Journal of Machine Learning Research 2 (2001) 213-242 Submitted 3/01; Published 12/01

A New Approximate Maximal Margin Classification Algorithm

Claudio Gentile GENTILE@DSI.UNIMI.IT
Dipartimento di Scienze dell'Informazione
Universita' di Milano
Via Comelico 39,
20135 Milano, Italy

Editors: Nello Cristianini, John Shawe-Taylor and Bob Williamson

Abstract

A new incremental learning algorithm is described which approximates the maximal margin hyperplane w.r.t. norm $p \geq 2$ for a set of linearly separable data. Our algorithm, called ALMA_p (Approximate Large Margin algorithm w.r.t. norm p), takes $O\left(\frac{(p-1)}{\alpha^2\,\gamma^2}\right)$ corrections to separate the data with p-norm margin larger than $(1-\alpha)\,\gamma$, where γ is the (normalized) p-norm margin of the data. ALMA_p avoids quadratic (or higher-order) programming methods. It is very easy to implement and is as fast as on-line algorithms, such as Rosenblatt's Perceptron algorithm. We performed extensive experiments on both real-world and artificial datasets. We compared ALMA_2 (i.e., ALMA_p with $p = 2$) to standard Support vector Machines (SVM) and to two incremental algorithms: the Perceptron algorithm and Li and Long's ROMMA. The accuracy levels achieved by ALMA_2 are superior to those achieved by the Perceptron algorithm and ROMMA, but slightly inferior to SVM's. On the other hand, ALMA_2 is quite faster and easier to implement than standard SVM training algorithms. When learning sparse target vectors, ALMA_p with $p > 2$ largely outperforms Perceptron-like algorithms, such as ALMA_2.

Keywords: Binary Classification, Large Margin, Support Vector Machines, On-line Learning

1. Introduction

Vapnik's Support Vector Machines (SVM) are a statistical model of data that simultaneously minimizes model complexity and data fitting error (Vapnik, 1998). SVM have attracted a lot of interest and have spurred voluminous work in Machine Learning, both theoretical and experimental. The remarkable generalization ability exhibited by SVM can be explained through margin-based VC theory (e.g., Shawe-Taylor et al., 1998; Anthony and Bartlett, 1999; Vapnik, 1998; Cristianini and Shawe-Taylor, 2000, and references therein).

At the core of SVM lies the problem of finding the so-called *maximal margin hyperplane*. Briefly, given a set linearly separable data, the maximal margin hyperplane classifies all data correctly and maximizes the minimal distance (the margin) between the data and the hyperplane. If euclidean norm is used to measure the distance then computing the maximal margin hyperplane corresponds to the, by now classical, SVM training problem (Cortes and Vapnik, 1995). This task is naturally formulated as a quadratic programming problem.

©2001 Claudio Gentile.

If an arbitrary norm p is used then such a task turns to a more general mathematical programming problem (e.g., Mangasarian, 1997; Nachbar et al., 1993) to be solved by general purpose (and computationally intensive) optimization methods. This more general task naturally arises in feature selection problems when the target to be learned is sparse (i.e., when the target has many irrelevant features). This is often the case in a number of natural language processing problems (e.g., Golding and Roth, 1996; Dagan et al., 1997).

A fair amount of recent work on SVM centers on finding simple and efficient methods to solve maximal margin hyperplane problems (e.g., Osuna et al., 1997; Joachims, 1998; Friess et al., 1998; Platt, 1998; Kowalczyk, 1999; Keerthi et al., 1999; Li and Long, 1999). This paper follows that trend, giving two main contributions. The first contribution is a new efficient algorithm which *approximates* the maximal margin hyperplane w.r.t. norm p to any given accuracy. We call this algorithm ALMA_p (Approximate Large Margin algorithm w.r.t. norm p). ALMA_p is naturally viewed as an *on-line* algorithm, i.e., as an algorithm which processes the examples one at a time. A distinguishing feature of ALMA_p is that its relevant parameters (such as the learning rate) are dynamically adjusted over time. In this sense, ALMA_p is a refinement of the on-line algorithms recently introduced by Auer et al. (2001).

On-line algorithms are useful when the examples become available to the learning algorithm one at a time, but also when the training set is too large to consider all examples at once. ALMA_2 (i.e., ALMA_p with $p = 2$) is a perceptron-like algorithm; the operations it performs can be expressed as dot products, so that we can replace them by kernel functions (Aizerman et al., 1964). ALMA_2 approximately solves the SVM training problem, avoiding quadratic programming. Unlike previous approaches (Cortes and Vapnik, 1995; Osuna et al., 1997; Joachims, 1998; Friess et al., 1998; Platt, 1998), our algorithm operates directly on (an approximation to) the primal maximal margin problem, instead of its (Wolfe) dual. ALMA_p is more similar to algorithms such as Li and Long's ROMMA (Li and Long, 1999) and the one analyzed by Kowalczyk (1999) and Keerthi et al. (1999). However, it seems those algorithms have been specifically designed for euclidean norm. Unlike those algorithms, ALMA_p remains computationally efficient when measuring the margin through a generic norm p.

As far as theoretical performance is concerned, ALMA_2 achieves essentially the same bound on the number of corrections as the one obtained by a version of Li and Long's ROMMA. In the case when p is logarithmic in the dimension of the instance space (Gentile and Littlestone, 1999) ALMA_p yields results similar to multiplicative algorithms, such as Littlestone's Winnow (Littlestone, 1988) and the Weighted Majority algorithm (Littlestone and Warmuth, 1994; Grove et al., 2001). The associated margin-dependent generalization bounds are very close to those obtained by estimators based on linear programming (e.g., Mangasarian, 1968; Anthony and Bartlett, 1999, Chap. 14).

The second contribution of this paper is an experimental investigation of ALMA_p on both real-world and artificial datasets. In our experiments we emphasized the accuracy performance achieved by ALMA_p as a fully on-line algorithm, i.e., after just one sweep through the training examples. Following Freund and Schapire (1999), the hypotheses produced by ALMA_p during training are combined via Helmbold and Warmuth's (1995) leave-one out scheme to make a *voted* hypothesis. We ran ALMA_2 with kernels on the real-world datasets and ALMA_p with $p > 2$ *without* kernels on artificial datasets. The real-world datasets are well-known Optical Character Recognition (OCR) benchmarks. On these datasets we followed the experimental setting described by Cortes and Vapnik (1995), Freund and Schapire (1999), Li and Long (1999) and Platt et al. (1999). We compared our algorithm to standard SVM, to the Perceptron algorithm and to ROMMA. We found that ALMA_2 generalizes quite better than both ROMMA and the Perceptron algorithm, but slightly worse than SVM. On the other hand, ALMA_2 is as fast and easy to implement as the other Perceptron-like algorithms. Hence, compared to standard algorithms, training SVM with ALMA_2 saves a considerable amount of time.

In the experiments with the artificial datasets we have been mainly interested in comparing the accuracy of Perceptron-like algorithms, such as ALMA_2, to the accuracy of non-Perceptron-like algorithms, such as ALMA_p with $p > 2$. When learning sparse target vectors with ALMA_p, the performance gap between $p = 2$ and $p > 2$ is big. This is mainly due to the different convergence speed of the two kinds of algorithms (see also the paper by Kivinen et al., 1997).

The next section defines our major notation and recalls some basic preliminaries. In Section 3 we describe ALMA_p and claim its theoretical properties. Section 4 describes our experiments. Concluding remarks and open problems are given in the last section.

2. Preliminaries and notation

This section defines our major notation and recalls some basic preliminaries.

An *example* is a pair $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is an *instance* belonging to a given *instance space* $\mathcal{X} \subseteq \mathcal{R}^n$ and $y \in \{-1, +1\}$ is the binary *label* associated with $\boldsymbol{x}$. A weight vector $\boldsymbol{w} = (w_1, ..., w_n) \in \mathcal{R}^n$ represents an n-dimensional hyperplane passing through the origin. It is natural to associate with $\boldsymbol{w}$ a linear threshold classifier with threshold zero: $\boldsymbol{w} : \boldsymbol{x} \to \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x}) = 1$ if $\boldsymbol{w} \cdot \boldsymbol{x} \geq 0$ and $= -1$ otherwise. When $p \geq 1$ we denote by $||\boldsymbol{w}||_p$ the p-norm of $\boldsymbol{w}$, i.e., $||\boldsymbol{w}||_p = (\sum_{i=1}^n |w_i|^p)^{1/p}$ (also, $||\boldsymbol{w}||_\infty = \lim_{p\to\infty} (\sum_{i=1}^n |w_i|^p)^{1/p} = \max_i |w_i|$). We say that q is *dual* to p if $\frac{1}{p} + \frac{1}{q} = 1$ holds. For instance, the 1-norm is dual to the ∞-norm and the 2-norm is self-dual. In this paper we assume that p and q are some pair of dual values, with $p \geq 2$. We use p-norms for instances and q-norms for weight vectors. For the sake of simplifying notation throughout this paper we use normalized instances $\hat{\boldsymbol{x}} = \boldsymbol{x}/||\boldsymbol{x}||_p$, where the norm p will be clear from the surrounding context. The (normalized) p-norm margin (or just the margin, if p is clear from the context) of a hyperplane $\boldsymbol{w}$ with $||\boldsymbol{w}||_q \leq 1$ on example $(\boldsymbol{x}, y)$ is defined as $y\, \boldsymbol{w} \cdot \hat{\boldsymbol{x}}$. If this margin is positive[1] then $\boldsymbol{w}$ classifies $(\boldsymbol{x}, y)$ correctly. Notice that from Hölder's inequality we have $|\boldsymbol{w} \cdot \hat{\boldsymbol{x}}| \leq ||\boldsymbol{w}||_q \, ||\hat{\boldsymbol{x}}||_p \leq 1$. Hence $y\, \boldsymbol{w} \cdot \hat{\boldsymbol{x}} \in [-1, 1]$.

Our goal is to approximate the maximal p-norm margin hyperplane for a set of examples (the training set). For this purpose, we use terminology and analytical tools from the on-line learning literature. We focus on an on-line learning model introduced by Littlestone (1988) and Angluin (1988). An on-line learning algorithm processes the examples one at a time in *trials*. In each trial, the algorithm observes an instance $\boldsymbol{x}$ and is required to predict the label y associated with $\boldsymbol{x}$. We denote the prediction by $\hat{y}$. The prediction $\hat{y}$ combines the current instance $\boldsymbol{x}$ with the current internal state of the algorithm. In our case this state

1. We assume that $\boldsymbol{w} \cdot \boldsymbol{x} = 0$ yields a wrong classification, independent of y.

is essentially a weight vector $\boldsymbol{w}$, representing the algorithm's current hypothesis about the maximal margin hyperplane. After the prediction is made, the true value of y is revealed and the algorithm suffers a *loss*, measuring the "distance" between the prediction $\hat{y}$ and the label y. Then the algorithm updates its internal state.

In this paper the prediction $\hat{y}$ can be seen as the linear function $\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x}$ and the loss is a margin-based 0-1 Loss: the loss of $\boldsymbol{w}$ on example $(\boldsymbol{x}, y)$ is 1 if $y\, \boldsymbol{w} \cdot \hat{\boldsymbol{x}} \le (1-\alpha)\,\gamma$ and 0 otherwise, for suitably chosen $\alpha, \gamma \in [0,1]$. Therefore, if $||\boldsymbol{w}||_q \le 1$ the algorithm incurs positive loss if and only if $\boldsymbol{w}$ classifies $(\boldsymbol{x}, y)$ with (p-norm) margin not larger than $(1-\alpha)\,\gamma$. The on-line algorithms are typically *loss driven*, i.e., they do update their internal state only in those trials where they suffer a positive loss. We call a *correction* a trial where this happens. In the special case when $\alpha = 1$ a correction is a *mistaken* trial and a loss driven algorithm turns to a *mistake driven* (Littlestone, 1988) algorithm.

Throughout the paper we use the subscript t for $\boldsymbol{x}$ and y to denote the instance and the label processed in trial t. We use the subscript k for those variables, such as the algorithm's weight vector $\boldsymbol{w}$, which are updated only within a correction. In particular, $\boldsymbol{w}_k$ denotes the algorithm's weight vector after $k-1$ corrections (so that $\boldsymbol{w}_1$ is the initial weight vector). The goal of the on-line algorithm is to bound the cumulative loss (i.e., the total number of corrections or mistakes) it suffers on an arbitrary sequence of examples $S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T))$. Consider the special case when S is linearly separable with margin γ. If we pick $\alpha < 1$ then a bounded loss clearly implies convergence in a finite number of steps to (an approximation of) the maximal margin hyperplane for S. When a training set is linearly separable with margin γ and hyperplane $\boldsymbol{w}$ is such that $y\, \boldsymbol{w} \cdot \hat{\boldsymbol{x}} \ge (1-\alpha)\,\gamma$ for any $(\boldsymbol{x}, y)$ in the training set we sometimes say that $\boldsymbol{w}$ is an *α-approximation* to the maximal margin hyperplane (for that training set).

Remark 1 *Our definition of margin is restricted to zero-threshold linear classifiers, i.e., to hyperplanes passing through the origin. The usual definition of margin in SVM literature (Cortes and Vapnik, 1995) actually considers the more general non-zero threshold linear classifiers. The threshold of an SVM maximal margin hyperplane is sometimes called the* bias *term of the SVM. Restricting margin analyses to zero-threshold hyperplanes loses only a constant factor (e.g., Cristianini and Shawe-Taylor, 2000). Nonetheless, in practical applications such a constant factor might make a significant difference.*

3. The approximate large margin algorithm ALMA$_p$

ALMA$_p$ is a large margin variant of the p-norm Perceptron algorithm[2] (Grove et al., 2001; Gentile and Littlestone, 1999), and is similar in spirit to the variable learning rate algorithms introduced by Auer et al. (2001). We analyze ALMA$_p$ by giving upper bounds on the number of corrections. We do not resort to the proof techniques developed by (Auer et al., 2001), as they seem to give rise to suboptimal results when applied to the algorithm described here.

The theoretical contribution of this paper is Theorem 3 below. This theorem has two parts. Part 1 bounds the number of corrections in the linearly separable case. In the special case when $p = 2$ this bound is very similar to the one proven by Li and Long for a version of

2. The p-norm Perceptron algorithm is a generalization of the classical Perceptron algorithm (Rosenblatt, 1962; Block, 1962; Novikov, 1962): p-norm Perceptron is actually Perceptron when $p = 2$.

Algorithm ALMA$_p(\alpha; B, C)$
with $\alpha \in (0,1]$, B, $C > 0$.
Initialization:
Initial weight vector $\boldsymbol{w}_1 = \mathbf{0}$; $k = 1$.
For $t = 1, ..., T$ **do**:

- Get example $(\boldsymbol{x}_t, y_t) \in \mathcal{X} \times \{-1, +1\}$;
- Update weights as follows:

Set $\gamma_k = B\sqrt{p-1}\,\frac{1}{\sqrt{k}}$;

If $y_t\, \boldsymbol{w}_k \cdot \hat{\boldsymbol{x}}_t \le (1-\alpha)\,\gamma_k$ **then :**
$$\eta_k = \frac{C}{\sqrt{p-1}}\,\frac{1}{\sqrt{k}},$$
$$\boldsymbol{w}'_k = \mathbf{f}^{-1}(\mathbf{f}(\boldsymbol{w}_k) + \eta_k\, y_t\, \hat{\boldsymbol{x}}_t),$$
$$\boldsymbol{w}_{k+1} = \boldsymbol{w}'_k / \max\{1, ||\boldsymbol{w}'_k||_q\},$$
$$k \leftarrow k+1.$$

Figure 1: The approximate large margin algorithm ALMA$_p$.

ROMMA (called aggressive ROMMA). Part 2 holds for an arbitrary sequence of examples. A bound which is very close to the one proven by Grove et al. (2001) and Gentile and Littlestone (1999) for the (constant learning rate) p-norm Perceptron algorithm is obtained as a special case.

In order to define our algorithm, we need to recall the following mapping $\mathbf{f}$ (Gentile and Littlestone, 1999) (a p-indexing for $\mathbf{f}$ is understood): $\mathbf{f} : \mathcal{R}^n \to \mathcal{R}^n$, $\mathbf{f} = (f_1, ..., f_n)$, where

$$f_i(\boldsymbol{w}) = \frac{\operatorname{sign}(w_i)\,|w_i|^{q-1}}{||\boldsymbol{w}||_q^{q-2}}, \quad \boldsymbol{w} = (w_1, ..., w_n) \in \mathcal{R}^n.$$

Observe that $p = q = 2$ yields the identify function. The (unique) inverse $\mathbf{f}^{-1}$ of $\mathbf{f}$ is (Gentile and Littlestone, 1999) $\mathbf{f}^{-1} : \mathcal{R}^n \to \mathcal{R}^n$, $\mathbf{f}^{-1} = (f_1^{-1}, ..., f_n^{-1})$, where

$$f_i^{-1}(\boldsymbol{\theta}) = \frac{\operatorname{sign}(\theta_i)\,|\theta_i|^{p-1}}{||\boldsymbol{\theta}||_p^{p-2}}, \quad \boldsymbol{\theta} = (\theta_1, ..., \theta_n) \in \mathcal{R}^n,$$

namely, $\mathbf{f}^{-1}$ is obtained from $\mathbf{f}$ by replacing q with p. It is easy to check that $\mathbf{f}$ is the gradient of the scalar function $\frac{1}{2}||\cdot||_q^2$, while $\mathbf{f}^{-1}$ is the gradient of the (dual) function $\frac{1}{2}||\cdot||_p^2$. The following simple property of $\mathbf{f}$ will be useful.

Lemma 2 *(Gentile and Littlestone, 1999) The function* $\mathbf{f}$ *maps vectors with a given q-norm to vectors with the same p-norm, i.e., for any* $\boldsymbol{w} \in \mathcal{R}^n$ *we have* $||\mathbf{f}(\boldsymbol{w})||_p = ||\boldsymbol{w}||_q$. ■

ALMA$_p$ is described in Figure 1. The algorithm is parameterized by $\alpha \in (0,1]$, $B > 0$ and $C > 0$. Parameter α measures the degree of approximation to the optimal margin

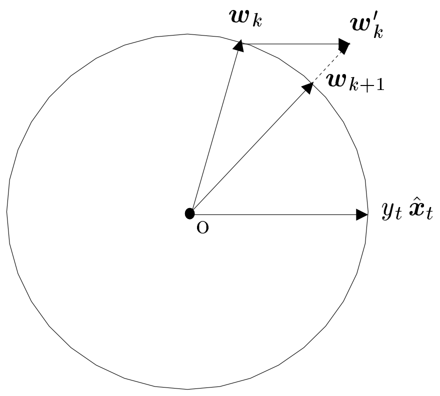


Figure 2: The update rule of ALMA$_p$ when $p = q = 2$. The circle is a two-dimensional $\mathcal{W}$.

hyperplane, while B and C might be considered as tuning parameters. Their use will be made clear in Theorem 3. Let $\mathcal{W}$ be the q-norm unit ball, i.e., $\mathcal{W} = \{\boldsymbol{w} \in \mathcal{R}^n : ||\boldsymbol{w}||_q \leq 1\}$. ALMA$_p$ maintains a vector $\boldsymbol{w}_k$ of n weights in $\mathcal{W}$. It starts from $\boldsymbol{w}_1 = \mathbf{0}$. At time t the algorithm processes example $(\boldsymbol{x}_t, y_t)$. If the current weight vector $\boldsymbol{w}_k$ classifies $(\boldsymbol{x}_t, y_t)$ with (normalized) margin not larger than $(1-\alpha)\,\gamma_k$ then a correction occurs. Here γ_k is intended as the current approximation to the unknown maximal margin (denoted by γ^* in Theorem 3) on the data. The update rule[3] has two main steps. The first step gives $\boldsymbol{w}'_k$ through the classical update of a (p-norm) perceptron-like algorithm (notice, however, that the learning rate η_k scales with k, the number of corrections occurred so far). The second step gives $\boldsymbol{w}_{k+1}$ by *projecting*[4] $\boldsymbol{w}'_k$ onto $\mathcal{W}$: $\boldsymbol{w}_{k+1} = \boldsymbol{w}'_k/||\boldsymbol{w}'_k||_q$ if $||\boldsymbol{w}'_k||_q > 1$ and $\boldsymbol{w}_{k+1} = \boldsymbol{w}'_k$ otherwise. The projection step makes the new weight vector $\boldsymbol{w}_{k+1}$ belong to $\mathcal{W}$. Figure 2 gives a graphical representation of the update rule.

It is worth discussing at this point how ALMA$_p$ is qualitatively different from previous on-line algorithms, such as the (p-norm) Perceptron algorithm and ROMMA. First, we notice that ALMA$_p$ maintains bounded weight vectors, but it uses a decaying learning rate. This is actually qualitatively similar to the standard (p-norm) Perceptron algorithm, where weight vectors are unbounded but the learning rate is kept constant. In fact, in both cases later updating instances have less influence on the direction of the current weight vector than earlier instances. On the other hand, unlike the (p-norm) Perceptron algorithm, ALMA$_p$ is sensitive to margins, i.e., the current weight vector gets updated even if the current margin is positive but smaller than desired. Compared to aggressive ROMMA, ALMA$_p$ requires the accuracy parameter α be fixed ahead of time; if α is not close to zero, this tends to make ALMA$_p$'s corrections less frequent than aggressive ROMMA's (see Section 4.2).

3. In the degenerate case that $\boldsymbol{x}_t = \mathbf{0}$ no update takes place.
4. From the proof of Theorem 3 the reader can see that the only way we exploit this projection step is through the condition $||\boldsymbol{w}_k||_q \leq 1$ for all k. Therefore if we replaced the update rule $\boldsymbol{w}_{k+1} = \boldsymbol{w}'_k/\max\{1, ||\boldsymbol{w}'_k||_q\}$ in Figure 1 by the simpler rule $\boldsymbol{w}_{k+1} = \boldsymbol{w}'_k/||\boldsymbol{w}'_k||_q$ Theorem 3 would still hold. The advantage of using the former rule is computational, as it requires less weight updating.

We now claim the theoretical properties of ALMA$_p$. The following theorem has two parts. In part 1 we treat the separable case. Here we prove that a special choice of parameters B and C gives rise to an algorithm which computes an α-approximation to the maximal margin hyperplane, for any given accurary α. In part 2 we show that if a suitable relationship between B and C is satisfied then a bound on the number of corrections can be proven in the general (nonseparable) case. The bound of part 2 is in terms of the margin-based quantity $\mathcal{D}_\gamma(\boldsymbol{u};(\boldsymbol{x},y)) = \max\{0, \gamma - y\,\boldsymbol{u}\cdot\hat{\boldsymbol{x}}\}$, $\gamma > 0$. (Here a p-indexing for $\mathcal{D}_\gamma$ is understood). $\mathcal{D}_\gamma$ is called *deviation* by Freund and Schapire (1999) and *linear hinge loss* by Gentile and Warmuth (2001).

Notice that B and C in part 1 do not meet the requirements given in part 2. On the other hand, in the separable case B and C chosen in part 2 do not yield, for any small α, an α-approximation to the maximal margin hyperplane.

Theorem 3 *Let $\mathcal{X} = \mathcal{R}^n$, $\mathcal{W} = \{\boldsymbol{w} \in \mathcal{R}^n : ||\boldsymbol{w}||_q \le 1\}$, $S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)) \in (\mathcal{X}\times\{-1,+1\})^T$, and $\mathcal{M}$ be the set of corrections of* ALMA$_p(\alpha; B, C)$ *running on S (i.e., the set of trials t such that $y_t\,\boldsymbol{w}_k\cdot\hat{\boldsymbol{x}}_t \le (1-\alpha)\,\gamma_k$).*

1. Let $\gamma^* = \max_{\boldsymbol{w}\in\mathcal{W}} \min_{t=1,...,T} y_t\,\boldsymbol{w}\cdot\hat{\boldsymbol{x}}_t > 0$. Then ALMA$_p(\alpha; \sqrt{8}/\alpha, \sqrt{2})$ *achieves the following bound*[5] *on $|\mathcal{M}|$:*

$$|\mathcal{M}| \le \frac{2\,(p-1)}{(\gamma^*)^2}\left(\frac{2}{\alpha}-1\right)^2 + \frac{8}{\alpha} - 4 = O\left(\frac{p-1}{\alpha^2\,(\gamma^*)^2}\right). \tag{1}$$

Furthermore, throughout the run of ALMA$_p(\alpha; \sqrt{8}/\alpha, \sqrt{2})$ *we have $\gamma_k \ge \gamma^*$. Hence (1) is also an upper bound on the number of trials t such that $y_t\,\boldsymbol{w}_k\cdot\hat{\boldsymbol{x}}_t \le (1-\alpha)\,\gamma^*$.*

2. Let the parameters B and C in Figure 1 satisfy the equation[6]

$$C^2 + 2\,(1-\alpha)\,B\,C = 1.$$

Then for any $\boldsymbol{u}\in\mathcal{W}$, ALMA$_p(\alpha; B, C)$ *achieves the following bound on $|\mathcal{M}|$, holding for any $\gamma > 0$, where $\rho^2 = \frac{p-1}{C^2\,\gamma^2}$:*

$$|\mathcal{M}| \le \frac{1}{\gamma}\sum_{t\in\mathcal{M}} D_\gamma(\boldsymbol{u};(\boldsymbol{x}_t,y_t)) + \frac{\rho^2}{2} + \sqrt{\frac{\rho^4}{4} + \frac{\rho^2}{\gamma}\sum_{t\in\mathcal{M}} D_\gamma(\boldsymbol{u};(\boldsymbol{x}_t,y_t)) + \rho^2}.$$

Observe that when $\alpha = 1$ the above inequality turns to a bound on the number of mistaken trials. *In such a case the value of γ_k (in particular, the value of B) is immaterial, while C is forced to be 1.*

Proof. We assume throughout this proof that the k-th correction occurs on example $(\boldsymbol{x}_t, y_t)$. We use the following shorthand notation: $\boldsymbol{\theta}_k = \mathbf{f}(\boldsymbol{w}_k)$, $\boldsymbol{\theta}'_k = \mathbf{f}(\boldsymbol{w}'_k) = \boldsymbol{\theta}_k + \eta_k\,y_t\,\hat{\boldsymbol{x}}_t$ and $N_{k+1} = \max\{1, ||\boldsymbol{w}'_k||_q\}$. We now consider the two parts separately.

5. We did not optimize the constants here.
6. Notice that B and C in part 1 do not satisfy this equation.

1. Let $\gamma_k^* = y_t\, \boldsymbol{u} \cdot \hat{\boldsymbol{x}}_t$, where $\boldsymbol{u}$ is the maximal margin hyperplane for the whole sequence S. We study how fast the quantity $\boldsymbol{u} \cdot \boldsymbol{\theta}_k$ increases from correction to correction. From the update rule of Figure 1 we have

$$\boldsymbol{u} \cdot \boldsymbol{\theta}_{k+1} = \frac{\boldsymbol{u} \cdot \boldsymbol{\theta}_k + \eta_k\, y_t\, \boldsymbol{u} \cdot \hat{\boldsymbol{x}}_t}{N_{k+1}} = \frac{\boldsymbol{u} \cdot \boldsymbol{\theta}_k + \eta_k\, \gamma_k^*}{N_{k+1}}. \tag{2}$$

Observe that $\boldsymbol{u} \cdot \boldsymbol{\theta}_k \geq 0$ for any k, since the data are linearly separable.

We need to find an upper bound on the normalization factor N_{k+1}. To this end, we focus on the square N_{k+1}^2. By virtue of Lemma 2 we can write

$$\begin{aligned} N_{k+1}^2 &= \max\{1, ||\boldsymbol{w}'_k||_q^2\} \\ &= \max\{1, ||\boldsymbol{\theta}'_k||_p^2\} \\ &= \max\{1, ||\boldsymbol{\theta}_k + \eta_k\, y_t\, \hat{\boldsymbol{x}}_t||_p^2\}. \end{aligned}$$

Furthermore,

$$\begin{aligned} ||\boldsymbol{\theta}_k + \eta_k\, y_t\, \hat{\boldsymbol{x}}_t||_p^2 &\leq ||\boldsymbol{\theta}_k||_p^2 + \eta_k^2\,(p-1) + 2\,\eta_k\, y_t\, \mathbf{f}^{-1}(\boldsymbol{\theta}_k) \cdot \hat{\boldsymbol{x}}_t \\ &= ||\boldsymbol{w}_k||_q^2 + \eta_k^2\,(p-1) + 2\,\eta_k\, y_t\, \boldsymbol{w}_k \cdot \hat{\boldsymbol{x}}_t \\ &\leq 1 + \eta_k^2\,(p-1) + 2\,(1-\alpha)\,\eta_k\,\gamma_k \\ &= 1 + 2\,A/k, \end{aligned}$$

where the first inequality is essentially proven by Grove et al. (2001) (see also Lemma 2 in the paper by Gentile and Littlestone, 1999), the first equality is again an application of Lemma 2, the second inequality derives from Figure 1, and the last equality derives from Figure 1 by setting $A = 4/\alpha - 3$. Thus we conclude that

$$N_{k+1} \leq \sqrt{1 + 2\,A/k}.$$

Now, we set for brevity $m = |\mathcal{M}|$, $\rho_k = \frac{\sqrt{p-1}}{\gamma_k^*}$ and $\rho = \frac{\sqrt{p-1}}{\gamma^*}$. Since γ_k^* and γ^* are p-norm margins with $\gamma^* \leq \gamma_k$ we have $\rho_k \geq \rho \geq 1$. We plug the bound on N_{k+1} back into (2), unwrap the resulting recurrence and take into account that $\boldsymbol{w}_1 = \boldsymbol{\theta}_1 = \mathbf{0}$. We yield

$$\boldsymbol{u} \cdot \boldsymbol{\theta}_{m+1} \geq \sum_{k=1}^{m} s_k \prod_{j=k+1}^{m} r_j,$$

where

$$\begin{aligned} r_j &= \frac{1}{\sqrt{1 + 2\,A/j}}, \\ s_k &= \frac{1}{\rho_k}\, \frac{1}{\sqrt{A + k/2}} \end{aligned}$$

and the product $\prod_{j=k+1}^{m} r_j$ is assumed to be 1 if $k = m$. From Hölder's inequality and Lemma 2 it follows that

$$\boldsymbol{u} \cdot \boldsymbol{\theta}_{m+1} \leq ||\boldsymbol{u}||_q\, ||\boldsymbol{\theta}_{m+1}||_p = ||\boldsymbol{u}||_q\, ||\boldsymbol{w}_{m+1}||_q \leq 1.$$

Therefore we have obtained:

$$1 \geq \sum_{k=1}^{m} s_k \prod_{j=k+1}^{m} r_j. \tag{3}$$

We use this inequality to compute an upper bound on the number of corrections m. We proceed by lower bounding the RHS of (3), as a function of m.

We first lower bound s_k by $\frac{1}{\rho}\frac{1}{\sqrt{A+m/2}}$. Next, considering the product $\prod_{j=k+1}^{m} r_j$, we can write

$$\begin{aligned}
-\ln \prod_{j=k+1}^{m} r_j &= \frac{1}{2}\sum_{j=k+1}^{m} \ln\left(1+\frac{2\,A}{j}\right) \\
&\leq \frac{1}{2}\sum_{j=k+1}^{m} \frac{2\,A}{j} \\
&\leq A\int_{k}^{m} \frac{1}{j}\,dj \\
&= A\,\ln\frac{m}{k}.
\end{aligned}$$

This is equivalent to $\prod_{j=k+1}^{m} r_j \geq \left(\frac{k}{m}\right)^A$. Therefore (3) implies

$$\begin{aligned}
\rho &\geq \sum_{k=1}^{m} \frac{(k/m)^A}{\sqrt{A+m/2}} \\
&\geq \int_{k=0}^{m} \frac{(k/m)^A}{\sqrt{A+m/2}}\,dk \\
&= \frac{1}{A+1}\,\frac{m}{\sqrt{A+m/2}}.
\end{aligned}$$

Solving for m gives

$$\begin{aligned}
m &\leq \frac{\rho^2\,(A+1)^2}{4} + \sqrt{\frac{\rho^4\,(A+1)^4}{16} + \rho^2\,(A+1)^2\,A} \\
&\leq \frac{\rho^2\,(A+1)^2}{4} + \rho\,(A+1)^{3/2}\sqrt{\frac{\rho^2\,(A+1)}{16}+1} \\
&\leq \frac{\rho^2\,(A+1)^2}{4} + \rho\,(A+1)^{3/2}\left(\frac{\sqrt{\rho^2\,(A+1)}}{4} + \frac{2}{\sqrt{\rho^2\,(A+1)}}\right) \\
&= \frac{\rho^2\,(A+1)^2}{2} + 2\,(A+1) \qquad (4) \\
&= 2\rho^2\left(\frac{2}{\alpha}-1\right)^2 + \frac{8}{\alpha} - 4 \\
&= \frac{2\,(p-1)}{(\gamma^*)^2}\left(\frac{2}{\alpha}-1\right)^2 + \frac{8}{\alpha} - 4,
\end{aligned}$$

where the third inequality uses $\sqrt{x+1} \le \sqrt{x} + \frac{1}{2\sqrt{x}}$, for $x > 0$. This proves that the number of corrections m made by the algorithm is upper bounded as in (1). In order to show that this is also an upper bound on the number of trials t such that $y_t\, \boldsymbol{w}_k \cdot \hat{\boldsymbol{x}}_t \le (1-\alpha)\,\gamma^*$, it suffices to prove that $\gamma_k \ge \gamma^*$ for $k = 1, ..., m$. Recalling Figure 1, we see that

$$
\begin{aligned}
\gamma_k &= \frac{\sqrt{8}\,\rho_k\,\gamma_k^*}{\alpha\,\sqrt{k}} \\
&\ge \frac{\sqrt{8}\,\rho\,\gamma^*}{\alpha\,\sqrt{m}} \\
&\ge \frac{\sqrt{8}\,\rho\,\gamma^*}{\alpha\,\sqrt{\rho^2\,\frac{(A+1)^2}{2} + 2\,(A+1)}} \\
&\ge \frac{\gamma^*}{\alpha\,\sqrt{\frac{(A+1)^2}{16} + \frac{A+1}{4}}} \\
&= \frac{\gamma^*}{\sqrt{1 - \frac{\alpha^2}{4}}} \\
&\ge \gamma^*,
\end{aligned}
$$

where the second inequality is (4), the third inequality follows from $\rho \ge 1$ and the last equality follows from the definition of A. This concludes the proof of part 1.

2. The proof proceeds along the same lines as the proof of part 1. Thus we only sketch the main steps. Let $\boldsymbol{u}$ be an arbitrary vector in $\mathcal{W}$. We can write

$$
N_{k+1}\,\boldsymbol{u} \cdot \boldsymbol{\theta}_{k+1} = \boldsymbol{u} \cdot \boldsymbol{\theta}_k + \eta_k\,y_t \boldsymbol{u} \cdot \boldsymbol{x}_t, \tag{5}
$$

where

$$
\begin{aligned}
N_{k+1}^2 &= \max\{1, ||\boldsymbol{\theta}_k + \eta_k\,y_t\,\hat{\boldsymbol{x}}_t||_p^2\} \\
&\le \max\{1, ||\boldsymbol{\theta}_k||_p^2 + \eta_k^2\,(p-1) + 2\,(1-\alpha)\,\eta_k\,\gamma_k \\
&\le \max\{1, 1 + \eta_k^2\,(p-1) + 2\,(1-\alpha)\,\eta_k\,\gamma_k\} \\
&= 1 + \frac{C^2 + 2\,(1-\alpha)\,B\,C}{k} \\
&= 1 + \frac{1}{k}.
\end{aligned}
$$

From (5) and the value of η_k we obtain

$$
\sqrt{k+1}\,\boldsymbol{u} \cdot \boldsymbol{\theta}_{k+1} \ge \sqrt{k}\,\boldsymbol{u} \cdot \boldsymbol{\theta}_k + \frac{C}{\sqrt{p-1}}\,y_t\,\boldsymbol{u} \cdot \hat{\boldsymbol{x}}_t.
$$

Unwrapping, using the two inequalities $\boldsymbol{u} \cdot \boldsymbol{\theta}_{m+1} \le 1$ and $y_t \boldsymbol{u} \cdot \hat{\boldsymbol{x}}_t \ge \gamma - D_\gamma(\boldsymbol{u}; (\boldsymbol{x}_t, y_t))$, and rearraning yields

$$
\sqrt{m+1}\,\frac{\sqrt{p-1}}{C} + \sum_{t \in \mathcal{M}} D_\gamma(\boldsymbol{u}; (\boldsymbol{x}_t, y_t)) \ge m\,\gamma,
$$

holding for any $\gamma > 0$ and any $\boldsymbol{u} \in \mathcal{W}$. Solving for m gives the desired inequality. This concludes the proof. ■

Some remarks are in order at this point.

Remark 4 *It is worth emphasizing that the difference between parts 1 and 2 in Theorem 3 is mainly theoretical. For instance, the condition $C^2 + 2(1-\alpha)BC = 1$ in Part 2 is satisfied even by $B = \frac{1}{2\sqrt{\alpha}}$ and $C = \sqrt{\alpha}$, for any $\alpha \in (0,1]$. It is not hard to see that in the separable case this setting yields the bounds $|\mathcal{M}| \le \frac{1+\sqrt{5}}{2}\frac{p-1}{(\gamma^*)^2\,\alpha}$ and $\gamma_k \ge \gamma^*/3$ for all k (independent of α). Hence, if we set $\alpha = 1/2$ we obtain a hyperplane whose margin on the data is at least $\gamma^*/6$ after no more than $(1+\sqrt{5})\frac{p-1}{(\gamma^*)^2}$ corrections. This degree of data fitting could be enough for many practical purposes. In fact, one should not heavily rely on Theorem 3 to choose the "best" tuning for B and C in $\text{ALMA}_p(\alpha; B, C)$, since many of the constants occurring in the statement of that theorem are just an artifact of our analysis.*

Remark 5 *When $p = 2$ the computations performed by ALMA_p essentially involve only dot products (recall that $p = 2$ yields $q = 2$ and $\mathbf{f} = \mathbf{f}^{-1} =$ identity). Thus the generalization of ALMA_2 to the kernel case is quite standard (we just replace every dot product between instances by a kernel dot product). In fact, the linear combination $\boldsymbol{w}_{k+1}\cdot\boldsymbol{x}$ can be computed recursively, since*

$$\boldsymbol{w}_{k+1}\cdot\boldsymbol{x} = \frac{\boldsymbol{w}_k\cdot\boldsymbol{x} + \eta_k\, y_t\hat{\boldsymbol{x}}_t\cdot\boldsymbol{x}}{N_{k+1}}.$$

Here the denominator N_{k+1} equals $\max\{1, ||\boldsymbol{w}'_k||_2\}$ and the norm $||\boldsymbol{w}'_k||_2$ is again computed recursively by

$$||\boldsymbol{w}'_k||_2^2 = ||\boldsymbol{w}'_{k-1}||_2^2/N_k^2 + 2\,\eta_k\, y_t\boldsymbol{w}_k\cdot\hat{\boldsymbol{x}}_t + \eta_k^2,$$

where the dot product $\boldsymbol{w}_k\cdot\hat{\boldsymbol{x}}_t$ is taken from the k-th correction (the trial where the k-th weight update did occur), and the normalization of instances $\hat{\boldsymbol{x}} = \boldsymbol{x}/||\boldsymbol{x}||_2$ is computed as $\hat{\boldsymbol{x}} = \boldsymbol{x}/\sqrt{\boldsymbol{x}\cdot\boldsymbol{x}}$.

Remark 6 *ALMA_p with $p > 2$ is useful when learning* sparse *hyperplanes, namely those hyperplanes having only a few relevant components. Gentile and Littlestone (1999) observe that setting $p = 2\ln n$ makes a p-norm algorithm similar to a purely multiplicative algorithm such as Winnow and the Weighted Majority algorithm (Littlestone, 1988; Littlestone and Warmuth, 1994). The performance of such algorithms is ruled by a limiting pair of dual norms, i.e., the infinity norm of instances and the 1-norm of weight vectors. Likewise, ALMA_p with $p = 2\ln n$ becomes a multiplicative approximate maximal margin classification algorithm, where the margin is meant to be an ∞-norm margin. To see this, observe that $||\boldsymbol{x}||_p \le n^{1/p}||\boldsymbol{x}||_\infty$ for any $\boldsymbol{x} \in \mathcal{R}^n$. Hence $p = 2\ln n$ yields $||\boldsymbol{x}||_{(2\ln n)} \le \sqrt{e}\,||\boldsymbol{x}||_\infty$. Also, $||\boldsymbol{w}||_1 \le 1$ implies $||\boldsymbol{w}||_q \le 1$ for any $q > 1$. Thus if $||\boldsymbol{w}||_1 \le 1$ the $(2\ln n)$-norm margin $\frac{y\,\boldsymbol{w}\cdot\boldsymbol{x}}{||\boldsymbol{x}||_{(2\ln n)}}$ is actually bounded from below by the ∞-norm margin $\frac{y\,\boldsymbol{w}\cdot\boldsymbol{x}}{||\boldsymbol{x}||_\infty}$ divided by $\sqrt{e}$. The bound in part 1 of Theorem 3 becomes $|\mathcal{M}| = O\left(\frac{\ln n}{\alpha^2(\gamma^*)^2}\right)$, where $\gamma^* = \max_{\boldsymbol{w}\,:\,||\boldsymbol{w}||_1\le 1}\min_{t=1,\dots,T}\frac{y_t\,\boldsymbol{w}\cdot\boldsymbol{x}_t}{||\boldsymbol{x}_t||_\infty}$. The associated margin-based generalization bounds are very similar to those obtained by classifiers based on linear programming (Mangasarian, 1968; Anthony and Bartlett, 1999, Chap. 14).*

Remark 7 ALMA$_p$ *and its analysis could be modified to handle the case when the margin is not normalized, i.e., when the margin of hyperplane* $\boldsymbol{w}$ *on example* $(\boldsymbol{x}, y)$ *is defined to be just* $y\,\boldsymbol{w} \cdot \boldsymbol{x}$. *We only need to introduce a new variable, call it* X_k, *which stores the maximal norm of the instances seen in past corrections, and then normalize the new instance* $\boldsymbol{x}_t$ *in the update rule by* X_k. *The resulting algorithm and the corresponding analysis would be slightly more complicated than the one we gave in Theorem 3. As a matter of fact, our initial experiments with* ALMA$_p$*-like algorithms were performed with the unnormalized margin version. Such experiments, which are not reported in this paper, show that the unnormalized margin version of* ALMA$_p$ *is fairly sensitive to example ordering (in particular, the algorithm is quite sensitive to the position of outliers in the stream of examples). This is one of the reasons why in this paper we only treat the normalized margin version.*

4. Experimental results

To see how our algorithm works in practice, we tested it on a number of classification datasets, both real-world and artificial. We used ALMA$_2$ with kernels on the real datasets and ALMA$_p$ with $p = 2,\ 6,\ 10$ *without* kernels on the artificial ones. The real-world datasets are well-known OCR benchmarks: the USPS dataset (e.g., Le Cun et al., 1995), the MNIST dataset[7], and the UCI Letter dataset (Blake et al., 1998). The artificial datasets consist of examples generated by some random process according to the rules described in Section 4.4.

For the sake of comparison, we tended to follow previous experimental setups, such as those described by Cortes and Vapnik (1995), Freund and Schapire (1999), Friess et al. (1998), Li and Long (1999) and Platt et al. (1999). We reduced an N-class problem to a set of N binary problems, according to the so-called *one-versus-rest* scheme. That is, we trained the algorithms once for each of the N classes. When training on the i-th class all the examples with label i are considered positive (labelled $+1$) and all other examples are negative (labelled -1). Classification is made according to the maximum output of the N binary classifiers. There are many other ways of combining binary classifiers into a multiclass classifier. We refer the reader to the work by Dietterich and Bakiri (1995), Platt et al. (1999), Allwein et al. (2000) and to references therein. Yet another method for facing multiclass classification (which does not explicitly reduce to binary) is mentioned in Section 5.

Our experimental results are summarized in Tables 1 through 6 and in Figures 3, 4 and 5. Following Freund and Schapire (1999), the output of a binary classifier is based on either the *last* hypothesis produced by the algorithms (denoted by "last" throughout this section) or Helmbold and Warmuth's (1995) leave-one-out *voted* hypothesis (denoted by "voted"). In our experiments we actually used the variant called *average* by Freund and Schapire (1999). We denote it by "average" or "avg", for brevity. This variant gave rise to slight accuracy improvements compared to "voted".[8] When using "last", the output of the i-th

7. It can be downloaded from Y. LeCun's home page: http://www.research.att.com/~yann/ocr/mnist/.
8. Freund and Schapire (1999) seem to use the average variant without any theoretical justification. When using margin sensitive classification algorithms, such as ALMA$_p$ with $\alpha < 1$, one can prove a bound on the expected generalization error of "avg" by first proving a bound on the expected hinge loss (Gentile and Warmuth, 2001) and then applying a simple convexity argument (Kivinen and Warmuth, 1997).

binary classifier on a new instance $\boldsymbol{x}$ is

$$\mathrm{output}_i(\boldsymbol{x}) = \boldsymbol{w}^{(i)}_{m^{(i)}+1} \cdot \boldsymbol{x},$$

where $\boldsymbol{w}^{(i)}_{m^{(i)}+1}$ is the last weight vector produced during training by the i-th classification algorithm (namely, after $m^{(i)}$ corrections); when using "voted" or "avg" we need to store the sequence of prediction vectors $\boldsymbol{w}^{(i)}_1$, $\boldsymbol{w}^{(i)}_2$, ..., $\boldsymbol{w}^{(i)}_{m^{(i)}+1}$, as well as the number of trials the corresponding prediction vector survives until it gets updated. Let us denote by $c^{(i)}_k$ the number of trials the k-th weight vector of the i-th classifier survives. Then if we use "voted" the output of the i-th classifier on instance $\boldsymbol{x}$ is

$$\mathrm{output}_i(\boldsymbol{x}) = \sum_{k=1}^{m^{(i)}+1} c^{(i)}_k \,\mathrm{sign}(\boldsymbol{w}^{(i)}_k \cdot \boldsymbol{x});$$

if we use "avg" the output of the i-th classifier on instance $\boldsymbol{x}$ is

$$\mathrm{output}_i(\boldsymbol{x}) = \left(\sum_{k=1}^{m^{(i)}+1} c^{(i)}_k \,\boldsymbol{w}^{(i)}_k \right) \cdot \boldsymbol{x}.$$

In all cases the predicted label $\hat{y}$ associated with $\boldsymbol{x}$ is

$$\hat{y} = \mathrm{argmax}_{i=1...N}\ \mathrm{output}_i(\boldsymbol{x}).$$

We trained the algorithms by cycling up to 3 times ("epochs") over the training set. All the results shown in Tables 1–6 and in Figures 3–5 are averaged over 10 random permutations of the training sequences.

In Tables 1–6 the columns marked "TestErr" give the fraction of misclassified examples in the test set. The columns marked "Correct" (or "Corr", for brevity) give the total number of corrections occurred in the training phase for the N labels (recall that for Perceptron and ALMA$_p$ with $\alpha = 1$ a correction is the same as a mistaken trial).

In Figures 3–5 we plotted a number of *margin distribution* graphs (Schapire et al., 1998) yielded when running ALMA$_p$ on various datasets. For binary classification tasks the margin distribution of a (binary) classifier $\boldsymbol{w}$ with $||\boldsymbol{w}||_q \le 1$ is the fraction of examples $(\boldsymbol{x}, y) \in \mathcal{X} \times \{-1, +1\}$ in the training set whose margin $y\, \boldsymbol{w} \cdot \hat{\boldsymbol{x}}$ is at most s, as a function of $s \in [-1, +1]$. For an N-class problem solved via the one-versus-rest scheme, it is natural to define the margin as the difference between the output of the classifier associated with the correct label and the maximal output of any other classifier. This value lies in $[-1, +1]$ once weight vectors and instances are properly normalized. Also, the margin is positive if and only if the example is correctly classified.

In our experiments no special attention has been paid to tune scaling factors and/or noise-control parameters. As far as parameters B and C is concerned, we have set $C = \sqrt{2}$ (as in Theorem 3, part 1) and $B = \frac{1}{\alpha}$, where α is chosen in the set $\{1.0,\ 0.95,\ 0.9,\ 0.8,\ 0.5\}$. Notice that with this parameterization the weight update condition $y_t\, \boldsymbol{w}_k \cdot \hat{\boldsymbol{x}}_t \le (1-\alpha)\,\gamma_k$ in Figure 1 actually becomes $y_t\, \boldsymbol{w}_k \cdot \hat{\boldsymbol{x}}_t \le \beta\, \frac{\sqrt{p-1}}{\sqrt{k}}$, where $\beta \in \{0,\ 0.0526,\ 0.111,\ 0.25,\ 1\}$.

		1 Epoch		*2 Epochs*		*3 Epochs*	
		TestErr	Correct	TestErr	Correct	TestErr	Correct
Perceptron	last	6.03%	1148	5.45%	1395	5.30%	1515
	avg	5.49%	1148	5.24%	1395	5.00%	1515
$\text{ALMA}_2(1.0)$	last	6.42%	1250	5.62%	1557	5.38%	1713
	avg	5.51%	1250	4.93%	1557	5.06%	1713
$\text{ALMA}_2(0.95)$	last	5.72%	1752	5.05%	2087	4.85%	2239
	avg	5.18%	1752	4.68%	2087	4.73%	2239
$\text{ALMA}_2(0.9)$	last	5.43%	2251	5.06%	2606	4.90%	2746
	avg	5.06%	2251	4.72%	2606	4.77%	2746

Table 1: Experimental results on USPS database. We used Gaussian kernels with width $\sigma = 3.5$. "TestErr" denotes the fraction of misclassified patterns in the test set, while "Correct" denotes the total number of training corrections for the 10 labels. $\text{ALMA}_2(\alpha)$ is shorthand for $\text{ALMA}_2(\alpha; \frac{1}{\alpha}, \sqrt{2})$. Recall that averaging takes place during the testing phase. Thus the number of corrections of "last" is the same as the number of corrections of "avg".

Below we are using the shorthand $\text{ALMA}_p(\alpha)$ to denote $\text{ALMA}_p(\alpha; \frac{1}{\alpha}, \sqrt{2})$. Thus, for example, $\text{ALMA}_p(1.0)$ denotes $\text{ALMA}_p(1; 1, \sqrt{2})$.

We made no preprocessing on the data (beyond the implicit preprocessing performed by the kernels). All our experiments have been run on a PC with a single Pentium® III MMX processor running at 447 Mhz. The running times we will be mentioning are measured on this machine.

The rest of this section describes the experiments in some detail.

4.1 Experiments with USPS dataset

The USPS (US Postal Service) dataset has 7291 training patterns and 2007 test patterns. Each pattern is a 16×16 vector representing a digitalized image of a handwritten digit, along with a {0,1,...,9}-valued label. The components of such vectors lie in $[-1, +1]$.

This is a well-known SVM benchmark. The accuracy results achieved by SVM range from 4.2% (obtained by Cortes and Vapnik, 1995, after suitable data smoothing) to 4.4% (reported by Schölkopf et al., 1999) to 4.7% (reported by Platt et al., 1999, with no data preprocessing). The best accuracy results we are aware of are those obtained by Simard, et al. (1993). They yield a test error of 2.7% by using a notion of distance between patterns that encodes specific prior knowledge about OCR problems, such as invariance to translation and rotation.

We ran both the Perceptron algorithm and ALMA_2. Following Schölkopf et al. (1997, 1999), Platt et al. (1999) and Friess et al. (1998), we used the Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{y}||_2^2}{2\,\sigma^2}\right)$. To choose the best width σ, we ran the Perceptron algorithm and $\text{ALMA}_2(1.0)$ for one epoch. We used 5-fold cross validation on the training set across the range $[0.5, 10.0]$ with step 0.5. The best σ for both algorithms turned out to be $\sigma = 3.5$. In Tables 1 and 2 only the best ($\sigma = 3.5$) results are displayed. Observe that

	digit	0	1	2	3	4	5	6	7	8	9
ALMA$_2(1.0)$											
1 Epoch	Corr	99	46	129	147	142	156	101	102	169	159
	SV	99	46	129	147	142	156	101	102	169	159
2 Epochs	Corr	121	58	164	183	174	195	127	128	206	201
	SV	121	55	159	182	168	193	126	124	202	195
3 Epochs	Corr	138	67	174	204	193	208	145	137	228	219
	SV	138	61	173	197	184	205	142	132	224	210
ALMA$_2(0.95)$											
1 Epoch	Corr	149	79	192	203	190	214	148	144	226	207
	SV	149	79	192	203	190	214	148	144	226	207
2 Epochs	Corr	176	94	226	242	226	255	174	168	277	249
	SV	175	89	224	238	221	253	173	163	272	243
3 Epochs	Corr	186	105	235	256	243	273	188	184	297	272
	SV	185	94	233	250	234	268	184	173	288	254
ALMA$_2(0.9)$											
1 Epoch	Corr	200	110	246	255	243	279	193	182	287	256
	SV	200	110	246	255	243	279	193	182	287	256
2 Epochs	Corr	224	128	284	294	286	320	224	209	331	306
	SV	223	123	282	290	280	314	220	201	323	294
3 Epochs	Corr	230	137	294	312	300	334	235	226	349	329
	SV	229	126	288	305	289	328	227	214	337	309
SVM	(SV)	219	91	316	309	288	340	213	206	304	250

Table 2: Experimental results on USPS database. “Corr” denotes the total number of training corrections for the 10 labels, while “SV” denotes the number of “support vectors” for the 10 labels. ALMA$_2(\alpha)$ is shorthand for ALMA$_2(\alpha; \frac{1}{\alpha}, \sqrt{2})$. Clearly, for one epoch “Corr” = “SV”.

using a Gaussian kernel makes the normalization of instances $\hat{\boldsymbol{x}} = \boldsymbol{x}/||\boldsymbol{x}||_2$ immaterial, since $K(\boldsymbol{x}, \boldsymbol{x}) = 1$ for any $\boldsymbol{x}$.

Table 1 gives test error and number of corrections for the Perceptron algorithm and ALMA_2 with different values of α. Table 2 gives statistics for ALMA_2 on the ten digits. Here "Corr" denotes the number of corrections while "SV" denotes the number of "support vectors", i.e., the number of examples that are actually involved in computing the prediction function for each of the ten classes. For the sake of comparison, we also give the number of support vectors yielded by SVM, as reported by Schölkopf et al. (1999). The standard deviations related to our averages are reasonably small; those concerning test errors are about 0.12%.

On this dataset $\text{ALMA}_2(1.0)$ and the Perceptron algorithm perform comparably. The accuracy of $\text{ALMA}_2(\alpha)$ improves significantly when we shrink the value of α, whereas the computed solution gets less and less sparse.

As in the experiments performed by Freund and Schapire (1999) and Li and Long (1999), the accuracy of the classifiers tends to get better as we increase the number of training epochs. However, training ALMA_2 "avg" for more than two epochs seems to hurt performance somewhat. This might be due to the fast convergence of this algorithm (notice that the accuracy obtained by SVM is quite close).

We found the accuracy of 5.06% for $\text{ALMA}_2(0.9)$ "avg" fairly remarkable, considering that it has been obtained by sweeping through the examples just once for each of the ten classes. Indeed, the algorithm is quite fast: training for one epoch the ten binary classifiers of $\text{ALMA}_2(0.9)$ takes on average only 6.5 minutes.

The reader might want to compare this performance to the similar accuracy of 5.00% achieved by the "average" Perceptron algorithm run for three epochs. Despite Perceptron's solution is sparser than $\text{ALMA}_2(0.9)$'s, it is worth saying that running Perceptron for three epochs takes about twice as long as training $\text{ALMA}_2(0.9)$ for one epoch.

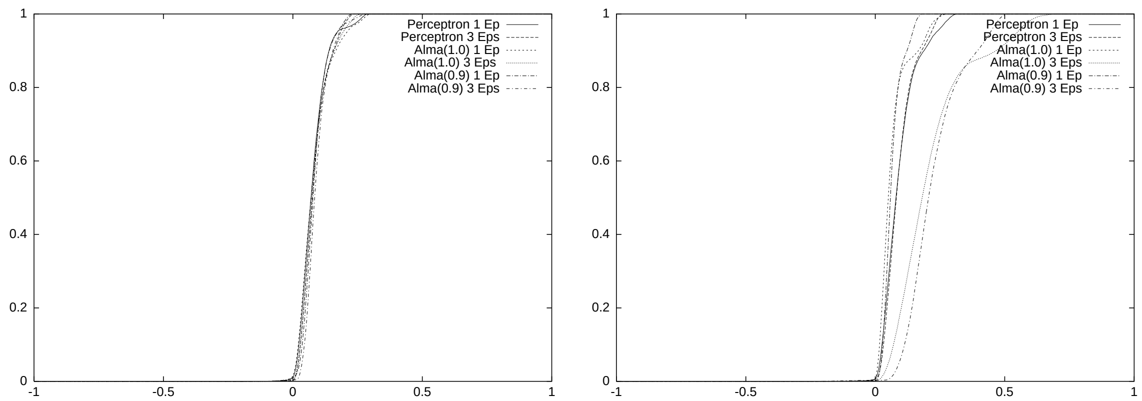
We plot in Figure 3 some of the margin distribution graphs obtained. The value of the curves at a given point $s \in [-1, +1]$ gives the fraction of patterns in the training set whose margin (after training) is at most s. $\text{ALMA}_2(0.9)$ tends to increase the number of examples with a strictly positive margin. This is actually more evident with the "average" variant (plots on the right) than with the "last" variant (plots on the left).

4.2 Experiments with the MNIST dataset

Each example in MNIST dataset is a 28×28 matrix, along with a {0,1,...,9}-valued label. Each entry in this matrix is a value in {0,1,...,255}, representing a grey level. The database has 60000 training examples and 10000 test examples.

The best accuracy results for this dataset are those obtained by Le Cun et al. (1995) through boosting on top of the neural net LeNet4. They reported a test error rate of 0.7%. A soft margin SVM achieved an error rate of 1.1% (Cortes and Vapnik, 1995).

In this subsection we are comparing to SVM, the Perceptron algorithm and the Perceptron-like algorithm ROMMA (Li and Long, 1999). We followed closely the experimental setting described by Cortes and Vapnik (1995), Freund and Schapire (1999), Li and Long (1999). We used a polynomial kernel K of the form $K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y})^d$. At the time of writing the conference version of this paper (Gentile, 2001) we set the degree d of



"last" "average"

Figure 3: Some of the margin distribution functions yielded by the Perceptron algorithm and by ALMA$_2$, run for 1 and 3 epochs on USPS dataset.

the kernel to 4. According to Freund and Schapire (1999) this choice was best. The same choice was made by Cortes and Vapnik (1995) and Li and Long (1999). Later on we found that significant improvements could be obtained by a larger d.

We give results for ALMA$_2$ with $d = 4, 5, 6$. We have not investigated any careful tuning of scaling factors. In particular, we have not determined the best instance scaling factor s for our algorithm (this corresponds to using the kernel $K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y}/s)^d)$. In our experiments we set $s = 255$. This was actually the best choice made by Li and Long (1999) for the Perceptron algorithm.

The experimental results are given in Tables 3 and 4. Table 3 gives test error and number of corrections for the Perceptron algorithm, ROMMA and ALMA$_2(\alpha)$ with $\alpha = 1.0,\ 0.9,\ 0.8$.

The first four rows of Table 1 summarize some of the results obtained by Freund and Schapire (1999), Li and Long (1999) and Li (2000). The first two rows refer to the Perceptron algorithm, while the third and the fourth rows refer to the original ROMMA[9] and the best noise-controlled version of ROMMA, called "aggressive ROMMA". Both the Perceptron algorithm and ROMMA have been run with a degree 4 polynomial kernel. Our own experimental results are given in the subsequent rows.

Table 4 is analogous to Table 2 and refers to ALMA$_2$ with a degree 4 polynomial kernel. In the last row of Table 4 we give the number of support vectors yielded by the soft-margin SVM employed by Cortes and Vapnik (1995). Again, the standard deviations about the averages we report in these tables are not large. Those concerning test errors range in (0.03%, 0.09%).

Among these Perceptron-like algorithms, ALMA$_2$ "avg" seems to be the most accurate. Again, we would like to emphasize the good accuracy performance yielded by ALMA$_2$ on the first epoch. Notice that if $d = 6$ ALMA$_2(0.9)$ "avg" gets 1.48%.

9. According to Li and Long (1999), ROMMA's last hypothesis seems to perform better than ROMMA's voted hypothesis.

		1 Epoch		2 Epochs		3 Epochs	
		TestErr	Correct	TestErr	Correct	TestErr	Correct
Perceptron	last	2.71%	7901	2.14%	10421	2.03%	11787
	voted	2.23%	7901	1.86%	10421	1.76%	11787
ROMMA (last)		2.48%	7963	1.96%	9995	1.79%	10971
agg-ROMMA (last)		2.05%	30088	1.76%	44495	1.67%	58583
ALMA$_2$(1.0)							
$d=4$	last	2.52%	7454	2.01%	9658	1.86%	10934
	avg	1.77%	7454	1.52%	9658	1.47%	10934
$d=5$	last	2.40%	7105	1.86%	9048	1.65%	10004
	avg	1.67%	7105	1.46%	9048	1.39%	10004
$d=6$	last	2.35%	7001	1.83%	8782	1.67%	9633
	avg	1.64%	7001	1.48%	8782	1.36%	9633
ALMA$_2$(0.9)							
$d=4$	last	2.10%	9911	1.74%	12711	1.64%	14244
	avg	1.69%	9911	1.49%	12711	1.40%	14244
$d=5$	last	1.93%	10373	1.64%	12700	1.49%	13820
	avg	1.59%	10373	1.39%	12700	1.32%	13820
$d=6$	last	1.84%	11652	1.53%	13712	1.45%	14598
	avg	1.48%	11652	1.32%	13712	1.27%	14598
ALMA$_2$(0.8)							
$d=4$	last	1.98%	12810	1.72%	16464	1.60%	18528
	avg	1.68%	12810	1.44%	16464	1.35%	18528

Table 3: Experimental results on MNIST database. The results have been obtained through the polynomial kernel $K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y}/255)^d$. The Perceptron algorithm and ROMMA use $d = 4$, while ALMA$_2$ uses $d = 4, 5, 6$. "TestErr" denotes the fraction of misclassified patterns in the test set, while "Correct" denotes the total number of training corrections for the 10 labels.

ALMA$_2$ is quite fast. Training for one epoch the ten binary classifiers of ALMA$_2$(1.0) with $d = 4$ takes on average 2.3 hours and the corresponding testing time is on average about 40 minutes; training for one epoch the ten binary classifiers of ALMA$_2$(0.9) with $d = 6$ takes on average 4.3 hours, while testing takes on average 1.2 hours.

There seems to be no big difference in accuracy between ALMA$_2$(0.9) and ALMA$_2$(0.8) when $d = 4$. This suggested us not to run ALMA$_2$(0.8) with $d > 4$. Also, we have not run ALMA$_2$ for more than 3 epochs. But it seems reasonable to expect the accuracy of ALMA$_2$(0.9) "avg" and ALMA$_2$(0.8) "avg" to get closer and closer to the one achieved by SVM.

4.3 Experiments with UCI Letter dataset

The UCI Letter dataset has 20000 patterns divided into 26 classes (the letters 'A' through 'Z'). The instance vectors have 16 integer attributes (statistical moments and edge counts)

	digit	0	1	2	3	4	5	6	7	8	9
$\text{ALMA}_2(1.0)$											
1 Epoch	Corr	441	373	738	914	715	792	504	727	1076	1174
	SV	441	373	738	914	715	792	504	727	1076	1174
2 Epochs	Corr	572	501	952	1181	915	1016	654	933	1412	1522
	SV	549	467	932	1134	887	993	635	902	1353	1463
3 Epochs	Corr	642	583	1071	1335	1024	1143	719	1076	1604	1737
	SV	616	517	1016	1237	978	1087	697	995	1481	1616
$\text{ALMA}_2(0.9)$											
1 Epoch	Corr	611	492	998	1206	970	1065	695	948	1420	1506
	SV	611	492	998	1206	970	1065	695	948	1420	1506
2 Epochs	Corr	776	656	1269	1548	1212	1353	876	1224	1829	1968
	SV	741	597	1205	1434	1153	1282	821	1139	1719	1832
3 Epochs	Corr	861	753	1401	1721	1354	1489	963	1384	2073	2245
	SV	801	647	1301	1575	1247	1394	896	1252	1857	1986
$\text{ALMA}_2(0.8)$											
1 Epoch	Corr	814	637	1303	1534	1271	1395	906	1214	1819	1917
	SV	814	637	1303	1534	1271	1395	906	1214	1819	1917
2 Epochs	Corr	1023	841	1652	1986	1585	1755	1152	1586	2370	2514
	SV	953	742	1528	1817	1476	1627	1064	1423	2134	2273
3 Epochs	Corr	1140	965	1846	2231	1768	1956	1279	1788	2684	2871
	SV	1026	803	1642	1951	1593	1744	1135	1532	2318	2438
SVM	(SV)	1379	989	1958	1900	1224	2024	1527	2064	2332	2765

Table 4: Experimental results on MNIST database, when using a polynomial kernel of degree $d = 4$. "Corr" denotes the total number of training corrections for the 10 labels, while "SV" denotes the number of "support vectors" for the 10 labels.

		1 Epoch TestErr	Correct	*2 Epochs* TestErr	Correct	*3 Epochs* TestErr	Correct
Perceptron	last	6.18%	5010	4.50%	6131	4.15%	7001
	avg	4.83%	5010	3.70%	6131	3.33%	7001
$\text{ALMA}_2(1.0)$	last	7.00%	5484	4.92%	6685	4.45%	7194
	avg	4.82%	5484	3.87%	6685	3.47%	7194
$\text{ALMA}_2(0.9)$	last	4.90%	8312	3.85%	9644	3.50%	10178
	avg	3.85%	8312	3.10%	9644	3.02%	10178
$\text{ALMA}_2(0.8)$	last	4.20%	11258	3.55%	13003	3.27%	13673
	avg	3.60%	11258	2.97%	13003	2.80%	13673

Table 5: Experimental results on UCI Letter database. We used a "poly-Gaussian" kernel (see main text). "TestErr" denotes the fraction of misclassified patterns in the test set, while "Correct" denotes the total number of training corrections for the 26 labels.

extracted from raster scan images of machine printed letters of 20 different fonts. The attributes have values in $\{0, 1, ..., 15\}$. A standard split is to consider the first 16000 patterns as training set and the remaining 4000 patterns as test set.

The best accuracy results we are aware of are those obtained by Schwenk and Bengio (2000) by boosting suitable neural network architectures. They reported an error rate of 1.5%. According to Platt et al. (1999), a Gaussian kernel SVM achieves 2.2%. This accuracy difference might actually be due to different preprocessing.

This is a dataset where Perceptron-like algorithms such as ALMA_2 did not work as well as we would have liked. We ran both the Perceptron algorithm and ALMA_2. Both algorithms exhibited a somewhat slow convergence. Besides, they both failed to converge to SVM's accuracy level. We tried to speed up convergence by using a "poly-Gaussian" kernel of the form $K(\boldsymbol{x}, \boldsymbol{y}) = \left(1 + \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{y}||_2^2}{2\,\sigma^2}\right)\right)^d$. This kernel corresponds to a linear combination of d Gaussian kernels with different width parameters σ. We set $d = 5$ to make the kernel flexible enough. Again, in order to determine the best σ, we ran the Perceptron algorithm and $\text{ALMA}_2(1.0)$ for one epoch, using 4-fold cross-validation on the training set across the range [0.5, 10.0], with step 0.5. The best σ for Perceptron was 4.0, while the best σ for ALMA_2 turned out to be 3.0. The experiments are summarized in Table 5, where only the best results are shown.

The conclusions we can draw from this table are similar to those for Table 1 and Table 3. The main difference is that the test error achieved by ALMA_2 after one epoch ($\text{ALMA}_2(0.8)$ "avg" achieves 3.60%) is significantly worse than SVM's (2.2%). The accuracy of ALMA_2 tends to improve after the first epoch (it reaches 2.80% after three epochs), but it stabilizes around 2.7%, no matter how many epochs one trains the algorithm for. We observed a similar behavior with the Perceptron algorithm (with an even slower convergence). This phenomenon might be due to the lack of SVM's bias term.

As far as running time is concerned, training $\text{ALMA}_2(1.0)$ for one epoch takes about 2.5 minutes, while training $\text{ALMA}_2(0.8)$ for one epoch takes about 5.2 minutes.

4.4 Experiments with ALMA_p on artificial datasets

We tested ALMA_p, $p \geq 2$, without kernels on medium-size artificial datasets. The datasets are about binary classification tasks and have been generated at random according to the following rules. We first generated a target vector $\boldsymbol{u} \in \{-1, 0, +1\}^{300}$, where the first s components are selected independently at random in $\{-1, +1\}$ and the remaining $300 - s$ components are 0. The value s is intended as a measure of the *sparsity* of target vector $\boldsymbol{u}$. In our experiments we set $s = 3,\ 10,\ 100,\ 300$. We also added labelling noise with rate ϵ. In our experiments $\epsilon = 0.0,\ 0.05,\ 0.10,\ 0.15$. For a given target vector $\boldsymbol{u}$ and a given noise rate ϵ, we randomly generated 10000 training examples and 10000 test examples. The instance vectors $\boldsymbol{x}_t$ have 300 components with values chosen in $[-1, +1]$. The training set is generated as follows. We picked $\boldsymbol{x}_t \in [-1, +1]^{300}$ at random. If $\boldsymbol{u} \cdot \boldsymbol{x}_t \geq 1$ then a $+1$ label is associated with $\boldsymbol{x}_t$. If $\boldsymbol{u} \cdot \boldsymbol{x}_t \leq -1$ then a -1 label is associated with $\boldsymbol{x}_t$. The labels so obtained are then flipped with probability ϵ. If $|\boldsymbol{u} \cdot \boldsymbol{x}_t| < 1$ then $\boldsymbol{x}_t$ is rejected and a new vector $\boldsymbol{x}_t$ is drawn. The test set instances $\boldsymbol{x}_t$ are again chosen at random in $[-1, +1]^{300}$, the corresponding labels equal[10] $\text{sign}(\boldsymbol{u} \cdot \boldsymbol{x}_t)$. We did not force a large margin on the test set.

On each of these $4 \times 4 = 16$ datasets we ran $\text{ALMA}_p(\alpha)$ (both "last" and "avg") for one epoch, with $p = 2,\ 6,\ 10$ and $\alpha = 1.0,\ 0.9,\ 0.8,\ 0.5$. The accuracy results (test errors) are shown in Table 6. These experiments had the purpose of investigating the behavior of ALMA_p on extreme scenarios. The differences in performance are big and sometimes even huge. In Table 6 we report what we believe are some of the most interesting results. We picked 6 out of the 16 datasets. The columns are marked according to the values of ϵ and s.

On sparse target datasets ($s = 3$ in Table 6) ALMA_6 and ALMA_{10} largely outperform ALMA_2. On these datasets the accuracy of all algorithms improves as α is made smaller. Correspondingly, the number of corrections (not shown in Table 6) increases. Like in the experiments of the previous subsections, there is a natural trade-off between the number of corrections the algorithms make and the accuracy of the resulting hypotheses. To give an idea of this trade-off, we report three results: on the "$\epsilon = 0.0,\ s = 3$" dataset $\text{ALMA}_2(1.0)$ makes on average 142 corrections, while $\text{ALMA}_2(0.5)$ makes 2720 corrections; on the same dataset, $\text{ALMA}_{10}(1.0)$ makes on average only 14 corrections, whereas $\text{ALMA}_{10}(0.5)$ makes 1050; on the "$\epsilon = 0.15,\ s = 3$" dataset $\text{ALMA}_{10}(1.0)$ makes on average 2656 corrections while $\text{ALMA}_{10}(0.5)$ makes on average 3594 corrections. The reader might want to compare the accuracy results for these three cases. For any given value of α, there is essentially no difference between $\text{ALMA}_{10}(\alpha)$ and $\text{ALMA}_6(\alpha)$ (the test errors shown in Table 6 are meaningful up to about 1%).

On dense target datasets ($s = 300$) $\text{ALMA}_p(\alpha)$ with $p = 2$ is best. Again, accuracy improves by shrinking α, but it degrades as we increase p.

In Figure 4 we plot the margin distribution graphs obtained on 4 of the 16 datasets generated. In these plots we put emphasis on the comparison between the two extreme cases $\alpha = 1.0$ and $\alpha = 0.5$. The performance gap between $\text{ALMA}_2(\alpha)$ and $\text{ALMA}_6(\alpha)$ on the two "$s = 3$" datasets (plots on the left) is clearly reflected by the different behavior of the corresponding margin distribution functions. The plots on the right, on the other

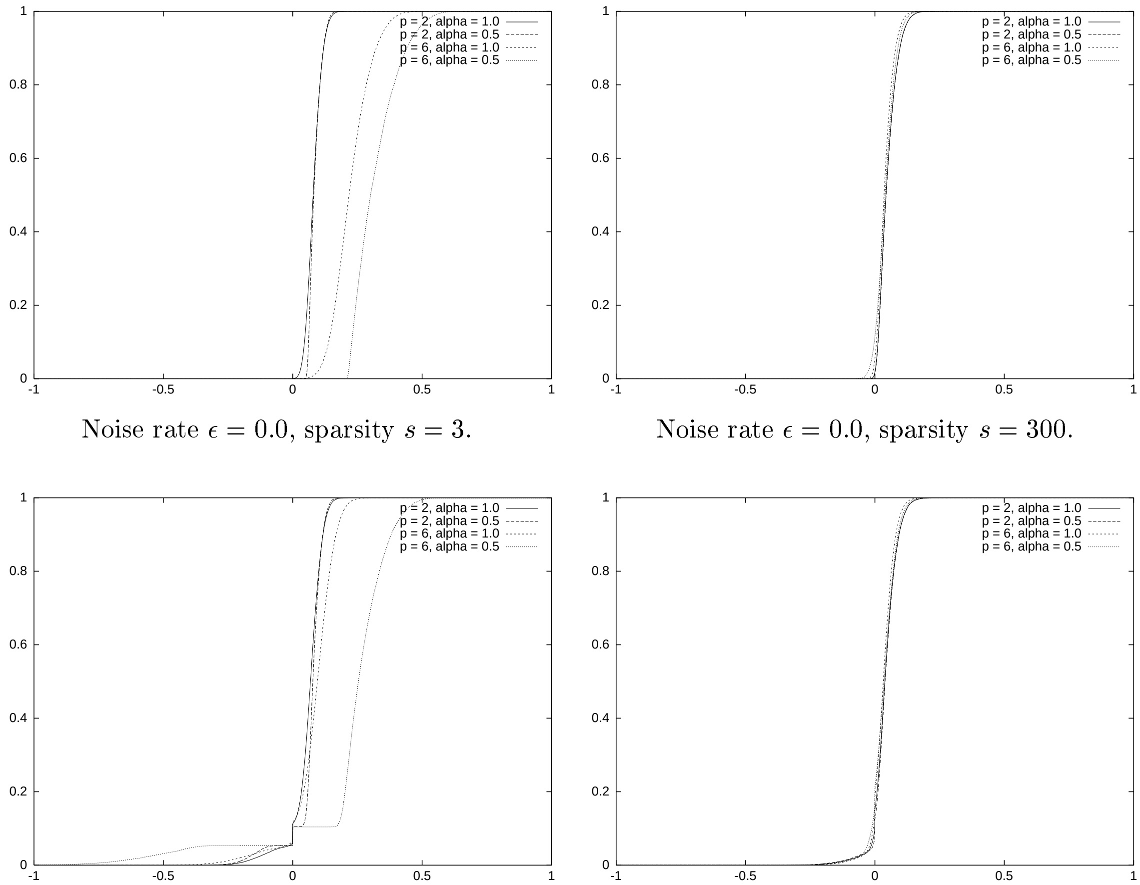
10. Clearly, the absence of noise in the test examples is not a real loss of generality here, as an independent noise rate ϵ in the test set would essentially be added to the noise-free test error rates of the algorithms.

dataset	$\epsilon = 0.0$ $s = 3$ (TestErr)	$\epsilon = 0.0$ $s = 300$ (TestErr)	$\epsilon = 0.1$ $s = 3$ (TestErr)	$\epsilon = 0.1$ $s = 300$ (TestErr)	$\epsilon = 0.15$ $s = 3$ (TestErr)	$\epsilon = 0.15$ $s = 300$ (TestErr)
ALMA$_2$(1.0)						
last	11.9%	8.3%	26.6%	17.2%	29.6%	22.1%
avg	10.9%	5.0%	16.6%	10.6%	18.7%	12.5%
ALMA$_2$(0.8)						
last	7.0%	7.8%	21.4%	13.6%	23.7%	16.2%
avg	4.9%	4.4%	11.5%	8.0%	14.0%	9.7%
ALMA$_2$(0.5)						
last	4.9%	9.0%	12.7%	11.5%	15.0%	13.6%
avg	2.5%	4.9%	5.4%	7.1%	7.1%	8.2%
ALMA$_6$(1.0)						
last	9.6%	12.0%	28.4%	18.8%	28.6%	22.5%
avg	8.5%	9.4%	17.4%	14.7%	20.0%	17.1%
ALMA$_6$(0.8)						
last	1.5%	12.9%	17.0%	16.0%	19.5%	17.5%
avg	1.2%	8.7%	8.8%	12.2%	11.0%	14.5%
ALMA$_6$(0.5)						
last	0.5%	18.7%	5.6%	20.7%	7.8%	22.0%
avg	0.3%	15.9%	2.2%	18.6%	3.1%	20.2%
ALMA$_{10}$(1.0)						
last	10.7%	13.9%	26.1%	21.9%	30.2%	24.9%
avg	8.3%	14.1%	16.9%	18.0%	18.8%	19.8%
ALMA$_{10}$(0.8)						
last	1.2%	15.3%	16.1%	19.3%	18.3%	21.0%
avg	0.9%	13.8%	7.4%	16.9%	9.3%	19.0%
ALMA$_{10}$(0.5)						
last	0.8%	25.7%	4.6%	27.3%	6.8%	28.6%
avg	0.5%	25.0%	1.3%	26.2%	1.9%	26.9%

Table 6: Results of experiments on artificially generated datasets. Recall that ϵ denotes the amount of labelling noise, while s is the number of nonzero components of target vector $\boldsymbol{u}$.

hand, are somewhat less informative. When learning a dense ("$s = 300$") target, a single training epoch is probably not sufficient to differentiate algorithms' performance through their margin properties.

In all our experiments the average hypothesis vector "avg" is substantially more accurate than "last". This tends to be even more evident on the noisy datasets "$\epsilon = 0.15$". Again, such a big performance difference after just one training epoch is hardly explained via a margin analysis. Figure 5 contains a somewhat disappointing attempt to relate the performance gap between "last" and "avg" to different margin properties. A better theoretical explaination for the resistance of an "avg"-like classifier to labelling noise is provided by Servedio (1999).



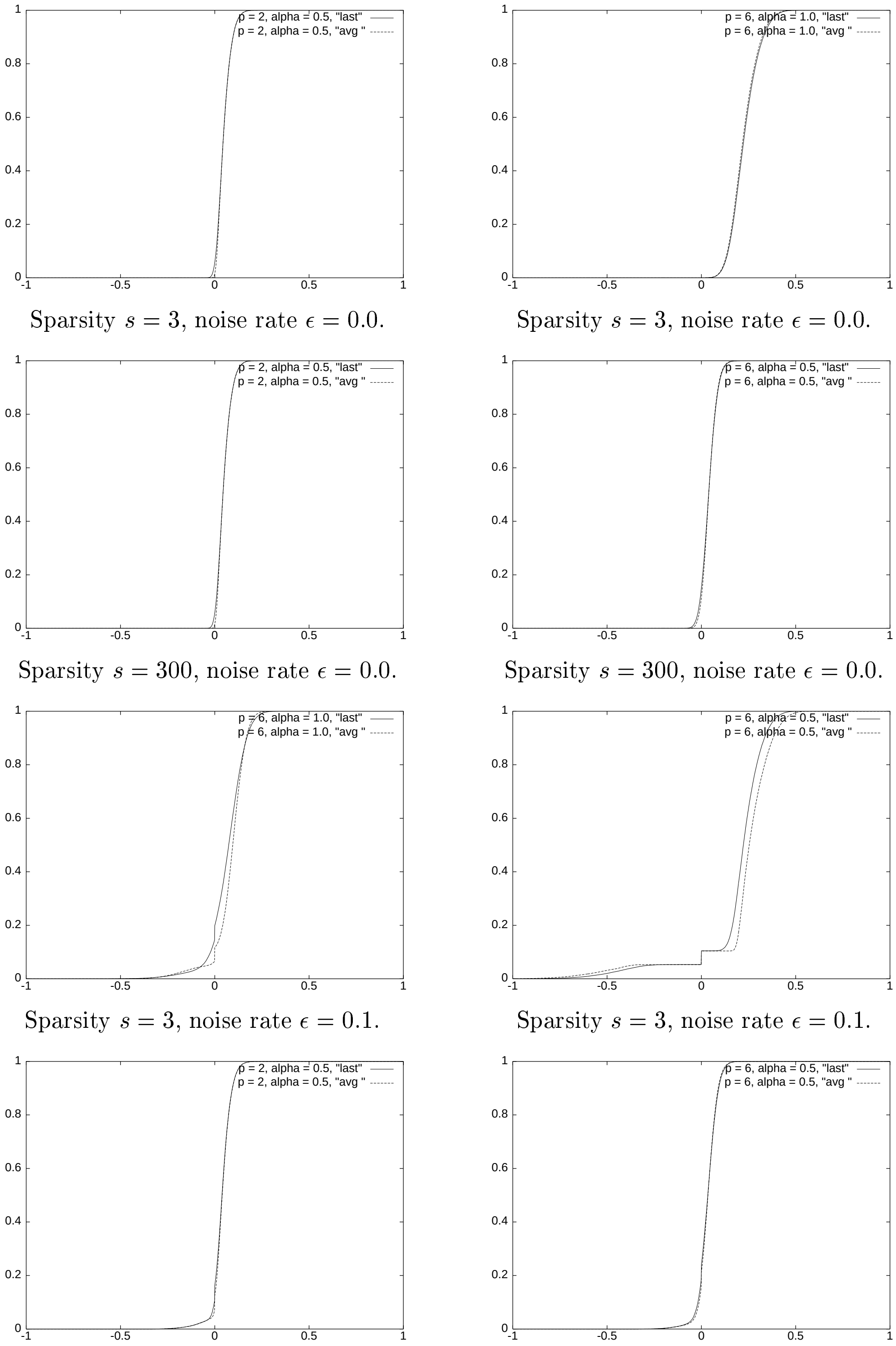
Noise rate $\epsilon = 0.0$, sparsity $s = 3$.

Noise rate $\epsilon = 0.0$, sparsity $s = 300$.

Noise rate $\epsilon = 0.1$, sparsity $s = 3$.

Noise rate $\epsilon = 0.1$, sparsity $s = 300$.

Figure 4: Margin distributions yielded by $\text{ALMA}_p(\alpha)$ "avg" with $p = 2, 6$ and $\alpha = 1.0, 0.5$, run for one epoch on 4 of the 16 datasets generated.



Sparsity $s = 3$, noise rate $\epsilon = 0.0$. Sparsity $s = 3$, noise rate $\epsilon = 0.0$.

Sparsity $s = 300$, noise rate $\epsilon = 0.0$. Sparsity $s = 300$, noise rate $\epsilon = 0.0$.

Sparsity $s = 3$, noise rate $\epsilon = 0.1$. Sparsity $s = 3$, noise rate $\epsilon = 0.1$.

Sparsity $s = 300$, noise rate $\epsilon = 0.1$. Sparsity $s = 300$, noise rate $\epsilon = 0.1$.

Figure 5: Margin distributions yielded by $\text{ALMA}_p(\alpha)$ with $p = 2, 6$ and $\alpha = 1.0, 0.5$, run for one epoch on 4 of the 16 datasets generated. The plots compare "last" and "avg" variants of ALMA_p.

4.5 Discussion and summary

Our empirical results show how accuracy and running time (as well as sparsity) can be traded-off against each other in a transparent way. In all our experiments the "avg" hypothesis significantly outperforms the "last" hypothesis, though the accuracy levels of our algorithm are in general slightly inferior to those achieved by SVM. On the other hand, our algorithm is quite faster and easier to implement than previous implementations of SVM, such as those given by Platt (1998), Joachims (1998) and Friess et al. (1998).

One of the most relevant features of ALMA_2 is that its approximate solution relies on fewer support vectors than SVM's solution. As a matter of fact, it often happens that ALMA_2 yields a good test error after just one training epoch. This obviously makes our algorithm attractive for an on-line learning framework (notice that the N binary classifiers in the one-versus-rest scheme can be trained "in parallel"). Still, a significant accuracy improvement can be observed when we run Perceptron-like algorithms for more than one epoch. For $\text{ALMA}_2(\alpha)$ with $\alpha < 1$ this improvement might be explained by observing that sweeping through the training set more than once tends to increase the margin over the training examples (this is also shown somewhat by the margin distributions plotted in Figure 3). However, we do not have an explaination for this phenomenon when we run $\text{ALMA}_2(1.0)$ or the simple Perceptron algorithm.

Finally, the experimental results in section 4.4 seem to be significantly captured by our theoretical analysis (Theorem 3). On sparse target datasets, the performance gap between $\text{ALMA}_2(\alpha)$ and $\text{ALMA}_p(\alpha)$ with $p > 2$ is big. This gap tends to be magnified as α gets smaller. There is again a trade-off between accuracy achieved on the test set and number of corrections made during the training phase. This trade-off is ruled by the interplay between the value of p in ALMA_p and the sparsity of the underlying target vector.

5. Conclusions and open problems

We have introduced a new incremental learning algorithm, called ALMA_p, which approximates the maximal p-norm margin hyperplane for a set of linearly separable data. ALMA_p avoids quadratic or higher-order programming methods. Unlike previous approaches (Cortes and Vapnik, 1995; Friess et al., 1998; Joachims, 1998 and Platt, 1998), our algorithm works directly with (an approximation to) the primal maximal margin problem, instead of its dual. Via this approach, we are able to avoid computationally intensive mathematical programming methods. ALMA_p is more similar to algorithms such as ROMMA (Li and Long, 1999) and the one analyzed by Kowalczyk (1999) and Keerthi et al. (1999). However, unlike those algorithms, ALMA_p remains computationally efficient when measuring the margin through a generic norm p. The theoretical properties of ALMA_p have been given in Theorem 3. We have proven upper bounds on the number of corrections in both the separable and the non-separable cases. ALMA_2 is a Perceptron-like algorithm. All its operations essentially involve only dot products. Hence one can replace those dot products by kernel dot products. We have tested ALMA_2 with kernel functions on three OCR benchmarks and have shown that our algorithm has very interesting accuracy levels after just one training epoch. Compared to SVM's, ALMA_2's approximate solution is a bit less accurate, but it is also significantly sparser. ALMA_2's training time is substantially shorter than SVM's. This makes our algorithm attractive for learning very large datasets. We have also tested ALMA_p with $p \geq 2$ on artificial datasets. Setting $p > 2$ is useful when learning sparse target vectors (this is not rare in text processing tasks). In such problems the practical superiority of ALMA_p with $p > 2$ over ALMA_2 is largely predicted by our theoretical analysis.

There are many directions in which this work could be extended. In the following we briefly discuss four of them.

1. It is not clear to us whether the convergence analysis provided by Theorem 3, part 1, is optimal. In fact, in the case when the margin γ^* is known to the algorithm we can prove a bound on $|\mathcal{M}|$ which scales with α as $\frac{1}{\alpha} \ln \frac{1}{\alpha}$ (instead of $\frac{1}{\alpha^2}$ occurring in both the bound of Theorem 3 and in the analysis by Kowalczyk, 1999, and Li and Long, 1999). We should stress that when γ^* is known a direct on-line analysis based on Bregman divergences (such as the one performed by Auer et al., 2001) still gives dependence $\frac{1}{\alpha^2}$. This is actually the reason why in the proof of Theorem 3 we do not use a Bregman divergence measure of progress.

2. We would like to see if a variant of ALMA_p exists which computes after a finite number of steps an α-approximation to the fixed (non-zero) threshold maximal margin hyperplane. This is an important question, as a relevant practical drawback of our algorithm is its inability to handle SVM's bias term.

3. Running Perceptron-like algorithms, such as Freund and Schapire's voted Perceptron algorithm, Li and Long's ROMMA, and our ALMA_p, for more than one epoch tends to increase performance. This phenomenon does not seem to fully depend on the margin properties of the algorithms. We would like to gain a deep theoretical understanding of this experimental evidence.

4. The way we handle multiclass classification problems is to reduce to a set of binary problems. As a matter of fact, natural multiclass versions Perceptron-like algorithms do exist (e.g., Duda and Hart, 1973, Chap. 5). As in the one-versus-rest scheme, these algorithms associate one weight vector classifier with each class and predict according to the maximum output of these classifiers. Again, margin is defined as the difference between the output of the classifier associated with the correct label and the output of any other classifier. However, only two weight vectors are updated within a correction. These are just the two weight vectors involved in computing the above margin. These multiclass algorithms can be motivated (Gentile and Warmuth, 2001) within Kivinen and Warmuth's multidimensional regression framework (Kivinen and Warmuth, 1998). The on-line analysis of such algorithms is a fairly straightforward extension of the analysis for the binary case. We are not aware of any thorough experimental investigation of these multiclass on-line algorithms. It would be interesting to see how they do compare in practice to the other multiclass methods available in the literature.

Acknowledgments

We would like to thank to Nicolò Cesa-Bianchi, Nigel Duffy, Dave Helmbold, Adam Kowalczyk, Yi Li, Nick Littlestone and Dale Schuurmans for valuable conversations and email exchange. We would also like to thank the anonymous reviewers for their comments which helped us to improve the presentation of this paper. The author has been partially supported by ESPRIT Working Group EP 27150, Neural and Computational Learning II (NeuroCOLT II).

References

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

E. L. Allwein, R. E. Schapire and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations.* CMU, 1999.

P. Auer, N. Cesa Bianchi and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, forthcoming. Preliminary version in *Proceedings of 13th Annu. Conf. on Comput. Learning Theory*, pages 107–117, Palo Alto, CA, 2000.

C. Blake, E. Keogh and C. Merz. Uci repository of machine learning databases. Technical report, Dept. of Information and Computer Sciences, University of California, Irvine, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

H. D. Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34:123–135, 1962.

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

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

I. Dagan, Y. Karov and D. Roth. Mistake-driven learning in text categorization. In *Proceedings of 2nd Conference on Empirical Methods in Natural Language Processing*, pages 55–63. Association for Computational Linguistics, Somerset, New Jersey, 1997.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Journal of Machine Learning*, 37(3):277–296, 1999.

T.-T. Friess, N. Cristianini and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proceedings of 15th International Conference in Machine Learning*, pages 188–196. Morgan Kaufmann, San Mateo, CA, 1998.

C. Gentile and N. Littlestone. The robustness of the p-norm algorithms. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 1–11. ACM, 1999.

C. Gentile and M. K. Warmuth. Linear hinge loss and average margin. Unpublished. Preliminary version in *Proc. Advances in Neural Information Processing Systems 11*, pages 225–231, MIT Press, Cambridge, MA, 1999, 2001.

C. Gentile. A new approximate maximal margin classification algorithm. In *T. K. Leen, T. G. Dietterich, and V. Tresp editors, Advances in Neural Information Processing Systems 13*, pages 500–506. MIT Press, Cambridge, MA, 2001.

A. R. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. In *Proceedings of 13th International Conference in Machine Learning*, pages 182–190. Morgan Kaufmann, San Mateo, CA,, 1996.

A. J. Grove, N. Littlestone and D. Schuurmans. General convergence results for linear discriminant updates. *Journal of Machine Learning*, 43(3):173–210, 2001.

D. P. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50(3):551–573, 1995.

T. Joachims. Making large-scale support vector machines learning practical. In *B. Scholkopf, C. Burges and A. Smola (eds.): Advances in kernel methods: support vector machines.* MIT Press, Cambridge, MA, 2000.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya and K.R.K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. Technical report, Indian Institute of Science, ISL-99-03, 1999.

J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning.* forthcoming. Preliminary version in *Proc. Advances in Neural Information Processing Systems 10*, pages 287–293, MIT Press, Cambridge, MA, 1998.

J. Kivinen, M. K. Warmuth and P. Auer. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97:325–343, 1997.

A. Kowalczyk. *Maximal margin perceptron.* MIT Press, Cambridge, MA, 1999.

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

Y. Le Cun, L. J. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, S. Sackinger, P. Simard and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *Proceedings of ICANN 1995*, pages 53–60, 1995.

Y. Li. *From support vector machines to large margin classifiers.* PhD thesis, School of Computing, National University of Singapore, 2000.

Y. Li and P. Long. The relaxed online maximum margin algorithm. *Journal of Machine Learning.* forthcoming. Preliminary version in *S. A. Solla, T. K. Leen and K. R. Muller editors, Advances in Neural Information Processing Systems 12*, pages 498–504, MIT Press, Cambridge, MA, 2000.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

O. Mangasarian. Multi-surface method of pattern separation. *IEEE Trans. on Information Theory*, 14:801–807, 1968.

O. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 42(1):183–201, 1997.

P. Nachbar, J. A. Nossek and J. Strobl. The generalized adatron algorithm. In *Proceedings of 1993 IEEE ISCAS*, pages 2152–2155, 1993.

A. B. J. Novikov. On convergence proofs on perceptrons. In *Proc. of the Symposium on the Mathematical Theory of Automata, vol. XII*, pages 615–622, 1962.

E. Osuna, R. Freund and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of IEEE NNSP'97*, 1997.

J. C. Platt. *Fast training of support vector machines using sequential minimal optimization.* MIT Press, Cambridge, MA, 1998.

J. C. Platt, N. Cristianini and J. Shawe-Taylor. Large margin dags for multiclass classification. In *S. A. Solla, T. K. Leen and K. R. Muller editors, Advances in Neural Information Processing Systems 12*, pages 547–553. MIT Press, Cambridge, MA, 1999.

F. Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms.* Spartan Books, Washington, D.C., 1962.

R. E. Schapire, P. Bartlett, Y. Freund and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

B. Scholkopf, S. Mika, C.J.C. Burges, P. Knirsch, K. Muller, G. Ratsch and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Trans. on Neural Network*, 10(5): 1000–1017, 1999.

B. Scholkopf, K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. on Signal Processing*, 45:2758–2765, 1997.

H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

R. A. Servedio. On pac learning using winnow, perceptron, and a perceptron-like algorithm. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 296–307. ACM, 1999.

J. Shawe-Taylor, P. Bartlett, R. Williamson and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5):1926–1940, 1998.

P. Simard, Y. LeCun and J. Denker. Efficient pattern recognition using a new transformation distance. In *S. Hanson, J. Cowan, and L. Giles, editors, Advances in Neural Information Processing Systems, volume 5.* Morgan Kaufmann, 1993.

V. Vapnik. *Statistical learning theory.* J. Wiley & Sons, New York, 1998.