Simple Online Learning with Approximate Max Margin and Fast Convergence

Kai Zhao KZHAO@GC.CUNY.EDU
Computer Science Program, Graduate Center, CUNY

Liang Huang HUANG@CS.QC.CUNY.EDU
Department of Computer Science, Queens College, CUNY

Abstract

We present a family of simple online learning algorithms called "Aggressive MIRA" (AMIRA) which includes MIRA and Passive-Aggressive as two extreme special cases. Theoretically, we show that these algorithms have a provable approximation ratio of the optimal margin, and are guaranteed to converge after finite number of updates. The convergence rate $O(1/\epsilon)$ is faster than the $O(1/\epsilon^2)$ rate found in most previous work, being the first algorithm to converge in $O(1/\epsilon)$ time with an approximation guarantee. These algorithms also generalize easily to handle multiclass problems and kernels. Empirically, our algorithms lead to larger margins, faster convergence, and better accuracies than previous algorithms such as Pegasos, ROMMA, and ALMA on both synthetic and real datasets.

1. Introduction

Support vector machines (SVM) (Cortes & Vapnik, 1995) is one of the most powerful machine learning algorithms because it outputs the separating hyperplane with maximum margin. However, it is a batch learning algorithm whose training procedure involves large-scale quadratic programming and is known to be slow even with decomposition methods, generally scaling between quadratic and cubic in the number of examples (Platt, 1998).

On the other hand, online learning methods such as Rosenblatt's Perceptron (Rosenblatt, 1959) and its variants such as the voted and averaged Perceptron (Freund & Schapire, 1999) and Margin-Infused Relaxation Algorithm (MIRA) (Crammer & Singer, 2003) offer much simpler implementation and faster convergence, but the resulting margin (and accuracy) from these algorithms is often inferior to that of SVM. To address this problem, several authors have proposed fancier versions of Perceptron-like algorithms that provably approximate maximum margin classifiers, including most notably ROMMA (Li & Long, 2002), ALMA (Gentile, 2002), and PUMMA (Ishibashi et al., 2008). However, these algorithms are significantly harder to implement than the original Perceptron, and more importantly, their convergence rate is much slower, being $O(\frac{1}{\epsilon^2})$ times slower than the original Perceptron in order to achieve a geometric margin of $(1-\epsilon)\gamma$ where γ is the optimal geometric margin.[1]

Table 1. Comparison of algorithms in terms of approximation ratios and convergence rates. Note that Pegasos (Shalev-Shwartz et al., 2007) approximates the SVM objective function (the sum of weight norm and hinge-loss on errors) and is thus not comparable to other algorithms which only approximates the max margin. Our work is the only one that converges in $O(1/\epsilon)$ with a provable approximation ratio.

Algorithm	approx.	approx. ratio	convergence
Perceptron	-	-	$O(1)$
ALMA	max margin	$1-\epsilon$	$O(1/\epsilon^2)$
ROMMA	max margin	$1-\epsilon$	$O(1/\epsilon^2)$
our AMIRA	max margin	$\frac{1-\epsilon}{2-\epsilon}$	$O(1/\epsilon)$
Pegasos	SVM objective	$1-\epsilon$	$\tilde{O}(1/\epsilon)$

We instead propose a family of simpler online learning algorithms which include MIRA and Passive-Aggressive (PA)[2] (Crammer et al., 2006) as two extreme cases. While MIRA updates whenever there is a prediction mistake and PA updates whenever the functional margin is less than one, our algorithm, called Aggressive MIRA or AMIRA, has a parameter ϵ $(0 \le \epsilon \le 1)$, and aggressively triggers an update whenever the functional margin is less than $1-\epsilon$.[3]

[1] In this paper we use two kinds of margins: the (unnormalized) **functional** margin $y(\mathbf{w}\cdot\mathbf{x})$, and the (normalized) **geometric** margin $\frac{y(\mathbf{w}\cdot\mathbf{x})}{\|\mathbf{w}\|}$. The approximation ratios are about the latter.

[2] Here we only consider the vanilla PA, not PA-I or PA-II. The latter two have additional parameters which make them harder to use in practice.

[3] In the following sections, we sometimes call $1-\epsilon$ the *ag-*

Last updated: October 5, 2013.

Thus MIRA is the most conservative case ($\epsilon = 1$) and PA the most aggressive case ($\epsilon = 0$) in this new framework. We make the following contributions:

1. We provide a unified proof of convergence for AMIRA using very different techniques from the original proofs in MIRA and PA, greatly simplifying the convergence proofs for MIRA and PA as a by-product. We also bound the number of mistakes of AMIRA for inseparable datasets.
2. We present a margin-approximation theorem that the final geometric margin from AMIRA(ϵ) is at least $\frac{1-\epsilon}{2-\epsilon}\gamma$, after at most $\frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2}$ updates ($R = \max_i \|\mathbf{x}_i\|$).
3. We show that empirically, the approximation ratio is much better than the above theoretical analysis, leading to a margin of at least $(1-\epsilon)\gamma$ on synthetic data.
4. We also show that on synthetic data our convergence rate, being $O(1/\epsilon)$, is indeed much faster than ROMMA's $O(1/\epsilon^2)$ in practice (see Table 1). To the best of our knowledge, AMIRA is the first online algorithm that converges in $O(1/\epsilon)$ with a provable approximation ratio on margin (by contrast, the famous Pegasos algorithm (Shalev-Shwartz et al., 2007) approximates a different objective function and converges in $\tilde{O}(1/\epsilon)$, which is still slower than $O(1/\epsilon)$).
5. Finally we show that AMIRA extends easily to incorporate kernels, which leads to better accuracies and faster convergence than previous methods (Pegasos, ROMMA, ALMA, etc.) on real datasets such as USPS and MNIST.

2. The Algorithm

We start with AMIRA in the standard binary classification setting, and then generalize it for multiclass problems.

2.1. Binary AMIRA

In the standard online binary classification setting, as shown in Algorithm 1, at time t, AMIRA considers each training example $(\mathbf{x}, y) \in D \subset \mathbb{R}^n \times \{-1, +1\}$ sequentially and performs two steps:

1. calculate functional margin $y(\mathbf{w}\cdot\mathbf{x})$, which represents the unnormalized distance of $\mathbf{x}$ to the separating hyperplane described by current weight vector $\mathbf{w}$, and
2. if the functional margin is not larger than $1-\epsilon$, which we call a "violation", update current weight vector:

$$\mathbf{w}' \leftarrow \mathbf{w} + \frac{y - \mathbf{w}\cdot\mathbf{x}}{\|\mathbf{x}\|^2}\mathbf{x} \quad (1)$$

gressiveness parameter of the algorithm.

Algorithm 1 Binary-AMIRA$_\epsilon$

Input: training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and parameter ϵ
Output: weight vector $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
repeat
  for each example $(\mathbf{x}, y)$ **in** D **do**
    if $y(\mathbf{w}\cdot\mathbf{x}) \le 1-\epsilon$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \frac{y-\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{x}\|^2}\mathbf{x}$
until converged

The update procedure in the second step has a very clear geometric interpretation: it searches for the new weight vector $\mathbf{w}'$ with minimum change from current weight vector $\mathbf{w}$ so that $\mathbf{w}'$ falls into the half space described by $y(\mathbf{w}\cdot\mathbf{x}) - 1 \ge 0$. This is actually a quadratic programming problem:

$$\mathbf{w} \leftarrow \mathbf{argmin}\|\mathbf{w}' - \mathbf{w}\|^2$$
$$\text{s.t. } y(\mathbf{w}'\cdot\mathbf{x}) \ge 1.$$

From the geometric intuition in Figure 1 we can see the change should connect the end point of $\mathbf{w}$ and its projection on the hyperplane $\mathbf{w}\cdot\mathbf{x} - y = 0$, which indicates:

$$\Delta\mathbf{w} = \mathbf{w}' - \mathbf{w} = \frac{y - \mathbf{w}\cdot\mathbf{x}}{\|\mathbf{x}\|^2}\mathbf{x},$$

from which we get Eq. 1 as the analytical solution to the quadratic programming problem.

Note that MIRA and PA are two extreme special cases of AMIRA, for $\epsilon = 1$ and $\epsilon = 0$ respectively (see Figure 1).

2.2. 1-best Multiclass AMIRA

In multiclass setting, each training example $\mathbf{x}$ is associated with a gold label $y \in \{1, 2, \ldots, M\}$ for M classes. We generalize our problem similar to (Crammer & Singer, 2003) with some slight changes. The differences are:

1. instead of using a weight matrix of shape $M \times n$ for M classes, we use a long weight vector $\mathbf{w} \in \mathbb{R}^{nM}$ that is a concatenation of weight vectors:

   $$\mathbf{w} = (\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(M)}),$$

   where $\mathbf{w}^{(i)}$ is used to calculate the functional margin for training example with label i;
2. for a given training example $\mathbf{x}$ and a label y, we define feature map function $\mathbf{\Phi}$ as

   $$\mathbf{\Phi}(\mathbf{x}, y) = (\mathbf{0}^{(1)}, \ldots, \mathbf{0}^{(y-1)}, \mathbf{x}, \mathbf{0}^{(y+1)}, \ldots, \mathbf{0}^{(M)}).$$

   such that $\mathbf{w}\cdot\mathbf{\Phi}(\mathbf{x}, y) = \mathbf{w}^{(y)}\cdot\mathbf{x}$.

   We also define that, with a given training example $\mathbf{x}$, the difference between two feature vectors for labels y and z as $\Delta\mathbf{\Phi}$:

   $$\Delta\mathbf{\Phi}(\mathbf{x}, y, z) = \mathbf{\Phi}(\mathbf{x}, y) - \mathbf{\Phi}(\mathbf{x}, z).$$

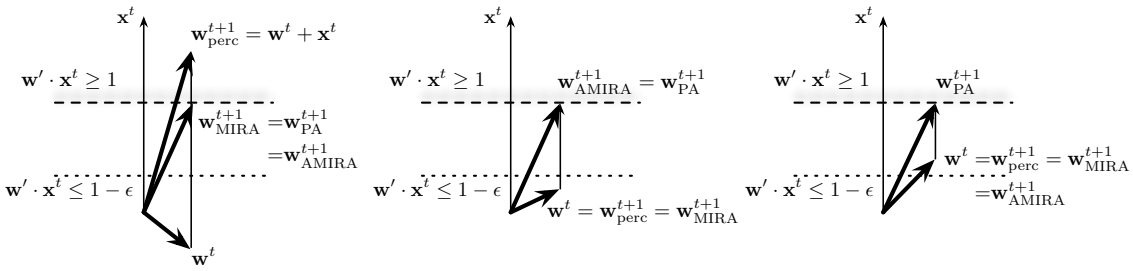

Figure 1. AMIRA updates compared to MIRA and PA. Left: Current weight vector $\mathbf{w}^t$ does not separate example $\mathbf{x}^t$ with a positive margin. Perceptron, AMIRA, MIRA and PA all update. The updates of AMIRA, MIRA, and PA guarantee a new margin of 1. Middle: Current weight vector $\mathbf{w}^t$ separates example $\mathbf{x}^t$ with a positive margin smaller than $1 - \epsilon$. Perceptron and MIRA will not update. AMIRA and PA update to guarantee a new margin of 1. Right: Current weight vector $\mathbf{w}^t$ separates example $\mathbf{x}^t$ with a positive margin larger than $1 - \epsilon$ but smaller than 1. Only PA will update to guarantee a new margin of 1.

Algorithm 2 1-best-AMIRA$_\epsilon$

Input: training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and parameter ϵ
Output: weight vector $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
repeat
  for each example $(\mathbf{x}, y)$ **in** D **do**
    $z \leftarrow \operatorname{argmax}_{z:z\in\{1,\ldots,M\}\setminus\{y\}} \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, z)$
    if $\mathbf{w} \cdot \Delta\mathbf{\Phi}(\mathbf{x}, y, z) \leq 1 - \epsilon$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \frac{1-\mathbf{w}\cdot\Delta\mathbf{\Phi}(\mathbf{x},y,z)}{\|\Delta\mathbf{\Phi}(\mathbf{x},y,z)\|^2}\Delta\mathbf{\Phi}(\mathbf{x}, y, z)$
until converged

We present 1-best AMIRA in Algorithm 2. The goal of 1-best AMIRA is to find the weight vector $\mathbf{w}$ so that, for each training example $\mathbf{x}$, it separates the feature maps $\mathbf{\Phi}(\mathbf{x}, y)$ and $\mathbf{\Phi}(\mathbf{x}, z)$ by some positive functional margin, where y is the gold label, and z is any incorrect label.

1-best AMIRA follows a scheme similar to binary AMIRA:

1. first it finds the best label z from candidate labels, which are all possible labels except the gold label y,
2. if current weight vector $\mathbf{w}$ can not separate the feature maps of y and z with a functional margin larger than $1 - \epsilon$, which is a violation, update the weight vector.

The geometric interpretation for the update of 1-best AMIRA are similar to binary AMIRA, except that the normal vector of the separating hyperplane is $\Delta\mathbf{\Phi}(\mathbf{x}, y, z)$, instead of $\mathbf{x}$. We get similar analytical solution for updating:

$$\mathbf{w}' \leftarrow \mathbf{w} + \frac{1 - \mathbf{w} \cdot \Delta\mathbf{\Phi}(\mathbf{x}, y, z)}{\|\Delta\mathbf{\Phi}(\mathbf{x}, y, z)\|^2}\Delta\mathbf{\Phi}(\mathbf{x}, y, z) \quad (2)$$

2.3. k-best Multiclass AMIRA

We further extend 1-best AMIRA to k-best AMIRA that fixes multiple "violations" in one update (see Algorithm 3).

Algorithm 3 k-best-AMIRA$_\epsilon$

Input: training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and parameter ϵ
Output: weight vector $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
repeat
  for each example $(\mathbf{x}, y)$ **in** D **do**
    $Z \leftarrow k\text{-best}_{z:z\in\{1,\ldots,K\}\setminus\{y\}} \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, z)$
    $Z \leftarrow \{z \in Z | \mathbf{w} \cdot \Delta\mathbf{\Phi}(\mathbf{x}, y, z) \leq 1 - \epsilon\}$
    if $Z \neq \emptyset$ **then**
      $\mathbf{w} \leftarrow \operatorname{argmin}_{\mathbf{w}':\forall z\in Z, \mathbf{w}'\cdot\Delta\mathbf{\Phi}(\mathbf{x},y,z)\geq 1} \|\mathbf{w}' - \mathbf{w}\|^2$
until converged

This kind of update resembles the k-best MIRA in (McDonald et al., 2005).

We call $\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, z)$ the score of label z, which is actually the signed functional margin. Instead of finding only the highest-scoring incorrect label as in 1-best AMIRA, k-best AMIRA finds k highest-scoring incorrect labels, and then checks whether current weight vector $\mathbf{w}$ separates every incorrect label and the gold label with a functional margin larger than $1 - \epsilon$. Any violations found will be put into a violation set Z. Our k-best AMIRA will update weight vector to fix all violations in Z. In practice we usually choose k to be identical to the number of classes M.

Here the update procedure in k-best AMIRA is a standard quadratic programming problem, which also has a clear geometric interpretation: it searches the weight vector $\mathbf{w}'$ with the minimum change from current weight vector $\mathbf{w}$ so that $\mathbf{w}'$ falls into a convex region, which is the intersection of half spaces described by linear violation constraints:

$$\forall z \in Z, \mathbf{w} \cdot \Delta\mathbf{\Phi}(\mathbf{x}, y, z) \geq 1$$

However, we do not have an analytical solution for this problem. In practice we use a very simple algorithm, Hildreth algorithm (Hildreth, 1957), to solve it.

3. Convergence and Margin Analysis

We first present a simple proof for the convergence of binary AMIRA for separable datasets, and then generalize it to multiclass classification (1-best and k-best AMIRA). For all the three variants of AMIRA$_\epsilon$, our proof guarantees the number of mistakes in the training is at most $\frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2}$.

This proof resembles the classical convergence proof of Perceptron (Novikoff, 1962), which is quite different from but much simpler than the original proofs given in (Crammer & Singer, 2003) for MIRA and (Crammer et al., 2006) for PA.

From the convergence proof, we find a margin approximation theorem that for AMIRA$_\epsilon$ that the output margin is at least $\frac{1-\epsilon}{2-\epsilon}\gamma$ where γ is the optimal margin.

We also analyze the number of mistakes bound of AMIRA for inseparable datasets.

3.1. Convergence of Binary AMIRA

We denote R to be the maximal radius of training examples: $R = \max_i \|\mathbf{x}_i\|$. We also assume the training set D is separable by a unit oracle vector $\mathbf{u}$ with optimal margin

$$\gamma = \max_{\mathbf{u}:\|\mathbf{u}\|=1} \min_{(\mathbf{x},y)\in D} y(\mathbf{u}\cdot\mathbf{x}).$$

Theorem 1 (convergence of binary AMIRA)**.** *For a separable dataset D with optimal margin γ and radius R, binary AMIRA (Algorithm 1) will terminate after t updates where $t \le \frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2}$.*

Proof. Let $\mathbf{w}^t$ be the weight vector **before** the t^{th} update. Suppose the t^{th} update happens on example $(\mathbf{x}^t, y^t)$.

We know that:

1. $y^t(\mathbf{u}\cdot\mathbf{x}^t) \ge \gamma$ (margin on unit vector)
2. $y^t(\mathbf{w}^t\cdot\mathbf{x}^t) \le 1-\epsilon$ (violation)
3. $\|\mathbf{x}^t\|^2 \le R^2$ (radius)

For convenience we denote $a^t = y^t(\mathbf{w}^t\cdot\mathbf{x}^t)$, and rewrite the update equation Eq. 1:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + y^t\frac{1-a^t}{\|\mathbf{x}^t\|^2}\mathbf{x}^t \tag{3}$$

We bound $\|\mathbf{w}^{t+1}\|$ in two directions:

1. Dot product both sides of Eq. 3 with $\mathbf{u}$:

$$\begin{aligned}\mathbf{u}\cdot\mathbf{w}^{t+1} &= \mathbf{u}\cdot\mathbf{w}^t + \frac{1-a^t}{\|\mathbf{x}^t\|^2}y^t(\mathbf{u}\cdot\mathbf{x}^t)\\ &\ge \mathbf{u}\cdot\mathbf{w}^t + \frac{1-a^t}{\|\mathbf{x}^t\|^2}\gamma \qquad \text{(margin)}\\ &\ge \sum_t \frac{1-a^t}{\|\mathbf{x}^t\|^2}\gamma \qquad \text{(by induction)}\end{aligned}$$

Denote $\frac{1-a^t}{\|\mathbf{x}^t\|^2}$ as d^t, we have $\mathbf{u}\cdot\mathbf{w}^{t+1} \ge \gamma\sum_t d^t$.

Since for any two vectors $\mathbf{a}$ and $\mathbf{b}$ we have $\|\mathbf{a}\|\|\mathbf{b}\| \ge \mathbf{a}\cdot\mathbf{b}$, thus $\|\mathbf{w}^{t+1}\| = \|\mathbf{u}\|\|\mathbf{w}^{t+1}\| \ge \mathbf{u}\cdot\mathbf{w}^{t+1}$, which indicates

$$\|\mathbf{w}^{t+1}\| \ge \sum_t d^t\gamma \tag{4}$$

2. Take the norm of both sides of Eq. 3:

$$\begin{aligned}\|\mathbf{w}^{t+1}\|^2 =& \|\mathbf{w}^t + y^t\frac{1-a^t}{\|\mathbf{x}^t\|^2}\mathbf{x}^t\|^2\\ =& \|\mathbf{w}^t\|^2 + \frac{(1-a^t)(1+a^t)}{\|\mathbf{x}^t\|^2}\\ \le& \|\mathbf{w}^t\|^2 + \frac{(1-a^t)(2-\epsilon)}{\|\mathbf{x}^t\|^2} \qquad \text{(violation)}\\ \le& (2-\epsilon)\sum_t \frac{1-a^t}{\|\mathbf{x}^t\|^2} = (2-\epsilon)\sum_t d^t \end{aligned} \tag{5}$$

Combining above result with Eq. 4, we have

$$\sum_t d^t \le \frac{2-\epsilon}{\gamma^2} \tag{6}$$

From violation and radius properties, we also have

$$\sum_t d^t = \sum_t \frac{1-a^t}{\|\mathbf{x}^t\|^2} \ge \sum_t \frac{\epsilon}{R^2} = t\frac{\epsilon}{R^2}$$

Combining it with Eq. 6, we have $t \le \frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2}$ □

Immediately from the upper bound of $\|\mathbf{w}^t\|$ (Eq. 5), and the upper bound of $\sum_t d^t$ (Eq. 6), we have following corollary:

Corollary 1. *For any t, we have $\|\mathbf{w}^t\| \le \frac{2-\epsilon}{\gamma}$.*

3.2. Convergence of k-best AMIRA

We also give a convergence proof for k-best AMIRA, which can be viewed as a generalization for 1-best AMIRA (we thus omit the proofs for 1-best AMIRA). This proof is nearly in parallel with the convergence proof of binary AMIRA, except that it leverages the Karush-Kuhn-Tucker (KKT) conditions for quadratic programming.

We first assume the training set D is separable by feature map $\mathbf{\Phi}$ and unit oracle vector $\mathbf{u}$ with optimal margin

$$\gamma = \max_{\mathbf{u}:\|\mathbf{u}\|=1} \min_{(\mathbf{x},y)\in D, z\neq y} \mathbf{u}\cdot\Delta\mathbf{\Phi}(\mathbf{x},y,z).$$

Let $R = \max_{(\mathbf{x},y)\in D, z\neq y} \Delta\mathbf{\Phi}(\mathbf{x}, y, z)$ be the radius.

Let $\mathbf{w}^t$ be the weight vector **before** the t^{th} update in Algorithm 3. Suppose the t^{th} update is for training example $\mathbf{x}^t$ and gold label y^t, the violation set is $Z^t = \{z_1^t, z_2^t, \ldots, z_{|Z^t|}^t\}$.

For convenience we use $\Delta\mathbf{\Phi}_i^t$ to denote $\Delta\mathbf{\Phi}(\mathbf{x}^t, y^t, z_i^t)$.

We present the update in quadratic programming form:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{argmin}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2$$
$$\text{s.t. } \forall z_i \in Z, \mathbf{w}^{t+1}\cdot\Delta\mathbf{\Phi}_i^t \geq 1 \quad (7)$$
$$\text{where } Z = \{z_i|\mathbf{w}^t\cdot\Delta\mathbf{\Phi}_i^t \leq 1-\epsilon\} \quad (8)$$

Consider the Lagrangian:

$$\mathcal{L} = \|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 + \sum_{i=1}^{|Z|}\eta_i^t(1 - \mathbf{w}^{t+1}\cdot\Delta\mathbf{\Phi}_i^t)$$

The KKT conditions require the solution $\mathbf{w}^{t+1}$ to satisfy:

1. **Stationarity**

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t, \quad (9)$$

which indicates that the weight vector change can be expressed as a linear combination of $\Delta\mathbf{\Phi}_i^t$.

2. **Complementary Slackness**

$$\forall i, \eta_i^t(\mathbf{w}^{t+1}\cdot\Delta\mathbf{\Phi}_i^t - 1) = 0$$

We call constraint i an "active constraint" if $\eta_i^t \neq 0$. For an active constraint i, we have

$$\mathbf{w}^{t+1}\cdot\Delta\mathbf{\Phi}_i^t = 1 \quad (10)$$

From above two properties we have following lemma:

Lemma 1. *The sum of Lagrangian multipliers at t^{th} update, denoted as $d^t = \sum_i \eta_i^t$, has lower bound $\frac{\epsilon}{R^2}$.*

Proof. For all active constraints in t^{th} update, w.l.o.g. we assume they are the first $\bar{k}$ constraints in Z, from the complementary slackness (Eq. 10) we have for $1 \leq i \leq \bar{k}$,

$$\mathbf{w}^{t+1}\cdot\Delta\mathbf{\Phi}_i^t = 1,$$

Scale constraint equation i with η_i^t and sum over all constraints:

$$\sum_i \eta_i^t = \mathbf{w}^{t+1}\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t$$
$$= (\mathbf{w}^t + \sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t)\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t \quad \text{(Stationarity)}$$
$$= \mathbf{w}^t\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t + (\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t)^2 \quad (11)$$
$$= \mathbf{w}^t\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t + (\sum_i \eta_i^t)^2(\sum_i \frac{\eta_i^t}{\sum_j \eta_j^t}\Delta\mathbf{\Phi}_i^t)^2$$
$$\leq \mathbf{w}^t\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t + (\sum_i \eta_i^t)^2R^2 \quad \text{(Jensen's)}$$
$$\leq (1-\epsilon)\sum_i \eta_i^t + (\sum_i \eta_i^t)^2R^2 \quad \text{(by Eq. 8)}$$

which indicates $d^t = \sum_i \eta_i^t \geq \frac{\epsilon}{R^2}$ □

Theorem 2 (convergence of k-best AMIRA)**.** *For a separable dataset D with margin γ and radius R, k-best AMIRA (Algorithm 3) will make t updates where $t \leq \frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2}$.*

Proof. 1. Dot product both sides of Eq. 9 with $\mathbf{u}$, similar to the proof for Theorem 1 we have

$$\mathbf{u}\cdot\mathbf{w}^{t+1} \geq \gamma\sum_t d^t \quad (12)$$

2. On the other hand

$$\|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w} + \sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t\|^2$$
$$= \|\mathbf{w}^t\|^2 + 2\mathbf{w}^t\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t + \|\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t\|^2$$
$$= \|\mathbf{w}^t\|^2 + \mathbf{w}^t\sum_i \eta_i^t\Delta\mathbf{\Phi}_i^t + \sum_i \eta_i^t \quad \text{(by Eq. 11)}$$
$$\leq \|\mathbf{w}^t\|^2 + (1-\epsilon)\sum_i \eta_i^t + \sum_i \eta_i^t \quad \text{(violation, Eq. 8)}$$
$$= (2-\epsilon)\sum_t\sum_i \eta_i^t = (2-\epsilon)\sum_t d^t$$

Combining above result with Eq. 12 gives an upper bound on the sum of d^ts:

$$\sum_t d^t \leq \frac{2-\epsilon}{\gamma^2} \quad (13)$$

Lemma 1 provides a lower bound of d^t, with which we have:

$$t \leq \frac{2-\epsilon}{\epsilon}\frac{R^2}{\gamma^2} \quad (14)$$

which is identical to the number of updates bound given in the convergence proof of binary AMIRA. □

Similar to the proof of binary AMIRA, we can also get an upper bound on the norm of weight vector:

Corollary 2. *For any t, we have $\|\mathbf{w}^t\| \leq \frac{2-\epsilon}{\gamma}$.*

These two convergence proofs unveil some interesting properties of AMIRA:

1. When $\epsilon = 1$ (i.e., MIRA), AMIRA has the same convergence rate as Perceptron.

2. Convergence rate slows down as ϵ approaches 0.

3. When $\epsilon = 0$ (i.e., PA), this result suggests that it may not converge, which is observed in practice (on synthetic data; see Section 4.1).[4]

4. The most distinctive feature of the result is that unlike Perceptron, $\|\mathbf{w}^t\|$ no longer depends on either t or R, instead, there is a constant upper bound depending only on ϵ and γ. This explains why empirically MIRA-learned weights are always very small.

3.3. Margin Analysis

With Corollary 1 and Corollary 2 on the upper bound of the weight vector norm, and since after convergence the margin on each example is at least $1-\epsilon$, we get following theorem:

Theorem 3 (margin approximation)**.** *The margin after AMIRA convergence is*

$$\frac{1-\epsilon}{\|\mathbf{w}\|} \geq \frac{1-\epsilon}{2-\epsilon}\gamma.$$

Theorem 3 points out that AMIRA is a $\frac{1-\epsilon}{2-\epsilon}$ approximation of SVM. This approximation is monotonically better and approaching $\frac{1}{2}$ as ϵ approaches 0.

However, empirically this approximation ratio seems to be better than $1-\epsilon$. When ϵ is close to 0, the margin is very close to the optimal margin given by SVM, with the cost of slower convergence.

This result also hints that MIRA has no guarantee of margin after convergence, which is consistent with our experiments in Section 4.1 and Figure 2.

3.4. Inseparable Case

For inseparable case, we adapt the analysis from (Freund & Schapire, 1999; Collins, 2002), and get the following theorems. We omit the proofs for conciseness.

For binary AMIRA on inseparable set D, and a particular unit vector $\mathbf{u}$ and margin γ, the hinge loss for a training example $\mathbf{x}^t$ is defined as

$$e^t = \max\{0, \gamma - y^t \mathbf{u} \cdot \mathbf{x}^t\}.$$

[4] The authors of PA (Crammer et al., 2006) do provide a proof on the bound on the *sum of losses*, but that does *not* imply a bound on the *number of mistakes*, e.g. $\sum_i 1/i^2$ v.s. $\sum_i 1$.

Theorem 4 (Number of Mistakes Bound for Binary AMIRA on Inseparable Set)**.** *For the first pass of binary AMIRA over an inseparable training set D,*

$$\#\ of\ mistakes \leq \min_{\mathbf{u},\gamma} \frac{2-\epsilon}{\epsilon}\frac{(R+\mathcal{D})^2}{\gamma^2},$$

where $\mathcal{D} = \sqrt{\sum_t e^t}$.

For k-best AMIRA we similarly define the hinge loss as

$$e^t = \max\{0, \gamma - \min_z \mathbf{u} \cdot \Delta\mathbf{\Phi}(\mathbf{x}^t, y^t, z)\}.$$

Then we have:

Theorem 5 (Number of Mistakes Bound for k-best AMIRA on Inseparable Set)**.** *For the first pass of k-best AMIRA over an inseparable training set D,*

$$\#\ of\ mistakes \leq \min_{\mathbf{u},\gamma} \frac{2-\epsilon}{\epsilon}\frac{(R+\mathcal{D})^2}{\gamma^2},$$

where $\mathcal{D} = \sqrt{\sum_t e^t}$.

3.5. About the Bias Term

In above analysis of binary AMIRA we chose to omit the bias term b. With the bias term the prediction function for training example $\mathbf{x}$ becomes $\mathbf{w} \cdot \mathbf{x} + b$. The bias term can be helpful in some applications. To adapt this bias term, we can simply augment the dimensionality by adding one extra feature, which is fixed to be 1, to each training example $\mathbf{x}$. At the same time we also add one dimension to the weight vector $\mathbf{w}$ to represent b.

With the extra bias dimension, all our proofs are still sound as is. The only slight difference is that we are now minimizing function $\|\mathbf{w}\|^2 + b^2$, instead of the original objective function $\|\mathbf{w}\|^2$. However, our experiments on synthetic data (see Section 4.1) show this change has little effects on the margin after convergence. For experiments on real datasets we do not use this bias term.

4. Experiments

We conduct experiments on two different types of datasets: a synthetic dataset to illustrate the properties in our proof, and natural datasets including USPS and MNIST to evaluate the performance of AMIRA. We also compare our result to existing online learning algorithms including Perceptron, ROMMA, ALMA, and an online version of Pegasos[5].

4.1. Synthetic Data

We examine binary AMIRA, Perceptron, Pegasos, and ROMMA over small linear separable binary synthetic

[5] For Pegasos we set its minibatch size to 1, making it a pure online algorithm.

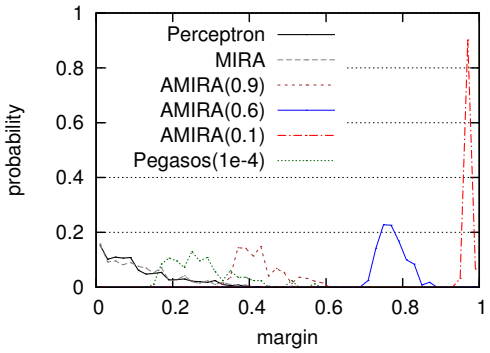

Figure 2. Distribution of margin with varying parameter ϵ (the number in parentheses) for AMIRA.

datasets. Each dataset consists of $10,000$ 10-dimensional vectors. Each element of the vector is a real value between 0 and 10 inclusively. For each dataset, we first randomly generate a separating hyperplane $\bar{\mathbf{w}} \cdot \mathbf{x} + b = 0$ that goes through the geometrical center of the 10-dimensional space. We run an SVM with linear kernel and no slacks, and from the separating hyperplane given by SVM we calculate the optimal separating margin γ. We implement SVM based on LIBSVM(Chang & Lin, 2011).

We do not augment the dimension for the bias term in SVM. After getting γ we augment the dimension by 1 as in Section 3.5. In the following experiments we measure the margin in unaugmented space. We observe that although we do not directly optimize the objective function ($\|\mathbf{w}\|^2$) for margin, we can still achieve large margin.

Our first experiment on these small datasets aims to show the margin achieved by our AMIRA with varying ϵ parameter. For comparison we also test the margin achieved by Perceptron and Pegasos with $\lambda = 1 \times 10^{-4}$. We plot the distribution of converged margins of Perceptron, Pegasos, and binary AMIRA with different ϵs, based on 500 random runs, as shown in Figure 2.

In Figure 2, the margins achieved by Perceptron and MIRA (AMIRA with $\epsilon = 1$) are mostly concentrated around margin ~ 0.1. This confirms our analysis that MIRA, like Perceptron, does not guarantee any separating margin in Theorem 3. Pegasos also does not give a large margin, partly because Pegasos does not directly optimize the margin, but the objective function with losses. AMIRA shows best performance: an averaged margin of ~ 0.4 for $\epsilon = 0.9$, and an averaged margin of ~ 0.95 for $\epsilon = 0.1$.

We then run binary AMIRA with varying ϵ between 0 and 1. We run 100 times and measure the average margin (see Figure 3 (a)) and the average number of updates performed (see Figure 3 (b)) when the algorithms reach convergence.

We compare our AMIRA to a parameterized version of ROMMA, which updates when current model does not separate the example by a margin of $1 - \epsilon$. When $\epsilon = 1$ we get the vanilla ROMMA(Li & Long, 2002).

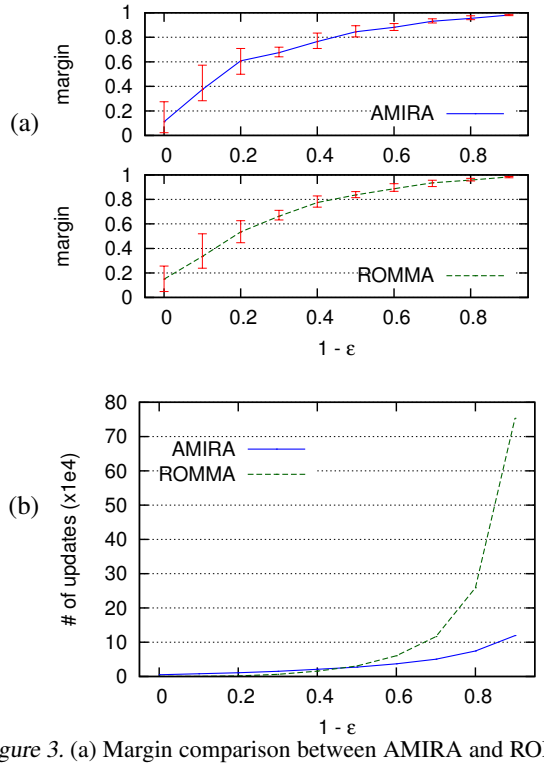

Figure 3. (a) Margin comparison between AMIRA and ROMMA with varying aggressiveness $1 - \epsilon$. Error bars represent the maximal and minimal margins achieved at a given ϵ. (b) Number of updates comparison between AMIRA and ROMMA with varying aggressiveness $1 - \epsilon$.

From the results, we first observe that, empirically our AMIRA achieves margin significantly better than the lower bound given in Theorem 3. The margin is almost the same as the margin of ROMMA, as in Figure 3 (a).

Also, we see that AMIRA can reach a margin of 0.9 with $\epsilon = 0.4$, which suggests that even without a very aggressive $1 - \epsilon$, AMIRA still can produce an acceptable margin.

Lastly, Figure 3 (b) demonstrates that our AMIRA converges much faster than ROMMA. This is consistent with the theoretical analysis that the number of updates for AMIRA is bounded by $\frac{1}{\epsilon}\frac{R^2}{\gamma^2}$, while ROMMA by $\frac{1}{\epsilon^2}\frac{R^2}{\gamma^2}$. We also find that, importantly, PA ($\epsilon = 0$) does *not* converge, which is consistent with the theory (see Section 3.2).

4.2. USPS

USPS dataset consists of $7,291$ training examples and $2,007$ testing examples. Each example is a 16×16 pixel image corresponding to a hand-written digit, which is the image's label. We do no preprocessing before training.

Table 2. Error rates (%) on USPS dataset. The number in parentheses after each algorithm name denotes its parameter. The column titled "convg." shows the highest accuracies achieved in first 5 epochs, with the numbers in parentheses denoting the peak epoch. The column titled "# updates" shows the number of updates for each algorithm after first 5 epochs. For Perceptron we also list the results of averaged Perceptron as in (Freund & Schapire, 1999), which significantly improves the accuracies, compared to the vanilla one. For k-best AMIRA, $k = 10$.

Epoch	1	3	convg.	# updates
Perceptron	7.40	6.18	6.16 (5)	1,726
Averaged Perceptron	6.80	5.97	5.93 (4)	1,726
ROMMA	7.06	6.15	6.15 (3)	1,703
Aggressive ROMMA	7.17	6.11	6.11 (3)	1,714
ALMA (0.9)	4.72	4.54	4.53 (2)	13,682
Pegasos ($\lambda = 5e-6$)	6.18	4.69	**4.51** (5)	30,473
bin MIRA	7.32	6.21	6.21 (3)	1,716
bin AMIRA (0.9)	4.62	**4.51**	**4.51** (3)	11,928
bin AMIRA (0.5)	5.02	5.00	5.00 (3)	27,201
k-best MIRA	7.19	6.15	6.15 (3)	667
k-best AMIRA (0.85)	**4.56**	4.60	4.56 **(1)**	2,585
k-best AMIRA (0.5)	4.92	4.91	4.87 (2)	4,084
SVM		4.70		-

To better classify the examples, we use a Gaussian kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2).$$

The parameter of Gaussian kernel σ is set to 3.5, which is same to the one used in (Gentile, 2002).

For a similar Gaussian kernel with $\sigma = 3.58$, (Platt et al., 2000) reports an accuracy of 4.70% using SVM with parameter $C = 100$ and no data preprocessing. We use this as a naive baseline.

We reimplemented ROMMA, ALMA, and Pegasos for comparison following the standard one-vs-rest scheme. The parameters in ALMA and Pegasos are chosen based on the suggestions in their papers, and slight changes are made for better accuracies. As a result, we achieve higher accuracies for ALMA than the reported numbers in (Gentile, 2002).

Because the training of our AMIRA depends on the ordering of the training examples, we shuffle the whole training set at each training iteration. Our error rate results are based on the average of 10 runs.

Table 2 presents the percentages of misclassified testing examples for Perceptron, ROMMA, ALMA, Pegasos, AMIRA, and SVM. Following (Li et al., 2002), the performances are reported after 1st and 3rd epochs. The highest accuracies achieved in the first 5 epochs are also reported.

We observe binary AMIRA$_{0.9}$ achieves best accuracy after 3 epochs, which is also achieved by Pegasos some epochs later. However, Pegasos achieves this accuracy by very aggrevessive updates and takes significantly more updates and longer time.

Table 3. Error rates (%) on MNIST dataset. The numbers in parentheses denote parameters for the algorithms. Note that (Schölkopf et al., 1997) report an accuracy of 1.40% with polynomial kernel of degree 5 using an SVM with parameter $C = 10$. This is not included because it uses a different kernel.

Epoch	1	2	3	4	# updates
Perceptron	2.87	2.26	2.12	2.26	12,607
Avg. Perceptron	**2.22**	1.80	1.71	1.69	12,607
ROMMA	2.53	1.82	1.76	1.68	10,466
Aggr. ROMMA	2.45	1.89	1.71	1.70	10,436
Pegasos ($\lambda = 500$)	2.33	1.92	1.85	1.82	118,615
bin MIRA	2.65	2.02	1.77	1.69	11,091
bin AMIRA(0.9)	2.27	**1.79**	**1.64**	**1.59**	12,969

On the other hand, k-best AMIRA converges much faster than other algorithms. For example k-best AMIRA$_{0.85}$ converges at 1st epoch with the lowest error rate, outperforming any other competitors at the same time. Also, k-best AMIRA needs significantly less updates for each iteration, because in k-best AMIRA only one classifier is trained, while the one-vs-rest scheme needs 10.

4.3. MNIST

MNIST dataset consists of $60,000$ training examples and $10,000$ testing examples. Similar to USPS, each example represents a hand-written digit, stored in a 28×28 integer matrix. We use the standard one-vs-rest scheme and do no preprocessing before training.

We use a polynomial kernel of degree 4. Also, following the suggestion from (Li & Long, 2002), we slightly modify the kernel function to handle the noise:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{255 \times 255} + 1)^4 + \delta_{ij}\lambda$$

where δ_{ij} is the Kronecker delta function that $\delta_{ij} = 1$ if and only if $i = j$. Parameter λ is set to 5×10^5 in our experiments, which is consistent with (Li & Long, 2002).

Table 3 reports the error rates on testing set after averaging on 5 runs. At each run we shuffle the training examples. Note that for ROMMA we achieve better accuracies than the reported numbers in (Li & Long, 2002). For Pegasos we use a much larger λ than in the USPS experiments, which is due to a different kernel and the Kronecker delta function.

In this dataset binary AMIRA$_{0.9}$ outperforms any other competitors after 2nd epoch in accuracy and the speed (# of updates) is comparable to Perceptron and ROMMA.

5. Conclusions and Future Work

We have presented a family of simple online learning algorithms, AMIRA, that can achieve approximate maximal

margin. We give a proof showing that, within finite number of steps, AMIRA is guaranteed to converge with a lower bound on convergence margin, for parameter $\epsilon > 0$. As a by-product, we also simplified convergence proofs for MIRA and PA as special cases. Theoretically AMIRA is the first to converge in $O(\frac{1}{\epsilon})$ among those online algorithms that have provable approximation ratios of the optimal margin, compared to previous methods' $O(\frac{1}{\epsilon^2})$. Empirically we observe our method leads to similar margin and higher accuracy compared to existing methods and converges faster.

For future work we would like to extend our k-best AMIRA to structured problems,which might benefit from the guaranteed geometrical margin of AMIRA.

References

Chang, Chih-Chung and Lin, Chih-Jen. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904.

Collins, Michael. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 2002.

Cortes, Corinna and Vapnik, Vladimir. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

Crammer, Koby and Singer, Yoram. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, March 2003. ISSN 1532-4435.

Crammer, Koby, Dekel, Ofer, Keshet, Joseph, Shalev-Shwartz, Shai, and Singer, Yoram. Online passive aggressive algorithms. *The Journal of Machine Learning Research*, 2006.

Freund, Yoav and Schapire, Robert. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Gentile, Claudio. A new approximate maximal margin classification algorithm. *The Journal of Machine Learning Research*, 2:213–242, 2002.

Hildreth, Clifford. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4(1):79–85, 1957. ISSN 1931-9193. doi: 10.1002/nav.3800040113.

Ishibashi, Kosuke, Hatano, Kohei, and Takeda, Masayuki. Online learning of approximate maximum p-norm margin classifiers with bias. In *Proceedings of the 21st COLT*, 2008.

Li, Yaoyong, Zaragoza, Hugo, Herbrich, Ralf, Shawe-Taylor, John, and Kandola, Jaz. The perceptron algorithm with uneven margins. In *Machine Learning-International Workshop then Conference*, 2002.

Li, Yi and Long, Philip M. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1):361–387, 2002.

McDonald, Ryan, Crammer, Koby, and Pereira, Fernando. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*, 2005.

Novikoff, A. B. On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata*, 1962.

Platt, John C. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.

Platt, John C, Cristianini, Nello, and Shawe-Taylor, John. Large margin dags for multiclass classification. *Advances in neural information processing systems*, 12(3): 547–553, 2000.

Rosenblatt, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 1959.

Schölkopf, B, Simard, P, Vapnik, V, and Smola, AJ. Improving the accuracy and speed of support vector machines. In *Proceedings of NIPS 1996*, volume 9, pp. 375. The MIT Press, 1997.

Shalev-Shwartz, Shai, Singer, Yoram, and Srebro, Nathan. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th ICML*, 2007.