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

Kosuke Ishibashi, Kohei Hatano and Masayuki Takeda
Department of Informatics, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
{k-ishi, hatano, takeda}@i.kyushu-u.ac.jp

Abstract

We propose a new online learning algorithm which 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 show 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., [32, 34, 10]) guarantee that linear classifiers with large margin over training data have small generalization error with high probability. The Support Vector Machine (SVM) [5] 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 [28, 29, 16, 8, 17]. Another popular approach is to apply online learning algorithms. Online learning algorithms such as Perceptron [31, 27, 26] and its variants [1, 11, 22, 13] 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, e.g., Kernel Adatron [12], Voted Perceptron [11], Max Margin Perceptron [21], ROMMA [22], ALMA [13], NORMA [19], LASVM [4], MICRA [35], and Pegasos [33].

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 [10]. 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 so as to limit the maximum 2-norm of instances to be one. Then, the corresponding hyperplane $\tilde{\boldsymbol{u}} = (\boldsymbol{u}, -b/R)$ over the augmented space, in which the maximum norm of instances is bounded by $\tilde{R}$, has margin

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

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 [22] 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 [15, 14]. 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 [23], 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 $O(k \ln n)$ mistakes, while Perceptron needs $\Omega(kn)$ mistakes [20].

We show that PUMMA, given a parameter δ ($0 < \delta \leq 1$) and $p \geq 2$, 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 [13] for $p > 2$, 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 formulation [9] for $p = 2$, as well as previous Perceptron-like online learning algorithms. Further, we extend PUMMA to deal with 2-norm soft margin formulation for $p > 2$. Note that in standard implementations of the SVM [16, 8, 17], the 1-norm soft margin formulation (see, e.g., [10]) is preferred since it often requires less computation time. However, in general, both soft margin formulations are incomparable in terms of generalization ability, which depends on data and choices of kernels. For online-based implementations of the SVM with 1-norm soft margin see LASVM [4] and Pegasos [33].

There are other related works. For $p = 2$, previous algorithms such as Kernel Adatron [12], NPA [18], SMO algorithm [29], Max Margin Perceptron [21], and LASVM [4] can find bias directly as well. However, the first three algorithms 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, but its upperbound of updates is $\ln(R/\gamma)$ times worse than that of PUMMA . For LASVM, there is no theoretical analysis of its convergence rate. For $p = \infty$, ROME algorithm [24] 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. For a more general convex optimization technique which includes ROMMA as a special case, see [3].

In our preliminary experiments, PUMMA converges faster than previous online algorithms over artificial dataset, especially when the underlying linear classifier has large bias. In particular, for $p = O(\ln n)$, PUMMA is from 2 to 10 times faster than ALMA. Over real datasets, PUMMA often outperforms previous online algorithms.

2 Preliminaries

2.1 Norm

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

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

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 $\boldsymbol{x}_t \in \mathbb{R}^n$, 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 $f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ for some weight vector $\boldsymbol{w} \in \mathbb{R}^n$ and *bias* $b \in \mathbb{R}$, where $\mathrm{sign}(a) = +1$ if $a \geq 0$, otherwise $\mathrm{sign}(a) = -1$. 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 ([25]) Let $V = \{\boldsymbol{v} \in \mathbb{R}^n \mid \boldsymbol{w} \cdot \boldsymbol{v} + b = 0\}$. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$,

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

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

Based on Lemma 1, the p-norm (geometric) *margin* of a hyperplane $(\boldsymbol{w}, b)$ over an example $(\boldsymbol{x}, y)$ is defined as

$$\frac{y(\boldsymbol{w} \cdot \boldsymbol{x} + b)}{\|\boldsymbol{w}\|_q}.$$

For any sequence of examples $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T))$ ($T \geq 1$), the *margin* of a hyperplane $(\boldsymbol{w}, b)$ over S is defined as the minimum margin of examples in S. 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

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

[1] More generally, this lemma holds for an arbitrary norm and its dual norm.

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

Lemma 2 ([30, 7]) 1. $F^{**} = F$.

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

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

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

By Lemma 2 and some calculations, we obtain the following property.

Lemma 3 ([14]) 1. The inverse $\boldsymbol{f}^{-1}$ of $\boldsymbol{f}$ is given as

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

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

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

Finally, we will use the following bound later.

Proposition 1 ([15, 14]) Let $\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_p^2$ with $p \geq 2$ and let $\boldsymbol{g} = \nabla \boldsymbol{G}$. Then it holds for any $\boldsymbol{x}$ and $\boldsymbol{a}$ that

$$\boldsymbol{G}(\boldsymbol{\theta} + \boldsymbol{a}) \leq \boldsymbol{G}(\boldsymbol{\theta}) + \boldsymbol{g}(\boldsymbol{\theta}) \cdot \boldsymbol{a} + \frac{(p-1)}{2}\|\boldsymbol{a}\|_p^2.$$

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 $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_T))$ is formulated as follows:

$$\min_{\boldsymbol{w}, b} \frac{1}{2}\|\boldsymbol{w}\|_q^2, \quad (1)$$
$$\text{subject to :}$$
$$y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t + b) \geq 1 \ \ (1 \leq t \leq T),$$

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 [6]. 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 [22]. Their algorithm, ROMMA, is an online learning algorithm that finds approximate 2-norm maximum margin hyperplanes without bias. At each trial t, given an instance $\boldsymbol{x}_t$, ROMMA predicts $\hat{y}_t = \mathrm{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$ such that

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

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 $y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j) \geq 1$ ($j = 1, \ldots, t-2$).

Our algorithm, PUMMA, generalizes ROMMA in two folds: (i) PUMMA can maximize any p-norm margin with $p \geq 2$. (ii) PUMMA can directly learns non-homogeneous hyperplanes. PUMMA takes δ ($0 \leq \delta < 1$) and p ($p \geq 2$) as parameters. For initialization, it requires initial weight vector $\boldsymbol{w}_0 = \boldsymbol{0} \in \mathbb{R}^n$ and positive and negative instances $\boldsymbol{x}_1^{pos}$ and $\boldsymbol{x}_1^{neg}$, 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.

Then, given a sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{t-1}, y_{t-1}))$ of examples and an instance $\boldsymbol{x}_t$, PUMMA predicts $\hat{y}_t = \mathrm{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t + b_t)$, where $\boldsymbol{w}_t$ and b_t is given as follows:

$$(\boldsymbol{w}_t, b_t) = \arg\min_{\boldsymbol{w}, b} \frac{1}{2}\|\boldsymbol{w}\|_q^2, \quad (3)$$
$$\text{subject to :}$$
$$\boldsymbol{w} \cdot \boldsymbol{x}_t^{pos} + b \geq 1, \ \ \boldsymbol{w} \cdot \boldsymbol{x}_t^{neg} + b \leq -1$$
$$\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{w}_{t-1}) \geq \|\boldsymbol{w}_{t-1}\|_q^2,$$

where $q = 1/(1 - 1/p)$, $\boldsymbol{x}_t^{pos}$ and $\boldsymbol{x}_t^{neg}$ are the last positive and negative examples which incur updates, respectively. If $y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t + b_t) < 1 - \delta$, PUMMA$_p(\delta)$ updates $(\boldsymbol{x}_{t+1}^{pos}, \boldsymbol{x}_{t+1}^{neg}) = (\boldsymbol{x}_t, \boldsymbol{x}_t^{neg})$, if $y_t = +1$, and $(\boldsymbol{x}_{t+1}^{pos}, \boldsymbol{x}_{t+1}^{neg}) = (\boldsymbol{x}_t^{pos}, \boldsymbol{x}_t)$, 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 $\boldsymbol{v} = \boldsymbol{w}_{t-1}$, $\boldsymbol{\theta} = \boldsymbol{f}(\boldsymbol{w}_{t-1})$, $\boldsymbol{x}^{pos} = \boldsymbol{x}_t^{pos}$ and $\boldsymbol{x}^{neg} = \boldsymbol{x}_t^{neg}$. Let L be the Lagrangian, that is,

$$L(\boldsymbol{w}, b, \alpha, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|_q^2 + \sum_{\ell \in \{pos, neg\}} \alpha^\ell\{1 - y^\ell(\boldsymbol{w} \cdot \boldsymbol{x}^\ell + b)\} + \beta(\|\boldsymbol{v}\|_q^2 - \boldsymbol{\theta} \cdot \boldsymbol{w}), \quad (4)$$

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

$$\frac{\partial L}{\partial w_i} = \boldsymbol{f}(\boldsymbol{w})_i - \sum_{\ell \in \{pos, neg\}} y^\ell \alpha^\ell x_i^\ell - \beta\theta_i, \ \text{ and } \quad (5)$$

$$\frac{\partial L}{\partial b} = \alpha^{pos} - \alpha^{neg}. \quad (6)$$

Since the solution $(\boldsymbol{w}^*, b^*)$ must enforce the partial derivatives (5) and (6) to be zero, the vector $\boldsymbol{w}^*$ is specified as

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

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

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

```
PUMMA_p(δ)
begin
  1. (Initialization) Get examples $(\boldsymbol{x}_1^{pos}, +1)$ and $(\boldsymbol{x}_1^{neg}, -1)$. Let $\boldsymbol{w}_0 = (0, \ldots, 0) \in \mathbb{R}^n$.
  2. For $t = 1$ to $T$,
     (a) Receive an instance $\boldsymbol{x}_t$.
     (b) Let
             $(\boldsymbol{w}_t, b_t) = \arg\min_{\boldsymbol{w}, b} \frac{1}{2}\|\boldsymbol{w}\|_q^2,$
             subject to :
             $(\boldsymbol{w} \cdot \boldsymbol{x}_t^{pos} + b) \geq 1$
             $(\boldsymbol{w} \cdot \boldsymbol{x}_t^{neg} + b) \leq -1$
             $\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{w}_{t-1}) \geq \|\boldsymbol{w}_{t-1}\|_q^2.$
     (c) Predict $\hat{y}_t = \mathrm{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t + b_t)$.
     (d) Receive the label $y_t$. If $y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t + b_t) < 1 - \delta$, update
             $(\boldsymbol{x}_{t+1}^{pos}, \boldsymbol{x}_{t+1}^{neg}) = \begin{cases} (\boldsymbol{x}_t, \boldsymbol{x}_t^{neg}) & ,(y_t = +1) \\ (\boldsymbol{x}_t^{pos}, \boldsymbol{x}_t) & ,(y_t = -1). \end{cases}$
         Otherwise, let
             $(\boldsymbol{x}_{t+1}^{pos}, \boldsymbol{x}_{t+1}^{neg}) = (\boldsymbol{x}_t^{pos}, \boldsymbol{x}_t^{neg}).$
end.
```

Figure 1: The description of PUMMA .

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

$$\alpha(1 - \boldsymbol{w}^* \cdot \boldsymbol{x}^{pos} - b^*) = 0, \tag{7}$$

$$\alpha(1 + \boldsymbol{w}^* \cdot \boldsymbol{x}^{neg} + b^*) = 0, \tag{8}$$

$$1 - \boldsymbol{w}^* \cdot \boldsymbol{x}^{pos} - b^* \leq 0, \tag{9}$$

$$1 + \boldsymbol{w}^* \cdot \boldsymbol{x}^{neg} + b^* \leq 0, \tag{10}$$

$$\alpha \geq 0, \tag{11}$$

$$\beta(\|\boldsymbol{v}\|_q^2 - \boldsymbol{w}^* \cdot \boldsymbol{\theta}) = 0, \tag{12}$$

$$\|\boldsymbol{v}\|_q^2 - \boldsymbol{w}^* \cdot \boldsymbol{\theta} \leq 0, \tag{13}$$

$$\text{and } \beta \geq 0. \tag{14}$$

We show that $\alpha > 0$ by contradiction. Assuming that $\alpha = 0$, we have $\boldsymbol{w}^* = \boldsymbol{f}(\beta\boldsymbol{\theta}) = \beta\boldsymbol{v}$. Then the conditions (12), (13) and (14) implies $\beta = 1$ and thus $\boldsymbol{w}^* = \boldsymbol{v}$. However, the conditions (9) or (10) cannot be satisfied for $\boldsymbol{w}^* = \boldsymbol{v}$, which is a contradiction.

Now we consider two cases. (i) Suppose that $\beta = 0$. Then, since $\alpha > 0$ and the conditions (7) and (8) hold, the vector $\boldsymbol{w}^*$ is given as

$$\boldsymbol{w}^* = \alpha\boldsymbol{f}^{-1}(\boldsymbol{z}), \tag{15}$$

where $\alpha = 2/\|\boldsymbol{z}\|_p^2$. (ii) Otherwise, i.e., if $\beta > 0$, by the conditions (7), (8), and (12),

$$\boldsymbol{w}^* = \boldsymbol{f}^{-1}(\alpha\boldsymbol{z} + \beta\boldsymbol{v}), \tag{16}$$

where α and β where α and β satisfies the following equations

$$\begin{cases} \boldsymbol{f}^{-1}(\alpha\boldsymbol{z} + \beta\theta) \cdot \boldsymbol{z} = 2, \\ \boldsymbol{f}^{-1}(\alpha\boldsymbol{z} + \beta\theta) \cdot \boldsymbol{\theta} = \|\boldsymbol{v}\|_q^2. \end{cases} \tag{17}$$

That is, the optimal solution $\boldsymbol{w}^*$ satisfies the constraints of the problem(3) with equality. In this case, the solution can be obtained by maximizing its Lagrange dual L^* which is defined as

$$L^*(\alpha, \beta) = \min_{\boldsymbol{w}, b} L(\boldsymbol{w}, b, \alpha, \beta).$$

Further, with some calculations, L^* is computed as

$$L^*(\alpha, \beta) = -\frac{1}{2}\|\alpha\boldsymbol{z} + \beta\boldsymbol{\theta}\|_p^2 + 2\alpha + \beta\|\boldsymbol{\theta}\|_p^2. \tag{18}$$

Then, Note that the partial derivatives of L^* are

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

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

Since L^* is concave, the equations (17) is satisfied if and only if L^* is maximized. So, given an initial assignment (α_0, β_0), we can approximate (α, β) by repeating the Newton update

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

for sufficiently many steps, where

$$\frac{\partial^2 L^*}{\partial^2 \alpha} = \sum_i \boldsymbol{f}^{-1'}(\alpha\boldsymbol{z} + \beta\boldsymbol{\theta})_i z_i^2,$$

$$\frac{\partial^2 L^*}{\partial\beta\partial\alpha} = \sum_i \boldsymbol{f}^{-1'}(\alpha\boldsymbol{z} + \beta\boldsymbol{\theta})_i z_i \theta_i,$$

$$\frac{\partial^2 L^*}{\partial\alpha\partial\beta} = \sum_i \boldsymbol{f}^{-1'}(\alpha\boldsymbol{z} + \beta\boldsymbol{\theta})_i z_i \theta_i,$$

$$\frac{\partial^2 L^*}{\partial^2 \beta} = \sum_i \boldsymbol{f}^{-1'}(\alpha\boldsymbol{z} + \beta\boldsymbol{\theta})_i \theta_i^2,$$

and

$$\begin{aligned} \boldsymbol{f}^{-1'}(\boldsymbol{\theta})_i &= \frac{\partial \boldsymbol{f}^{-1}(\boldsymbol{\theta})}{\partial \theta_i} \\ &= -(p-2)\frac{|\theta_i|^{2(p-1)}}{\|\boldsymbol{\theta}\|_p^{2p-2}} + (p-1)\frac{|\theta_i|^{p-2}}{\|\boldsymbol{\theta}\|_p^{p-2}}. \end{aligned}$$

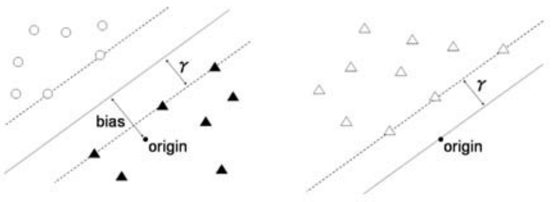
In our implementation, we set initial values as $\alpha_0 = 0$ and $\beta_0 = 1$.

In particular, for $p = 2$, it holds that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}^{-1} = \boldsymbol{x}$. So, we have the following analytical solution for equations (17):

$$\alpha = \frac{\|\boldsymbol{v}\|^2(2 - \boldsymbol{v} \cdot \boldsymbol{z})}{\|\boldsymbol{v}\|^2\|\boldsymbol{z}\|^2 - (\boldsymbol{v} \cdot \boldsymbol{z})^2} \text{ and}$$

$$\beta = \frac{\|\boldsymbol{v}\|^2\|\boldsymbol{z}\|^2 - 2(\boldsymbol{v} \cdot \boldsymbol{z})}{\|\boldsymbol{v}\|^2\|\boldsymbol{z}\|^2 - (\boldsymbol{v} \cdot \boldsymbol{z})^2}. \tag{19}$$

Figure 2: Illustration of the implicit reduction which preserves the margin. Each pair of positive and negative examples in the original space (left) corresponds to a positive example in the new space (right).



As a summary, in order to obtain the solution $\boldsymbol{w}^*$, we first assume the case (i) and check whether the condition $\boldsymbol{w}^* \cdot \boldsymbol{\theta} > \|\boldsymbol{v}\|_q^2$ holds or not. If it does, the solution is given as (15). Otherwise, the case (ii) holds and the solution is (19) for $p = 2$, or we apply Newton method for $p > 2$.

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

$$b^* = -\frac{\boldsymbol{w}^* \cdot \boldsymbol{x}^{pos} + \boldsymbol{w}^* \cdot \boldsymbol{x}^{neg}}{2}. \tag{20}$$

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 $\boldsymbol{z} = (\boldsymbol{x}_t^{pos} - \boldsymbol{x}_t^{neg})/2$ 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 $\mathbb{R}^n$, 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 $(\boldsymbol{u}, b)$ such that $\boldsymbol{u} \cdot \boldsymbol{x}^{pos} + b \geq 1$ for each $\boldsymbol{x}^{pos} \in \mathcal{X}^{pos}$, and $\boldsymbol{u} \cdot \boldsymbol{x}^{neg} + b \leq -1$ for each $\boldsymbol{x}^{neg} \in \mathcal{X}^{neg}$. Then we consider the set

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

That is, from a set of positive and negative instances, we define the set of positive instances. Then, the following property holds for $\mathcal{Z}$.

Theorem 2 Fix any p satisfying $2 \leq p < \infty$. Let $(\boldsymbol{u}, b)$ be the maximum p-norm hyperplane over $\mathcal{X}$. Then, $\boldsymbol{u}$ is the maximum p-norm hyperplane over $\mathcal{Z}$ as well. Also, the opposite holds for some b.

Proof: Let $\boldsymbol{u}'$ be the maximum p-norm hyperplane over $\mathcal{Z}$. Note that $\boldsymbol{u} \cdot \boldsymbol{z} \geq 1$ for each $\boldsymbol{z} \in \mathcal{Z}$ (See Figure 2). So, we have $\|\boldsymbol{u}\|_q^2 \geq \|\boldsymbol{u}'\|_q^2$ for q s.t. $1/p + 1/q = 1$. Now let $b' = \boldsymbol{u}' \cdot (\tilde{\boldsymbol{x}}^{pos} + \tilde{\boldsymbol{x}}^{neg})/2$, where $\tilde{\boldsymbol{x}}^{pos}$ and $\tilde{\boldsymbol{x}}^{neg}$ satisfies $\boldsymbol{u}' \cdot (\tilde{\boldsymbol{x}}^{pos} - \tilde{\boldsymbol{x}}^{neg}) = 2$, for any $\boldsymbol{x}^{pos} \in \mathcal{X}^{pos}$. Note that such a pair $(\tilde{\boldsymbol{x}}^{pos}, \tilde{\boldsymbol{x}}^{neg})$ always exists since $\boldsymbol{u}'$ is the maximum p-norm margin hyperplane. Then, we have

$$\begin{aligned} \boldsymbol{u}' \cdot \boldsymbol{x}^{pos} + b' &= \boldsymbol{u}' \cdot \tilde{\boldsymbol{x}}^{pos} + b' + \boldsymbol{u}' \cdot (\boldsymbol{x}^{pos} - \tilde{\boldsymbol{x}}^{pos}) \\ &= \frac{\boldsymbol{u}' \cdot (\tilde{\boldsymbol{x}}^{pos} - \tilde{\boldsymbol{x}}^{neg})}{2} + \boldsymbol{u}' \cdot (\boldsymbol{x}^{pos} - \tilde{\boldsymbol{x}}^{pos}) \\ &= 1 + \boldsymbol{u}' \cdot (\boldsymbol{x}^{pos} - \tilde{\boldsymbol{x}}^{neg} - \tilde{\boldsymbol{x}}^{pos} + \tilde{\boldsymbol{x}}^{neg}) \\ &\geq 1 + 2 - 2 = 1. \end{aligned}$$

Similarly, it holds for any $\boldsymbol{x}^{neg} \in \mathcal{X}^{neg}$ that $\boldsymbol{u}' \cdot \boldsymbol{x}^{neg} + b' \leq -1$. So, we get $\|\boldsymbol{u}'\|_q^2 \geq \|\boldsymbol{u}\|_q^2$. Finally, since the function $\|\cdot\|_q^2$ $(1 < q \leq 2)$ is strictly convex, the minimum is unique. Therefore we obtain $\boldsymbol{u} = \boldsymbol{u}'$. ■

This theorem ensures that finding the maximum margin hyperplane with bias can be reduced to finding those without bias over pairs of positive and negative instances. 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 $\boldsymbol{x}^{pos}$ and $\boldsymbol{x}^{neg}$, PUMMA constructs a positive instance $\boldsymbol{z} = (\boldsymbol{x}^{pos} - \boldsymbol{x}^{neg})/2$ and train ROMMA with $\boldsymbol{z}$ for a trial. Then PUMMA receives a weight vector $\boldsymbol{w}$ and set bias b as $b = -(\boldsymbol{w} \cdot (\boldsymbol{x}^{pos} + \boldsymbol{x}^{neg}))/2$. If PUMMA makes a mistake (or does not have enough margin) over a new instance, it updates $\boldsymbol{z}$ 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 an algorithm must output a weight vector whose support vector is $\boldsymbol{z}$. 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. First of all, by the KKT conditions for equations (7) and (8), the following property holds:

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

$$\boldsymbol{w}_t \cdot \boldsymbol{x}_t^{pos} + b_t = 1 \text{ and } \boldsymbol{w}_t \cdot \boldsymbol{x}_t^{neg} + b_t = -1.$$

Then we prove that the optimal solution of the offline optimization problem (1) is a feasible solution of the PUMMA's optimization problem (3).

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

Proof: For convenience of the proof, we denote $\theta_t = \boldsymbol{f}(\boldsymbol{w}_t)$. Without loss of generality, we can assume that an update is made at each trial $t \geq 1$. The proof for the first inequality is done by induction on t. For $t = 1$, the vector is written as $\boldsymbol{w}_1 = \boldsymbol{f}^{-1}(\boldsymbol{\theta}_1)$, where $\boldsymbol{\theta}_1 = \alpha(\boldsymbol{x}_1^{pos} - \boldsymbol{x}_1^{neg})$ for some $\alpha \geq 0$. By the definition of $\boldsymbol{u}$ and b, it holds that $\boldsymbol{u} \cdot \boldsymbol{x}_1^{pos} + b \geq 1$ and $\boldsymbol{u} \cdot \boldsymbol{x}_1^{neg} + b \leq -1$, respectively. So, we obtain

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{\theta}_1 &= \alpha(\boldsymbol{u} \cdot \boldsymbol{x}_1^{pos} - \boldsymbol{u} \cdot \boldsymbol{x}_1^{neg}) \\ &\geq \alpha(1 - b + 1 + b) = 2\alpha. \end{aligned}$$

On the other hand, by Lemma 4, we have

$$\|\boldsymbol{w}_1\|_q^2 = \boldsymbol{w}_1 \cdot \boldsymbol{\theta}_1 = \alpha \boldsymbol{w}_1 \cdot (\boldsymbol{x}_1^{pos} - \boldsymbol{x}_1^{neg}) = 2\alpha,$$

which shows $\boldsymbol{u} \cdot \boldsymbol{\theta}_1 \geq \|\boldsymbol{w}_1\|_q^2$.

Suppose that for $t < t'$, the statement is true. Then, there are two cases: (i) $\boldsymbol{w}_{t'} \cdot \boldsymbol{\theta}_{t'-1} = \|\boldsymbol{w}_{t'-1}\|_q^2$, and $\boldsymbol{w}_{t'} = \boldsymbol{f}^{-1}(\boldsymbol{\theta}_{t'})$, where $\boldsymbol{\theta}_{t'} = \alpha(\boldsymbol{x}_{t'}^{pos} - \boldsymbol{x}_{t'}^{neg}) + \beta \boldsymbol{\theta}_{t'-1}$ for some α and β, or (ii)$\boldsymbol{w}_{t'} \cdot \boldsymbol{\theta}_{t'-1} > \|\boldsymbol{w}_{t'-1}\|_q^2$, and $\boldsymbol{w}_{t'} = \boldsymbol{f}^{-1}(\boldsymbol{\theta}_{t'})$, where $\boldsymbol{\theta}_{t'} = \alpha(\boldsymbol{x}_t^{pos} - \boldsymbol{x}_t^{neg})$. For the case (ii), the proof follows the same argument for $t = 1$, so we only consider the case (i). By the inductive assumption, we have

$$\begin{aligned}\boldsymbol{u} \cdot \boldsymbol{\theta}_{t'} &= \alpha(\boldsymbol{u} \cdot \boldsymbol{x}_{t'}^{pos} - \boldsymbol{u} \cdot \boldsymbol{x}_{t'}^{neg}) + \beta \boldsymbol{u} \cdot \boldsymbol{\theta}_{t'-1} \\ &\geq 2\alpha + \beta \|\boldsymbol{w}_{t'-1}\|_q^2\end{aligned}$$

By Lemma 4,

$$\begin{aligned}\|\boldsymbol{w}_{t'}\|_q^2 &= \boldsymbol{w}_{t'} \cdot \boldsymbol{\theta}_{t'} \\ &= \boldsymbol{w}_{t'} \cdot \alpha(\boldsymbol{x}_{t'}^{pos} - \boldsymbol{x}_{t'}^{neg}) + \beta \boldsymbol{w}_{t'} \boldsymbol{\theta}_{t'-1} \\ &= 2\alpha + \beta \|\boldsymbol{w}_{t'-1}\|_q^2.\end{aligned}$$

So, we get $\boldsymbol{u} \cdot \boldsymbol{\theta}_{t'} \geq \|\boldsymbol{w}_{t'}\|_q^2$ and thus we prove the first inequality. The second inequality holds immediately since both $(\boldsymbol{u}, b)$ and $(\boldsymbol{w}_t, b_t)$ satisfy the same constraints in (3) and $(\boldsymbol{w}_t, b_t)$ minimizes the norm by definition. ■

Next, prove the following lemma:

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

$$\|\boldsymbol{w}_{t+1}\|_q^2 - \|\boldsymbol{w}_t\|_q^2 \geq \frac{\delta^2}{2(p-1)R^2},$$

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

Proof: By the weak duality theorem (see, e.g.,[6]), the optimum of the problem (3) is bounded below by the Lagrangian dual $L^*(\alpha, \beta)$ in (18) for any $\alpha \geq 0$ and $\beta \geq 0$. Therefore, using the notations in the derivation of update,

$$\frac{1}{2}\|\boldsymbol{w}^*\|_q^2 - \frac{1}{2}\|\boldsymbol{v}\|_q^2 \geq L^*(\alpha, \beta) - \frac{1}{2}\|\boldsymbol{v}\|_q^2.$$

So, by using Proposition 1 and letting $\beta = 1$, we have

$$\begin{aligned}&L^*(\alpha, 1) - \frac{1}{2}\|\boldsymbol{v}\|_q^2 \\ =& -\boldsymbol{G}(\boldsymbol{\theta} + \alpha \boldsymbol{z}) + \boldsymbol{G}(\boldsymbol{\theta}) + 2\alpha \\ \geq& -\boldsymbol{g}(\boldsymbol{\theta}) \cdot \alpha \boldsymbol{z} - \frac{(p-1)}{2}\alpha^2 \|\boldsymbol{z}\|_p^2 + 2\alpha \\ =& -\alpha \boldsymbol{v} \cdot \boldsymbol{z} - \frac{(p-1)}{2}\alpha^2 \|\boldsymbol{z}\|_p^2 + 2\alpha.\end{aligned}$$

The right hand side of the inequality above is maximized if

$$\alpha = \frac{2 - \boldsymbol{v} \cdot \boldsymbol{z}}{(p-1)\|\boldsymbol{z}\|_p^2}. \qquad (21)$$

Note that α is positive since $\boldsymbol{v} \cdot \boldsymbol{z} \leq 2 - \delta$. Subsisting (21),

$$L^*(\alpha, 1) - \frac{1}{2}\|\boldsymbol{v}\|_q^2 \geq \frac{(2 - \boldsymbol{v} \cdot \boldsymbol{z})^2}{2(p-1)\|\boldsymbol{z}\|_p^2} \geq \frac{\delta^2}{2(p-1)R^2}.$$

■

Now we are ready to prove our main result.

Theorem 3 Suppose that for a sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{w}_T, y_T))$, there exists a hyperplane $(\boldsymbol{u}, b) \in \mathbb{R}^n \times \mathbb{R}$ such that $y_t(\boldsymbol{u} \cdot \boldsymbol{x}_t + b) \geq 1$ for $t = 1, \ldots, T$ and the hyperplane $(\boldsymbol{u}, b)$ has p-norm margin γ over S. Further, let $R = \max_{t=1,\ldots,T} \|\boldsymbol{x}_t\|_p$. (i) Then the number of updates made by PUMMA$_p(\delta)$ is at most

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

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

Proof: As in Lemma 5, without loss of generality, we assume that PUMMA updates for $t = 1, \ldots, M (M \leq T)$. By Lemma 5, we have $\|\boldsymbol{w}_t\|_q \leq \|\boldsymbol{u}\|_q$ for $t \geq 1$. Further, by Lemma 6, it holds that after M updates

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

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

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

■

Since it holds that $\|\boldsymbol{x}\|_p \leq \|\boldsymbol{x}\|_1$ for $p \geq 1$ and $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_p \leq n^{1/p}\|\boldsymbol{x}\|_\infty$, we obtain the following corollary (A similar result was shown in [13]).

Corollary 4 Assume that for a sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{w}_T, y_T))$, there exists a hyperplane $(\boldsymbol{u}, b) \in \mathbb{R}^n \times \mathbb{R}$ such that $y_t(\boldsymbol{u} \cdot \boldsymbol{x}_t + b) \geq 1$ for $t = 1, \ldots, T$ and the hyperplane $(\boldsymbol{u}, b)$ has ∞-norm margin γ over S. Further, let $R = \max_{t=1,\ldots,T} \|\boldsymbol{x}_t\|_\infty$. Then, by setting $p = c \ln n$ $(c > 0)$, (i) the number of updates made by PUMMA$_p(\delta)$ is at most

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

(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 Kernel and Soft Margin Extensions

4.1 Kernel Extension

As well as SVM, ROMMA and other Perceptron-like online algorithms, PUMMA can use kernel functions for $p = 2$. Note that, at trial t, the weight vector $\boldsymbol{w}_t$ is written as

$$\boldsymbol{w}_t = \sum_{j=1}^{t-1}\left(\prod_{n=j+1}^{t-1}\alpha_n\right)\beta_j \boldsymbol{z}_j,$$

thus an inner product $\boldsymbol{w}_t \cdot \boldsymbol{x}_t$ is given as a weighted sum of inner products $\boldsymbol{x}_j \cdot \boldsymbol{x}_{j'}$ between instances since $\boldsymbol{z}_j = \boldsymbol{x}_j^{pos} - \boldsymbol{x}_j^{neg}$. Therefore, we can apply kernel methods by replacing each inner product $\boldsymbol{x}_j \cdot \boldsymbol{x}_{j'}$ with $K(\boldsymbol{x}_j, \boldsymbol{x}_{j'})$ for some kernel K. More practically, we can compute the inner products between $\boldsymbol{w}_t$ and a mapped instance using the recurrence $\boldsymbol{w}_t = \alpha_t(\boldsymbol{x}_t^{pos} - \boldsymbol{x}_t^{neg}) + \beta_t \boldsymbol{w}_{t-1}$.

4.2 2-norm Soft Margin Extension

In order to apply PUMMA to linearly inseparable data, as in [21, 22], we employ the 2-norm soft margin minimization [9, 10], which is formulated as follows: Given a sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T))$ and letting $\mathcal{S}$ be the *set* of examples in S,

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^q + \frac{C}{2} \sum_{(\boldsymbol{x}, y)\in\mathcal{S}} \xi^2_{\boldsymbol{x}}, \tag{22}$$

subject to

$$y(\boldsymbol{w}\cdot\boldsymbol{x} + b) \geq 1 - \xi_{\boldsymbol{x}} \ \ ((\boldsymbol{x}, y) \in \mathcal{S}),$$

where the constant $C > 0$ is given as a parameter. Here, we implicitly assume that labels are consistent, i.e., if $\boldsymbol{x}_t = \boldsymbol{x}_{t'}$ then $y_t = y_{t'}$. So we drop y from the subscript of ξ.

For $p = 2$, it is well known that this formulation is equivalent to the 2-norm minimization problem over linearly separable examples in an augmented space:

$$\min_{\tilde{\boldsymbol{w}}, b,} \frac{1}{2}\|\tilde{\boldsymbol{w}}\|^2,$$

subject to:

$$y(\tilde{\boldsymbol{w}}\cdot\tilde{\boldsymbol{x}} + b) \geq 1 \ (\boldsymbol{x} \in \mathcal{S}),$$

where $\tilde{\boldsymbol{w}} = (\boldsymbol{w}, \sqrt{C}\boldsymbol{\xi})$, $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \frac{y}{\sqrt{C}}\boldsymbol{e}_{\boldsymbol{x}})$ for each $(\boldsymbol{x}, y) \in \mathcal{S}$, and each $\boldsymbol{e}_{\boldsymbol{x}}$ is a unit vector in $\mathbb{R}^{|\mathcal{S}|}$ whose element corresponding to $\boldsymbol{x}$ is 1 and other elements are set to 0. To use a kernel function K with this soft margin formulation, we just modify K as follows:

$$\tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{\Delta_{ij}}{C}, \tag{23}$$

where $\Delta_{ij} = 1$ if $i = j$, otherwise $\Delta_{ij} = 0$.

For $p > 2$, we modify PUMMA so that, given $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{t-1}, y_{t-1}))$ and an instance $\boldsymbol{x}_t$, it predicts $\hat{y}_t = \mathrm{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t + b_t)$, where $(\boldsymbol{w}_t, b_t, \boldsymbol{\xi}_t)$ is specified as follows:

$$(\boldsymbol{w}_t, b_t, \boldsymbol{\xi}_t) = \arg\min_{\boldsymbol{w}, b, \xi} \frac{1}{2}\|\boldsymbol{w}\|^2_q + \frac{C}{2} \sum_{(\boldsymbol{x}, y)\in\mathcal{M}_t} \xi^2_{\boldsymbol{x}}, \tag{24}$$

subject to:

$$\begin{aligned} &\boldsymbol{w}\cdot\boldsymbol{x}^{pos}_t + b \geq 1 - \xi^{pos}_t, && (25)\\ &\boldsymbol{w}\cdot\boldsymbol{x}^{neg}_t + b \leq -1 + \xi^{neg}_t, \\ &\boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{w}_{t-1}) + C \sum_{(\boldsymbol{x}, y)\in\mathcal{M}_{t-1}} \xi_{\boldsymbol{x}}\xi_{t-1,\boldsymbol{x}} \geq \\ &\|\boldsymbol{w}_{t-1}\|^2_q + C \sum_{(\boldsymbol{x}, y)\in M_{t-1}} \xi^2_{t-1,\boldsymbol{x}}, \end{aligned}$$

where $\mathcal{M}_t$ denotes the set of examples in S which have incurred updates of PUMMA in $t-1$ trials, $\xi^{pos}_t = \xi_{\boldsymbol{x}^{pos}_t}$ and $\xi^{neg}_t = \xi_{\boldsymbol{x}^{neg}_t}$. Then the modified PUMMA update $\boldsymbol{x}^{pos}_{t+1}$ or $\boldsymbol{x}^{neg}_t$ if $y_{t+1}(\boldsymbol{w}_{t+1}\cdot\boldsymbol{x}_{t+1} + b_{t+1}) < 1 - \delta - \xi_{\boldsymbol{x}_{t+1}}$, where $\xi_{\boldsymbol{x}_{t+1}} = \xi_{\boldsymbol{x}'_t}$ if $\boldsymbol{x}_{t+1} = \boldsymbol{x}_{t'}$ such that $(\boldsymbol{x}_{t'}, y_{t'}) \in \mathcal{M}_t$. Otherwise, $\xi_{\boldsymbol{x}_{t+1}} = 0$.

Solution The Lagrangian function is given as

$$\begin{aligned} &L(\boldsymbol{w}, b, \boldsymbol{\xi}, \alpha, \beta) \\ &= \frac{1}{2}\|\boldsymbol{w}\|^2_q + \frac{C}{2} \sum_{(\boldsymbol{x}, y)\in\mathcal{M}_t} \xi^2_{\boldsymbol{x}} \\ &+ \sum_{\ell\in\{pos,neg\}} \alpha^\ell(1 - \xi^\ell_t - y^\ell_t \boldsymbol{w}\cdot\boldsymbol{x}^\ell_t) \\ &+ \beta\Bigg(\|\boldsymbol{w}_{t-1}\|^2_q - \boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{w}_{t-1}) \\ &+ C\sum_{\boldsymbol{x}\in\mathcal{M}_{t-1}} {\xi_{t-1,\boldsymbol{x}}}^2 - C\sum_{\boldsymbol{x}\in\mathcal{M}_{t-1}} \xi_{\boldsymbol{x}}\xi_{t-1,\boldsymbol{x}} \Bigg). \end{aligned}$$

To simplify descriptions, without loss of generality, we assume that $\boldsymbol{x}_t$ is a positive instance. Note that every solution is as same as when x_t is a negative one. As done in the separable case, by using KKT conditions, we consider the following two cases:

(i)Suppose that $\beta = 0$, $\alpha > 0$. Then the optimal solution $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*)$ is given as

$$\begin{aligned} \boldsymbol{w}^* &= \alpha\boldsymbol{f}^{-1}(\boldsymbol{z}), \\ \alpha &= \frac{2}{\frac{2}{C} + \|\boldsymbol{z}\|^2_p}, \\ \xi^{pos*}_t &= \xi^{neg*}_t = \frac{\alpha}{C}, \\ \xi^*_{\boldsymbol{x}} &= 0 \quad (\boldsymbol{x}\in\mathcal{M}_t\backslash\{\boldsymbol{x}^{pos}_t, \boldsymbol{x}^{neg}_t\}), \end{aligned}$$

where $\boldsymbol{z} = \boldsymbol{x}^{pos}_t - \boldsymbol{x}^{neg}_t$. (ii)Otherwise, $\beta \neq 0$, $\alpha > 0$. Let $\boldsymbol{\theta} = \boldsymbol{f}(\boldsymbol{w}_{t-1})$. Then we have

$$\begin{aligned} \boldsymbol{w}^* &= \boldsymbol{f}^{-1}(\alpha\boldsymbol{z} + \beta\boldsymbol{\theta}), \\ \xi^{pos*}_t &= \begin{cases} \frac{\alpha}{C}, & \text{if } (\boldsymbol{x}_t, y_t) \notin \mathcal{M}_t, \\ \frac{\alpha}{C} + \beta\xi^{neg}_{t-1}, & \text{if } (\boldsymbol{x}_t, y_t) \in \mathcal{M}_t, \end{cases} \\ \xi^{neg*}_t &= \frac{\alpha}{C} + \beta\xi^{neg}_{t-1}, \\ \xi_{\boldsymbol{x}} &= \beta\xi_{t-1,\boldsymbol{x}} \quad (\boldsymbol{x}\in\mathcal{M}_t\backslash\{\boldsymbol{x}^{pos}_t, \boldsymbol{x}^{neg}_t\}), \end{aligned}$$

where α and β are the maximizers of the Lagrange dual

$$\begin{aligned} L^*(\alpha, \beta) &= \min \boldsymbol{w}, b, \boldsymbol{\xi} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \alpha, \beta) \\ &= \frac{1}{2}\|\alpha\boldsymbol{z} + \beta\boldsymbol{\theta}\|^2_p \\ &- 2\alpha + \frac{\alpha^2}{C} + \alpha\beta\xi^{neg}_{t-1} \\ &- \beta\|\boldsymbol{\theta}\|^2_p - C(\beta - \frac{\beta^2}{2}) \sum_{\boldsymbol{x}\in M_{t-1}} \xi^2_{\boldsymbol{x},j}. \end{aligned}$$

Again, we can approximate (α, β) by repeating the Newton update

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

for sufficiently many steps, where

$$\frac{\partial^2 L^*}{\partial^2 \alpha} = \sum_i \boldsymbol{f}^{-1'}(\alpha \boldsymbol{z} + \beta \boldsymbol{\theta})_i z_i^2 + \frac{2}{C}$$

$$\frac{\partial^2 L^*}{\partial \beta \partial \alpha} = \frac{\partial^2 L^*}{\partial \alpha \partial \beta} = \sum_i \boldsymbol{f}^{-1'}(\alpha \boldsymbol{z} + \beta \boldsymbol{\theta})_i z_i \theta_i + \xi_{t-1}^{neg}$$

$$\frac{\partial^2 L^*}{\partial^2 \beta} = \sum_i \boldsymbol{f}^{-1'}(\alpha \boldsymbol{z} + \beta \boldsymbol{\theta})_i \theta_i^2 + C \sum_{\boldsymbol{x} \in M_{t-1}} \xi_{\boldsymbol{x},j}^2 .$$

As in the case without soft margin, in order to acquire the solution $\boldsymbol{w}^*$ and $\boldsymbol{\xi}^*$, we first assume the case (i) and check whether the third constraint of the problem (24) holds with strict inequality or not. If it does, then the case (i) is true. Otherwise, the case (ii) holds. Finally, the bias b^* is given as

$$b^* = -\frac{\boldsymbol{w}^* \cdot \boldsymbol{x}^{pos} + \boldsymbol{w}^* \cdot \boldsymbol{x}^{neg} + (\xi_t^{pos} - \xi_t^{neg})}{2}.$$

By the same argument as Section 3, we obtain the following:

Theorem 5 For a sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{w}_T, y_T))$, let $(\boldsymbol{u}, b, \boldsymbol{\xi}) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{|S|}$ be the optimal solution of the problem (22). Further, let $R = \max_{t=1,\ldots,T} \|\boldsymbol{x}_t\|_p$. (i) Then the number of updates made by PUMMA$_p(\delta)$ is at most

$$O\left(\frac{\{(p-1)R^2 + \frac{2}{C}\} \left(\|\boldsymbol{u}\|_q^2 + \sum_{(\boldsymbol{x},y) \in S} \xi_{\boldsymbol{x}}^2 \right)}{\delta^2} \right).$$

(ii) PUMMA$_p(\delta)$ outputs a hypothesis with p-norm margin whose objective value for the problem (22) is at most $\frac{1}{(1-\delta)^2}$ times the optimum after at most the updates above.

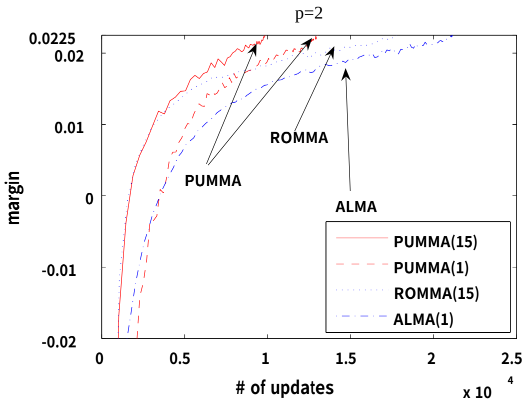
5 Experiments

5.1 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 $f(\boldsymbol{x}) = \text{sign}(x_{i_1} + \cdots + x_{i_k} + k - 2r + 1)$ for some $i_1, \ldots, i_k$ s.t. $1 \le i_1 \le i_2 \le \cdots \le i_k \le n$, 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 $R = \max ||\boldsymbol{x}||_p$. Note that one can choose different values other than $-R$, say, 1. However, as remarked in [10], such a choice for the value in the extra dimension increases the number of iterations by $O(R^2)$ times when the underlying hyperplane has large bias. So our choice seems to be fair.

We set parameters so that each algorithm is guaranteed to achieve at least 0.9 times the maximum p-norm margin.



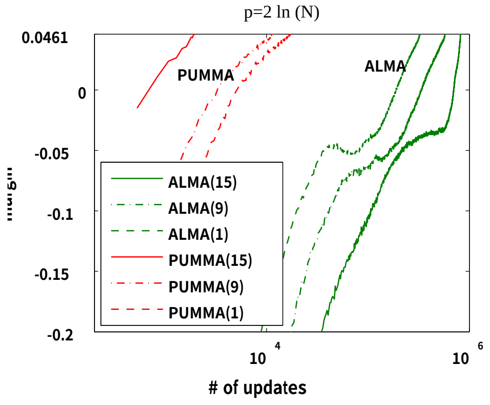


Figure 3: Number of updates and margin over artificial data set in the case $p = 2$ (upper) and $p = 2\ln(n)$ (lower). We set x-axes log scale since the numbers of updates of ALMA are quite larger than PUMMA 's. And we hide the result of the case $p = 2$ and $b = 9$ since we make the figure easy to view.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

That is, we set $\alpha = 0.1$ (note the parameter α is defined differently in [13]) 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, 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 represented in Figure 3 and 4. We observe that PUMMA converges faster. PUMMA 's computation time is quite shorter than that of ALMA, although it uses Newton method in each update, 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 α and β by solving the optimization problem directly.

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

5.2 Experiments over some UCI datasets

We compare PUMMA with some other learning algorithms over the real datasets. The algorithms are SVM^{light} [16], MICRA [35], and ROMMA [22]. 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 [29]. 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, we use the following modified inner product

$$IP(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \frac{\Delta_{ij}}{C}.$$

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 modify SVM^{light} so as not to optimize 1-norm soft margin, and we change the inner product so that it optimizes 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 [35]. 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 $C = 1$ for all algorithms. In order to converge faster, we use the following heuristics for each online algorithm.

Active Set We try to improve the order of given examples to feed for each online algorithm. First, we give all the examples to each online learning algorithm once. Then, we make a new dataset called "active set" , containing the examples which causes updates. After that, we give each example in the active set to the algorithm. If the example doesn't cause any updates, we remove the example from the active set, and we repeat this procedure until the active set becomes empty. Finally, we give all the examples again and check if the algorithm makes any updates. If some updates occur, we construct an active set again and repeat the whole procedure.

We run each algorithm and we measure its real computation time as well as its 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 and obtained margin. As can be seen, PUMMA converges quite faster than ROMMA. On the other hand, PUMMA converges slower than MICRA. However, the parameters of MICRA are quite sensitive to datasets and nontrivial to tune appropriately. The results on all the real data set show that SVM^{light} is the fastest, whereas MICRA is reported to be faster than SVM^{light} over some datasets and with tuned parameters [35]. Note that this might be due to our selection of active sets which is different from theirs.

5.3 Experiments over MNIST dataset

Next, we compare these algorithms over MNIST dataset. Since the dataset is not linearly separable, we use polynomial kernel and 2-norm soft margin as follows.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(1 + \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{s}\right)^d + \frac{\Delta_{i,j}}{C}.$$

Since computing kernels is time-consuming, we use some extra heuristics in addition to our active set selection.

Kernel Cache Since we have to compute kernel values of the same examples repeatedly, we memorize them in a cache matrix. In the cache matrix, each row memorizes the kernel values of a support vector and all the examples, where a support vector is an instance which causes an update. The length of each row equals to the number of training instances. The number of rows depends on the memory size. When the new kernel value of a support vector and an instance is required, we search the cached value in the cache matrix. If we fails, we calculate the value and store it in the cache matrix. To do this, we search the row of the corresponding support vector. We store the value if we succeed, or we make the new row otherwise. If the matrix is full, we replace the least referenced row by the new row.

Inner Product Cache In our experiments, we keep giving examples to each online leaning algorithms until they

Table 2: Computation time (sec.) and obtained margin (denoted as γ') on MNIST datasets.

	SVM^{light}		SVM^{light} w/o bias		PUMMA		ROMMA	
class	sec.	γ'	sec.	γ'	sec.	γ'	sec.	γ'
0	256.51	1.339	164.83	1.155	373.48	1.330	218.97	1.150
1	152.10	0.712	119.62	0.712	291.54	0.706	231.82	0.708
2	413.43	0.810	309.77	0.765	1674.08	0.804	870.58	0.761
3	566.84	0.763	384.17	0.722	2654.19	0.757	2296.34	0.719
4	333.04	0.650	267.65	0.629	905.16	0.645	505.11	0.626
5	428.36	0.672	301.99	0.664	1480.44	0.667	1007.91	0.661
6	246.47	0.941	184.39	0.880	534.80	0.934	308.18	0.876
7	322.90	0.621	304.54	0.611	860.89	0.616	584.36	0.608
8	694.17	0.810	437.48	0.727	5648.12	0.804	5074.51	0.723
9	599.78	0.558	399.08	0.541	5290.33	0.554	6057.92	0.538
avg.	401.36	0.788	287.35	0.741	1971.30	0.782	1715.57	0.737

make no update on all the examples. Assume that at trial $t = t_1, t_2 (t_1 < t_2)$, the weight vector $\boldsymbol{w}_t$ is updated by the same example x_{t_1}. The weight vector $\boldsymbol{w}_{t_2}$ is written as

$$\begin{aligned}\boldsymbol{w}_{t_2} &= \sum_{j=1}^{t_2-1}\left(\prod_{k=j+1}^{t_2-1}\alpha_k\right)\beta_j\boldsymbol{z}_j \\ &= \left(\prod_{k=t_1}^{t_2-1}\alpha_k\right)\boldsymbol{w}_{t_1} + \sum_{j=t_1}^{t_2-1}\left(\prod_{n=j+1}^{t_2-1}\alpha_n\right)\beta_j\boldsymbol{z}_j.\end{aligned}$$

So, if we memorize the inner product $\boldsymbol{w}_{t_1}\cdot\boldsymbol{x}_{t_1}$, we can calculate $\boldsymbol{w}_{t_2}\cdot\boldsymbol{x}_{t_1}$ easier. This technique is efficient when we use kernel.

Halving δ It is reported that by decreasing δ in a dynamical way, ROMMA converges faster [22]. Similar to their approach, we shrink the parameter δ by halving repeatedly. More precisely, we set $\delta = 1$ at first, and halve δ when the algorithm makes no update for all the examples. We repeat this procedure until δ is as small as we require. Note that if δ is smaller than the required value δ_{target}, we set $\delta = \delta_{\text{target}}$. When we use kernels, this halving heuristics can reduce support vectors in the early stage of learning, which contributes faster convergence.

MNIST dataset contains $60,000$ matrix and labels. Each (28×28) matrix represents the image of the hand written digit. The value of each element is in $\{0,\cdots,255\}$, which denotes the density. Each label takes the value $\{0,\cdots,9\}$. MNIST dataset has 10 classes. Since each algorithm can deal with only binary class, we change each label so that one class is positive and the others are negative. Then we get 10 binary labeled datasets.

We run three learning algorithms, SVM^{light}, ROMMA and PUMMA on these datasets until they converge. We omit the evaluation of MICRA since it needs careful tuning of parameters to converge fast. We record the real computation time and margin. Note that we use our heuristics for ROMMA and PUMMA . And we set some kernel parameters, $s = 1100^2$, $d = 5$ and $C = 1/30$ as in [22]. We set $\delta_{\text{target}} = 0.01$ and use 1 GB kernel cache. We also run SVM^{light} with the same size of cache memory, but its caching strategy is different from ours. The experiments on MNIST dataset are conducted on the same machine as the experiments on UCI dataset.

The results are shown in Table 2. PUMMA gains higher margin than ROMMA over almost all of the datasets. On the other hand, PUMMA requires more computation time. This seems to be due to the fact that ROMMA solves the different optimization problem, i.e., maximization of margin without bias. We observe the same tendency between SVM^{light} with and without bias. Further, computation times of PUMMA are worse than SVM^{light}. But, PUMMA and ROMMA might be improved if we employ a different strategy for active set selection.

6 Conclusion and Future work

In this paper, we propose PUMMA which obtains the maximum p-norm margin classifier with bias approximately. Our algorithm often runs faster than previous online learning algorithms when the underlying linear classifier has large bias, by taking advantage of finding bias directly.

Although the worst case upperbound on iterations of our algorithm is the same as those of previous algorithms, our experiments over artificial datasets suggest that our iteration bound might be better. For example, when the target function is a r-of-k threshold function, iteration bound of PUMMA is $O(k^2 \ln n)$ with $p = O(\ln n)$. However, in our experiments, PUMMA seems to converge in $O(rk \ln n)$ iterations, which is the best upperbound obtained by Winnow when k and r are known a priori. Unfortunately, we have not yet succeeded in proving better iteration bounds. It is still open if there exists an online learning algorithms that learns r-of-k threshold functions in $O(rk \ln n)$ updates without knowing k and r [23].

So far PUMMA or ALMA approximates ∞-norm margin indirectly by setting $p = O(\ln n)$. Developing an adaptive online algorithm that directly maximizes ∞-norm margin is also an open problem. One of the 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 applica-

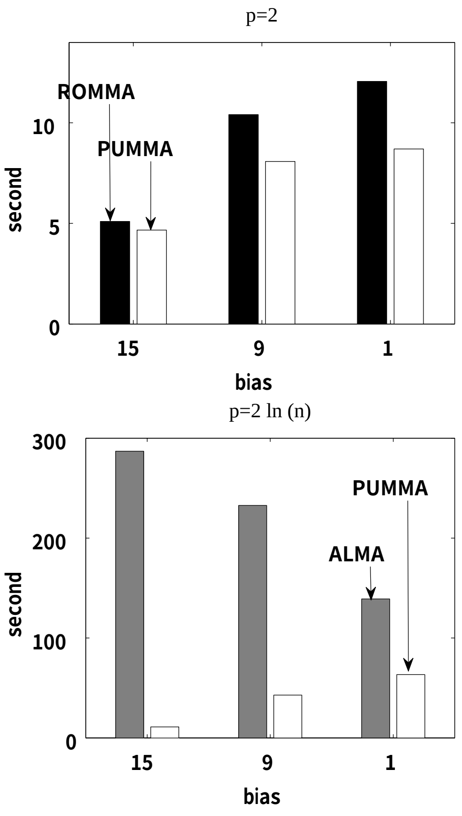


Figure 4: Computation time over artificial data set in the case $p = 2$ (upper) and $p = 2\ln(n)$ (lower).

tions.

Acknowledgments

We thank anonymous referees for helpful comments.

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] H. H. Bauschke and P. L. Combettes. A weak-to-strong convergence principle for Fejér-monotone methods in hilbelt spaces. *Mathematics of Operations Research*, 26(2):248–264, 2001.

[4] A. Bordes, S. Ertekin, J. Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

[5] 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.

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

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

[8] 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.

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

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

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

[12] 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.

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

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

[15] 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.

[16] 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.

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

[18] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.

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

[20] 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.

[21] 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.

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

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

[24] P. M. Long and X. Wu. Mistake bounds for maximum entropy discrimination. In *Advances in Neural Infor-*

mation Processing Systems 17, pages 833–840, 2004.

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

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

[27] 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.

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

[29] 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.

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.