, y_t)
3) Compute $\hat{y}(x_t) = \sum_{i \in \mathcal{S}} \alpha_i K(x_t, x_i) + b$
4) If $y_t \hat{y}(x_t) \leq 0$ then $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}, \quad \alpha_t \leftarrow y_t$
5) Return to step 2.

Such *Online Learning Algorithms* require very little memory because the examples are processed one by one and can be discarded after being examined.

Iterations such that $y_t \hat{y}(x_t) < 0$ are called *mistakes* because they correspond to patterns misclassified by the perceptron decision boundary. The algorithm then modifies the decision boundary by inserting the misclassified pattern into the kernel expansion. When a solution exists, Novikoff's Theorem (Novikoff, 1962) states that the algorithm converges after a finite number of mistakes, or equivalently after inserting a finite number of support vectors. Noisy data sets are more problematic.

Large Margin Kernel Perceptrons The success of Support Vector Machines has shown that large classification margins were desirable. On the other hand, the Kernel Perceptron (Section 2.2) makes no attempt to achieve large margins because it happily ignores training examples that are very close to being misclassified.

Many authors have proposed to close the gap with online kernel classifiers by providing larger margins. The Averaged Perceptron (Freund and Schapire, 1998) decision rule is the majority vote of all the decision rules obtained after each iteration of the Kernel Perceptron algorithm. This choice provides a bound comparable to those offered in support of SVMs. Other algorithms (Frieß et al., 1998; Gentile, 2001; Li and Long, 2002; Crammer and Singer, 2003) explicitely construct larger margins. These algorithms modify the decision boundary whenever a training example is either misclassified or classified with an insufficient margin. Such examples are then inserted into the kernel expansion with a suitable coefficient. Unfortunately, this change significantly increases the number of mistakes and therefore the number of support vectors. The increased computational cost and the potential overfitting undermines the positive effects of the increased margin.

Kernel Perceptrons with Removal Step This is why Crammer et al. (2004) suggest an additional step for *removing* support vectors from the kernel expansion (2). The Budget Perceptron performs very nicely on relatively clean data sets.

Budget Kernel Perceptron (β, N)
1) $\mathcal{S} \leftarrow \emptyset, \quad b \leftarrow 0.$
2) Pick a random example (x_t, y_t)
3) Compute $\hat{y}(x_t) = \sum_{i \in \mathcal{S}} \alpha_i K(x_t, x_i) + b$
4) If $y_t \hat{y}(x_t) \leq \beta$ then,
   4a) $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}, \quad \alpha_t \leftarrow y_t$
   4b) If $|\mathcal{S}| > N$ then $\mathcal{S} \leftarrow \mathcal{S} - \{ \arg\max_{i \in \mathcal{S}} y_i (\hat{y}(x_i) - \alpha_i K(x_i, x_i)) \}$
5) Return to step 2.

Online kernel classifiers usually experience considerable problems with noisy data sets. Each iteration is likely to cause a mistake because the best achievable misclassification rate for such problems is high. The number of support vectors increases very rapidly and potentially causes overfitting and poor convergence. More sophisticated support vector removal criteria avoid this drawback (Weston et al., 2005). This modified algorithm outperforms all other *online* kernel classifiers on noisy data sets and matches the performance of Support Vector Machines with less support vectors.

3. Online Support Vector Machines

This section proposes a novel online algorithm named LASVM that converges to the SVM solution. This algorithm furthers ideas first presented by Bordes and Bottou (2005). Unlike this previous

work, LASVM relies on the traditional "soft margin" SVM formulation, handles noisy data sets, and is nicely related to the SMO algorithm. Experimental evidence on multiple data sets indicates that it reliably reaches competitive test error rates after performing a single pass over the training set. It uses less memory and trains significantly faster than state-of-the-art SVM solvers.

3.1 Quadratic Programming Solvers for SVMs

Sequential Direction Search Efficient numerical algorithms have been developed to solve the SVM QP problem (5). The best known methods are the Conjugate Gradient method (Vapnik, 1982, pages 359–362) and the Sequential Minimal Optimization (Platt, 1999). Both methods work by making successive searches along well chosen directions.

Each direction search solves the restriction of the SVM problem to the half-line starting from the current vector α and extending along the specified direction u. Such a search yields a new feasible vector $\alpha + \lambda^* u$, where

$$\lambda^* = \arg\max W(\alpha + \lambda u) \quad \text{with} \quad 0 \le \lambda \le \phi(\alpha, u). \tag{6}$$

The upper bound $\phi(\alpha, u)$ ensures that $\alpha + \lambda u$ is feasible as well:

$$\phi(\alpha, u) \;=\; \min \left\{ \begin{array}{cl} 0 & \text{if} \;\; \sum_k u_k \neq 0 \\ (B_i - \alpha_i)/u_i & \text{for all } i \text{ such that } u_i > 0 \\ (A_j - \alpha_j)/u_j & \text{for all } j \text{ such that } u_j < 0. \end{array} \right\} \tag{7}$$

Calculus shows that the optimal value is achieved for

$$\lambda^* = \min \left\{ \; \phi(\alpha, u) \;,\; \frac{\sum_i g_i u_i}{\sum_{i,j} u_i u_j K_{ij}} \; \right\} \tag{8}$$

where $K_{ij} = K(x_i, x_j)$ and $g = (g_1 \ldots g_n)$ is the gradient of $W(\alpha)$, and

$$g_k \;=\; \frac{\partial W(\alpha)}{\partial \alpha_k} \;=\; y_k - \sum_i \alpha_i K(x_i, x_k) \;=\; y_k - \hat{y}(x_k) + b. \tag{9}$$

Sequential Minimal Optimization Platt (1999) observes that direction search computations are much faster when the search direction u mostly contains zero coefficients. At least two coefficients are needed to ensure that $\sum_k u_k = 0$. The *Sequential Minimal Optimization* (SMO) algorithm uses search directions whose coefficients are all zero except for a single $+1$ and a single -1.

Practical implementations of the SMO algorithm (Chang and Lin, 2001-2004; Collobert and Bengio, 2001) usually rely on a small positive tolerance $\tau > 0$. They only select directions u such that $\phi(\alpha, u) > 0$ and $u'g > \tau$. This means that we can move along direction u without immediately reaching a constraint and increase the value of $W(\alpha)$. Such directions are defined by the so-called τ-*violating pair* (i, j):

$$(i, j) \text{ is a } \tau\text{-}\textit{violating pair} \iff \left\{ \begin{array}{c} \alpha_i < B_i \\ \alpha_j > A_j \\ g_i - g_j > \tau. \end{array} \right.$$

SMO Algorithm

1) Set $\alpha \leftarrow 0$ and compute the initial gradient g (equation 9)
2) Choose a τ-violating pair (i, j). Stop if no such pair exists.
3) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$
   $\alpha_i \leftarrow \alpha_i + \lambda\,, \quad \alpha_j \leftarrow \alpha_j - \lambda$
   $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \{1 \ldots n\}$
4) Return to step (2)

The above algorithm does not specify how exactly the τ-violating pairs are chosen. Modern implementations of SMO select the τ-violating pair (i, j) that maximizes the directional gradient $u'g$. This choice was described in the context of Optimal Hyperplanes in both (Vapnik, 1982, pages 362–364) and (Vapnik et al., 1984).

Regardless of how exactly the τ-violating pairs are chosen, Keerthi and Gilbert (2002) assert that the SMO algorithm stops after a finite number of steps. This assertion is correct despite a slight flaw in their final argument (Takahashi and Nishi, 2003).

When SMO stops, no τ-violating pair remain. The corresponding α is called a τ-*approximate solution*. Proposition 13 in appendix A establishes that such approximate solutions indicate the location of the solution(s) of the SVM QP problem when the tolerance τ become close to zero.

3.2 Online LASVM

This section presents a novel online SVM algorithm named LASVM. There are two ways to view this algorithm. LASVM is an online kernel classifier sporting a support vector removal step: vectors collected in the current kernel expansion can be removed during the online process. LASVM also is a reorganization of the SMO sequential direction searches and, as such, converges to the solution of the SVM QP problem.

Compared to basic kernel perceptrons (Aizerman et al., 1964; Freund and Schapire, 1998), the LASVM algorithm features a removal step and gracefully handles noisy data. Compared to kernel perceptrons with removal steps (Crammer et al., 2004; Weston et al., 2005), LASVM converges to the known SVM solution. Compared to a traditional SVM solver (Platt, 1999; Chang and Lin, 2001-2004; Collobert and Bengio, 2001), LASVM brings the computational benefits and the flexibility of online learning algorithms. Experimental evidence indicates that LASVM matches the SVM accuracy after a single sequential pass over the training examples.

This is achieved by alternating two kinds of direction searches named PROCESS and REPROCESS. Each direction search involves a pair of examples. Direction searches of the PROCESS kind involve at least one example that is not a support vector of the current kernel expansion. They potentially can change the coefficient of this example and make it a support vector. Direction searches of the REPROCESS kind involve two examples that already are support vectors in the current kernel expansion. They potentially can zero the coefficient of one or both support vectors and thus remove them from the kernel expansion.

Building Blocks The LASVM algorithm maintains three essential pieces of information: the set $\mathcal{S}$ of potential support vector indices, the coefficients α_i of the current kernel expansion, and the partial derivatives g_i defined in (9). Variables α_i and g_i contain meaningful values when $i \in \mathcal{S}$ only.

The coefficient α_i are assumed to be null if $i \notin \mathcal{S}$. On the other hand, set $\mathcal{S}$ might contain a few indices i such that $\alpha_i = 0$.

The two basic operations of the Online LASVM algorithm correspond to steps 2 and 3 of the SMO algorithm. These two operations differ from each other because they have different ways to select τ-violating pairs.

The first operation, PROCESS, attempts to insert example $k \notin \mathcal{S}$ into the set of current support vectors. In the online setting this can be used to process a new example at time t. It first adds example $k \notin \mathcal{S}$ into $\mathcal{S}$ (step 1-2). Then it searches a second example in $\mathcal{S}$ to find the τ-violating pair with maximal gradient (steps 3-4) and performs a direction search (step 5).

LASVM PROCESS(k)

1) Bail out if $k \in \mathcal{S}$.

2) $\alpha_k \leftarrow 0\,, \quad g_k \leftarrow y_k - \sum_{s\in\mathcal{S}} \alpha_s K_{ks}\,, \quad \mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$

3) If $y_k = +1$ then
$\quad i \leftarrow k\,, \quad j \leftarrow \arg\min_{s\in S} g_s$ with $\alpha_s > A_s$
else
$\quad j \leftarrow k\,, \quad i \leftarrow \arg\max_{s\in S} g_s$ with $\alpha_s < B_s$

4) Bail out if (i, j) is not a τ-violating pair.

5) $\lambda \leftarrow \min\left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$
$\alpha_i \leftarrow \alpha_i + \lambda\,, \quad \alpha_j \leftarrow \alpha_j - \lambda$
$g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \mathcal{S}$

The second operation, REPROCESS, removes some elements from $\mathcal{S}$. It first searches the τ-violating pair of elements of $\mathcal{S}$ with maximal gradient (steps 1-2), and performs a direction search (step 3). Then it removes blatant non support vectors (step 4). Finally it computes two useful quantities: the bias term b of the decision function (2) and the gradient δ of the most τ-violating pair in $\mathcal{S}$.

LASVM REPROCESS

1) $i \leftarrow \arg\max_{s\in\mathcal{S}} g_s$ with $\alpha_s < B_s$
$j \leftarrow \arg\min_{s\in\mathcal{S}} g_s$ with $\alpha_s > A_s$

2) Bail out if (i, j) is not a τ-violating pair.

3) $\lambda \leftarrow \min\left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$
$\alpha_i \leftarrow \alpha_i + \lambda\,, \quad \alpha_j \leftarrow \alpha_j - \lambda$
$g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \mathcal{S}$

4) $i \leftarrow \arg\max_{s\in\mathcal{S}} g_s$ with $\alpha_s < B_s$
$j \leftarrow \arg\min_{s\in\mathcal{S}} g_s$ with $\alpha_s > A_s$
For all $s \in \mathcal{S}$ such that $\alpha_s = 0$
$\quad$ If $y_s = -1$ and $g_s \geq g_i$ then $\mathcal{S} = \mathcal{S} - \{s\}$
$\quad$ If $y_s = +1$ and $g_s \leq g_j$ then $\mathcal{S} = \mathcal{S} - \{s\}$

5) $b \leftarrow (g_i + g_j)/2\,, \quad \delta \leftarrow g_i - g_j$

Online LASVM After initializing the state variables (step 1), the Online LASVM algorithm alternates PROCESS and REPROCESS a predefined number of times (step 2). Then it simplifies the kernel expansion by running REPROCESS to remove all τ-violating pairs from the kernel expansion (step 3).

LASVM

1) **Initialization**:
   Seed $\mathcal{S}$ with a few examples of each class.
   Set $\alpha \leftarrow 0$ and compute the initial gradient g (equation 9)
2) **Online Iterations**:
   Repeat a predefined number of times:
   - Pick an example k_t
   - Run PROCESS(k_t).
   - Run REPROCESS once.
3) **Finishing**:
   Repeat REPROCESS until $\delta \leq \tau$.

LASVM can be used in the online setup where one is given a continuous stream of fresh random examples. The online iterations process fresh training examples as they come. LASVM can also be used as a stochastic optimization algorithm in the offline setup where the complete training set is available before hand. Each iteration randomly picks an example from the training set.

In practice we run the LASVM online iterations in epochs. Each epoch sequentially visits all the randomly shuffled training examples. After a predefined number P of epochs, we perform the finishing step. A single epoch is consistent with the use of LASVM in the online setup. Multiple epochs are consistent with the use of LASVM as a stochastic optimization algorithm in the offline setup.

Convergence of the Online Iterations Let us first ignore the finishing step (step 3) and assume that online iterations (step 2) are repeated indefinitely. Suppose that there are remaining τ-violating pairs at iteration T.

a.) If there are τ-violating pairs (i, j) such that $i \in \mathcal{S}$ and $j \in \mathcal{S}$, one of them will be exploited by the next REPROCESS.

b.) Otherwise, if there are τ-violating pairs (i, j) such that $i \in \mathcal{S}$ or $j \in \mathcal{S}$, each subsequent PROCESS has a chance to exploit one of them. The intervening REPROCESS do nothing because they bail out at step 2.

c.) Otherwise, all τ-violating pairs involve indices outside $\mathcal{S}$. Subsequent calls to PROCESS and REPROCESS bail out until we reach a time $t > T$ such that $k_t = i$ and $k_{t+1} = j$ for some τ-violating pair (i, j). The first PROCESS then inserts i into $\mathcal{S}$ and bails out. The following REPROCESS bails out immediately. Finally the second PROCESS locates pair (i, j).

This case is not important in practice. There usually is a support vector $s \in \mathcal{S}$ such that $A_s < \alpha_s < B_s$. We can then write $g_i - g_j = (g_i - g_s) + (g_s - g_j) \leq 2\tau$ and conclude that we already have reached a 2τ-approximate solution.

The LASVM online iterations therefore work like the SMO algorithm. Remaining τ-violating pairs is sooner or later exploited by either PROCESS or REPROCESS. As soon as a τ-approximate solution is reached, the algorithm stops updating the coefficients α. Theorem 18 in the appendix gives more precise convergence results for this stochastic algorithm.

The finishing step (step 3) is only useful when one limits the number of online iterations. Running LASVM usually consists in performing a predefined number P of epochs and running the finishing step. Each epoch performs n online iterations by sequentially visiting the randomly shuffled training examples. Empirical evidence suggests indeed that a *single epoch* yields a classifier almost as good as the SVM solution.

Computational Cost of LASVM Both PROCESS and REPROCESS require a number of operations proportional to the number S of support vectors in set $\mathcal{S}$. Performing P epochs of online iterations requires a number of operations proportional to $nP\bar{S}$. The average number $\bar{S}$ of support vectors scales no more than linearly with n because each online iteration brings at most one new support vector. The asymptotic cost therefore grows like n^2 at most. The finishing step is similar to running a SMO solver on a SVM problem with only S training examples. We recover here the n^2 to n^3 behavior of standard SVM solvers.

Online algorithms access kernel values with a very specific pattern. Most of the kernel values accessed by PROCESS and REPROCESS involve only support vectors from set $\mathcal{S}$. Only PROCESS on a new example x_{k_t} accesses S fresh kernel values $K(x_{k_t}, x_i)$ for $i \in \mathcal{S}$.

Implementation Details Our LASVM implementation reorders the examples after every PROCESS or REPROCESS to ensure that the current support vectors come first in the reordered list of indices. The kernel cache records truncated rows of the reordered kernel matrix. SVMLight (Joachims, 1999) and LIBSVM (Chang and Lin, 2001-2004) also perform such reorderings, but do so rather infrequently (Joachims, 1999). The reordering overhead is acceptable during the online iterations because the computation of fresh kernel values takes much more time.

Reordering examples during the finishing step was more problematic. We eventually deployed an adaptation of the *shrinking* heuristic (Joachims, 1999) for the finishing step only. The set $\mathcal{S}$ of support vectors is split into an active set $\mathcal{S}_a$ and an inactive set $\mathcal{S}_i$. All support vectors are initially active. The REPROCESS iterations are restricted to the active set $\mathcal{S}_a$ and do not perform any reordering. About every 1000 iterations, support vectors that hit the boundaries of the box constraints are either removed from the set $\mathcal{S}$ of support vectors or moved from the active set $\mathcal{S}_a$ to the inactive set $\mathcal{S}_i$. When all τ-violating pairs of the active set are exhausted, the inactive set examples are transferred back into the active set. The process continues as long as the merged set contains τ-violating pairs.

3.3 MNIST Experiments

The Online LASVM was first evaluated on the MNIST[1] handwritten digit data set (Bottou et al., 1994). Computing kernel values for this data set is relatively expensive because it involves dot products of 784 gray level pixel values. In the experiments reported below, all algorithms use the same code for computing kernel values. The ten binary classification tasks consist of separating each digit class from the nine remaining classes. All experiments use RBF kernels with $\gamma = 0.005$

1. This data set is available at `http://yann.lecun.com/exdb/mnist`.

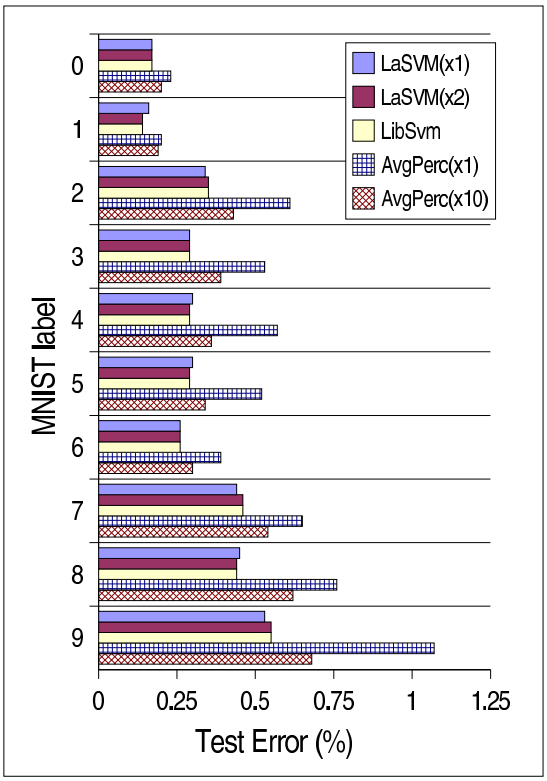

Figure 1: Compared test error rates for the ten MNIST binary classifiers.

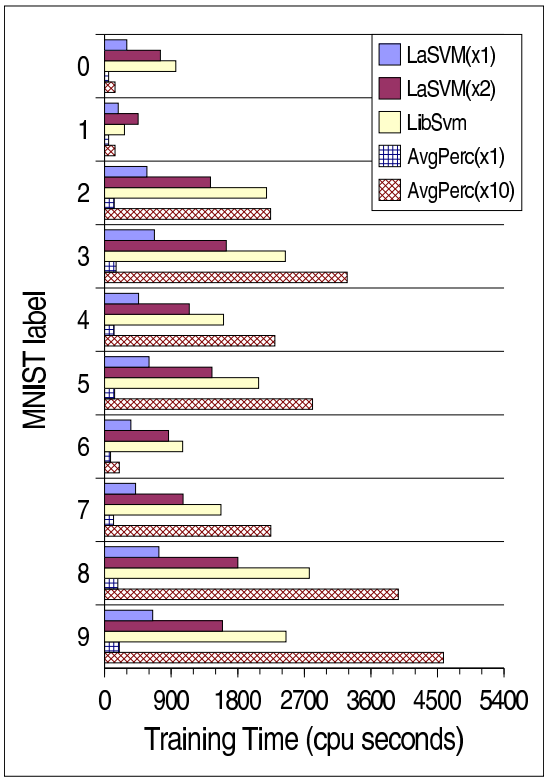

Figure 2: Compared training times for the ten MNIST binary classifiers.

and the same training parameters $C = 1000$ and $\tau = 0.001$. Unless indicated otherwise, the kernel cache size is 256MB.

LASVM versus Sequential Minimal Optimization Baseline results were obtained by running the state-of-the-art SMO solver LIBSVM (Chang and Lin, 2001-2004). The resulting classifier accurately represents the SVM solution.

Two sets of results are reported for LASVM. The LASVM$\times$1 results were obtained by performing a single epoch of online iterations: each training example was processed exactly once during a single sequential sweep over the training set. The LASVM$\times$2 results were obtained by performing two epochs of online iterations.

Figures 1 and 2 show the resulting test errors and training times. LASVM$\times$1 runs about three times faster than LIBSVM and yields test error rates very close to the LIBSVM results. Standard paired significance tests indicate that these small differences are not significant. LASVM$\times$2 usually runs faster than LIBSVM and very closely tracks the LIBSVM test errors.

Neither the LASVM$\times$1 or LASVM$\times$2 experiments yield the exact SVM solution. On this data set, LASVM reaches the exact SVM solution after about five epochs. The first two epochs represent the bulk of the computing time. The remaining epochs run faster when the kernel cache is large

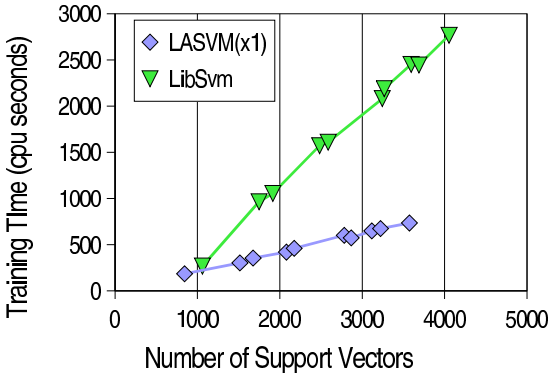

Figure 3: Training time as a function of the number of support vectors.

Algorithm	Error	Time
LIBSVM	**1.36%**	17400s
LASVM×1	1.42%	**4950s**
LASVM×2	**1.36%**	12210s

Figure 4: Multiclass errors and training times for the MNIST data set.

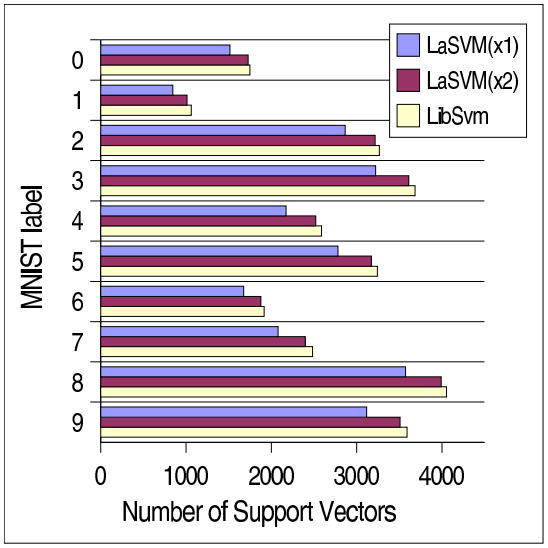

Figure 5: Compared numbers of support vectors for the ten MNIST binary classifiers.

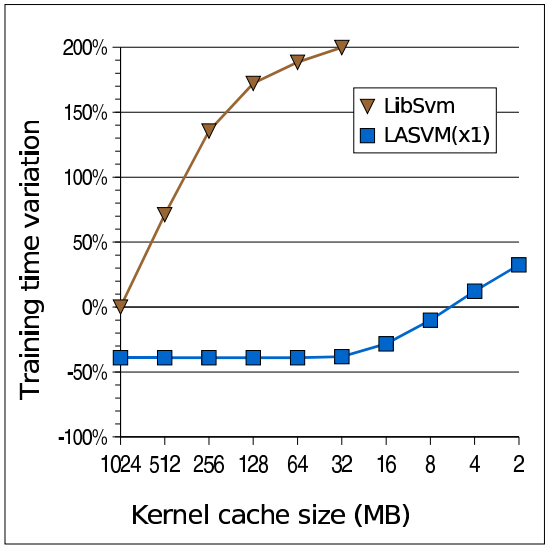

Figure 6: Training time variation as a function of the cache size. Relative changes with respect to the 1GB LIBSVM times are averaged over all ten MNIST classifiers.

enough to hold all the dot products involving support vectors. Yet the overall optimization times are not competitive with those achieved by LIBSVM.

Figure 3 shows the training time as a function of the final number of support vectors for the ten binary classification problems. Both LIBSVM and LASVM×1 show a linear dependency. The Online LASVM algorithm seems more efficient overall.

Figure 4 shows the multiclass error rates and training times obtained by combining the ten classifiers using the well known 1-versus-rest scheme (Schölkopf and Smola, 2002). LASVM×1 provides almost the same accuracy with much shorter training times. LASVM×2 reproduces the LIBSVM accuracy with slightly shorter training time.

Figure 5 shows the resulting number of support vectors. A single epoch of the Online LASVM algorithm gathers most of the support vectors of the SVM solution computed by LIBSVM. The first iterations of the Online LASVM might indeed ignore examples that later become support vectors. Performing a second epoch captures most of the missing support vectors.

LASVM versus the Averaged Perceptron The computational advantage of LASVM relies on its apparent ability to match the SVM accuracies after a single epoch. Therefore it must be compared with algorithms such as the Averaged Perceptron (Freund and Schapire, 1998) that provably match well known upper bounds on the SVM accuracies. The AVGPERC×1 results in Figures 1 and 2 were obtained after running a single epoch of the Averaged Perceptron. Although the computing times are very good, the corresponding test errors are not competitive with those achieved by either LIBSVM or LASVM. Freund and Schapire (1998) suggest that the Averaged Perceptron approaches the actual SVM accuracies after 10 to 30 epochs. Doing so no longer provides the theoretical guarantees. The AVGPERC×10 results in Figures 1 and 2 were obtained after ten epochs. Test error rates indeed approach the SVM results. The corresponding training times are no longer competitive.

Impact of the Kernel Cache Size These training times stress the importance of the kernel cache size. Figure 2 shows how the AVGPERC×10 runs much faster on problems 0, 1, and 6. This is happening because the cache is large enough to accomodate the dot products of all examples with all support vectors. Each repeated iteration of the Average Perceptron requires very few additional kernel evaluations. This is much less likely to happen when the training set size increases. Computing times then increase drastically because repeated kernel evaluations become necessary.

Figure 6 compares how the LIBSVM and LASVM×1 training times change with the kernel cache size. The vertical axis reports the relative changes with respect to LIBSVM with one gigabyte of kernel cache. These changes are averaged over the ten MNIST classifiers. The plot shows how LASVM tolerates much smaller caches. On this problem, *LASVM with a 8MB cache runs slightly faster than LIBSVM with a 1024MB cache.*

Useful orders of magnitude can be obtained by evaluating how large the kernel cache must be to avoid the systematic recomputation of dot-products. Following the notations of Section 2.1, let n be the number of examples, S be the number of support vectors, and $R \leq S$ the number of support vectors such that $0 < |\alpha_i| < C$.

- In the case of LIBSVM, the cache must accommodate about nR terms: the examples selected for the SMO iterations are usually chosen among the R free support vectors. Each SMO iteration needs n distinct dot-products for each selected example.

- To perform a *single* LASVM epoch, the cache must only accommodate about SR terms: since the examples are visited only once, the dot-products computed by a PROCESS operation can only be reutilized by subsequent REPROCESS operations. The examples selected by REPROCESS are usually chosen amont the R free support vectors; for each selected example, REPROCESS needs one distinct dot-product per support vector in set $\mathcal{S}$.

- To perform *multiple* LASVM epochs, the cache must accommodate about nS terms: the dot-products computed by processing a particular example are reused when processing the same example again in subsequent epochs. This also applies to multiple Averaged Perceptron epochs.

An efficient single epoch learning algorithm is therefore very desirable when one expects S to be much smaller than n. Unfortunately, this may not be the case when the data set is noisy. Section 3.4 presents results obtained in such less favorable conditions. Section 4 then proposes an active learning method to contain the growth of the number of support vectors, and recover the full benefits of the online approach.

3.4 Multiple Data Set Experiments

Further experiments were carried out with a collection of standard data sets representing diverse noise conditions, training set sizes, and input dimensionality. Figure 7 presents these data sets and the parameters used for the experiments.

Kernel computation times for these data sets are extremely fast. The data either has low dimensionality or can be represented with sparse vectors. For instance, computing kernel values for two Reuters documents only involves words common to both documents (excluding stop words). The Forest experiments use a kernel implemented with hand optimized assembly code (Graf et al., 2005).

Figure 8 compares the solutions returned by LASVM$\times$1 and LIBSVM. The LASVM$\times$1 experiments call the kernel function much less often, but do not always run faster. The fast kernel computation times expose the relative weakness of our kernel cache implementation. The LASVM$\times$1 accuracies are very close to the LIBSVM accuracies. The number of support vectors is always slightly smaller.

LASVM$\times$1 essentially achieves consistent results over very diverse data sets, after performing one single epoch over the training set only. In this situation, the LASVM PROCESS function gets only once chance to take a particular example into the kernel expansion and potentially make it a support vector. The conservative strategy would be to take all examples and sort them out during the finishing step. The resulting training times would always be worse than LIBSVM's because the finishing step is itself a simplified SMO solver. Therefore LASVM online iterations are able to very quickly discard a large number of examples with a high confidence. This process is not perfect because we can see that the LASVM$\times$1 number of support vectors are smaller than LIBSVM's. Some good support vectors are discarded erroneously.

Figure 9 reports the relative variations of the test error, number of support vectors, and training time measured before and after the finishing step. The online iterations pretty much select the right support vectors on clean data sets such as "Waveform", "Reuters" or "USPS", and the finishing step does very little. On the other problems the online iterations keep much more examples as potential support vectors. The finishing step significantly improves the accuracy on noisy data sets such as "Banana", "Adult" or "USPS+N", and drastically increases the computation time on data sets with complicated decision boundaries such as "Banana" or "Forest".

	Train Size	Test Size	γ	C	Cache	τ	Notes
Waveform[1]	4000	1000	0.05	1	40M	0.001	Artificial data, 21 dims.
Banana[1]	4000	1300	0.5	316	40M	0.001	Artificial data, 2 dims.
Reuters[2]	7700	3299	1	1	40M	0.001	Topic "moneyfx" vs. rest.
USPS[3]	7329	2000	2	1000	40M	0.001	Class "0" vs. rest.
USPS+N[3]	7329	2000	2	10	40M	0.001	10% training label noise.
Adult[3]	32562	16282	0.005	100	40M	0.001	As in (Platt, 1999).
Forest[3] (100k)	100000	50000	1	3	512M	0.001	As in (Collobert et al., 2002).
Forest[3] (521k)	521012	50000	1	3	1250M	0.01	As in (Collobert et al., 2002).

[1] `http://mlg.anu.edu.au/~raetsch/data/index.html`
[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578`
[3] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases`

Figure 7: Data Sets discussed in Section 3.4.

	LIBSVM				LASVM×1			
Data Set	**Error**	**SV**	**KCalc**	**Time**	**Error**	**SV**	**KCalc**	**Time**
Waveform	8.82%	1006	4.2M	3.2s	**8.68%**	**948**	**2.2M**	**2.7s**
Banana	9.96%	873	6.8M	9.9s	9.98%	869	6.7M	10.0s
Reuters	2.76%	1493	11.8M	**24s**	2.76%	1504	**9.2M**	31.4s
USPS	0.41%	236	1.97M	**13.5s**	0.43%	**201**	**1.08M**	15.9s
USPS+N	**0.41%**	2750	63M	305s	0.53%	**2572**	**20M**	**178s**
Adult	14.90%	11327	1760M	1079s	14.94%	11268	**626M**	**809s**
Forest (100k)	**8.03%**	43251	27569M	14598s	8.15%	**41750**	**18939M**	**10310s**
Forest (521k)	4.84%	124782	316750M	159443s	4.83%	122064	**188744M**	**137183s**

Figure 8: Comparison of LIBSVM versus LASVM×1: Test error rates (Error), number of support vectors (SV), number of kernel calls (KCalc), and training time (Time). Bold characters indicate significative differences.

	Relative Variation		
Data Set	**Error**	**SV**	**Time**
Waveform	-0%	-0%	+4%
Banana	-79%	-74%	+185%
Reuters	0%	-0%	+3%
USPS	0%	-2%	+0%
USPS+N%	-67%	-33%	+7%
Adult	-13%	-19%	+80%
Forest (100k)	-1%	-24%	+248%
Forest (521k)	-2%	-24%	+84%

Figure 9: Relative variations of test error, number of support vectors and training time measured before and after the finishing step.

3.5 The Collection of Potential Support Vectors

The final step of the REPROCESS operation computes the current value of the kernel expansion bias b and the stopping criterion δ:

$$\begin{aligned} g_{\max} &= \max_{s \in \mathcal{S}} g_s \;\; \text{with } \alpha_s < B_s & \qquad b &= \frac{g_{\max} + g_{\min}}{2} \\ g_{\min} &= \min_{s \in \mathcal{S}} g_s \;\; \text{with } \alpha_s > A_s & \qquad \delta &= g_{\max} - g_{\min}. \end{aligned} \tag{10}$$

The quantities $g_{\min}$ and $g_{\max}$ can be interpreted as bounds for the decision threshold b. The quantity δ then represents an uncertainty on the decision threshold b.

The quantity δ also controls how LASVM collects potential support vectors. The definition of PROCESS and the equality (9) indicate indeed that PROCESS(k) adds the support vector x_k to the kernel expansion if and only if

$$y_k \hat{y}(x_k) < 1 + \frac{\delta}{2} - \tau. \tag{11}$$

When α is optimal, the uncertainty δ is zero, and this condition matches the Karush-Kuhn-Tucker condition for support vectors $y_k \hat{y}(x_k) \leq 1$.

Intuitively, relation (11) describes how PROCESS collects potential support vectors that are compatible with the current uncertainty level δ on the threshold b. Simultaneously, the REPROCESS operations reduce δ and discard the support vectors that are no longer compatible with this reduced uncertainty.

The online iterations of the LASVM algorithm make equal numbers of PROCESS and REPROCESS for purely heuristic reasons. Nothing guarantees that this is the optimal proportion. The results reported in Figure 9 clearly suggest to investigate this arbitrage more closely.

Variations on REPROCESS Experiments were carried out with a slightly modified LASVM algorithm: instead of performing a single REPROCESS, the modified online iterations repeatedly run REPROCESS until the uncertainty δ becomes smaller than a predefined threshold $\delta_{\max}$.

Figure 10 reports comparative results for the "Banana" data set. Similar results were obtained with other data sets. The three plots report test error rates, training time, and number of support vectors as a function of $\delta_{\max}$. These measurements were performed after one epoch of online iterations without finishing step, and after one and two epochs followed by the finishing step. The corresponding LIBSVM figures are indicated by large triangles on the right side of the plot.

Regardless of $\delta_{\max}$, the SVM test error rate can be replicated by performing two epochs followed by a finishing step. However, this does not guarantee that the optimal SVM solution has been reached.

Large values of $\delta_{\max}$ essentially correspond to the unmodified LASVM algorithm. Small values of $\delta_{\max}$ considerably increases the computation time because each online iteration calls REPROCESS many times in order to sufficiently reduce δ. Small values of $\delta_{\max}$ also remove the LASVM ability to produce a competitive result after a single epoch followed by a finishing step. The additional optimization effort discards support vectors more aggressively. Additional epochs are necessary to recapture the support vectors that should have been kept.

There clearly is a sweet spot around $\delta_{\max} = 3$ when one epoch of online iterations alone almost match the SVM performance and also makes the finishing step very fast. This sweet spot is difficult to find in general. If $\delta_{\max}$ is a little bit too small, we must make one extra epoch. If $\delta_{\max}$ is a little

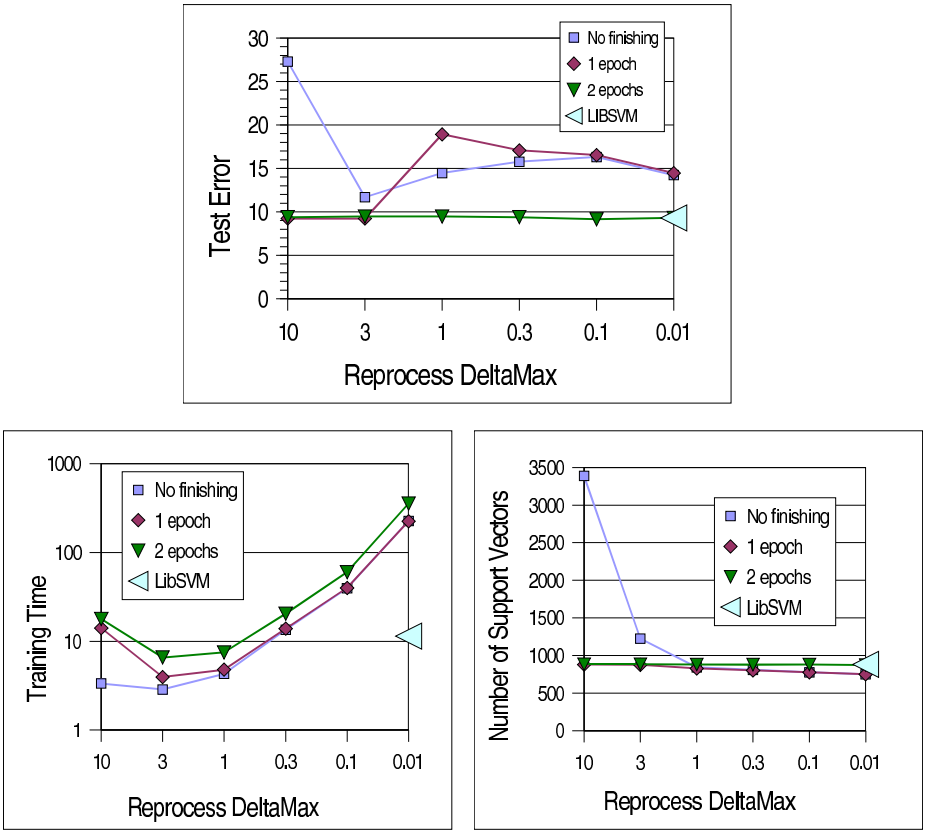

Figure 10: Impact of additional REPROCESS measured on "Banana" data set. During the LASVM online iterations, calls to REPROCESS are repeated until $\delta < \delta_{max}$.

bit too large, the algorithm behaves like the unmodified LASVM. Short of a deeper understanding of these effects, the unmodified LASVM seems to be a robust compromise.

SimpleSVM The right side of each plot in Figure 10 corresponds to an algorithm that optimizes the coefficient of the current support vectors at each iteration. This is closely related to the SimpleSVM algorithm (Vishwanathan et al., 2003). Both LASVM and the SimpleSVM update a current kernel expansion by adding or removing one or two support vectors at each iteration. The two key differences are the numerical objective of these updates and their computational costs.

Whereas each SimpleSVM iteration seeks the optimal solution of the SVM QP problem restricted to the current set of support vectors, the LASVM online iterations merely attempt to improve the value of the dual objective function $W(\alpha)$. As a a consequence, LASVM needs a finishing step and the SimpleSVM does not. On the other hand, Figure 10 suggests that seeking the optimum at each iteration discards support vectors too aggressively to reach competitive accuracies after a single epoch.

Each SimpleSVM iteration updates the current kernel expansion using rank 1 matrix updates (Cauwenberghs and Poggio, 2001) whose computational cost grows as the square of the number of support vectors. LASVM performs these updates using SMO direction searches whose cost grows

linearly with the number of examples. Rank 1 updates make good sense when one seeks the optimal coefficients. On the other hand, all the kernel values involving support vectors must be stored in memory. The LASVM direction searches are more amenable to caching strategies for kernel values.

4. Active Selection of Training Examples

The previous section presents LASVM as an Online Learning algorithm or as a Stochastic Optimization algorithm. In both cases, the LASVM online iterations pick random training examples. The current section departs from this framework and investigates more refined ways to select an informative example for each iteration.

This departure is justified in the offline setup because the complete training set is available beforehand and can be searched for informative examples. It is also justified in the online setup when the continuous stream of fresh training examples is too costly to process, either because the computational requirements are too high, or because it is inpractical to label all the potential training examples.

In particular, we show that selecting informative examples yields considerable speedups. Furthermore, training example selection can be achieved without the knowledge of the training example labels. In fact, excessive reliance on the training example labels can have very detrimental effects.

4.1 Gradient Selection

The most obvious approach consists in selecting an example k such that the PROCESS operation results in a large increase of the dual objective function. This can be approximated by choosing the example which yields the τ-violating pair with the largest gradient. Depending on the class y_k, the PROCESS(k) operation considers pair (k, j) or (i, k) where i and j are the indices of the examples in $\mathcal{S}$ with extreme gradients:

$$i = \arg\max_{s \in \mathcal{S}} g_s \;\text{ with } \alpha_s < B_s \;, \qquad j = \arg\min_{s \in \mathcal{S}} g_s \;\text{ with } \alpha_s > A_s.$$

The corresponding gradients are $g_k - g_j$ for positive examples and $g_i - g_k$ for negative examples. Using the expression (9) of the gradients and the value of b and δ computed during the previous REPROCESS (10), we can write:

$$\text{when } y_k = +1, \quad g_k - g_j = y_k g_k - \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} = 1 + \frac{\delta}{2} - y_k \hat{y}(x_k)$$
$$\text{when } y_k = -1, \quad g_i - g_k = \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} + y_k g_k = 1 + \frac{\delta}{2} - y_k \hat{y}(x_k).$$

This expression shows that the *Gradient Selection Criterion* simply suggests to pick the most misclassified example

$$k_G \;=\; \arg\min_{k \notin \mathcal{S}} \; y_k \hat{y}(x_k). \tag{12}$$

4.2 Active Selection

Always picking the most misclassified example is reasonable when one is very confident of the training example labels. On noisy data sets, this strategy is simply going to pick mislabelled examples or examples that sit on the wrong side of the optimal decision boundary.

When training example labels are unreliable, a conservative approach chooses the example k_A that yields the strongest minimax gradient:

$$k_A = \arg\min_{k \notin \mathcal{S}} \max_{y=\pm 1} y\hat{y}(x_k) = \arg\min_{k \notin \mathcal{S}} |\hat{y}(x_k)|. \quad (13)$$

This *Active Selection Criterion* simply chooses the example that comes closest to the current decision boundary. Such a choice yields a gradient approximatively equal to $1+\delta/2$ regardless of the true class of the example.

Criterion (13) does not depend on the labels y_k. The resulting learning algorithm only uses the labels of examples that have been selected during the previous online iterations. This is related to the *Pool Based Active Learning* paradigm (Cohn et al., 1990).

Early active learning literature, also known as *Experiment Design* (Fedorov, 1972), contrasts the passive learner, who observes examples (x,y), with the active learner, who constructs queries x and observes their labels y. In this setup, the active learner cannot beat the passive learner because he lacks information about the input pattern distribution (Eisenberg and Rivest, 1990). Pool-based active learning algorithms observe the pattern distribution from a vast pool of unlabelled examples. Instead of constructing queries, they incrementally select unlabelled examples x_k and obtain their labels y_k from an oracle.

Several authors (Campbell et al., 2000; Schohn and Cohn, 2000; Tong and Koller, 2000) propose incremental active learning algorithms that clearly are related to Active Selection. The initialization consists of obtaining the labels for a small random subset of examples. A SVM is trained using all the labelled examples as a training set. Then one searches the pool for the unlabelled example that comes closest to the SVM decision boundary, one obtains the label of this example, retrains the SVM and reiterates the process.

4.3 Randomized Search

Both criteria (12) and (13) suggest a search through all the training examples. This is impossible in the online setup and potentially expensive in the offline setup.

It is however possible to locate an approximate optimum by simply examining a small constant number of randomly chosen examples. The randomized search first samples M random training examples and selects the best one among these M examples. With probability $1-\eta^M$, the value of the criterion for this example exceeds the η-quantile of the criterion for all training examples (Schölkopf and Smola, 2002, theorem 6.33) regardless of the size of the training set. In practice this means that the best among 59 random training examples has 95% chances to belong to the best 5% examples in the training set.

Randomized search has been used in the offline setup to accelerate various machine learning algorithms (Domingo and Watanabe, 2000; Vishwanathan et al., 2003; Tsang et al., 2005). In the online setup, randomized search is the only practical way to select training examples. For instance, here is a modification of the basic LASVM algorithm to select examples using the Active Selection Criterion with Randomized Search:

LASVM + Active Example Selection + Randomized Search

1) **Initialization**:
   Seed $\mathcal{S}$ with a few examples of each class.
   Set $\alpha \leftarrow 0$ and $g \leftarrow 0$.

2) **Online Iterations**:
   Repeat a predefined number of times:
   - Pick M random examples $s_1 \ldots s_M$.
   - $k_t \leftarrow \underset{i=1\ldots M}{\arg\min} \; |\hat{y}(x_{s_i})|$
   - Run PROCESS(k_t).
   - Run REPROCESS once.

3) **Finishing**:
   Repeat REPROCESS until $\delta \leq \tau$.

Each online iteration of the above algorithm is about M times more computationally expensive that an online iteration of the basic LASVM algorithm. Indeed one must compute the kernel expansion (2) for M fresh examples instead of a single one (9). This cost can be reduced by heuristic techniques for adapting M to the current conditions. For instance, we present experimental results where one stops collecting new examples as soon as $\mathcal{M}$ contains five examples such that $|\hat{y}(x_s)| < 1+\delta/2$.

Finally the last two paragraphs of appendix A discuss the convergence of LASVM with example selection according to the gradient selection criterion or the active selection criterion. The gradient selection criterion always leads to a solution of the SVM problem. On the other hand, the active selection criterion only does so when one uses the sampling method. In practice this convergence occurs very slowly. The next section presents many reasons to prefer the intermediate kernel classifiers visited by this algorithm.

4.4 Example Selection for Online SVMs

This section experimentally compares the LASVM algorithm using different example selection methods. Four different algorithms are compared:

- RANDOM example selection randomly picks the next training example among those that have not yet been PROCESSed. This is equivalent to the plain LASVM algorithm discussed in Section 3.2.

- GRADIENT example selection consists in sampling 50 random training examples among those that have not yet been PROCESSed. The sampled example with the smallest $y_k \hat{y}(x_k)$ is then selected.

- ACTIVE example selection consists in sampling 50 random training examples among those that have not yet been PROCESSed. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.

- AUTOACTIVE example selection attempts to adaptively select the sampling size. Sampling stops as soon as 5 examples are within distance $1+\delta/2$ of the decision boundary. The maximum sample size is 100 examples. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.

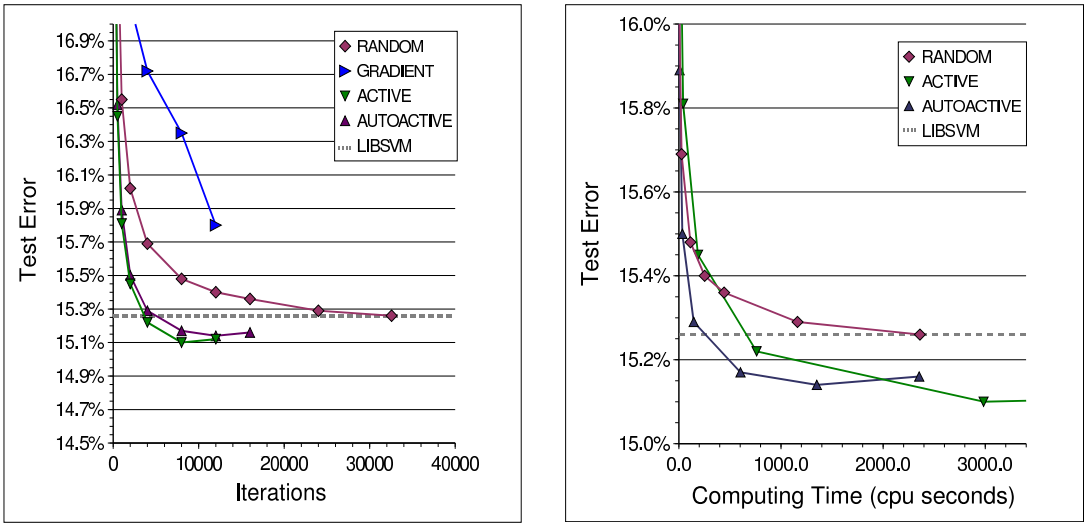

Figure 11: Comparing example selection criteria on the Adult data set. Measurements were performed on 65 runs using randomly selected training sets. The graphs show the error measured on the remaining testing examples as a function of the number of iterations and the computing time. The dashed line represents the LIBSVM test error under the same conditions.

Adult Data Set We first report experiments performed on the "Adult" data set. This data set provides a good indication of the relative performance of the Gradient and Active selection criteria under noisy conditions.

Reliable results were obtained by averaging experimental results measured for 65 random splits of the full data set into training and test sets. Paired tests indicate that test error differences of 0.25% on a single run are statistically significant at the 95% level. We conservatively estimate that average error differences of 0.05% are meaningful.

Figure 11 reports the average error rate measured on the test set as a function of the number of online iterations (left plot) and of the average computing time (right plot). Regardless of the training example selection method, all reported results were measured after performing the LASVM finishing step. More specifically, we run a predefined number of online iterations, save the LASVM state, perform the finishing step, measure error rates and number of support vectors, and restore the saved LASVM state before proceeding with more online iterations. Computing time includes the duration of the online iterations and the duration of the finishing step.

The GRADIENT example selection criterion performs very poorly on this noisy data set. A detailed analysis shows that most of the selected examples become support vectors with coefficient reaching the upper bound C. The ACTIVE and AUTOACTIVE criteria both reach smaller test error rates than those achieved by the SVM solution computed by LIBSVM. The error rates then seem to increase towards the error rate of the SVM solution (left plot). We believe indeed that continued iterations of the algorithm eventually yield the SVM solution.

Figure 12 relates error rates and numbers of support vectors. The RANDOM LASVM algorithm performs as expected: a single pass over all training examples replicates the accuracy and the num-

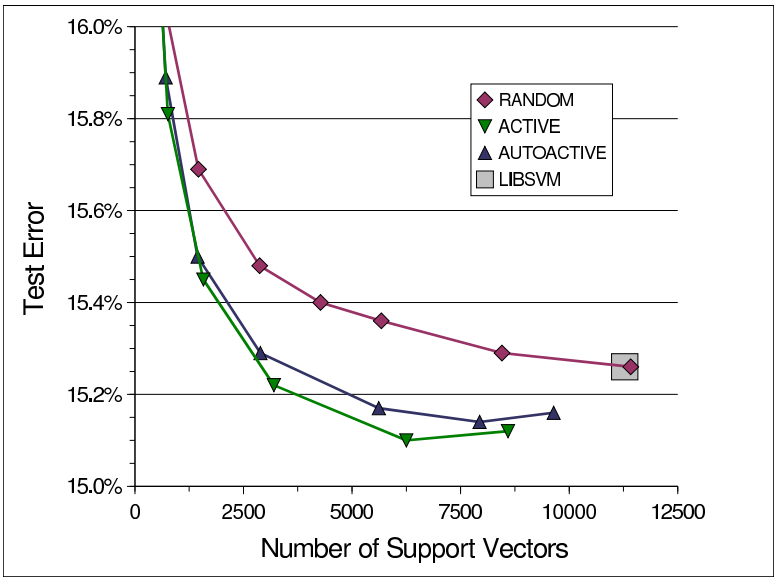

Figure 12: Comparing example selection criteria on the Adult data set. Test error as a function of the number of support vectors.

ber of support vectors of the LIBSVM solution. Both the ACTIVE and AUTOACTIVE criteria yield kernel classifiers with the same accuracy and much less support vectors. For instance, the AUTOACTIVE LASVM algorithm reaches the accuracy of the LIBSVM solution using 2500 support vectors instead of 11278. Figure 11 (right plot) shows that this result is achieved after 150 seconds only. This is about one fifteenth of the time needed to perform a full RANDOM LASVM epoch.[2]

Both the ACTIVE LASVM and AUTOACTIVE LASVM algorithms exceed the LIBSVM accuracy after a few iterations only. This is surprising because these algorithms only use the training labels of the few selected examples. They both outperform the LIBSVM solution by using only a small subset of the available training labels.

MNIST Data Set The comparatively clean MNIST data set provides a good opportunity to verify the behavior of the various example selection criteria on a problem with a much lower error rate.

Figure 13 compares the performance of the RANDOM, GRADIENT and ACTIVE criteria on the classification of digit "8" versus all other digits. The curves are averaged on 5 runs using different random seeds. All runs use the standard MNIST training and test sets. Both the GRADIENT and ACTIVE criteria perform similarly on this relatively clean data set. They require about as much computing time as RANDOM example selection to achieve a similar test error.

Adding ten percent label noise on the MNIST training data provides additional insight regarding the relation between noisy data and example selection criteria. Label noise was not applied to the testing set because the resulting measurement can be readily compared to test errors achieved by training SVMs without label noise. The expected test errors under similar label noise conditions can be derived from the test errors measured without label noise. Figure 14 shows the test errors achieved when 10% label noise is added to the training examples. The GRADIENT selection cri-

2. The timing results reported in Figure 8 were measured on a faster computer.

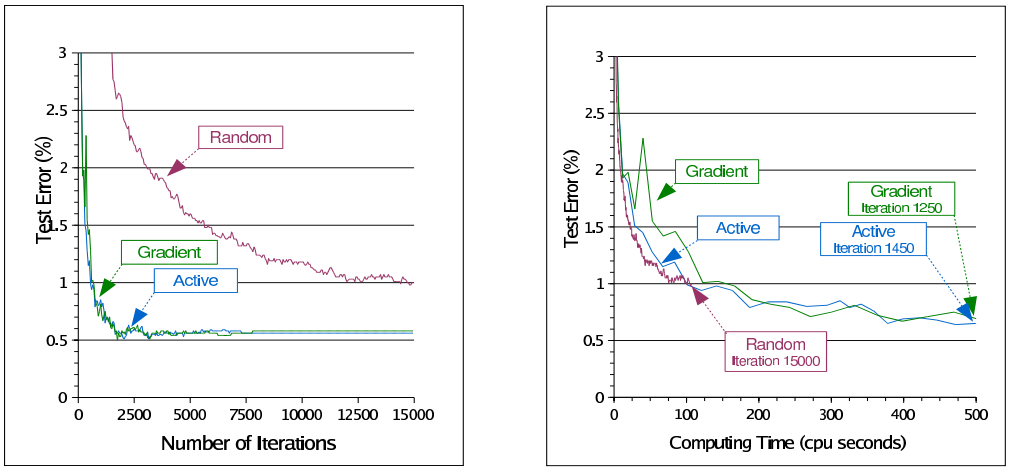

Figure 13: Comparing example selection criteria on the MNIST data set, recognizing digit "8" against all other classes. Gradient selection and Active selection perform similarly on this relatively noiseless task.

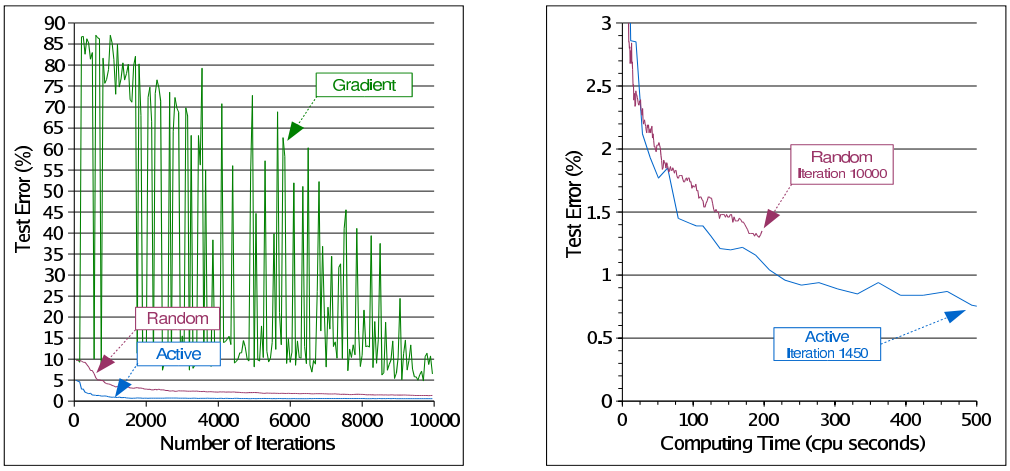

Figure 14: Comparing example selection criteria on the MNIST data set with 10% label noise on the training examples.

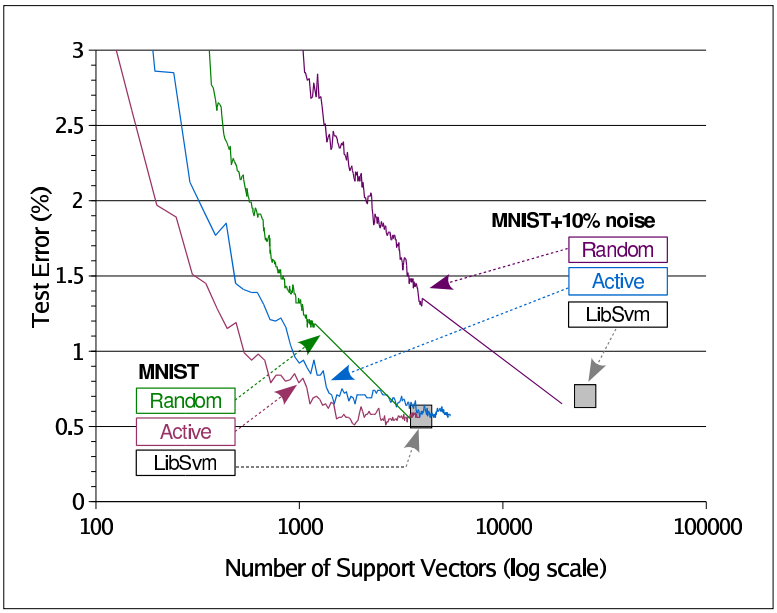

Figure 15: Comparing example selection criteria on the MNIST data set. Active example selection is insensitive to the artificial label noise.

terion causes a very chaotic convergence because it keeps selecting mislabelled training examples. The ACTIVE selection criterion is obviously undisturbed by the label noise.

Figure 15 summarizes error rates and number of support vectors for all noise conditions. In the presence of label noise on the training data, LIBSVM yields a slightly higher test error rate, and a much larger number of support vectors. The RANDOM LASVM algorithm replicates the LIBSVM results after one epoch. Regardless of the noise conditions, the ACTIVE LASVM algorithm reaches the accuracy and the number of support vectors of the LIBSVM solution obtained with clean training data. Although we have not been able to observe it on this data set, we expect that, after a very long time, the ACTIVE curve for the noisy training set converges to the accuracy and the number of support vectors achieved of the LIBSVM solution obtained for the noisy training data.

4.5 Online SVMs for Active Learning

The ACTIVE LASVM algorithm implements two dramatic speedups with respect to existing active learning algorithms such as (Campbell et al., 2000; Schohn and Cohn, 2000; Tong and Koller, 2000). First it chooses a query by sampling a small number of random examples instead of scanning all unlabelled examples. Second, it uses a single LASVM iteration after each query instead of fully retraining the SVM.

Figure 16 reports experiments performed on the Reuters and USPS data sets presented in table 7. The RETRAIN ACTIVE 50 and RETRAIN ACTIVE ALL select a query from 50 or all unlabeled examples respectively, and then retrain the SVM. The SVM solver was initialized with the solution from the previous iteration. The LASVM ACTIVE 50 and LASVM ACTIVE ALL do not retrain the SVM, but instead make a single LASVM iteration for each new labeled example.

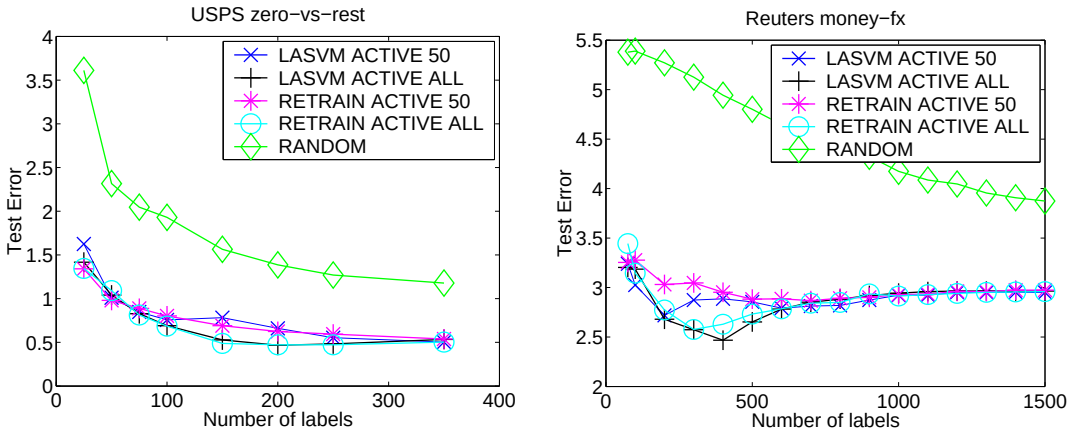

Figure 16: Comparing active learning methods on the USPS and Reuters data sets. Results are averaged on 10 random choices of training and test sets. Using LASVM iterations instead of retraining causes no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small.

All the active learning methods performed approximately the same, and were superior to random selection. Using LASVM iterations instead of retraining causes no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small. Yet the speedups are very significant: for 500 queried labels on the Reuters data set, the RETRAIN ACTIVE ALL, LASVM ACTIVE ALL, and LASVM ACTIVE 50 algorithms took 917 seconds, 99 seconds, and 9.6 seconds respectively.

5. Discussion

This work started because we observed that the data set sizes are quickly outgrowing the computing power of our calculators. One possible avenue consists of harnessing the computing power of multiple computers (Graf et al., 2005). Instead we propose learning algorithms that remain closely related to SVMs but require less computational resources. This section discusses their practical and theoretical implications.

5.1 Practical Significance

When we have access to an abundant source of training examples, the simple way to reduce the complexity of a learning algorithm consists of picking a random subset of training examples and running a regular training algorithm on this subset. Unfortunately this approach renounces the more accurate models that the large training set could afford. This is why we say, by reference to statistical efficiency, that an *efficient* learning algorithm should at least pay a brief look at every training example.

The online LASVM algorithm is very attractive because it matches the performance of a SVM trained on all the examples. More importantly, it achives this performance after a single epoch,

faster than a SVM (figure 2) and using much less memory than a SVM (figure 6). This is very important in practice because modern data storage devices are most effective when the data is accessed sequentially.

Active Selection of the LASVM training examples brings two additional benefits for practical applications. It achieves equivalent performances with significantly less support vectors, further reducing the required time and memory. It also offers an obvious opportunity to parallelize the search for informative examples.

5.2 Informative Examples and Support Vectors

By suggesting that all examples should not be given equal attention, we first state that all training examples are not equally informative. This question has been asked and answered in various contexts (Fedorov, 1972; Cohn et al., 1990; MacKay, 1992). We also ask whether these differences can be exploited to reduce the computational requirements of learning algorithms. Our work answers this question by proposing algorithms that exploit these differences and achieve very competitive performances.

Kernel classifiers in general distinguish the few training examples named support vectors. Kernel classifier algorithms usually maintain an active set of potential support vectors and work by iterations. Their computing requirements are readily associated with the training examples that belong to the active set. Adding a training example to the active set increases the computing time associated with each subsequent iteration because they will require additional kernel computations involving this new support vector. Removing a training example from the active set reduces the cost of each subsequent iteration. However it is unclear how such changes affect the number of subsequent iterations needed to reach a satisfactory performance level.

Online kernel algorithms, such as the kernel perceptrons usually produce different classifiers when given different sequences of training examples. Section 3 proposes an online kernel algorithm that converges to the SVM solution after many epochs. The final set of support vectors is intrinsically defined by the SVM QP problem, regardless of the path followed by the online learning process. Intrinsic support vectors provide a benchmark to evaluate the impact of changes in the active set of current support vectors. Augmenting the active set with an example that is not an intrinsic support vector moderately increases the cost of each iteration without clear benefits. Discarding an example that is an intrinsic support vector incurs a much higher cost. Additional iterations will be necessary to recapture the missing support vector. Empirical evidence is presented in Section 3.5.

Nothing guarantees however that the most informative examples are the support vectors of the SVM solution. Bakır et al. (2005) interpret Steinwart's theorem (Steinwart, 2004) as an indication that the number of SVM support vectors is asymptotically driven by the examples located on the wrong side of the optimal decision boundary. Although such outliers might play a useful role in the construction of a decision boundary, it seems unwise to give them the bulk of the available computing time. Section 4 adds explicit example selection criteria to LASVM. The Gradient Selection Criterion selects the example most likely to cause a large increase of the SVM objective function. Experiments show that it prefers outliers over honest examples. The Active Selection Criterion bypasses the problem by choosing examples without regard to their labels. Experiments show that it leads to competitive test error rates after a shorter time, with less support vectors, and using only the labels of a small fraction of the examples.

5.3 Theoretical Questions

The appendix provides a comprehensive analysis of the convergence of the algorithms discussed in this contribution. Such convergence results are useful but limited in scope. This section underlines some aspects of this work that would vastly benefit from a deeper theoretical understanding.

- Empirical evidence suggests that a single epoch of the LASVM algorithm yields misclassification rates comparable with a SVM. We also know that LASVM exactly reaches the SVM solution after a sufficient number of epochs. Can we theoretically estimate the expected difference between the first epoch test error and the many epoch test error? Such results exist for well designed online learning algorithms based on stochastic gradient descent (Murata and Amari, 1999; Bottou and LeCun, 2005). Unfortunately these results do not directly apply to kernel classifiers. A better understanding would certainly suggest improved algorithms.

- Test error rates are sometimes improved by active example selection. In fact this effect has already been observed in the active learning setups (Schohn and Cohn, 2000). This small improvement is difficult to exploit in practice because it requires very sensitive early stopping criteria. Yet it demands an explanation because it seems that one gets a better performance by using less information. There are three potential explanations: (*i*) active selection works well on unbalanced data sets because it tends to pick equal number of examples of each class (Schohn and Cohn, 2000), (*ii*) active selection improves the SVM loss function because it discards distant outliers, (*iii*) active selection leads to more sparse kernel expansions with better generalization abilities (Cesa-Bianchi et al., 2005). These three explanations may be related.

- We know that the number of SVM support vectors scales linearly with the number of examples (Steinwart, 2004). Empirical evidence suggests that active example selection yields transitory kernel classifiers that achieve low error rates with much less support vectors. What is the scaling law for this new number of support vectors?

- What is the minimal computational cost for learning n independent examples and achieving "optimal" test error rates? The answer depends of course of how we define these "optimal" test error rates. This cost intuitively scales at least linearly with n because one must pay a look at each example to fully exploit them. The present work suggest that this cost might be smaller than n times the reduced number of support vectors achievable with the active learning technique. This range is consistent with previous work showing that stochastic gradient algorithms can train a fixed capacity model in linear time (Bottou and LeCun, 2005). Learning seems to be much easier than computing the optimum of the empirical loss.

5.4 Future Directions

Progress can also be achieved along less arduous directions.

- Section 3.5 suggests that better convergence speed could be attained by cleverly modulating the number of calls to REPROCESS during the online iterations. Simple heuristics might go a long way.

- Section 4.3 suggests a heuristic to adapt the sampling size for the randomized search of informative training examples. This AUTOACTIVE heuristic performs very well and deserves further investigation.
- Sometimes one can generate a very large number of training examples by exploiting known invariances. Active example selection can drive the generation of examples. This idea was suggested in (Loosli et al., 2004) for the SimpleSVM.

6. Conclusion

This work explores various ways to speedup kernel classifiers by asking which examples deserve more computing time. We have proposed a novel online algorithm that converges to the SVM solution. LASVM reliably reaches competitive accuracies after performing a single pass over the training examples, outspeeding state-of-the-art SVM solvers. We have then shown how active example selection can yield faster training, higher accuracies and simpler models using only a fraction of the training examples labels.

Acknowledgments

Part of this work was funded by NSF grant CCR-0325463. We also thank Eric Cosatto, Hans-Peter Graf, C. Lee Giles and Vladimir Vapnik for their advice and support, Ronan Collobert and Chih-Jen Lin for thoroughly checking the mathematical appendix, and Sathiya Keerthi for pointing out reference (Takahashi and Nishi, 2003).

Appendix A. Convex Programming with Witness Families

This appendix presents theoretical elements about convex programming algorithms that rely on successive direction searches. Results are presented for the case where directions are selected from a well chosen finite pool, like SMO (Platt, 1999), and for the stochastic algorithms, like the online and active SVM discussed in the body of this contribution.

Consider a compact convex subset $\mathcal{F}$ of $\mathbb{R}^n$ and a concave function f defined on $\mathcal{F}$. We assume that f is twice differentiable with continuous derivatives. This appendix discusses the maximization of function f over set F:

$$\max_{x \in \mathcal{F}} f(x). \tag{14}$$

This discussion starts with some results about feasible directions. Then it introduces the notion of witness family of directions which leads to a more compact characterization of the optimum. Finally it presents maximization algorithms and establishes their convergence to approximate solutions

A.1 Feasible Directions

Notations Given a point $x \in \mathcal{F}$ and a direction $u \in \mathbb{R}^n_* = \mathbb{R}^n$, let

$$\begin{aligned} \phi(x,u) &= \max\{\lambda \geq 0 \,|\, x+\lambda u \in \mathcal{F}\} \\ f^*(x,u) &= \max\{f(x+\lambda u), x+\lambda u \in \mathcal{F}\}. \end{aligned}$$

In particular we write $\phi(x,0) = \infty$ and $f^*(x,0) = f(x)$.

Definition 1 *The cone of feasible directions in $x \in \mathcal{F}$ is the set*

$$\mathcal{D}_X = \{u \in \mathbb{R}^n \,|\, \phi(x,u) > 0\}.$$

All the points $x + \lambda u$, $0 \le \lambda \le \phi(x,u)$ belong to $\mathcal{F}$ because $\mathcal{F}$ is convex. Intuitively, a direction $u \neq 0$ is feasible in x when we can start from x and make a little movement along direction u without leaving the convex set $\mathcal{F}$.

Proposition 2 *Given $x \in \mathcal{F}$ and $u \in \mathbb{R}^n$,*

$$f^*(x,u) > f(x) \quad \Longleftrightarrow \quad \begin{cases} u'\nabla f(x) > 0 \\ u \in \mathcal{D}_X. \end{cases}$$

Proof Assume $f^*(x,u) > f(x)$. Direction $u \neq 0$ is feasible because the maximum $f^*(x,u)$ is reached for some $0 < \lambda^* \le \phi(x,u)$. Let $\nu \in [0,1]$. Since set $\mathcal{F}$ is convex, $x + \nu\lambda^* u \in \mathcal{F}$. Since function f is concave, $f(x + \nu\lambda^* u)) \ge (1-\nu) f(x) + \nu f^*(x,u)$. Writing a first order expansion when $\nu \to 0$ yields $\lambda^* u'\nabla f(x) \ge f^*(x,u) - f(x) > 0$. Conversely, assume $u'\nabla f(x) > 0$ and $u \neq 0$ is a feasible direction. Recall $f(x+\lambda u) = f(x) + \lambda u'\nabla f(x) + o(\lambda)$. Therefore we can choose $0 < \lambda_0 \le \phi(x,u)$ such that $f(x+\lambda_0 u) > f(x) + \lambda_0 u'\nabla f(x)/2$. Therefore $f^*(x,u) \ge f(x+\lambda_0 u) > f(x)$. ■

Theorem 3 (Zoutendijk (1960) page 22) *The following assertions are equivalent:*

i) *x is a solution of problem (14).*
ii) $\forall u \in \mathbb{R}^n \quad f^*(x,u) \le f(x)$.
iii) $\forall u \in \mathcal{D}_X \quad u'\nabla f(x) \le 0$.

Proof The equivalence between assertions (ii) and (iii) results from proposition 2. Assume assertion (i) is true. Assertion (ii) is necessarily true because $f^*(u,x) \le \max_{\mathcal{F}} f = f(x)$. Conversely, assume assertion (i) is false. Then there is $y \in \mathcal{F}$ such that $f(y) > f(x)$. Therefore $f^*(x, y-x) > f(x)$ and assertion (ii) is false. ■

A.2 Witness Families

We now seek to improve this theorem. Instead of considering all feasible directions in $\mathbb{R}^n$, we wish to only consider the feasible directions from a smaller set $\mathcal{U}$.

Proposition 4 *Let $x \in \mathcal{F}$ and $v_1 \ldots v_k \in \mathcal{D}_X$ be feasible directions. Every positive linear combination of $v_1 \ldots v_k$ (i.e. a linear combination with positive coefficients) is a feasible direction.*

Proof Let u be a positive linear combination of the v_i. Since the v_i are feasible directions there are $y_i = x + \lambda_i v_i \in \mathcal{F}$, and u can be written as $\sum_i \gamma_i (y_i - x)$ with $\gamma_i \ge 0$. Direction u is feasible because the convex $\mathcal{F}$ contains $(\sum \gamma_i y_i) / (\sum \gamma_i) = x + (1/\sum \gamma_i)\, u$. ■

Definition 5 *A set of directions $\mathcal{U} \subset \mathbb{R}^n_*$ is a "witness family for $\mathcal{F}$" when, for any point $x \in \mathcal{F}$, any feasible direction $u \in \mathcal{D}_\mathcal{X}$ can be expressed as a positive linear combination of a finite number of feasible directions $v_j \in \mathcal{U} \cap \mathcal{D}_x$.*

This definition directly leads to an improved characterization of the optima.

Theorem 6 *Let $\mathcal{U}$ be a witness family for convex set $\mathcal{F}$.*
The following assertions are equivalent:

i) x *is a solution of problem (14).*
ii) $\forall u \in \mathcal{U} \quad f^*(x,u) \leq f(x)$.
iii) $\forall u \in \mathcal{U} \cap \mathcal{D}_\mathcal{X} \quad u' \nabla f(x) \leq 0$.

Proof The equivalence between assertions (ii) and (iii) results from proposition 2. Assume assertion (i) is true. Theorem 3 implies that assertion (ii) is true as well. Conversely, assume assertion (i) is false. Theorem 3 implies that there is a feasible direction $u \in \mathbb{R}^n$ on point x such that $u' \nabla f(x) > 0$. Since $\mathcal{U}$ is a witness family, there are positive coefficients $\gamma_1 \ldots \gamma_k$ and feasible directions $v_1, \ldots, v_k \in \mathcal{U} \cap \mathcal{D}_\mathcal{X}$ such that $u = \sum \gamma_i v_i$. We have then $\sum \gamma_j v_j' \nabla f(x) > 0$. Since all coefficients γ_j are positive, there is at least one term j_0 such that $v_{j_0}' \nabla f(x) > 0$. Assertion (iii) is therefore false. ■

The following proposition provides an example of witness family for the convex domain $\mathcal{F}_\mathrm{s}$ that appears in the SVM QP problem (5).

Proposition 7 *Let $(e_1 \ldots e_n)$ be the canonical basis of $\mathbb{R}^n$. Set $\mathcal{U}_\mathrm{s} = \{e_i - e_j, i \neq j\}$ is a witness family for convex set $\mathcal{F}_\mathrm{s}$ defined by the constraints*

$$x \in \mathcal{F}_\mathrm{s} \quad \Longleftrightarrow \quad \left\{ \begin{array}{l} \forall\, i \quad A_i \leq x_i \leq B_i \\ \sum_i x_i = 0. \end{array} \right.$$

Proof Let $u \in \mathbb{R}^n_*$ be a feasible direction in $x \in \mathcal{F}_\mathrm{s}$. Since u is a feasible direction, there is $\lambda > 0$ such that $y = x + \lambda u \in \mathcal{F}_\mathrm{s}$. Consider the subset $\mathcal{B} \subset \mathcal{F}_\mathrm{s}$ defined by the constraints

$$z \in \mathcal{B} \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \forall\, i,\, A_i \leq \min(x_i, y_i) \leq z_i \leq \max(x_i, y_i) \leq B_i \\ \sum_i z_i = 0. \end{array} \right.$$

Let us recursively define a sequence of points $z(j) \in \mathcal{B}$. We start with $z(0) = x \in \mathcal{B}$. For each $t \geq 0$, we define two sets of coordinate indices $I_t^+ = \{i \,|\, z_i(t) < y_i\}$ and $I_t^- = \{j \,|\, z_j(t) > y_j\}$. The recursion stops if either set is empty. Otherwise, we choose $i \in I_t^+$ and $j \in I_t^-$ and define $z(t+1) = z(t) + \gamma(t)\, v(t) \in \mathcal{B}$ with $v(t) = e_i - e_j \in \mathcal{U}_\mathrm{s}$ and $\gamma(t) = \min(y_i - z_i(t), z_j(t) - y_j) > 0$. Intuitively, we move towards y along direction $v(t)$ until we hit the boundaries of set $\mathcal{B}$.

Each iteration removes at least one of the indices i or j from sets I_t^+ and I_t^-. Eventually one of these sets gets empty and the recursion stops after a finite number k of iterations. The other set is also empty because

$$\sum_{i \in I_k^+} |y_i - z_i(k)| - \sum_{i \in I_k^-} |y_i - z_i(k)| \;=\; \sum_{i=1}^n y_i - z_i(k) \;=\; \sum_{i=1}^n y_i - \sum_{i=1}^n z_i(k) \;=\; 0.$$

Therefore $z(k) = y$ and $\lambda u = y - x = \sum_t \gamma(t)\, v(t)$. Moreover the $v(t)$ are feasible directions on x because $v(t) = e_i - e_j$ with $i \in I_t^+ \subset I_0^+$ and $j \in I_t^- \subset I_0^-$. ■

Assertion (iii) in Theorem 6 then yields the following necessary and sufficient optimality criterion for the SVM QP problem (5):

$$\forall (i,j) \in \{1 \ldots n\}^2 \qquad x_i < B_i \text{ and } x_j > A_j \;\Rightarrow\; \frac{\partial f}{\partial x_i}(x) - \frac{\partial f}{\partial x_j}(x) \leq 0.$$

Different constraint sets call for different choices of witness family. For instance, it is sometimes useful to disregard the equality constraint in the SVM polytope $\mathcal{F}_s$. Along the lines of proposition 7, it is quite easy to prove that $\{\pm e_i,\, i = 1 \ldots n\}$ is a witness family. Theorem 6 then yields an adequate optimality criterion.

A.3 Finite Witness Families

This section deals with *finite witness families*. Theorem 9 shows that $\mathcal{F}$ is necessarily a convex polytope, that is a bounded set defined by a finite number of linear of linear equality and inequality constraints (Schrijver, 1986).

Proposition 8 *Let $\mathcal{C}_x = \{x + u\,,\, u \in \mathcal{D}_x\}$ for $x \in \mathcal{F}$. Then $\mathcal{F} = \bigcap_{x \in \mathcal{F}} \mathcal{C}_x$.*

Proof We first show that $\mathcal{F} \subset \bigcap_{x \in \mathcal{F}} \mathcal{C}_x$. Indeed $\mathcal{F} \subset \mathcal{C}_x$ for all x because every point $z \in \mathcal{F}$ defines a feasible direction $z - x \in \mathcal{D}_x$.

Conversely, Let $z \in \bigcap_{x \in \mathcal{F}} \mathcal{C}_x$ and assume that z does not belong to $\mathcal{F}$. Let $\hat{z}$ be the projection of z on $\mathcal{F}$. We know that $z \in \mathcal{C}_{\hat{z}}$ because $z \in \bigcap_{x \in \mathcal{F}} \mathcal{C}_x$. Therefore $z - \hat{z}$ is a feasible direction in $\hat{z}$. Choose $0 < \lambda < \phi(\hat{z}, z - \hat{z})$. We know that $\lambda < 1$ because z does not belong to $\mathcal{F}$. But then $\hat{z} + \lambda(z - \hat{z}) \in \mathcal{F}$ is closer to z than $\hat{z}$. This contradicts the definition of the projection $\hat{z}$. ■

Theorem 9 *Let $\mathcal{F}$ be a bounded convex set.*
If there is a finite witness family for $\mathcal{F}$, then $\mathcal{F}$ is a convex polytope.[3]

Proof Consider a point $x \in \mathcal{F}$ and let $\{v_1 \ldots v_k\} = \mathcal{U} \cap \mathcal{D}_x$. Proposition 4 and definition 5 imply that $\mathcal{D}_x$ is the polyhedral cone $\{z = \sum \gamma_i v_i,\, \gamma_i \geq 0\}$ and can be represented (Schrijver, 1986) by a finite number of linear equality and inequality constraints of the form $nz \leq 0$ where the directions n are unit vectors. Let $\mathcal{K}_x$ be the set of these unit vectors. Equality constraints arise when the set $\mathcal{K}_x$ contains both n and $-n$. Each set $\mathcal{K}_x$ depends only on the subset $\{v_1 \ldots v_k\} = \mathcal{U} \cap \mathcal{D}_x$ of feasible witness directions in x. Since the finite set $\mathcal{U}$ contains only a finite number of potential subsets, there is only a finite number of distinct sets $\mathcal{K}_x$.

Each set $\mathcal{C}_x$ is therefore represented by the constraints $nz \leq nx$ for $n \in \mathcal{K}_x$. The intersection $\mathcal{F} = \bigcap_{x \in \mathcal{F}} \mathcal{C}_x$ is then defined by all the constraints associated with $\mathcal{C}_x$ for any $x \in \mathcal{F}$. These constraints involve only a finite number of unit vectors n because there is only a finite number of distinct sets $\mathcal{K}_x$.

Inequalities defined by the same unit vector n can be summarized by considering only the most restrictive right hand side. Therefore $\mathcal{F}$ is described by a finite number of equality and inequality

constraints. Since $\mathcal{F}$ is bounded, it is a polytope. ■

A convex polytope comes with useful continuity properties.

Proposition 10 *Let $\mathcal{F}$ be a polytope, and let $u \in \mathbb{R}^n$ be fixed.*
Functions $x \mapsto \phi(x,u)$ and $x \mapsto f^*(x,u)$ are uniformly continous on $\mathcal{F}$.

Proof The polytope $\mathcal{F}$ is defined by a finite set of constraints $n\,x \leq b$. Let $\mathcal{K}_\mathcal{F}$ be the set of pairs (n,b) representing these constraints. Function $x \mapsto \phi(x,u)$ is a continuous on $\mathcal{F}$ because we can write:

$$\phi(x,u) = \min \left\{ \frac{b - n\,x}{n\,u} \quad \text{for all } (n,b) \in \mathcal{K}_\mathcal{F} \text{ such that } n\,u > 0 \right\}.$$

Function $x \mapsto \phi(x,u)$ is uniformly continuous because it is continuous on the compact $\mathcal{F}$.

Choose $\varepsilon > 0$ and let $x, y \in \mathcal{F}$. Let the maximum $f^*(x,u)$ be reached in $x + \lambda^* u$ with $0 \leq \lambda^* \leq \phi(x,u)$. Since f is uniformly continous on compact $\mathcal{F}$, there is $\eta > 0$ such that $|f(x+\lambda^* u) - f(y+\lambda' u)| < \varepsilon$ whenever $\|x - y + (\lambda^* - \lambda')u\| < \eta(1 + \|u\|)$. In particular, it is sufficient to have $\|x-y\| < \eta$ and $|\lambda^* - \lambda'| < \eta$. Since ϕ is uniformly continuous, there is $\tau > 0$ such that $|\phi(y,u) - \phi(x,u)| < \eta$ whenever $\|x-y\| < \tau$. We can then select $0 \leq \lambda' \leq \phi(y,u)$ such that $|\lambda^* - \lambda'| < \eta$. Therefore, when $\|x-y\| < \min(\eta,\tau)$, $f^*(x,u) = f(x+\lambda^* u) \leq f(y+\lambda' u) + \varepsilon \leq f^*(y,u) + \varepsilon$.

By reversing the roles of x and y in the above argument, we can similary establish that $f^*(y,u) \leq f^*(x,u) + \varepsilon$ when $\|x-y\| \leq \min(\eta,\tau)$. Function $x \mapsto f^*(x,u)$ is therefore uniformly continuous on $\mathcal{F}$. ■

A.4 Stochastic Witness Direction Search

Each iteration of the following algorithm randomly chooses a feasible witness direction and performs an optimization along this direction. The successive search directions u_t are randomly selected (step 2a) according to some distribution P_t defined on $\mathcal{U}$. Distribution P_t possibly depends on values observed before time t.

Stochastic Witness Direction Search (WDS)

1) Find an initial feasible point $x_0 \in \mathcal{F}$.
2) For each $t = 1, 2, \ldots,$
   2a) Draw a direction $u_t \in \mathcal{U}$ from a distribution P_t
   2b) If $u \in \mathcal{D}_{x_{t-1}}$ and $u_t' \nabla f(x_{t-1}) > 0$,
   $x_t \leftarrow \text{argmax } f(x)$ under $x \in \{x_{t-1} + \lambda u_t \in \mathcal{F}, \lambda \geq 0\}$
   otherwise
   $x_t \leftarrow x_{t-1}$.

Clearly the Stochastic WDS algorithm does not work if the distributions P_t always give probability zero to important directions. On the other hand, convergence is easily established if all feasible directions can be drawn with non zero minimal probability at any time.

3. We believe that the converse of Theorem 9 is also true.

Theorem 11 *Let f be a concave function defined on a compact convex set $\mathcal{F}$, differentiable with continuous derivatives. Assume $\mathcal{U}$ is a finite witness set for set $\mathcal{F}$, and let the sequence x_t be defined by the Stochastic WDS algorithm above. Further assume there is $\pi > 0$ such that $P_t(u) > \pi$ for all $u \in \mathcal{U} \cap \mathcal{D}_{X_{t-1}}$. All accumulation points of the sequence x_t are then solutions of problem (14) with probability* 1.

Proof We want to evaluate the probability of event Q comprising all sequences of selected directions $(u_1, u_2, \ldots)$ leading to a situation where x_t has an accumulation point x^* that is not a solution of problem (14).

For each sequence of directions $(u_1, u_2, \ldots)$, the sequence $f(x_t)$ is increasing and bounded. It converges to $f^* = \sup_t f(x_t)$. We have $f(x^*) = f^*$ because f is continuous. By Theorem 6, there is a direction $u \in \mathcal{U}$ such that $f^*(x^*, u) > f^*$ and $\phi(x^*, u) > 0$. Let x_{k_t} be a subsequence converging to x^*. Thanks to the continuity of ϕ, f^* and ∇f, there is a t_0 such that $f^*(x_{k_t}, u) > f^*$ and $\phi(x_{k_t}, u) > 0$ for all $k_t > t_0$.

Choose $\varepsilon > 0$ and let $Q_T \subset Q$ contain only sequences of directions such that $t_0 = T$. For any $k_t > T$, we know that $\phi(x_{k_t}, u) > 0$ which means $u \in \mathcal{U} \cap \mathcal{D}_{X_{k_t}}$. We also know that $u_{k_t} \neq u$ because we would otherwise obtain a contradiction $f(x_{k_t+1}) = f^*(x_{k_t}, u) > f^*$. The probability of selecting such a u_{k_t} is therefore smaller than $(1-\pi)$. The probability that this happens simultaneously for N distinct $k_t \geq T$ is smaller than $(1-\pi)^N$ for any N. We get $P(Q_T) \leq \varepsilon / T^2$ by choosing N large enough.

Then we have $P(Q) = \sum_T P(Q_T) \leq \varepsilon \left(\sum_T 1/T^2 \right) = K\varepsilon$. Hence $P(Q) = 0$ because we can choose ε as small as we want, We can therefore assert with probability 1 that all accumulation points of sequence x_t are solutions. ■

This condition on the distributions P_t is unfortunately too restrictive. The PROCESS and REPROCESS iterations of the Online LASVM algorithm (Section 3.2) only exploit directions from very specific subsets.

On the other hand, the Online LASVM algorithm only ensures that any remaining feasible direction at time T will eventually be selected with probability 1. Yet it is challenging to mathematically express that there is no coupling between the subset of time points t corresponding to a subsequence converging to a particular accumulation point, and the subset of time points t corresponding to the iterations where specific feasible directions are selected.

This problem also occurs in the deterministic Generalized SMO algorithm (Section 3.1). An asymptotic convergence proof (Lin, 2001) only exist for the important case of the SVM QP problem using a specific direction selection strategy. Following Keerthi and Gilbert (2002), we bypass this technical difficulty by defining a notion of approximate optimum and proving convergence in finite time. It is then easy to discuss the properties of the limit point.

A.5 Approximate Witness Direction Search

Definition 12 *Given a finite witness family $\mathcal{U}$ and the tolerances $\kappa > 0$ and $\tau > 0$, we say that x is a $\kappa\tau$-approximate solution of problem (14) when the following condition is verified:*

$$\forall u \in \mathcal{U}, \quad \phi(x,u) \leq \kappa \ \text{ or } \ u' \nabla f(x) \leq \tau.$$

A vector $u \in \mathbb{R}_n$ such that $\phi(x,u) > \kappa$ and $u' \nabla f(x) > \tau$ is called a $\kappa\tau$-violating direction in point x.

This definition is inspired by assertion (*iii*) in Theorem 6. The definition demands a *finite* witness family because this leads to proposition 13 establishing that $\kappa\tau$-approximate solutions indicate the location of actual solutions when κ and τ tend to zero.

Proposition 13 *Let $\mathcal{U}$ be a finite witness family for bounded convex set $\mathcal{F}$. Consider a sequence $x_t \in \mathcal{F}$ of $\kappa_t\tau_t$-approximate solutions of problem (14) with $\tau_t \to 0$ and $\kappa_t \to 0$. The accumulation points of this sequence are solutions of problem (14).*

Proof Consider an accumulation point x^* and a subsequence x_{k_t} converging to x^*. Define function

$$(x,\tau,\kappa) \;\mapsto\; \psi(x,\tau,\kappa,u) = \big(u'\nabla f(x) - \tau\big)\,\max\{0,\phi(x,u)-\kappa\}$$

such that u is a $\kappa\tau$-violating direction if and only if $\psi(x,\kappa,\tau,u) > 0$. Function ψ is continuous thanks to Theorem 9, proposition 10 and to the continuity of ∇f. Therefore, we have $\psi(x_{k_t},\kappa_{k_t},\tau_{k_t},u) \le 0$ for all $u \in \mathcal{U}$. Taking the limit when $k_t \to \infty$ gives $\psi(x^*,0,0,u) \le 0$ for all $u \in \mathcal{U}$. Theorem 6 then states that x^* is a solution. ■

The following algorithm introduces the two tolerance parameters $\tau > 0$ and $\kappa > 0$ into the Stochastic Witness Direction Search algorithm.

Approximate Stochastic Witness Direction Search

1) Find an initial feasible point $x_0 \in \mathcal{F}$.
2) For each $t = 1, 2, \ldots,$
   2a) Draw a direction $u_t \in \mathcal{U}$ from a probability distribution P_t
   2b) If u_t is a $\kappa\tau$-violating direction,
   $x_t \leftarrow \operatorname{argmax} f(x)$ under $x \in \{x_{t-1} + \lambda u_t \in \mathcal{F}, \lambda \ge 0\}$
   otherwise
   $x_t \leftarrow x_{t-1}$.

The successive search directions u_t are drawn from some unspecified distributions P_t defined on $\mathcal{U}$. Proposition 16 establishes that this algorithm always converges to some $x^* \in \mathcal{F}$ after a finite number of steps, regardless of the selected directions (u_t). The proof relies on the two intermediate results that generalize a lemma proposed by Keerthi and Gilbert (2002) in the case of quadratic functions.

Proposition 14 *If u_t is a $\kappa\tau$-violating direction in x_{t-1},*

$$\phi(x_t,u_t)\,u_t'\nabla f(x_t) = 0.$$

Proof Let the maximum $f(x_t) = f^*(x_{t-1},u_t)$ be attained in $x_t = x_{t-1} + \lambda^* u_t$ with $0 \le \lambda^* \le \phi(x_{t-1},u_t)$. We know that $\lambda^* \neq 0$ because u_t is $\kappa\tau$-violating and proposition 2 implies $f^*(x_{t-1},u_t) > f(x_{t-1})$. If λ^* reaches its upper bound, $\phi(x_t,u_t) = 0$. Otherwise x_t is an unconstrained maximum and $u_t'\nabla f(x_t) = 0$. ■

Proposition 15 *There is a constant $K > 0$ such that*

$$\forall t\,,\quad f(x_t) - f(x_{t-1}) \;\ge\; K\,\|x_t - x_{t-1}\|.$$

Proof The relation is obvious when u_t is not a $\kappa\tau$-violating direction in x_{t-1}. Otherwise let the maximum $f(x_t) = f^*(x_{t-1}, u_t)$ be attained in $x_t = x_{t-1} + \lambda^* u_t$.
Let $\lambda = \nu\lambda^*$ with $0 < \nu \leq 1$. Since x_t is a maximum,

$$f(x_t) - f(x_{t-1}) = f(x_{t-1} + \lambda^* u_t) - f(x_{t-1}) \geq f(x_{t-1} + \lambda u_t) - f(x_{t-1}).$$

Let H be the maximum over $\mathcal{F}$ of the norm of the Hessian of f.
A Taylor expansion with the Cauchy remainder gives

$$\left| f(x_{t-1} + \lambda u_t) - f(x_{t-1}) - \lambda u_t' \nabla f(x_{t-1}) \right| \leq \frac{1}{2}\lambda^2 \|u_t\|^2 H$$

or, more specifically,

$$f(x_{t-1} + \lambda u_t) - f(x_{t-1}) - \lambda u_t' \nabla f(x_{t-1}) \;\geq\; -\frac{1}{2}\lambda^2 \|u_t\|^2 H.$$

Combining these inequalities yields

$$f(x_t) - f(x_{t-1}) \geq f(x_{t-1} + \lambda u_t) - f(x_{t-1}) \geq \lambda u_t' \nabla f(x_{t-1}) - \frac{1}{2}\lambda^2 \|u_t\|^2 H.$$

Recalling $u_t' \nabla f(x_{t-1}) > \tau$, and $\lambda \|u_t\| = \nu \|x_t - x_{t-1}\|$, we obtain

$$f(x_t) - f(x_{t-1}) \geq \|x_t - x_{t-1}\| \left(\nu \frac{\tau}{U} - \nu^2 \frac{1}{2} DH \right)$$

where $U = \max_{u \in \mathcal{U}} \|u\|$ and D is the diameter of the compact convex $\mathcal{F}$.
Choosing $\nu = \min\left(1, \frac{\tau}{UDH}\right)$ then gives the desired result. ■

Proposition 16 *Assume $\mathcal{U}$ is a finite witness set for set $\mathcal{F}$. The Approximate Stochastic WDS algorithm converges to some $x^* \in \mathcal{F}$ after a finite number of steps.*

Proof Sequence $f(x_t)$ converges because it is increasing and bounded. Therefore it satisfies Cauchy's convergence criterion:

$$\forall \varepsilon > 0, \exists t_0, \forall t_2 > t_1 > t_0, \quad f(x_{t_2}) - f(x_{t_1}) = \sum_{t_1 < t \leq t_2} f(x_t) - f(x_{t-1}) < \varepsilon.$$

Using proposition 15, we can write

$$\forall \varepsilon > 0, \exists t_0, \forall t_2 > t_1 > t_0, \quad \|x_{t_2} - x_{t_1}\| \leq \sum_{t_1 < t \leq t_2} \|x_t - x_{t-1}\| \leq \sum_{t_1 < t \leq t_2} \frac{f(x_t) - f(x_{t-1})}{K} < \frac{\varepsilon}{K}.$$

Therefore sequence x_t satisfies Cauchy's condition and converges to some $x^* \in \mathcal{F}$.

Assume this convergence does not occur in a finite time. Since $\mathcal{U}$ is finite, the algorithm exploits at least one direction $u \in \mathcal{U}$ an infinite number of times. Therefore there is a strictly increasing sequence of positive indices k_t such that $u_{k_t} = u$ is $\kappa\tau$-violating in point $x_{k_t - 1}$. We have then

$\phi(x_{k_t-1},u) > \kappa$ and $u'\nabla f(x_{k_t-1}) > \tau$. By continuity we have $\phi(x^*,u) \geq \kappa$ and $u'\nabla f(x^*) \geq \tau$. On the other hand, proposition 14 states that $\phi(x_{k_t},u)\, u'\nabla f(x_{k_t}) = 0$. By continuity when $t \to 0$, we obtain the contradiction $\phi(x^*,u)\, u'\nabla f(x^*) = 0$. ■

In general, proposition 16 only holds for $\kappa > 0$ and $\tau > 0$. Keerthi and Gilbert (2002) assert a similar property for $\kappa = 0$ and $\tau > 0$ in the case of SVMs only. Despite a mild flaw in the final argument of the initial proof, this assertion is correct (Takahashi and Nishi, 2003).

Proposition 16 does not prove that the limit x^* is related to the solution of the optimization problem (14). Additional assumptions on the direction selection step are required. Theorem 17 addresses the deterministic case by considering trivial distributions P_t that always select a $\kappa\tau$-violating direction if such directions exist. Theorem 18 addresses the stochastic case under mild conditions on the distribution P_t.

Theorem 17 *Let the concave function f defined on the compact convex set $\mathcal{F}$ be twice differentiable with continuous second derivatives. Assume $\mathcal{U}$ is a finite witness set for set $\mathcal{F}$, and let the sequence x_t be defined by the Approximate Stochastic WDS algorithm above. Assume that step (2a) always selects a $\kappa\tau$-violating direction in x_{t-1} if such directions exist. Then x_t converges to a $\kappa\tau$-approximate solution of problem (14) after a finite number of steps.*

Proof Proposition 16 establishes that there is t_0 such that $x_t = x^*$ for all $t \geq t_0$. Assume there is a $\kappa\tau$-violating direction in x^*. For any $t > t_0$, step (2a) always selects such a direction, and step (2b) makes x_t different from $x_{t-1} = x^*$. This contradicts the definition of t_0. Therefore there are no $\kappa\tau$-violating direction in x^* and x^* is a $\kappa\tau$-approximate solution. ■

Example (SMO) The SMO algorithm (Section 3.1) is[4] an Approximate Stochastic WDS that always selects a $\kappa\tau$-violating direction when one exists. Therefore Theorem 17 applies.

Theorem 18 *Let the concave function f defined on the compact convex set $\mathcal{F}$ be twice differentiable with continuous second derivatives. Assume $\mathcal{U}$ is a finite witness set for set $\mathcal{F}$, and let the sequence x_t be defined by the Approximate Stochastic WDS algorithm above. Let p_t be the conditional probability that u_t is $\kappa\tau$-violating in x_{t-1} given that $\mathcal{U}$ contains such directions. Assume that* $\limsup p_t > 0$. *Then x_t converges with probability one to a $\kappa\tau$-approximate solution of problem (14) after a finite number of steps.*

Proof Proposition 16 establishes that for each sequence of selected directions u_t, there is a time t_0 and a point $x^* \in \mathcal{F}$ such that $x_t = x^*$ for all $t \geq t_0$. Both t_0 and x^* depend on the sequence of directions $(u_1, u_2, \ldots)$.

We want to evaluate the probability of event Q comprising all sequences of directions $(u_1, u_2, \ldots)$ leading to a situation where there are $\kappa\tau$-violating directions in point x^*. Choose $\varepsilon > 0$ and let $Q_T \subset Q$ contain only sequences of decisions $(u_1, u_2, \ldots)$ such that $t_0 = T$.

Since $\limsup p_t > 0$, there is a subsequence k_t such that $p_{k_t} \geq \pi > 0$. For any $k_t > T$, we know that $\mathcal{U}$ contains $\kappa\tau$-violating directions in $x_{k_t-1} = x^*$. Direction u_{k_t} is not one of them because this

4. Strictly speaking we should introduce the tolerance $\kappa > 0$ into the SMO algorithm. We can also claim that (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2003) have established proposition 16 with $\kappa = 0$ and $\tau > 0$ for the specific case of SVMs. Therefore Theorems 17 and 18 remain valid.

would make x_{k_t} different from $x_{k_t-1} = x^*$. This occurs with probability $1 - p_{k_t} \leq 1 - \pi < 1$. The probability that this happens simultaneously for N distinct $k_t > T$ is smaller than $(1-\pi)^N$ for any N. We get $P(Q_T) \leq \varepsilon/T^2$ by choosing N large enough.

Then we have $P(Q) = \sum_T P(Q_T) \leq \varepsilon \left(\sum_T 1/T^2\right) = K\varepsilon$. Hence $P(Q) = 0$ because we can choose ε as small as we want. We can therefore assert with probability 1 that $\mathcal{U}$ contains no $\kappa\tau$-violating directions in point x^*. ■

Example (LASVM) The LASVM algorithm (Section 3.2) is[5] an Approximate Stochastic WDS that alternates two strategies for selecting search directions: PROCESS and REPROCESS. Theorem 18 applies because $\limsup p_t > 0$.

Proof Consider a arbitrary iteration T corresponding to a REPROCESS.

Let us define the following assertions:

- A – There are τ-violating pairs (i, j) with both $i \in \mathcal{S}$ and $j \in \mathcal{S}$.
- B – A is false, but there are τ-violating pairs (i, j) with either $i \in \mathcal{S}$ or $j \in \mathcal{S}$.
- C – A and B are false, but there are τ-violating pairs (i, j).
- Q_t – Direction u_t is τ-violating in x_{t-1}.

A reasoning similar to the convergence discussion in Section 3.2 gives the following lower bounds (where n is the total number of examples).

$$\begin{array}{llll} P(Q_T|A) = 1 & & & \\ P(Q_T|B) = 0 & P(Q_{T+1}|B) \geq n^{-1} & & \\ P(Q_T|C) = 0 & P(Q_{T+1}|C) = 0 & P(Q_{T+2}|C) = 0 & P(Q_{T+3}|C) \geq n^{-2}. \end{array}$$

Therefore

$$\begin{array}{l} P(\, Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+2} \mid A \,) \geq n^{-2} \\ P(\, Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+2} \mid B \,) \geq n^{-2} \\ P(\, Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+2} \mid C \,) \geq n^{-2}. \end{array}$$

Since $p_t = P(Q_t \mid A \cup B \cup C)$ and since the events A, B, and C are disjoint, we have

$$p_T + p_{T+1} + p_{T+2} + p_{T+4} \geq P(\, Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+2} \mid A \cup B \cup C \,) \geq n^{-2}.$$

Therefore $\limsup p_t \geq \frac{1}{4} n^{-2}$. ■

Example (LASVM + Gradient Selection) The LASVM algorithm with Gradient Example Selection remains an Approximate WDS algorithm. Whenever Random Example Selection has a non zero probability to pick a τ-violating pair, Gradient Example Selection picks the a τ-violating pair with maximal gradient with probability one. Reasoning as above yields $\limsup p_t \geq 1$. Therefore Theorem 18 applies and the algorithm converges to a solution of the SVM QP problem.

Example (LASVM + Active Selection + Randomized Search) The LASVM algorithm with Active Example Selection remains an Approximate WDS algorithm. However it does not necessarily verify the conditions of Theorem 18. There might indeed be τ-violating pairs that do not involve the example closest to the decision boundary.

However, convergence occurs when one uses the Randomized Search method to select an example near the decision boundary. There is indeed a probability greater than $1/n^M$ to draw a sample

5. See footnote 4 discussing the tolerance κ in the case of SVMs.

containing M copies of the same example. Reasonning as above yields $\limsup p_t \geq \frac{1}{4} n^{-2M}$. Therefore, Theorem 18 applies and the algorithm eventually converges to a solution of the SVM QP problem.

In practice this convergence occurs very slowly because it involves very rare events. On the other hand, there are good reasons to prefer the intermediate kernel classifiers visited by this algorithm (see Section 4).

References

M. A. Aizerman, É. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

G. Bakır, L. Bottou, and J. Weston. Breaking SVM complexity with cross-training. In Lawrence Saul, Bernhard Schölkopf, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 81–88. MIT Press, 2005.

A. Bordes and L. Bottou. The Huller: a simple and efficient online SVM. In *Proceedings of the 16th European Conference on Machine Learning (ECML2005)*, Lecture Notes in Artificial Intelligence, to appear. Springer, 2005.

L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing.*, volume 2, pages 77–82, Jerusalem, October 1994. IEEE.

L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *Proceedings of ICML'2000*, 2000.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Processing Systems*, 2001.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 241–248. MIT Press, Cambridge, MA, 2005.

C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. Technical report, Computer Science and Information Engineering, National Taiwan University, 2001-2004. http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

D. Cohn, L. Atlas, and R. Ladner. Training connectionist networks with queries and selective sampling. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.

C. Domingo and O. Watanabe. MadaBoost: a modification of AdaBoost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory, COLT'00*, pages 180–189, 2000.

B. Eisenberg and R. Rivest. On the sample complexity of PAC learning using random and chosen examples. In M. Fulk and J. Case, editors, *Proceedings of the Third Annual ACM Workshop on Computational Learning Theory*, pages 154–162, San Mateo, CA, 1990. Kaufmann.

V. V. Fedorov. *Theory of Optimal Experiments.* Academic Press, New York, 1972.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, San Francisco, CA, 1998. Morgan Kaufmann.

T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel Adatron algorithm: a fast and simple learning procedure for support vector machines. In J. Shavlik, editor, *15th International Conf. Machine Learning*, pages 188–196. Morgan Kaufmann Publishers, 1998. See (Cristianini and Shawe-Taylor, 2000, section 7.2) for an updated presentation.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

H.-P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: The Cascade SVM. In Lawrence Saul, Bernhard Schölkopf, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In S. J. Hanson, J. D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

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*, pages 169–184, Cambridge, MA, 1999. MIT Press.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

Y. Li and P. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46:361–387, 2002.

C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001.

N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.

G. Loosli, S. Canu, S.V.N. Vishwanathan, A. J. Smola, and M. Chattopadhyay. Une boîte à outils rapide et simple pour les SVM. In Michel Liquière and Marc Sebban, editors, *CAp 2004 - Confrence d'Apprentissage*, pages 113–128. Presses Universitaires de Grenoble, 2004. ISBN 9-782706-112249.

D. J. C. MacKay. Information based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992.

N. Murata and S.-I. Amari. Statistical analysis of learning dynamics. *Signal Processing*, 74(1): 3–28, 1999.

N. J. Nilsson. *Learning machines: Foundations of Trainable Pattern Classifying Systems*. McGraw–Hill, 1965.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

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

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 839–846. Morgan Kaufmann, June 2000.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1986.

I. Steinwart. Sparseness of support vector machines—some asymptotically sharp bounds. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

N. Takahashi and T. Nishi. On termination of the SMO algorithm for support vector machines. In *Proceedings of International Symposium on Information Science and Electrical Engineering 2003 (ISEE 2003)*, pages 187–190, November 2003.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, California, 2000. Morgan Kaufmann.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Very large SVM training using core vector machines. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT'05)*. Society for Artificial Intelligence and Statistics, 2005.

V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, 1982.

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

V. N. Vapnik, T. G. Glaskova, V. A. Koscheev, A. I. Mikhailski, and A. Y. Chervonenkis. *Algorihms and Programs for Dependency Estimation*. Nauka, 1984. In Russian.

S. V. N. Vishwanathan, A. J. Smola, and M. Narasimha Murty. SimpleSVM. In *Proceedings of ICML 2003*, pages 760–767, 2003.

J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, pages 413–420. Society for Artificial Intelligence and Statistics, 2005.

G. Zoutendijk. *Methods of Feasible Directions*. Elsevier, 1960.