数理解析研究所講究録
第 1599 巻 2008 年 154-161

Online Learning of Approximate Maximum p-Norm Margin Classifiers with Bias

Kosuke Ishibashi, Kohei Hatano and Masayuki Takeda
Department of Informatics, Kyushu University
{k-ishi, hatano, takeda}@i.kyushu-u.ac.jp

Abstract

We propose a new online learning algorithm that provably approximates maximum margin classifiers with bias, where the margin is defined in terms of p-norm distance. Although learning of linear classifiers with bias can be reduced to learning of those without bias, the known reduction might lose the margin and slow down the convergence of online learning algorithms. Our algorithm, unlike previous online learning algorithms, implicitly uses a new reduction which preserves the margin and avoids such possible deficiencies. Our preliminary experiments shows that our algorithm runs much faster than previous algorithms especially when the underlying linear classifier has large bias.

1 Introduction

Large margin classification methods are quite popular among Machine Learning and related research areas. Various generalization bounds (e.g., [27, 29, 7]) guarantee that linear classifiers with large margin over training data have small generalization error with high probability. The Support Vector Machine (SVM) [3] is one of the most powerful among such methods. The central idea of SVM is to find the maximum 2-norm margin hyperplane over linearly separable data. Further, by using kernels and soft margin formulations, it can learn large margin hyperplane over linearly inseparable data as well. The problem of finding the maximum 2-norm margin hyperplane over data is formulated as a quadratic programming problem. So the task of SVM can be solved in polynomial time by using standard optimization methods.

On the other hand, solving quadratic programming problems is time-consuming, especially for huge data which is now common in many applications. This motivates many researches for making SVM more scalable. One of major approaches is to decompose the original quadratic programming problem into smaller problems which are to solve [23, 24, 13, 6, 14]. Another popular approach is to apply online learning algorithms. Online learning algorithms such as Perceptron [26, 22, 21] and its variants [1, 8, 31, 10] work in iterations, where at each iteration, they process only one instance and update their hypotheses successively. Online learning algorithms use less memory, and are easy to implement. Many online learning algorithms that find large margin classifiers have been proposed, including Kernel Adatron[9], Max Margin Perceptron [17] Voted Perceptron [8], ROMMA [31], ALMA [10], NORMA [15], MICRA [30], and Pegasos [28].

However, most of these online learning algorithms do not fully exploit the linear separability of data. More precisely, they are designed to learn homogeneous hyperplanes, i.e., hyperplanes that lie on the origin, and they cannot learn linear classifiers with bias directly. So, in order to learn linear classifiers with bias, typical online learning algorithms map instances from the original space $\mathbb{R}^n$ to an augmented space $\mathbb{R}^{n+1}$ with an extra dimension by using the mapping $\phi : \boldsymbol{x} \mapsto \tilde{\boldsymbol{x}} = (\boldsymbol{x}, -R)$, where R is the maximum 2-norm of instances [7]. Then, a hyperplane with bias $(\boldsymbol{w}, b)$ in the original space corresponds to the hyperplane without bias $\tilde{\boldsymbol{w}} = (\boldsymbol{w}, -b/R)$ in the augmented space since $\boldsymbol{w}\cdot\boldsymbol{x}+b = \tilde{\boldsymbol{w}}\cdot\tilde{\boldsymbol{x}}$. So, by using this mapping, learning linear classifiers with bias can be reduced to learning those without bias. But, this mapping weakens the guarantee of margin. Suppose that for a sequence of labeled examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ ($\boldsymbol{x}_t \in \mathbb{R}^n$ and $y_t \in \{-1, +1\}$ for $t = 1, \ldots, T$), there is a hyperplane with bias $(\boldsymbol{u}, b)$ that has margin $\gamma = \min_{t=1,\ldots,T} \frac{y_t(\boldsymbol{u}\cdot\boldsymbol{x}_t+b)}{\|\boldsymbol{u}\|_2 R}$, where instances are normalized by R.

Then, the corresponding hyperplane $\tilde{\boldsymbol{u}} = (\boldsymbol{u}, -b/R)$ over the augmented space has margin

$$\tilde{\gamma} = \frac{y(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{x}})}{\|\tilde{\boldsymbol{u}}\|_2 R} = \frac{y(\boldsymbol{u}\cdot\boldsymbol{x}+b)}{\|\tilde{\boldsymbol{u}}\|_2 R} > \frac{1}{2}\gamma,$$

since $\|\tilde{\boldsymbol{u}}\|_2^2 = \|\boldsymbol{u}\|^2 + b^2/R^2 \leq 2\|\boldsymbol{u}\|^2$, and $\|\tilde{\boldsymbol{x}}\|_2^2 \leq 2R$. Even though the loss of margin is at most by a constant factor, it might cause significant difference in prediction performance over practical applications.

In this paper, we propose a new online learning algorithm that approximately maximizes the margin. Our algorithm, PUMMA (P-norm Utilizing Maximum Margin Algorithm), is an extension of ROMMA [31] in two ways. First, PUMMA can optimize the bias directly by using an implicit reduction from learning of linear classifiers with bias to learning those without bias, instead of using the mapping ϕ.

Second, PUMMA can provably approximate the maximum p-norm margin classifier for $p \geq 2$. A benefit of maximizing p-norm margin is that we can find sparse linear classifiers quickly. Technically speaking, PUMMA is a variant of p-norm algorithm [12, 11]. It is known that, if we set $p = \infty$ or $p = O(\ln n)$, the p-norm algorithm behaves like online multiplicative update algorithms such as Winnow [18], which can converge exponentially faster than Perceptron, when the underlying linear classifier is sparse. For example, if the target concept is a k-disjunction over n boolean variables, Winnow can find a consistent hypothesis in $O(k \log n)$ mistakes, while Perceptron needs $\Omega(kn)$ mistakes [16].

We show that PUMMA, given a parameter δ $(0 < \delta \leq 1)$ and $p > 1$, finds a linear classifier which has p-norm margin at least $(1-\delta)\gamma$ in $O(\frac{(p-1)R^2}{\delta^2\gamma^2})$ updates, when there exists a hyperplane with p-norm margin γ that separates the given sequence of data. The worst-case iteration bound of PUMMA is as the same as those of typical Perceptron-like algorithms when p=2 and that of ALMA [10] for $p > 1$, PUMMA is potentially faster than these previous algorithms especially when the underlying linear classifier has large bias. For linearly inseparable data, PUMMA can use kernels and the 2-norm soft margin furmution for $p = 2$, as well as previous online learning algorithms.

There are several related works. Kernel Adatron [9], SMO algorithm [24], and Max Margin Perceptron [17] can find bias directly, too. However, Kernel Adatron and SMO are not suitable for the online setting since they need to store past examples to compute the bias. Max Margin Perceptron finds the same solution of our algorithm when $p = 2$, but its upperbound of updates is $\log(R/\gamma)$ times worse than that of PUMMA . ROME algorithm [19] is also similar to our present work. It is an online learning algorithm that finds an accurate linear classifier quickly when the margin of the underlying classifier is defined as ∞-norm distance. On the other hand, ROME requires prior knowledge of the margin and bias.

In our preliminary experiments, PUMMA often outperforms previous online algorithms over artificial and real data by taking advantage of computing the bias directly.

2 Preliminaries

2.1 Norm

For any vector $x \in \mathbb{R}^n$ and $p > 0$, p-norm $\|x\|_p$ of x is given as $(\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$. In particular, $\|x\|_\infty$ is given as $\|x\|_\infty = \max_i |x_i|$. It can be shown that, for any fixed $x \in \mathbb{R}^n$, the p-norm $\|x\|_p$ is decreasing with respect to p, i.e., $\|x\|_{p'} \leq \|x\|_p$ for any $0 < p \leq p'$. For $p > 1$, q-norm is *dual* to p-norm if $q = 1 - 1/p$. For $p \geq 1$ and q such that $1/p + 1/q = 1$, it is known that

$$\|x\|_\infty \leq \|x\|_p \leq \|x\|_1 \leq n^{1/p}\|x\|_\infty.$$

2.2 Online learning

We consider the standard setting of online learning of linear classifiers, in which learning proceeds in trials. At each trial t, the learner receives an instance $x_t \in \mathbb{R}^n$, and it predicts a label $\hat{y}_t \in \{-1, +1\}$. Then the learner receives the true label $y_t \in \{-1, +1\}$ and then it possibly updates its current hypothesis depending on the received label. In this paper, we assume that labels are determined by a linear classifier $f(x) = \text{sign}(w \cdot x + b)$ for some weight vector $w \in \mathbb{R}^n$ and *bias* $b \in \mathbb{R}$, where $\text{sign}(a) = +1$ if $a \geq 0$, otherwise $\text{sign}(a) = -1$. In particular, if $y_t \neq \hat{y}_t$, we say that the learner makes a *mistake*. A typical goal of online learning is to minimize the number of mistakes as small as possible. Most of known online algorithms are *mistake-driven*, that is, they update their hypotheses when they make a mistake.

The p-norm distance between a hyperplane and a point is computed as follows:

Lemma 1 (Mangasarian [20]) Let $V = \{v \in \mathbb{R}^n \mid w \cdot v + b = 0\}$. Then, for any $x \in \mathbb{R}^n$,

$$\min_{v \in V} \|x - v\|_p = \frac{|w \cdot x + b|}{\|w\|_q},$$

where $q = 1/(1 - 1/p)$.

Based on Lemma 1, the p-norm (geometric) *margin* of a hyperplane (w, b) over an example (x, y) is defined as $\frac{y(w \cdot x + b)}{\|w\|_q}$. For any sequence of examples $S = ((x_1, y_1), \ldots, (x_T, y_T))$ $(T \geq 1)$, the *margin* of a hyperplane (w, b) over S is defined as $\min_{t=1,\ldots,T} \frac{y_t(w \cdot x_t + b)}{\|w\|_q}$. The algorithms we consider update their hypotheses if not only they make a mistake, but also their hypotheses have insufficient margin. In this paper, the learner's goal is to minimize the number of updates in order to obtain a linear classifier with approximately maximum p-norm margin over the given sequence of examples.

2.3 Convex duality

We review the basic results on convex analysis. Let $F : \mathbb{R}^n \to \mathbb{R}$ be a strictly convex differentiable function. The *Legendre dual* of F, denoted as F^*, is defined by

$$F^*(\theta) = \sup_{w \in \mathbb{R}^n} (\theta \cdot w - F(w)).$$

It can be verified that F^* is also strictly convex and differentiable. Then the following lemma holds:

Lemma 2 ([25, 5]) 1. $F^{**} = F$.

2. $F(w) + F^*(\theta) = \theta \cdot w$ if and only if $\theta = \nabla F(w)$.
3. $\nabla F^* = (\nabla F)^{-1}$.

In particular, we use $F(w) = \frac{1}{2}\|w\|_q^2$ throughout this paper. Let $f = \nabla F$, that is,

$$f(w)_i = \frac{\text{sign}(w_i)|w_i|^{q-1}}{\|w\|_q^{q-2}}$$

By Lemma 2 and some calculations, we obtain the following property (which was originally proved by Gentile [11]).

Lemma 3 1. The inverse f^{-} of f is given as

$$f^{-1}(w)_i = \frac{\mathrm{sign}(w_i)|w_i|^{p-1}}{\|w\|_p^{p-2}},$$

where $1/p + 1/q = 1$.

2. $\|f(w)\|_p = \|w\|_q$.

3. $w \cdot f(w) = \|f(w)\|_p^2 = \|w\|_q^2$.

3 PUMMA

We consider the learning of maximum p-norm margin classifiers in the online learning setting. By Lemma 1, the problem of finding the maximum p-norm margin hyperplane over a sequence of labeled examples $S = ((x_1, y_1), \dots, (x_m, y_T))$ is formulated as follows:

$$\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2}\|w\|_q^2, \quad (1)$$
$$\text{subject to :}$$
$$y_t(w \cdot x_t^{pos} + b) \geq 1 \quad (1 \leq t \leq T),$$

where q is such that $1/p + 1/q = 1$. Since the problem (1) is a convex optimization problem with linear inequality constraints, it can be solved by optimization methods such as interior-point methods [4]. However, in the context of online learning, it is time-consuming to solve the problem (1) at each trial. Further, it is necessary to store all the past given examples.

For $p = 2$, Li and Long proposed an elegant solution of the problem (1) in the online learning setting [31]. Their algorithm, ROMMA, is an online learning algorithm which finds approximate 2-norm maximum margin hyperplanes without bias. At each trial t, given an instance x_t, ROMMA predicts $\hat{y}_t = \mathrm{sign}(w_t \cdot x_t)$ such that

$$w_t = \arg\min_{w} \frac{1}{2}\|w\|_2^2, \quad (2)$$
$$\text{subject to}$$
$$y_{t-1} w \cdot x_{t-1} \geq 1 \text{ and } w \cdot w_{t-1} \geq \|w_{t-1}\|_2^2.$$

It can be shown that the constraints of the problem (2) is *relaxed*, that is, the constraints of the problem (2) is weaker than those of the problem (1) when $p = 2$ and b_t is fixed with 0. In fact, the second constraint in (2) corresponds to the hyperspace that contains the polyhedron which representing the constraints $y_j(w \cdot x_j) \geq 1$ $(j = 1, \dots, t-2)$.

Our algorithm PUMMA generalizes ROMMA in two folds: (i) PUMMA can maximize any p-norm margin with $p > 1$. (ii) PUMMA can directly learns non-homogeneous hyperplanes. PUMMA takes δ $(0 \leq \delta < 1)$ and p $(p > 1)$ as parameters. For initialization, it requires initial weight vector $w_0 = 0 \in \mathbb{R}^n$ and positive and negative instances x_1^{pos} and x_1^{neg}, respectively. These two examples are easily obtained by keep predicting -1 until the first positive example appears and predicting $+1$ until the first negative example comes. If either a positive or negative example cannot be obtained, then the number of updates is at most 1.

```
PUMMA ($\delta$, $p$)
begin
  1. (Initialization) Get examples $(x_1^{pos}, +1)$ and $(x_1^{neg}, -1)$. Let $w_0 = (0,\dots,0) \in \mathbb{R}^n$.
  2. For $t = 1$ to $T$,
     (a) Receive an instance $x_t$.
     (b) Let
            $(w_t, b_t) = \arg\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2}\|w\|_q^2$,
            subject to :
            $(w \cdot x_t^{pos} + b) \geq 1$
            $(w \cdot x_t^{neg} + b) \leq -1$
            $w \cdot f(w_{t-1}) \geq \|w_{t-1}\|_q^2$.
     (c) Predict $\hat{y}_t = \mathrm{sign}(w_t \cdot x_t + b_t)$.
     (d) Receive the label $y_t$. If $y_t(w_t \cdot x_t + b_t) < 1 - \delta$, update
            $(x_{t+1}^{pos}, x_{t+1}^{neg}) = \begin{cases} (x_t, x_t^{neg}) & ,(y_t = +1) \\ (x_t^{pos}, x_t) & ,(y_t = -1). \end{cases}$
         Otherwise, let
            $(x_{t+1}^{pos}, x_{t+1}^{neg}) = (x_t^{pos}, x_t^{neg})$.
end.
```

Figure 1: The description of PUMMA .

Then, given a sequence $S = ((x_1, y_1), \dots, (x_{t-1}, y_{t-1}))$ of examples and an instance x_t, PUMMA predicts $\hat{y}_t = \mathrm{sign}(w_t \cdot x_t + b_t)$, where w_t and b_t is given as follows:

$$(w_t, b_t) = \arg\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2}\|w\|_q^2, \quad (3)$$
$$\text{subject to :}$$
$$w \cdot x_t^{pos} + b \geq 1, \quad w \cdot x_t^{neg} + b \leq -1$$
$$w \cdot f(w_{t-1}) \geq \|w_{t-1}\|_q^2,$$

where $q = 1/(1 - 1/p)$, x_t^{pos} (x_t^{neg}) is the last positive (negative) example which incur an update. If $y_t(w_t \cdot x_t + b_t) < 1 - \delta$, PUMMA$_p(\delta)$ updates $(x_{t+1}^{pos}, x_{t+1}^{neg}) = (x_t, x_t^{neg})$, if $y_t = +1$, and $(x_{t+1}^{pos}, x_{t+1}^{neg}) = (x_t^{pos}, x_t)$, otherwise.

3.1 Solution of the optimization problem (3)

Now we show the solution of the optimization problem (3). In this subsection, for simplicity, we denote $v = w_{t-1}$, $\theta = f(w_{t-1})$, $x^{pos} = x_t^{pos}$ and $x^{neg} = x_t^{neg}$. Let L be the Lagrangian, that is,

$$L(w, \alpha, \beta) = \frac{1}{2}\|w\|_q^2 + \sum_{\ell \in \{pos, neg\}} \alpha^\ell \{1 - y^\ell (w \cdot x^\ell)\}$$
$$+ \beta(\|v\|_q^2 - \theta \cdot w),$$

where $y^{pos} = +1$ and $y^{neg} = -1$. Then the partial derivative of L w.r.t. w_i and b is given respectively as

$$\frac{\partial L}{\partial w_i} = f(w)_i - \sum_{\ell \in \{pos,neg\}} \alpha^\ell x_i^\ell - \beta\theta_i, \text{ and} \tag{4}$$

$$\frac{\partial L}{\partial b} = \alpha^{pos} - \alpha^{neg}. \tag{5}$$

Since the solution (w^*, b^*) must enforce the partial derivatives (4) and (5) to be zero, the vector w^* is specified as

$$w^* = f^{-1}(\alpha z + \beta_i \theta),$$

where $\alpha = \alpha^{pos} = \alpha^{neg}$, $z = x^{pos} - x^{neg}$ and

$$f^{-1}(\theta)_i = \frac{\operatorname{sign}(\theta_i)|\theta_i|^{p-1}}{\|\theta\|_p^{p-2}}.$$

Further, by KKT conditions, the parameters α and β satisfy that

$$\begin{aligned}
&\alpha(2 - w^* \cdot z) = 0, && (6)\\
&2 - w^* \cdot z \le 0, && (7)\\
&\alpha \ge 0, && (8)\\
&\beta(\|v\|_q^2 - w^* \cdot \theta) = 0, && (9)\\
&\|v\|_q^2 - w^* \cdot \theta \le 0, && (10)\\
&\text{and } \beta \ge 0. && (11)
\end{aligned}$$

We show that $\alpha > 0$ by contradiction. Assuming that $\alpha = 0$, we have $w^* = f(\beta\theta) = \beta v$. Then the conditions (9), (10) and (11) implies $\beta = 1$ and thus $w^* = v$. However, the condition (7) cannot be satisfied for $w^* = v$, which is a contradiction.

Now we consider two cases. (i) Suppose that $\beta = 0$. Then, the vector w^* is given as

$$w^* = \alpha f^{-1}(z), \tag{12}$$

where $\alpha = 2/\|z\|_q^2$. (ii) Otherwise, i.e., if $\beta > 0$,

$$w^* = \alpha f^{-1}(\alpha z + \beta v), \tag{13}$$

where α and β satisfies the following equations

$$\begin{cases} f^{-1}(\alpha z + \beta\theta) \cdot z = 2 \\ f^{-1}(\alpha z + \beta\theta) \cdot \theta = \|v\|_q^2 \end{cases} \tag{14}$$

The solution of equations (14) can be obtained by using Newton method. Let

$$G(\alpha, \beta) = \frac{1}{2}\|\alpha z + \beta\theta'\|_p^2 - 2\alpha - \beta\|\theta'\|_p^2.$$

Note that the partial derivatives of G are

$$\frac{\partial G}{\partial \alpha} = f^{-1}(\alpha z + \beta\theta') \cdot z - 2$$

$$\frac{\partial G}{\partial \beta} = f^{1}(\alpha z + \beta\theta') \cdot \theta' - \|\theta'\|_p^2.$$

Since G is convex, the equations (14) is satisfied if and only if G is minimized. So, given an initial assignment (α_0, β_0), we can approximate (α, β) by repeating the Newton update

$$\begin{pmatrix}\alpha_{k+1}\\ \beta_{k+1}\end{pmatrix} = \begin{pmatrix}\alpha_k\\ \beta_k\end{pmatrix} - \nabla^2 G(\alpha, \beta)^{-1} \nabla G(\alpha_k, \beta_k)$$

for sufficiently many steps. In particular, for $p = 2$, it holds that $f(x) = f^{-1} = x$. So, we have the following analytical solution for equations (14):

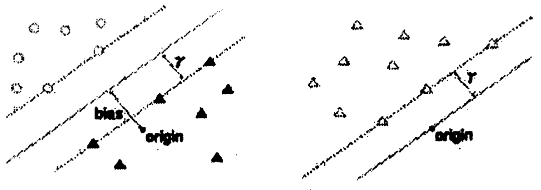
$$\begin{aligned}
\alpha &= \frac{\|v\|^2(2 - v \cdot z)}{\|v\|^2\|z\|^2 - (v \cdot z)^2} \text{ and}\\
\beta &= \frac{\|v\|^2\|z\|^2 - 2(v \cdot z)}{\|v\|^2\|z\|^2 - (v \cdot z)^2}.
\end{aligned} \tag{15}$$

As a summary, in order to obtain the solution w^*, we first assume the case (i) and check whether the condition $w^* \cdot \theta > \|v\|_q^2$ holds or not. If it does, the solution is given as (12). Otherwise, the case (ii) holds and the solution is (15) for $p = 2$, or we apply Newton method for $p > 1$.

In either case (i) or (ii), the bias b^* is given as

$$b^* = -\frac{w^* \cdot x^{pos} + w^* \cdot x^{neg}}{2}. \tag{16}$$

Figure 2: Illustration of the implicit reduction which preserves the margin.



3.2 Implicit reduction to learning classifiers without bias

We show an interpretation of PUMMA from the viewpoint of reduction. Let us fix $p = 2$. Then, it is easily verified that the update of PUMMA is identical to that of ROMMA for the instance $z = (x_t^{pos} - x_t^{neg})/2$ whose label is positive. This observation implies a reduction from learning linear classifiers with bias to learning of those without bias. Let $\mathcal{X} = \mathcal{X}^{pos} \cup \mathcal{X}^{neg}$ be a subset of $\mathbf{R}^n$, where $\mathcal{X}^{pos}$ and $\mathcal{X}^{neg}$ are positive and negative set of instances and $\mathcal{X}^{pos} \cap \mathcal{X}^{neg} = \emptyset$. Assume that there exists (u, b) such that $u \cdot x^{pos} + b \ge 1$ for each $x^{pos} \in \mathcal{X}^{pos}$, and $u \cdot x^{neg} + b \ge -1$ for each $x^{neg} \in \mathcal{X}^{neg}$. Then we consider the set

$$\mathcal{Z} = \left\{ \frac{x^{pos} - x^{neg}}{2} \mid x^{pos} \in \mathcal{X}^{pos},\ x^{neg} \in \mathcal{X}^{neg} \right\}.$$

That is, from a set of positive and negative instances, we define the set of positive instances. Note that $\mathcal{Z} \subset \mathbf{R}^n$ and $u \cdot z \ge 1$ for each $z \in \mathcal{Z}$ (See Figure 2). Further, if x^{pos} and x^{neg} are positive and negative support vectors of (u, b) respectively, then $z = (x^{pos} - x^{neg})$ is a support vector of u over Z. In addition, there is no pair of $(\tilde{x}^{pos}, \tilde{x}^{neg})$ of positive and negative non-support vectors of (u, b) such that $z = (\tilde{x}^{pos} - \tilde{x}^{neg})/2$. To see this, assume otherwise. Then, $\tilde{x}^{pos}$ and $\tilde{x}^{neg}$) are written as $\tilde{x}^{pos} = x^{pos} + \Delta$ and $\tilde{x}^{neg} = x^{neg} + \Delta$ for some $\Delta > 0$. Then, $u \cdot \tilde{x}^{pos} + b = 1 + u \cdot \Delta$ and $u \cdot \tilde{x}^{neg} + b = -1 + u \cdot \Delta$. By definition, $u \cdot \Delta \neq 0$, but then it follows that (u, b) misclassifies either $\tilde{x}^{pos}$ or $\tilde{x}^{neg}$, which contradicts the assumption [1]. Observe that this reduction does not reduce the margin.

PUMMA can be viewed as a "wrapper" algorithm of ROMMA equipped with this reduction. Given positive and negative instances x^{pos} and x^{neg}, PUMMA constructs a positive instance $z = (x^{pos} - x^{neg})/2$ and train ROMMA with z for a trial. Then PUMMA receives a weight vector w and set bias b as $b = -(w \cdot (x^{pos} - x^{neg}))/2$. If PUMMA makes a mistake (or does not have enough margin) over a new instance, it updates z and train ROMMA again.

It is possible to use any online learning algorithm that finds maximum margin linear classifier without bias as subroutines if it satisfies the following requirement: such a algorithm must output a weight vector whose support vector is z. However, most of known online algorithms maximizing the margin does not satisfy this requirement and ROMMA seems to be the only one satisfying the requirement so far.

3.3 Convergence proof

We prove an upperbound of updates made by PUMMA.

Lemma 4 For $t \geq 1$, it holds that

$$w_t \cdot x_t^{pos} + b_t = 1 \quad \text{and} \quad w_t \cdot x_t^{neg} + b_t = -1.$$

Lemma 5 Let $(u, b) \in \mathbb{R}^n \times \mathbb{R}$ be a hyperplane such that $y_j(u \cdot x_j + b) \geq 1$ for $j = 1, \ldots, t$. Then, it holds that $u \cdot \theta_t \geq \|w_t\|_q^2$ and $\|u\|_q \geq \|w_t\|_q$.

Proposition 1 Let $G(\theta) = \frac{1}{2}\|\theta\|_p^2$ with $p \geq 2$ and let $g = \nabla G$. Then it holds for any x and a that

$$G(\theta + a) \leq G(\theta) + g(\theta) \cdot a + (p-1)\|a\|_p^2$$

Lemma 6 For each trial $t \geq 1$ in which an update is incurred,

$$\|w_{t+1}\|_q^2 - \|w_t\|_q^2 \geq \frac{\delta^2}{2(p-1)R^2},$$

where $R = \max_{j=1,\ldots,t} \|x_j\|_p$.

Theorem 2 Suppose that for a sequence $S = ((x_1, y_1), \ldots, (w_T, y_T))$, there exists a hyperplane $(u, b) \in \mathbb{R}^n \times \mathbb{R}$ such that $y_t(u \cdot x_t + b) \geq 1$ for $t = 1, \ldots, T$ and the hyperplane (u, b) has p-norm margin γ over S. Further, let $R = \max_{t=1,\ldots,T} \|x_t\|_p$. (i) Then the number of updates made by PUMMA$_p(\delta)$ is at most

$$O\left(\frac{(p-1)R^2\|u\|_q^2}{\delta^2}\right).$$

(ii) PUMMA$_p(\delta)$ outputs a hypothesis with p-norm margin at least $(1-\delta)\gamma$ after at most the updates above.

Proof: W.l.o.g., we assume that PUMMA updates for $t = 1, \ldots, M (M \leq T)$. By Lemma 5, we have $\|w_t\| \leq \|u\|_q$ for $t \geq 1$. Further, by Lemma 6, it holds that after M updates

$$\|u\|_q^2 \geq \|w_T\|_q^2 \geq \frac{\delta^2 M}{2(p-1)R^2},$$

which implies $M \leq \frac{2\|u\|_q^2 R^2}{\delta^2}$. Further, after at most $\frac{2\|u\|_q^2 R^2}{\delta^2}$ updates, we have $y_t(w_t + b_t) \geq 1 - \delta$ for $t \geq T$. Then the achieved margin is at least

$$\frac{1-\delta}{\|w\|_q} \geq \frac{1-\delta}{\|u\|_q} = (1-\delta)\gamma.$$

■

Since $\|x\|_q \leq \|x\|_1$ for $q \leq 1$ and $\|x\|_p \leq n^{1/p}\|x\|_\infty$, we obtain the following corollary.

Corollary 3 Assume that for a sequence $S = ((x_1, y_1), \ldots, (w_T, y_T))$, there exists a hyperplane $(u, b) \in \mathbb{R}^n \times \mathbb{R}$ such that $y_t(u \cdot x_t + b) \geq 1$ for $t = 1, \ldots, T$ and the hyperplane (u, b) has ∞-norm margin γ over S. Further, let $R = \max_{t=1,\ldots,T} \|x_t\|_\infty$. Then, by setting $p = c \ln n$ $(c > 0)$, (i) the number of updates made by PUMMA$_p(\delta)$ is at most

$$O\left(\frac{R^2\|u\|_1^2 \log n}{\delta^2}\right).$$

(ii) PUMMA$_p(\delta)$ outputs a hypothesis with ∞-norm margin at least $\frac{1-\delta}{e^{1/c}}\gamma$ after at most the updates above.

4 Experiments

4.1 Experiments over artificial datasets

We examine PUMMA , ALMA and ROMMA over artificial datasets generated by sparse linear classifiers. Each artificial dataset consists of n-dimensional $\{-1, +1\}$-valued vectors with $n = 100$. Each vector is labeled with a r-of-k threshold function f, which is represented as $f(x) = \text{sign}(x_{i_1} + \cdots + x_{i_k} + k - 2r + 1)$ for some $i_1, \ldots, i_k$ s.t. $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$, and it outputs $+1$ if at least r of k relevant features have value $+1$, and outputs -1, otherwise.

For $k = 16$ and $r \in \{1, 4, 8\}$ (equivalently, the bias $b \in \{15, 9, 1\}$, respectively), we generate random 1000 examples labeled by the r-of-k threshold function, so that positive and negative examples are equally likely. For ALMA and ROMMA, we add an extra dimension with value $-R$ to each vector to learn linear classifiers with bias, where $R = \max\|x\|_p$. We set parameters so that each algorithm is guaranteed to achieve at least 0.9 times the maximum p-norm margin. That is, we set $\alpha = 0.1$ (note the parameter α is defined differently in [10]) for ALMA and $\delta = 0.1$ for ROMMA an PUMMA . We examine $p \in \{2, 2\ln n\}$.

We train each algorithm until its hypothesis converges by running it in epochs, where, in one epoch,

[1]Note that there might exist several pairs of support vectors corresponding to z. Imagine that $u \cdot \Delta = 0$ to see why.

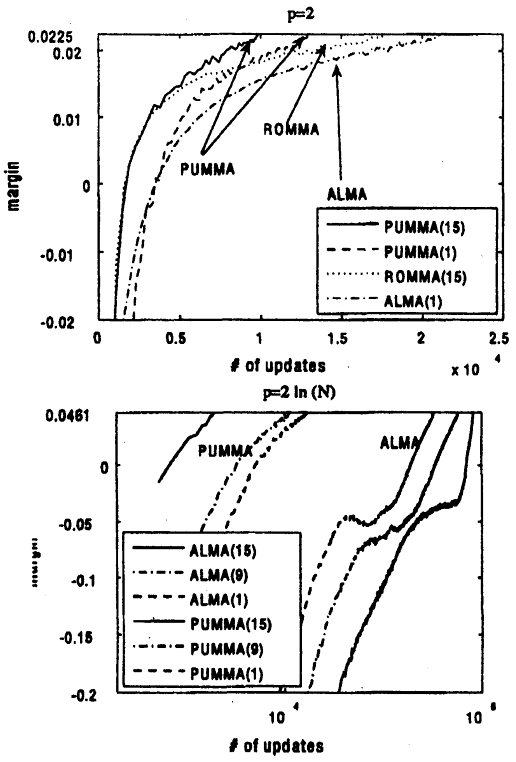


Figure 3: Number of updates and margin over artifitial data set in the case $p = 2$ (upper) and $p = 2\ln(N)$ (lower). We set x-axes log scale since the numbers of updates of ALMA are quite larger than PUMMA 's. And we hide the result of the case $p = 2$ and $b = 9$ since we make the figure easy to view. The parenthetical digits denote the value of bias.

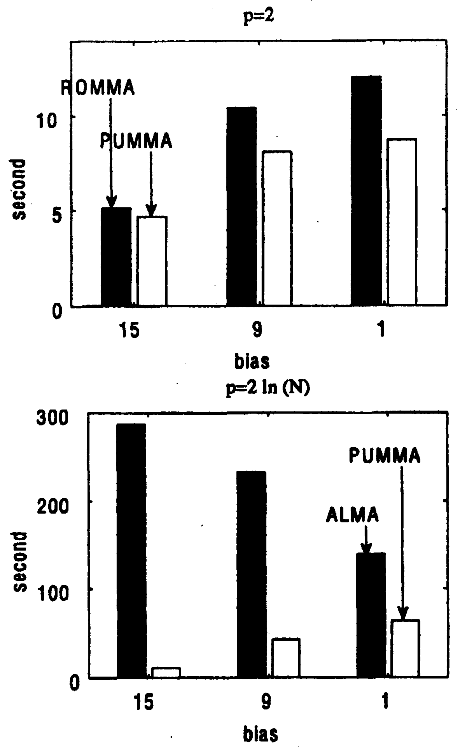


Figure 4: Comptation time over artifitial data set in the case $p = 2$ (upper) and $p = 2\ln(n)$ (lower).

we make each algorithm go through the whole training data once. At end of each epoch, for each algorithm, we record number of updates, margin incurred during the training and real computation time. Note that we measure the margin of each hypothesis over the original space. We execute these operations 10 times, changing the randomly generated data, and we average the results over 10 executions. The experiments are conducted on a 3.8 GHz Intel Xeon processor with 8 GB RAM running Linux. We use MATLAB for the experiments.

The results are presented in Figure 3 and Figure 4. We observe that PUMMA converges faster. Although PUMMA uses Newton method in each update, its computation time is quite shorter than that of ALMA. Note that we omit the result of ALMA in the case $p = 2$ since the result is worse than the others. For $p = 2$, we don't use Newton method in the execution of PUMMA because we have the analytical solution of the optimal value of $\alpha$ and $\beta$ by solving the optimization problem directly.

4.2 Experiments over real datasets

We compare PUMMA with other learning algorithms over some real datasets. The algorithms we compare are SVM^{light} [13], MICRA [30], and ROMMA [31]. We used the following datasets of UCI Machine Learning Repository [2]. (i) The ionosphere dataset consists of 351 instances which have 34 continuous attributes. (ii) The house-vote dataset consists of 435 instances which have 16 discrete attributes $\{y, n, ?\}$. We change these attributes to $\{1, -1, 0\}$. (iii) The adult dataset consists of 32561 instances which have 14 attributes. Among the attributes, 6 of them are discrete and the others are continuous. We change this 14 attributes to 123 binary attributes as Platt did in [24]. The name of dataset 'adult-mk' in Table 1 denotes a subset of the adult dataset which contains $1000 \times m$ instances. Note that all the datasets have binary class and we change the range of labels with $\{1, -1\}$.

To optimize the 2-norm soft margin for this linearly inseparable dataset, as in [7], we use the following modified inner product

$$IP(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \delta_{i,j}\lambda,$$

where $\delta_{i,j}$ is the Kronecker delta function which equals to 1 if $i = j$, and otherwise 0. We added a dimension which denotes the bias as in section 1 when we run MICRA and ROMMA which can't deal with bias directly.

We set $c = \infty$ for SVM^{light} to stop 1-norm soft margin working, and we change the inner product so

Table 1: Computation time (sec.) and obtained margin (denoted as γ') on real datasets.

	SVM^{light}		PUMMA		ROMMA		MICRA	
dataset	sec.	$10^2\gamma'$	sec.	$10^2\gamma'$	sec.	$10^2\gamma'$	sec.	$10^2\gamma'$
ionosphere	0.06	10.55	0.54	10.49	3.12	10.50	0.48	10.04
house-votes	0.03	17.42	0.26	17.31	0.62	17.36	0.09	16.51
adult-1k	0.47	4.95	5.40	4.50	15.83	4.91	2.34	4.03
adult-2k	2.13	3.40	25.38	3.37	82.70	3.38	5.61	2.81
adult-4k	9.33	2.40	159.54	2.38	496.52	2.38	55.91	2.00
adult-8k	232.42	1.69	807.46	1.67	2167.40	1.67	189.13	1.46
adult-16k	1271.06	1.20	3365.47	1.18	12503.62	1.18	2050.84	1.13
adult-full	5893.20	0.83	44480.59	0.82	71296.34	0.82	12394.86	0.79

that it deal with 2-norm soft margin. We set $\delta = 0.01$ for PUMMA and ROMMA to achieve 99% of the maximum margin. The parameters of MICRA are changed for each dataset as in [30]. But, parameters might not be completely the same as them because some datasets are different from those they used. Finally we set 2-norm soft margin parameter $\lambda = 1$ for all algorithms.

We run SVM^{light} and each online learning algorithm until it converges, and we measure the real computation time and the obtained margin. The experiments on real datasets are conducted on a 3.0 GHz Intel Xeon processor with 16 GB RAM running Linux. We implemented each algorithm in C.

Table 1 shows the real computation time. As can be seen, PUMMA converges quite faster than ROMMA. On the other hand, PUMMA converges slower than MICRA. But the parameters of MICRA must be changed for each dataset to get better result, and these parameters are sensitive and it is nontrivial to choose good parameters. The results on all the real data set show that SVM^{light} is the fastest, but MICRA is reported to be faster than SVM^{light} over some datasets and with tuned parameters [30]. We report that the computation time of PUMMA is comparable to SVM^{light}'s if $\delta = 0.1$. However, in this case, lower margins ar obtained by PUMMA.

5 Conclusion and Future work

In this paper, we propose PUMMA which obtains the maximum p-norm margin classifier with bias approximately. PUMMA runs much faster than previous online learning algorithms over both artificial and real datasets.

One of our future work is to extend our algorithm to handle 1-norm soft margin which is commonly used in SVM. Further, we would like to apply PUMMA to learning sparse classifiers in practical aplications.

References

[1] J. K. Anlauf and M. Biehl. The adatron; an adaptive perceptron algorithm. *Europhysics Letters*, 10:687–692, 1989.

[2] A. Asuncion and D. J. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, `http://mlearn.ics.uci.edu/MLRepository.html`, 2007.

[3] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

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

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machine*. Cambridge University Press, 2000.

[8] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–299, 1999.

[9] T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

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

[11] C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

[12] A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. In *Proceedings of the tenth anual conference of Computational learning theory*, pages 171–183, 1997.

[13] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in kernel methods - Support vector learning*, pages 169–184. MIT Press, 1999.

[14] T. Joachims. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[15] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[16] J. Kivinen, M. K. Warmuth, and P. Auer. The perceptron algorithm versus winnow: linear versus logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97(1-2):325–343, 1997.

[17] A. Kowalczyk. Maximum margin perceptron. In B. Scholkopf A. Smola, P. Bartlett and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 75–114. MIT Press, 2000.

[18] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[19] P. M. Long and X. Wu. Mistake bounds for maximum entropy discrimination. In *Advances in Neural Information Processing Systems 17*, pages 833–840, 2004.

[20] O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999.

[21] M. L. Minsky and S. A. Papert. *Perceptrons.* MIT Press, 1969.

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

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

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

[25] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, 1970.

[26] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1959.

[27] R. E. Schapire, Y. Freund, P. Bartlett, 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.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

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

[30] P. Tsampouka and J. Shawe-Taylor. Approximate maximum margin algorithms with rules controlled by the number of mistakes. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[31] Y.Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.