Journal of Machine Learning Research 7 (2006) 551–585 Submitted 5/05; Published 3/06

Online Passive-Aggressive Algorithms

Koby Crammer[*] CRAMMER@CIS.UPENN.EDU
Ofer Dekel OFERD@CS.HUJI.AC.IL
Joseph Keshet JKESHET@CS.HUJI.AC.IL
Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
Yoram Singer[†] SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Editor: Manfred K. Warmuth

Abstract

We present a family of margin based online learning algorithms for various prediction tasks. In particular we derive and analyze algorithms for binary and multiclass categorization, regression, uniclass prediction and sequence prediction. The update steps of our different algorithms are all based on analytical solutions to simple constrained optimization problems. This unified view allows us to prove worst-case loss bounds for the different algorithms and for the various decision problems based on a single lemma. Our bounds on the cumulative loss of the algorithms are relative to the smallest loss that can be attained by any fixed hypothesis, and as such are applicable to both realizable and unrealizable settings. We demonstrate some of the merits of the proposed algorithms in a series of experiments with synthetic and real data sets.

1. Introduction

In this paper we describe and analyze several online learning tasks through the same algorithmic prism. We first introduce a simple online algorithm which we call Passive-Aggressive (PA) for online binary classification (see also (Herbster, 2001)). We then propose two alternative modifications to the PA algorithm which improve the algorithm's ability to cope with noise. We provide a unified analysis for the three variants. Building on this unified view, we show how to generalize the binary setting to various learning tasks, ranging from regression to sequence prediction.

The setting we focus on is that of online learning. In the online setting, a learning algorithm observes instances in a sequential manner. After each observation, the algorithm predicts an outcome. This outcome can be as simple as a yes/no $(+/-)$ decision, as in the case of binary classification problems, and as complex as a string over a large alphabet. Once the algorithm has made a prediction, it receives feedback indicating the correct outcome. Then, the online algorithm may modify its prediction mechanism, presumably improving the chances of making an accurate prediction on subsequent rounds. Online algorithms are typically simple to implement and their analysis often provides tight bounds on their performance (see for instance Kivinen and Warmuth (1997)).

∗. Current affiliation: Department of Computer and Information Science, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104, USA.
†. Current affiliation: Google, Moutain View, CA 94043, USA.

Our learning algorithms use hypotheses from the set of linear predictors. While this class may seem restrictive, the pioneering work of Vapnik (1998) and colleagues demonstrates that by using Mercer kernels one can employ highly non-linear predictors and still entertain all the formal properties and simplicity of linear predictors. For concreteness, our presentation and analysis are confined to the linear case which is often referred to as the primal version (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002). As in other constructions of linear kernel machines, our paradigm also builds on the notion of margin.

Binary classification is the first setting we discuss in the paper. In this setting each instance is represented by a vector and the prediction mechanism is based on a hyperplane which divides the instance space into two half-spaces. The margin of an example is proportional to the distance between the instance and the hyperplane. The PA algorithm utilizes the margin to modify the current classifier. The update of the classifier is performed by solving a constrained optimization problem: we would like the new classifier to remain as close as possible to the current one while achieving at least a unit margin on the most recent example. Forcing a unit margin might turn out to be too aggressive in the presence of noise. Therefore, we also describe two versions of our algorithm which cast a tradeoff between the desired margin and the proximity to the current classifier.

The above formalism is motivated by the work of Warmuth and colleagues for deriving online algorithms (see for instance (Kivinen and Warmuth, 1997) and the references therein). Furthermore, an analogous optimization problem arises in support vector machines (SVM) for classification (Vapnik, 1998). Indeed, the core of our construction can be viewed as finding a support vector machine based on a single example while replacing the norm constraint of SVM with a proximity constraint to the current classifier. The benefit of this approach is two fold. First, we get a closed form solution for the next classifier. Second, we are able to provide a unified analysis of the cumulative loss for various online algorithms used to solve different decision problems. Specifically, we derive and analyze versions for regression problems, uniclass prediction, multiclass problems, and sequence prediction tasks.

Our analysis is in the realm of relative loss bounds. In this framework, the cumulative loss suffered by an online algorithm is compared to the loss suffered by a fixed hypothesis that may be chosen in hindsight. Our proof techniques are surprisingly simple and the proofs are fairly short and easy to follow. We build on numerous previous results and views. The mere idea of deriving an update as a result of a constrained optimization problem compromising of two opposing terms, has been largely advocated by Littlestone, Warmuth, Kivinen and colleagues (Littlestone, 1989; Kivinen and Warmuth, 1997). Online margin-based prediction algorithms are also quite prevalent. The roots of many of the papers date back to the Perceptron algorithm (Agmon, 1954; Rosenblatt, 1958; Novikoff, 1962). More modern examples include the ROMMA algorithm of Li and Long (2002), Gentile's ALMA algorithm (Gentile, 2001), the MIRA algorithm (Crammer and Singer, 2003b), and the NORMA algorithm (Kivinen et al., 2002). The MIRA algorithm is closely related to the work presented in this paper, and specifically, the MIRA algorithm for binary classification is identical to our basic PA algorithm. However, MIRA was designed for *separable* binary and multiclass problems whereas our algorithms also apply to nonseparable problems. Furthermore, the loss bounds derived in Crammer and Singer (2003b) are inferior and less general than the bounds derived in this paper. The NORMA algorithm also shares a similar view of classification problems. Rather than projecting the current hypothesis onto the set of constraints induced by the most recent example, NORMA's update rule is based on a stochastic gradient approach (Kivinen et al., 2002). Of all the work on online learning algorithms, the work by Herbster (2001) is probably the closest to the work

presented here. Herbster describes and analyzes a projection algorithm that, like MIRA, is essentially the same as the basic PA algorithm for the separable case. We surpass MIRA and Herbster's algorithm by providing bounds for both the separable and the nonseparable settings using a unified analysis. As mentioned above we also extend the algorithmic framework and the analysis to more complex decision problems.

The paper is organized as follows. In Sec. 2 we formally introduce the binary classification problem and in the next section we derive three variants of an online learning algorithm for this setting. The three variants of our algorithm are then analyzed in Sec. 4. We next show how to modify these algorithms to solve regression problems (Sec. 5) and uniclass prediction problems (Sec. 6). We then shift gears to discuss and analyze more complex decision problems. Specifically, in Sec. 7 we describe a generalization of the algorithms to multiclass problems and further extend the algorithms to cope with sequence prediction problems (Sec. 9). We describe experimental results with binary and multiclass problems in Sec. 10 and conclude with a discussion of future directions in Sec. 11.

2. Problem Setting

As mentioned above, the paper describes and analyzes several online learning tasks through the same algorithmic prism. We begin with binary classification which serves as the main building block for the remainder of the paper. Online binary classification takes place in a sequence of rounds. On each round the algorithm observes an instance and predicts its label to be either $+1$ or -1. After the prediction is made, the true label is revealed and the algorithm suffers an *instantaneous loss* which reflects the degree to which its prediction was wrong. At the end of each round, the algorithm uses the newly obtained instance-label pair to improve its prediction rule for the rounds to come.

We denote the instance presented to the algorithm on round t by $\mathbf{x}_t$, and for concreteness we assume that it is a vector in $\mathbb{R}^n$. We assume that $\mathbf{x}_t$ is associated with a unique label $y_t \in \{+1,-1\}$. We refer to each instance-label pair $(\mathbf{x}_t, y_t)$ as an *example*. The algorithms discussed in this paper make predictions using a classification function which they maintain in their internal memory and update from round to round. We restrict our discussion to classification functions based on a vector of weights $\mathbf{w} \in \mathbb{R}^n$, which take the form $\text{sign}(\mathbf{w} \cdot \mathbf{x})$. The magnitude $|\mathbf{w} \cdot \mathbf{x}|$ is interpreted as the degree of confidence in this prediction. The task of the algorithm is therefore to incrementally learn the weight vector $\mathbf{w}$. We denote by $\mathbf{w}_t$ the weight vector used by the algorithm on round t, and refer to the term $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ as the (signed) *margin* attained on round t. Whenever the margin is a positive number then $\text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t) = y_t$ and the algorithm has made a correct prediction. However, we are not satisfied by a positive margin value and would additionally like the algorithm to predict with high confidence. Therefore, the algorithm's goal is to achieve a margin of at least 1 as often as possible. On rounds where the algorithm attains a margin less than 1 it suffers an instantaneous loss. This loss is defined by the following *hinge-loss* function,

$$\ell\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \begin{cases} 0 & y(\mathbf{w}\cdot\mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w}\cdot\mathbf{x}) & \text{otherwise} \end{cases} \quad . \tag{1}$$

Whenever the margin exceeds 1, the loss equals zero. Otherwise, it equals the difference between the margin value and 1. We note in passing that the choice of 1 as the margin threshold below which a loss is suffered is rather arbitrary. In Sec. 5 we generalize the hinge-loss function in the context of regression problems, by letting the threshold be a user-defined parameter. We abbreviate the loss

suffered on round t by ℓ_t, that is, $\ell_t = \ell\big(\mathbf{w}_t;(\mathbf{x}_t,y_t)\big)$. The algorithms presented in this paper will be shown to attain a small *cumulative squared loss* over a given sequence of examples. In other words, we will prove different bounds on $\sum_{t=1}^{T}\ell_t^2$, where T is the length of the sequence. Notice that whenever a prediction mistake is made then $\ell_t^2 \geq 1$ and therefore a bound on the cumulative squared loss also bounds the number of prediction mistakes made over the sequence of examples.

3. Binary Classification Algorithms

In the previous section we described a general setting for binary classification. To obtain a concrete algorithm we must determine how to initialize the weight vector $\mathbf{w}_1$ and we must define the update rule used to modify the weight vector at the end of each round. In this section we present three variants of an online learning algorithm for binary classification. The pseudo-code for the three variants is given in Fig. 1. The vector $\mathbf{w}_1$ is initialized to $(0,\ldots,0)$ for all three variants, however each variant employs a different update rule. We focus first on the simplest of the three, which on round t sets the new weight vector $\mathbf{w}_{t+1}$ to be the solution to the following constrained optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) = 0. \tag{2}$$

Geometrically, $\mathbf{w}_{t+1}$ is set to be the projection of $\mathbf{w}_t$ onto the half-space of vectors which attain a hinge-loss of zero on the current example. The resulting algorithm is *passive* whenever the hinge-loss is zero, that is, $\mathbf{w}_{t+1} = \mathbf{w}_t$ whenever $\ell_t = 0$. In contrast, on those rounds where the loss is positive, the algorithm *aggressively* forces $\mathbf{w}_{t+1}$ to satisfy the constraint $\ell(\mathbf{w}_{t+1};(\mathbf{x}_t,y_t)) = 0$ regardless of the step-size required. We therefore name the algorithm *Passive-Aggressive* or *PA* for short.

The motivation for this update stems from the work of Helmbold et al. (Helmbold et al., 1999) who formalized the trade-off between the amount of progress made on each round and the amount of information retained from previous rounds. On one hand, our update requires $\mathbf{w}_{t+1}$ to correctly classify the current example with a sufficiently high margin and thus progress is made. On the other hand, $\mathbf{w}_{t+1}$ must stay as close as possible to $\mathbf{w}_t$, thus retaining the information learned on previous rounds.

The solution to the optimization problem in Eq. (2) has a simple closed form solution,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \;\text{ where }\; \tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}. \tag{3}$$

We now show how this update is derived using standard tools from convex analysis (see for instance (Boyd and Vandenberghe, 2004)). If $\ell_t = 0$ then $\mathbf{w}_t$ itself satisfies the constraint in Eq. (2) and is clearly the optimal solution. We therefore concentrate on the case where $\ell_t > 0$. First, we define the Lagrangian of the optimization problem in Eq. (2) to be,

$$\mathcal{L}(\mathbf{w},\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \;+\; \tau\big(1-y_t(\mathbf{w}\cdot\mathbf{x}_t)\big), \tag{4}$$

where $\tau \geq 0$ is a Lagrange multiplier. The optimization problem in Eq. (2) has a convex objective function and a single feasible affine constraint. These are sufficient conditions for Slater's condition to hold therefore finding the problem's optimum is equivalent to satisfying the Karush-Khun-Tucker

INPUT: aggressiveness parameter $C > 0$
INITIALIZE: $\mathbf{w}_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots$

- receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
- predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
- receive correct label: $y_t \in \{-1, +1\}$
- suffer loss: $\ell_t = \max\{0\ ,\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\}$
- update:
  1. set:
     $$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2} \qquad \text{(PA)}$$
     $$\tau_t = \min\left\{C\ ,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\} \qquad \text{(PA-I)}$$
     $$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \qquad \text{(PA-II)}$$
  2. update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$

Figure 1: Three variants of the Passive-Aggressive algorithm for binary classification.

conditions (Boyd and Vandenberghe, 2004). Setting the partial derivatives of $\mathcal{L}$ with respect to the elements of $\mathbf{w}$ to zero gives,

$$0 \ = \ \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \tau) \ = \ \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t \qquad \Longrightarrow \qquad \mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t . \tag{5}$$

Plugging the above back into Eq. (4) we get,

$$\mathcal{L}(\tau) \ = \ -\frac{1}{2}\tau^2 \|\mathbf{x}_t\|^2 \ + \ \tau\big(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\big).$$

Taking the derivative of $\mathcal{L}(\tau)$ with respect to τ and setting it to zero, we get,

$$0 \ = \ \frac{\partial \mathcal{L}(\tau)}{\partial \tau} \ = \ -\tau \|\mathbf{x}_t\|^2 \ + \ \big(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\big) \qquad \Longrightarrow \qquad \tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}.$$

Since we assumed that $\ell_t > 0$ then $\ell_t = 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)$. In summary, we can state a unified update for the case where $\ell_t = 0$ and the case where $\ell_t > 0$ by setting $\tau_t = \ell_t / \|\mathbf{x}_t\|^2$.

As discussed above, the PA algorithm employs an aggressive update strategy by modifying the weight vector by as much as needed to satisfy the constraint imposed by the current example. In certain real-life situations this strategy may also result in undesirable consequences. Consider for instance the common phenomenon of label noise. A mislabeled example may cause the PA algorithm to drastically change its weight vector in the wrong direction. A single mislabeled example can lead to several prediction mistakes on subsequent rounds. To cope with such problems, we present two variations on the PA update that employ gentler update strategies. We adopt the technique previously used to derive soft-margin classifiers (Vapnik, 1998) and introduce a non-negative slack variable ξ into the optimization problem defined in Eq. (2). This variable can be introduced in two different ways. First, we consider the update where the objective function scales linearly with

ξ, namely,

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) \le \xi \text{ and } \xi \ge 0. \tag{6}$$

Here C is a positive parameter which controls the influence of the slack term on the objective function. Specifically, we will show that larger values of C imply a more aggressive update step and we therefore refer to C as the *aggressiveness parameter* of the algorithm. We term the algorithm which results from this update *PA-I* .

Alternatively, we can have the objective function scale quadratically with ξ, resulting in the following constrained optimization problem,

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi^2 \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) \le \xi. \tag{7}$$

Note that the constraint $\xi \ge 0$ which appears in Eq. (6) is no longer necessary since ξ^2 is always non-negative. We term the algorithm which results from this update *PA-II* . As with PA-I , C is a positive parameter which governs the degree to which the update of PA-II is aggressive. The updates of PA-I and PA-II also share the simple closed form $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$, where

$$\tau_t = \min\left\{ C\ ,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2} \right\} \quad \text{(PA-I)} \qquad \text{or} \qquad \tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \quad \text{(PA-II)}. \tag{8}$$

A detailed derivation of the PA-I and PA-II updates is provided in Appendix A. It is worth noting that the PA-II update is equivalent to increasing the dimension of each $\mathbf{x}_t$ from n to $n+T$, setting $x_{n+t} = \sqrt{1/2C}$, setting the remaining $T-1$ new coordinates to zero, and then using the simple PA update. This technique was previously used to derive noise-tolerant online algorithms in (Klasner and Simon, 1995; Freund and Schapire, 1999). We do not use this observation explicitly in this paper, since it does not lead to a tighter analysis.

Up until now, we have restricted our discussion to linear predictors of the form $\text{sign}(\mathbf{w}\cdot\mathbf{x})$. We can easily generalize any of the algorithms presented in this section using Mercer kernels. Simply note that for all three PA variants,

$$\mathbf{w}_t = \sum_{i=1}^{t-1} \tau_t y_t \mathbf{x}_t,$$

and therefore,

$$\mathbf{w}_t \cdot \mathbf{x}_t = \sum_{i=1}^{t-1} \tau_t y_t (\mathbf{x}_i \cdot \mathbf{x}_t).$$

The inner product on the right hand side of the above can be replaced with a general Mercer kernel $K(\mathbf{x}_i,\mathbf{x}_t)$ without otherwise changing our derivation. Additionally, the formal analysis presented in the next section also holds for any kernel operator.

4. Analysis

In this section we prove *relative* loss bounds for the three variants of the PA algorithm presented in the previous section. Specifically, most of the theorems in this section relate the cumulative squared loss attained by our algorithms on any sequence of examples with the loss attained by an arbitrary

fixed classification function of the form $\text{sign}(\mathbf{u} \cdot \mathbf{x})$ on the same sequence. As previously mentioned, the cumulative squared hinge loss upper bounds the number of prediction mistakes. Our bounds essentially prove that, for any sequence of examples, our algorithms cannot do much worse than the best fixed predictor chosen in hindsight.

To simplify the presentation we use two abbreviations throughout this paper. As before we denote by ℓ_t the instantaneous loss suffered by our algorithm on round t. In addition, we denote by $\ell_t^\star$ the loss suffered by the arbitrary fixed predictor to which we are comparing our performance. Formally, let $\mathbf{u}$ be an arbitrary vector in $\mathbb{R}^n$, and define

$$\ell_t = \ell\big(\mathbf{w}_t; (\mathbf{x}_t, y_t)\big) \qquad \text{and} \qquad \ell_t^\star = \ell\big(\mathbf{u}; (\mathbf{x}_t, y_t)\big). \tag{9}$$

We begin with a technical lemma which facilitates the proofs in this section. With this lemma handy, we then derive loss and mistake bounds for the variants of the PA algorithm presented in the previous section.

Lemma 1 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$ and $y_t \in \{+1, -1\}$ for all t. Let τ_t be as defined by either of the three PA variants given in Fig. 1. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u} \in \mathbb{R}^n$,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof Define Δ_t to be $\|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$. We prove the lemma by summing Δ_t over all t in $1, \ldots, T$ and bounding this sum from above and below. First note that $\sum_t \Delta_t$ is a telescopic sum which collapses to,

$$\begin{aligned} \sum_{t=1}^{T} \Delta_t &= \sum_{t=1}^{T} \left(\|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2\right) \\ &= \|\mathbf{w}_1 - \mathbf{u}\|^2 - \|\mathbf{w}_{T+1} - \mathbf{u}\|^2. \end{aligned}$$

Using the facts that $\mathbf{w}_1$ is defined to be the zero vector and that $\|\mathbf{w}_{T+1} - \mathbf{u}\|^2$ is non-negative, we can upper bound the right-hand side of the above by $\|\mathbf{u}\|^2$ and conclude that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2. \tag{10}$$

We now turn to bounding Δ_t from below. If the minimum margin requirement is not violated on round t, i.e. $\ell_t = 0$, then $\tau_t = 0$ and therefore $\Delta_t = 0$. We can therefore focus only on rounds for which $\ell_t > 0$. Using the definition $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \tau_t \mathbf{x}_t$, we can write Δ_t as,

$$\begin{aligned} \Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + y_t \tau_t \mathbf{x}_t\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left(\|\mathbf{w}_t - \mathbf{u}\|^2 + 2\tau_t y_t (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t + \tau_t^2 \|\mathbf{x}_t\|^2\right) \\ &= -2\tau_t y_t (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t - \tau_t^2 \|\mathbf{x}_t\|^2. \end{aligned} \tag{11}$$

Since we assumed that $\ell_t > 0$ then $\ell_t = 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ or alternatively $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) = 1 - \ell_t$. In addition, the definition of the hinge loss implies that $\ell_t^\star \geq 1 - y_t(\mathbf{u} \cdot \mathbf{x}_t)$, hence $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq 1 - \ell_t^\star$. Using these two facts back in Eq. (11) gives,

$$\begin{aligned} \Delta_t &\geq 2\tau_t \left((1 - \ell_t^\star) - (1 - \ell_t) \right) - \tau_t^2 \|\mathbf{x}_t\|^2 \\ &= \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star \right). \end{aligned} \tag{12}$$

Summing Δ_t over all t and comparing the lower bound of Eq. (12) with the upper bound in Eq. (10) proves the lemma. ■

We first prove a loss bound for the PA algorithm in the separable case. This bound was previously presented by Herbster (2001) and is analogous to the classic mistake bound for the Perceptron algorithm due to Novikoff (1962). We assume that there exists some $\mathbf{u} \in \mathbb{R}^n$ such that $y_t(\mathbf{u} \cdot \mathbf{x}_t) > 0$ for all $t \in \{1, \ldots, T\}$. Without loss of generality we can assume that $\mathbf{u}$ is scaled such that that $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq 1$ and therefore $\mathbf{u}$ attains a loss of zero on all T examples in the sequence. With the vector $\mathbf{u}$ at our disposal, we prove the following bound on the cumulative squared loss of PA .

Theorem 2 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Assume that there exists a vector $\mathbf{u}$ such that $\ell_t^\star = 0$ for all t. Then, the cumulative squared loss of PA on this sequence of examples is bounded by,*

$$\sum_{t=1}^{T} \ell_t^2 \ \leq \ \|\mathbf{u}\|^2 R^2.$$

Proof Since $\ell_t^\star = 0$ for all t, Lemma 1 implies that,

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 \right) \ \leq \ \|\mathbf{u}\|^2. \tag{13}$$

Using the definition of τ_t for the PA algorithm in the left-hand side of the above gives,

$$\sum_{t=1}^{T} \ell_t^2 / \|\mathbf{x}_t\|^2 \ \leq \ \|\mathbf{u}\|^2.$$

Now using the fact that $\|\mathbf{x}_t\|^2 \leq R^2$ for all t, we get,

$$\sum_{t=1}^{T} \ell_t^2 / R^2 \ \leq \ \|\mathbf{u}\|^2.$$

Multiplying both sides of this inequality by R^2 gives the desired bound. ■

The remaining bounds we prove in this section do not depend on a separability assumption. In contrast to the assumptions of Thm. 2, the vector $\mathbf{u}$ which appears in the theorems below is an arbitrary vector in $\mathbb{R}^n$ and not necessarily a perfect separator. The first of the following theorems bounds the cumulative squared loss attained by the PA algorithm in the special case where all of

the instances in the input sequence are normalized so that $\|\mathbf{x}_t\|^2 = 1$. Although this assumption is somewhat restrictive, it is often the case in many practical applications of classification that the instances are normalized. For instance, certain kernel operators, such as the Gaussian kernel, imply that all input instances have a unit norm. See for example (Cristianini and Shawe-Taylor, 2000).

Theorem 3 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| = 1$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$ the cumulative squared loss of PA on this sequence of examples is bounded from above by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \left(\|\mathbf{u}\| + 2\sqrt{\textstyle\sum_{t=1}^{T} (\ell_t^\star)^2} \right)^2 .$$

Proof In the special case where $\|\mathbf{x}_t\|^2 = 1$, τ_t and ℓ_t are equal. Therefore, Lemma 1 gives us that,

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \|\mathbf{u}\|^2 + 2 \sum_{t=1}^{T} \ell_t \ell_t^\star .$$

Using the Cauchy-Schwartz inequality to upper bound the right-hand side of the above inequality, and denoting

$$L_T = \sqrt{\textstyle\sum_{t=1}^{T} \ell_t^2} \quad \text{and} \quad U_T = \sqrt{\textstyle\sum_{t=1}^{T} (\ell_t^\star)^2}, \tag{14}$$

we get that $L_T^2 \leq \|\mathbf{u}\|^2 + 2 L_T U_T$. The largest value of L_T for which this inequality is satisfied is the larger of the two values for which this inequality holds with equality. That is, to obtain an upper bound on L_T we need to find the largest root of the second degree polynomial $L_T^2 - 2 U_T L_T - \|\mathbf{u}\|^2$, which is,

$$U_T + \sqrt{U_T^2 + \|\mathbf{u}\|^2}.$$

Using the fact that $\sqrt{\alpha + \beta} \leq \sqrt{\alpha} + \sqrt{\beta}$, we conclude that

$$L_T \;\leq\; \|\mathbf{u}\| + 2 U_T . \tag{15}$$

Taking the square of both sides of this inequality and plugging in the definitions of L_T and U_T from Eq. (14) gives the desired bound. ■

Next we turn to the analysis of PA-I . The following theorem does not provide a loss bound but rather a mistake bound for the PA-I algorithm. That is, we prove a direct bound on the number of times $y_t \neq \mathrm{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ without using $\sum \ell_t^2$ as a proxy.

Theorem 4 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Then, for any vector $\mathbf{u} \in \mathbb{R}^n$, the number of prediction mistakes made by PA-I on this sequence of examples is bounded from above by,*

$$\max\left\{R^2, 1/C\right\} \left(\|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star \right),$$

where C is the aggressiveness parameter provided to PA-I (Fig. 1) .

Proof If PA-I makes a prediction mistake on round t then $\ell_t \geq 1$. Using our assumption that $\|\mathbf{x}_t\|^2 \leq R^2$ and the definition $\tau_t = \min\{\ell_t/\|\mathbf{x}_t\|^2, C\}$, we conclude that if a prediction mistake occurs then it holds that,

$$\min\{1/R^2, C\} \;\leq\; \tau_t \ell_t .$$

Let M denote the number of prediction mistakes made on the entire sequence. Since $\tau_t \ell_t$ is always non-negative, it holds that,

$$\min\{1/R^2, C\}\, M \;\leq\; \sum_{t=1}^{T} \tau_t \ell_t . \tag{16}$$

Again using the definition of τ_t, we know that $\tau_t \ell_t^\star \leq C \ell_t^\star$ and that $\tau_t \|\mathbf{x}_t\|^2 \leq \ell_t$. Plugging these two inequalities into Lemma 1 gives,

$$\sum_{t=1}^{T} \tau_t \ell_t \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star . \tag{17}$$

Combining Eq. (16) with Eq. (17), we conclude that,

$$\min\{1/R^2, C\}\, M \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star .$$

The theorem follows from multiplying both sides of the above by $\max\{R^2, 1/C\}$. ■

Finally, we turn to the analysis of PA-II . As before, the proof of the following theorem is based on Lemma 1.

Theorem 5 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_t)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\|^2 \leq R^2$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$ it holds that the cumulative squared loss of PA-II on this sequence of examples is bounded by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \left(R^2 + \frac{1}{2C}\right) \left(\|\mathbf{u}\|^2 \;+\; 2C \sum_{t=1}^{T} (\ell_t^\star)^2\right),$$

where C is the aggressiveness parameter provided to PA-II (Fig. 1) .

Proof Recall that Lemma 1 states that,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star\right).$$

Defining $\alpha = 1/\sqrt{2C}$, we subtract the non-negative term $(\alpha\tau_t - \ell_t^\star/\alpha)^2$ from each summand on the right-hand side of the above inequality, to get

$$\begin{aligned}
\|\mathbf{u}\|^2 \;&\geq\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star - (\alpha\tau_t - \ell_t^\star/\alpha)^2\right) \\
&=\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star - \alpha^2\tau_t^2 + 2\tau_t \ell_t^\star - (\ell_t^\star)^2/\alpha^2\right) \\
&=\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 (\|\mathbf{x}_t\|^2 + \alpha^2) - (\ell_t^\star)^2/\alpha^2\right).
\end{aligned}$$

Plugging in the definition of α, we obtain the following lower bound,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T}\left(2\tau_t\ell_t - \tau_t^2\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) - 2C(\ell_t^\star)^2\right).$$

Using the definition $\tau_t = \ell_t/(\|\mathbf{x}_t\|^2 + 1/(2C))$, we can rewrite the above as,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T}\left(\frac{\ell_t^2}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} - 2C(\ell_t^\star)^2\right).$$

Replacing $\|\mathbf{x}_t\|^2$ with its upper bound of R^2 and rearranging terms gives the desired bound. ■

We conclude this section with a brief comparison of our bounds to previously published bounds for the Perceptron algorithm. As mentioned above, the bound in Thm. 2 is equal to the bound of Novikoff (1962) for the Perceptron in the separable case. However, Thm. 2 bounds the cumulative squared hinge loss of PA , whereas Novikoff's bound is on the number of prediction mistakes. Gentile (2002) proved a mistake bound for the Perceptron in the nonseparable case which can be compared to our mistake bound for PA-I in Thm. 4. Using our notation from Thm. 4, Gentile bounds the number of mistakes made by the Perceptron by,

$$\tfrac{R^2\|\mathbf{u}\|^2}{2} \;+\; \textstyle\sum_{t=1}^{T}\ell_t^\star \;+\; \sqrt{R^2\|\mathbf{u}\|^2\textstyle\sum_{t=1}^{T}\ell_t^\star + \left(\tfrac{R^2\|\mathbf{u}\|^2}{2}\right)^2}.$$

At the price of a slightly loosening this bound, we can use the inequality $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ to get the simpler bound,

$$R^2\|\mathbf{u}\|^2 \;+\; \textstyle\sum_{t=1}^{T}\ell_t^\star \;+\; R\|\mathbf{u}\|\sqrt{\textstyle\sum_{t=1}^{T}\ell_t^\star}.$$

With $C = 1/R^2$, our bound in Thm. 4 becomes,

$$R^2\|\mathbf{u}\|^2 \;+\; 2\sum_{t=1}^{T}\ell_t^\star.$$

Thus, our bound is inferior to Gentile's when $R\|\mathbf{u}\| < \sqrt{\sum_{t=1}^{T}\ell_t^\star}$, and even then by a factor of at most 2.

The loss bound for PA-II in Thm. 5 can be compared with the bound of Freund and Schapire (1999) for the Perceptron algorithm. Using the notation defined in Thm. 5, Freund and Schapire bound the number of incorrect predictions made by the Perceptron by,

$$\left(R\|\mathbf{u}\| + \sqrt{\textstyle\sum_{t=1}^{T}(\ell_t^\star)^2}\right)^2.$$

It can be easily verified that the bound for the PA-II algorithm given in Thm. 5 exactly equals the above bound of Freund and Schapire when C is set to $\|\mathbf{u}\|/(2R\sqrt{\sum_t(\ell_t^\star)^2})$. Moreover, this is the optimal choice of C. However, we bound the cumulative squared hinge-loss of PA-II whereas the bound of Freund and Schapire is on the number of mistakes.

5. Regression

In this section we show that the algorithms described in Sec. 3 can be modified to deal with online regression problems. In the regression setting, every instance $\mathbf{x}_t$ is associated with a real target value $y_t \in \mathbb{R}$, which the online algorithm tries to predict. On every round, the algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^n$ and predicts a target value $\hat{y}_t \in \mathbb{R}$ using its internal regression function. We focus on the class of linear regression functions, that is, $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$ where $\mathbf{w}_t$ is the incrementally learned vector. After making a prediction, the algorithm is given the true target value y_t and suffers an instantaneous loss. We use the ε-insensitive hinge loss function:

$$\ell_\varepsilon\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \begin{cases} 0 & |\mathbf{w}\cdot\mathbf{x}-y| \le \varepsilon \\ |\mathbf{w}\cdot\mathbf{x}-y|-\varepsilon & \text{otherwise} \end{cases}, \tag{18}$$

where ε is a positive parameter which controls the sensitivity to prediction mistakes. This loss is zero when the predicted target deviates from the true target by less than ε and otherwise grows linearly with $|\hat{y}_t - y_t|$. At the end of every round, the algorithm uses $\mathbf{w}_t$ and the example $(\mathbf{x}_t, y_t)$ to generate a new weight vector $\mathbf{w}_{t+1}$, which will be used to extend the prediction on the next round.

We now describe how the various PA algorithms from Sec. 3 can be adapted to learn regression problems. As in the case of classification, we initialize $\mathbf{w}_1$ to $(0,\ldots,0)$. On each round, the PA regression algorithm sets the new weight vector to be,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_\varepsilon\big(\mathbf{w};(\mathbf{x}_t,y_t)\big) = 0, \tag{19}$$

In the binary classification setting, we gave the PA update the geometric interpretation of projecting $\mathbf{w}_t$ onto the linear half-space defined by the constraint $\ell\big(\mathbf{w};(\mathbf{x}_t,y_t)\big) = 0$. For regression problems, the set $\{\mathbf{w}\in\mathbb{R}^n : \ell_\varepsilon(\mathbf{w},\mathbf{z}_t) = 0\}$ is not a half-space but rather a hyper-slab of width 2ε. Geometrically, the PA algorithm for regression projects $\mathbf{w}_t$ onto this hyper-slab at the end of every round. Using the shorthand $\ell_t = \ell_\varepsilon(\mathbf{w}_t;(\mathbf{x}_t,y_t))$, the update given in Eq. (19) has a closed form solution similar to that of the classification PA algorithm of the previous section, namely,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \operatorname{sign}(y_t - \hat{y}_t)\tau_t\mathbf{x}_t \qquad \text{where} \qquad \tau_t = \ell_t/\|\mathbf{x}_t\|^2.$$

We can also obtain the PA-I and PA-II variants for online regression by introducing a slack variable into the optimization problem in Eq. (19), as we did for classification in Eq. (6) and Eq. (7). The closed form solution for these updates also comes out to be $\mathbf{w}_{t+1} = \mathbf{w}_t + \operatorname{sign}(y_t - \hat{y}_t)\tau_t\mathbf{x}_t$ where τ_t is defined as in Eq. (8). The derivations of these closed-form updates are almost identical to that of the classification problem in Sec. 3.

We now turn to the analysis of the three PA regression algorithms described above. We would like to show that the analysis given in Sec. 4 for the classification algorithms also holds for their regression counterparts. To do so, it suffices to show that Lemma 1 still holds for regression problems. After obtaining a regression version of Lemma 1, regression versions of Thm. 2 through Thm. 5 follow as immediate corollaries.

Lemma 6 *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be an arbitrary sequence of examples, where $\mathbf{x}_t\in\mathbb{R}^n$ and $y_t\in\mathbb{R}$ for all t. Let τ_t be as defined in either of the three PA variants for regression problems. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u}\in\mathbb{R}^n$,*

$$\sum_{t=1}^{T}\tau_t\left(2\ell_t - \tau_t\|\mathbf{x}_t\|^2 - 2\ell_t^\star\right) \;\le\; \|\mathbf{u}\|^2.$$

Proof The proof of this lemma follows that of Lemma 1 and therefore subtleties which were discussed in detail in that proof are omitted here. Again, we use the definition

$$\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$$

and the same argument used in Lemma 1 implies that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2,$$

We focus our attention on bounding Δ_t from below on those rounds where $\Delta_t \neq 0$. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned} \Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + \text{sign}(y_t - \hat{y}_t)\tau_t \mathbf{x}_t\|^2 \\ &= -\text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t - \tau_t^2 \|\mathbf{x}_t\|^2 \end{aligned}$$

We now add and subtract the term $\text{sign}(y_t - \hat{y}_t)2\tau_t y_t$ from the right-hand side above to get the bound,

$$\Delta_t \;\geq\; -\text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) + \text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{u} \cdot \mathbf{x}_t - y_t) - \tau_t^2 \|\mathbf{x}_t\|^2. \tag{20}$$

Since $\mathbf{w}_t \cdot \mathbf{x}_t = \hat{y}_t$, we have that $-\text{sign}(y_t - \hat{y}_t)(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) = |\mathbf{w}_t \cdot \mathbf{x}_t - y_t|$. We only need to consider the case where $\Delta_t \neq 0$, so $\ell_t = |\mathbf{w}_t \cdot \mathbf{x}_t - y_t| - \varepsilon$ and we can rewrite the bound in Eq. (20) as,

$$\Delta_t \;\geq\; 2\tau_t(\ell_t + \varepsilon) + \text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{u} \cdot \mathbf{x}_t - y_t) - \tau_t^2 \|\mathbf{x}_t\|^2.$$

We also know that $\text{sign}(y_t - \hat{y}_t)(\mathbf{u} \cdot \mathbf{x}_t - y_t) \geq -|\mathbf{u} \cdot \mathbf{x}_t - y_t|$ and that $-|\mathbf{u} \cdot \mathbf{x}_t - y_t| \geq -(\ell_t^\star + \varepsilon)$. This enables us to further bound,

$$\Delta_t \;\geq\; 2\tau_t(\ell_t + \varepsilon) - 2\tau_t(\ell_t^\star + \varepsilon) - \tau_t^2 \|\mathbf{x}_t\|^2 \;=\; \tau_t(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star).$$

Summing the above over all t and comparing to the upper bound discussed in the beginning of this proof proves the lemma. ■

6. Uniclass Prediction

In this section we present PA algorithms for the uniclass prediction problem. This task involves predicting a sequence of vectors $\mathbf{y}_1, \mathbf{y}_2, \cdots$ where $\mathbf{y}_t \in \mathbb{R}^n$. Uniclass prediction is fundamentally different than classification and regression as the algorithm makes predictions without first observing any external input (such as the instance $\mathbf{x}_t$). Specifically, the algorithm maintains in its memory a vector $\mathbf{w}_t \in \mathbb{R}^n$ and simply predicts the next element of the sequence to be $\mathbf{w}_t$. After extending this prediction, the next element in the sequence is revealed and an instantaneous loss is suffered. We measure loss using the following ε-insensitive loss function:

$$\ell_\varepsilon(\mathbf{w}; \mathbf{y}) \;=\; \begin{cases} 0 & \|\mathbf{w} - \mathbf{y}\| \leq \varepsilon \\ \|\mathbf{w} - \mathbf{y}\| - \varepsilon & \text{otherwise} \end{cases}. \tag{21}$$

As in the regression setting, ε is a positive user-defined parameter. If the prediction is within ε of the true sequence element then no loss is suffered. Otherwise the loss is proportional to the Euclidean

distance between the prediction and the true vector. At the end of each round $\mathbf{w}_t$ is updated in order to have a potentially more accurate prediction on where the next element in the sequence will fall. Equivalently, we can think of uniclass prediction as the task of finding a center-point $\mathbf{w}$ such that as many vectors in the sequence fall within a radius of ε from $\mathbf{w}$. At the end of this section we discuss a generalization of this problem, where the radius ε is also determined by the algorithm.

As before, we initialize $\mathbf{w}_1 = (0,\ldots,0)$. Beginning with the PA algorithm, we define the update for the uniclass prediction algorithm to be,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_\varepsilon(\mathbf{w};\mathbf{y}_t) = 0, \tag{22}$$

Geometrically, $\mathbf{w}_{t+1}$ is set to be the projection of $\mathbf{w}_t$ onto a ball of radius ε about $\mathbf{y}_t$. We now show that the closed form solution of this optimization problem turns out to be,

$$\mathbf{w}_{t+1} \;=\; \left(1-\frac{\ell_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\right)\mathbf{w}_t \;+\; \left(\frac{\ell_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\right)\mathbf{y}_t. \tag{23}$$

First, we rewrite the above equation and express $\mathbf{w}_{t+1}$ by,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}, \tag{24}$$

where $\tau_t = \ell_t$. In the Uniclass problem the KKT conditions are both sufficient and necessary for optimality. Therefore, we prove that Eq. (24) is the minimizer of Eq. (22) by verifying that the KKT conditions indeed hold. The Lagrangian of Eq. (22) is,

$$\mathcal{L}(\mathbf{w},\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + \tau\left(\|\mathbf{w}-\mathbf{y}_t\|-\varepsilon\right), \tag{25}$$

where $\tau \geq 0$ is a Lagrange multiplier. Differentiating with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero, we get the first KKT condition, stating that at the optimum $(\mathbf{w},\tau)$ must satisfy the equality,

$$0 \;=\; \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\tau) \;=\; \mathbf{w}-\mathbf{w}_t+\tau\frac{\mathbf{w}-\mathbf{y}_t}{\|\mathbf{w}-\mathbf{y}_t\|}. \tag{26}$$

In addition, an optimal solution must satisfy the conditions $\tau \geq 0$ and,

$$\tau\left(\|\mathbf{w}-\mathbf{y}_t\|-\varepsilon\right) \;=\; 0. \tag{27}$$

Clearly, $\tau_t \geq 0$. Therefore, to show that $\mathbf{w}_{t+1}$ is the optimum of Eq. (22) it suffices to prove that $(\mathbf{w}_{t+1},\tau_t)$ satisfies Eq. (26) and Eq. (27). These equalities trivially hold if $\ell_t = 0$ and therefore from now on we assume that $\ell_t > 0$. Plugging the values $\mathbf{w} = \mathbf{w}_{t+1}$ and $\tau = \tau_t$ in the right-hand side of Eq. (26) gives,

$$\mathbf{w}_{t+1}-\mathbf{w}_t+\tau_t\frac{\mathbf{w}_{t+1}-\mathbf{y}_t}{\|\mathbf{w}_{t+1}-\mathbf{y}_t\|} \;=\; \tau_t\left(\frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}+\frac{\mathbf{w}_{t+1}-\mathbf{y}_t}{\|\mathbf{w}_{t+1}-\mathbf{y}_t\|}\right). \tag{28}$$

Note that,

$$\begin{aligned}\mathbf{w}_{t+1}-\mathbf{y}_t &= \mathbf{w}_t+\tau_t\frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}-\mathbf{y}_t \;=\; (\mathbf{w}_t-\mathbf{y}_t)\left(1-\tau_t\frac{1}{\|\mathbf{y}_t-\mathbf{w}_t\|}\right)\\ &= \frac{\mathbf{w}_t-\mathbf{y}_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\left(\|\mathbf{w}_t-\mathbf{y}_t\|-\tau_t\right) \;=\; \frac{\varepsilon}{\|\mathbf{w}_t-\mathbf{y}_t\|}(\mathbf{w}_t-\mathbf{y}_t).\end{aligned} \tag{29}$$

Combining Eq. (29) with Eq. (28) we get that,

$$\mathbf{w}_{t+1} - \mathbf{w}_t + \tau_t \frac{\mathbf{w}_{t+1} - \mathbf{y}_t}{\|\mathbf{w}_{t+1} - \mathbf{y}_t\|} \;=\; 0,$$

and thus Eq. (26) holds for $(\mathbf{w}_{t+1}, \tau_t)$. Similarly,

$$\|\mathbf{w}_{t+1} - \mathbf{y}_t\| - \varepsilon \;=\; \varepsilon - \varepsilon \;=\; 0,$$

and thus Eq. (27) also holds. In summary, we have shown that the KKT optimality conditions hold for $(\mathbf{w}_{t+1}, \tau_t)$ and therefore Eq. (24) gives the desired closed-form update.

To obtain uniclass versions of PA-I and PA-II , we add a slack variable to the optimization problem in Eq. (22) in the same way as we did in Eq. (6) and Eq. (7) for the classification algorithms. Namely, the update for PA-I is defined by,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \quad \text{s.t.} \quad \|\mathbf{w} - \mathbf{y}_t\| \leq \varepsilon + \xi, \quad \xi \geq 0, \tag{30}$$

and the update for PA-II is,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{s.t.} \quad \|\mathbf{w} - \mathbf{y}_t\| \leq \varepsilon + \xi.$$

The closed form for these updates can be derived using the same technique as we used for deriving the PA update. The final outcome is that both PA-I and PA-II share the form of update given in Eq. (24), with τ_t set to be,

$$\tau_t = \min\{\, C \,,\, \ell_t \,\} \quad \text{(PA-I)} \qquad \text{or} \qquad \tau_t = \frac{\ell_t}{1 + \frac{1}{2C}} \quad \text{(PA-II)}.$$

We can extend the analysis of the three PA variants from Sec. 4 to the case of uniclass prediction. We do so by proving a uniclass version of Lemma 1. After proving this lemma, we discuss an additional technical difficulty which needs to be addressed so that Thm. 2 through Thm. 5 carry over smoothly to the uniclass case.

Lemma 7 *Let $\mathbf{y}_1, \ldots, \mathbf{y}_T$ be an arbitrary sequence of vectors, where $\mathbf{y}_t \in \mathbb{R}^n$ for all t. Let τ_t be as defined in either of the three PA variants for uniclass prediction. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u} \in \mathbb{R}^n$,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t - 2\ell_t^\star\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof We prove this lemma in much the same way as we did Lemma 1. We again use the definition, $\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$, along with the fact stated in Eq. (10) that

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2.$$

We now focus our attention on bounding Δ_t from below on those rounds where $\Delta_t \neq 0$. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned}
\Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left\| \left(1 - \frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) \mathbf{w}_t + \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) \mathbf{y}_t - \mathbf{u} \right\|^2 \\
&= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left\| (\mathbf{w}_t - \mathbf{u}) + \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{y}_t - \mathbf{w}_t) \right\|^2 \\
&= -2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{w}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 .
\end{aligned}$$

We now add and subtract $\mathbf{y}_t$ from the term $(\mathbf{w}_t - \mathbf{u})$ above to get,

$$\begin{aligned}
\Delta_t &= -2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{w}_t - \mathbf{y}_t + \mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 \\
&= 2\tau_t \|\mathbf{w}_t - \mathbf{y}_t\| - 2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 .
\end{aligned}$$

Now, using the Cauchy-Schwartz inequality on the term $(\mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t)$, we can bound,

$$\Delta_t \;\geq\; 2\tau_t \|\mathbf{w}_t - \mathbf{y}_t\| - 2\tau_t \|\mathbf{y}_t - \mathbf{u}\| - \tau_t^2 .$$

We now add and subtract $2\tau_t \varepsilon$ from the right-hand side of the above, to get,

$$\Delta_t \;\geq\; 2\tau_t \left(\|\mathbf{w}_t - \mathbf{y}_t\| - \varepsilon\right) - 2\tau_t \left(\|\mathbf{y}_t - \mathbf{u}\| - \varepsilon\right) - \tau_t^2 .$$

Since we are dealing with the case where $\ell_t > 0$, it holds that $\ell_t = \|\mathbf{w}_t - \mathbf{y}_t\| - \varepsilon$. By definition, $\ell_t^\star \geq \|\mathbf{u} - \mathbf{y}_t\| - \varepsilon$. Using these two facts, we get,

$$\Delta_t \;\geq\; 2\tau_t \ell_t - 2\tau_t \ell_t^\star - \tau_t^2 .$$

Summing the above inequality over all t and comparing the result to the upper bound in Eq. (10) gives the bound stated in the lemma. ■

As mentioned above, there remains one more technical obstacle which stands in the way of applying Thm. 2 through Thm. 5 to the uniclass case. This difficulty stems from the fact $\mathbf{x}_t$ is not defined in the uniclass whereas the term $\|\mathbf{x}\|^2$ appears in the theorems. This issue is easily resolved by setting $\mathbf{x}_t$ in the uniclass case to be an arbitrary vector of a unit length, namely $\|\mathbf{x}_t\|^2 = 1$. This technical modification enables us to write τ_t as $\ell_t / \|\mathbf{x}_t\|^2$ in the uniclass PA algorithm, as in the classification case. Similarly, τ_t can be defined as in the classification case for PA-I and PA-II . Now Thm. 2 through Thm. 5 can be applied verbatim to the uniclass PA algorithms.

Learning the Radius of the Uniclass Predictor In the derivation above we made the simplifying assumption that ε, the radius of our uniclass predictor, is fixed beforehand and that the online algorithm can only move its center, $\mathbf{w}$. We now show that learning ε and $\mathbf{w}$ in parallel is no harder than learning $\mathbf{w}$ alone. We do so by using a simple reduction argument. For technical reasons, we still require an upper bound on ε, which we denote by B. Although B is specified ahead of time, it can

be arbitrarily large and does not appear in our analysis. Typically, we will think of B as being far greater than any conceivable value of ε. Our goal is now to incrementally find $\mathbf{w}_t$ and ε_t such that,

$$\|\mathbf{w}_t - \mathbf{y}_t\| \leq \varepsilon_t, \tag{31}$$

as often as possible. Additionally, we would like ε_t to stay relatively small, since an extremely large value of ε_t would solve the problem in a trivial way. We do so by reducing this problem to a different uniclass problem where the radius is fixed and where $\mathbf{y}_t$ is in $\mathbb{R}^{n+1}$. That is, by adding an additional dimension to the problem, we can learn ε using the same machinery developed for fixed-radius uniclass problems. The reduction stems from the observation that Eq. (31) can be written equivalently as,

$$\|\mathbf{w}_t - \mathbf{y}_t\|^2 + (B^2 - \varepsilon_t^2) \;\leq\; B^2. \tag{32}$$

If we were to concatenate a 0 to the end of every $\mathbf{y}_t$ (thus increasing its dimension to $n+1$) and if we considered the $n+1$'th coordinate of $\mathbf{w}_t$ to be equivalent to $\sqrt{B^2 - \varepsilon_t^2}$, then Eq. (32) simply becomes $\|\mathbf{w}_t - \mathbf{y}_t\|^2 \;\leq\; B^2$. Our problem has reduced to a fixed-radius uniclass problem where the radius is set to B. $w_{1,n+1}$ should be initialized to B, which is equivalent to initializing $\varepsilon_1 = 0$. On each round, ε_t can be extracted from $\mathbf{w}_t$ by,

$$\varepsilon_t \;=\; \sqrt{B^2 - w_{t,n+1}^2}.$$

Since $w_{t+1,n+1}$ is defined to be a convex combination of $w_{t,n+1}$ and $y_{t,n+1}$ (where the latter equals zero), then $w_{t,n+1}$ is bounded in $(0,B]$ for all t and can only decrease from round to round. This means that the radius ε_t is always well defined and can only increase with t. Since the radius is initialized to zero and is now one of the learned parameters, the algorithm has a natural tendency to favor small radii. Let $\mathbf{u}$ denote the center of a fixed uniclass predictor and let ε denote its radius. Then the reduction described above enables us to prove loss bounds similar to those presented in Sec. 4, with $\|\mathbf{u}\|^2$ replaced by $\|\mathbf{u}\|^2 + \varepsilon^2$.

7. Multiclass Problems

We now address more complex decision problems. We first adapt the binary classification algorithms described in Sec. 3 to the task of *multiclass multilabel* classification. In this setting, every instance is associated with a set of labels Y_t. For concreteness we assume that there are k different possible labels and denote the set of all possible labels by $\mathcal{Y} = \{1,\ldots,k\}$. For every instance $\mathbf{x}_t$, the set of relevant labels Y_t is therefore a subset of $\mathcal{Y}$. We say that label y is *relevant* to the instance $\mathbf{x}_t$ if $y \in Y_t$. This setting is often discussed in text categorization applications (see for instance (Schapire and Singer, 2000)) where $\mathbf{x}_t$ represents a document and Y_t is the set of topics which are relevant to the document and is chosen from a predefined collection of topics. The special case where there is only a *single* relevant topic for each instance is typically referred to as *multiclass single-label* classification or multiclass categorization for short. As discussed below, our adaptation of the PA variants to multiclass multilabel settings encompasses the single-label setting as a special case.

As in the previous sections, the algorithm receives instances $\mathbf{x}_1, \mathbf{x}_2, \ldots$ in a sequential manner where each $\mathbf{x}_t$ belongs to an instance space $\mathcal{X}$. Upon receiving an instance, the algorithm outputs a score for each of the k labels in $\mathcal{Y}$. That is, the algorithm's prediction is a vector in $\mathbb{R}^k$ where each element in the vector corresponds to the score assigned to the respective label. This form of prediction is often referred to as label ranking. Predicting a label ranking is more general and

flexible than predicting the set of relevant labels Y_t. Special purpose learning algorithms such as AdaBoost.MR (Schapire and Singer, 1998) and adaptations of support vector machines (Crammer and Singer, 2003a) have been devised for the task of label ranking. Here we describe a reduction from online label ranking to online binary classification that deems label ranking as simple as binary prediction. We note that in the case of multiclass single-label classification, the prediction of the algorithm is simply set to be the label with the highest score.

For a pair of labels $r,s \in \mathcal{Y}$, if the score assigned by the algorithm to label r is greater than the score assigned to label s, we say that label r is *ranked* higher than label s. The goal of the algorithm is to rank every relevant label above every irrelevant label. Assume that we are provided with a set of d features $\phi_1,\ldots,\phi_d$ where each feature ϕ_j is a mapping from $\mathcal{X}\times\mathcal{Y}$ to the reals. We denote by $\Phi(\mathbf{x},y) = (\phi_1(\mathbf{x},y),\ldots,\phi_d(\mathbf{x},y))$ the vector formed by concatenating the outputs of the features, when each feature is applied to the pair $(\mathbf{x},y)$. The label ranking function discussed in this section is parameterized by a weight vector, $\mathbf{w}\in\mathbb{R}^d$. On round t, the prediction of the algorithm is the k-dimensional vector,

$$\Big((\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,1)),\ldots,(\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,k))\Big).$$

We motivate our construction with an example from the domain of text categorization. We describe a variant of the *Term Frequency - Inverse Document Frequency* (TF-IDF) representation of documents (Rocchio, 1971; Salton and Buckley, 1988). Each feature ϕ_j corresponds to a different word, denoted μ_j. Given a corpus of documents S, for every $\mathbf{x}\in S$ and for every potential topic y, the feature $\phi_j(\mathbf{x},y)$ is defined to be,

$$\phi_j(\mathbf{x},y) \;=\; \mathrm{TF}(\mu_j,\mathbf{x})\cdot\log\left(\frac{|S|}{\mathrm{DF}(\mu_j,y)}\right),$$

where $\mathrm{TF}(\mu_j,\mathbf{x})$ is the number of times μ_j appears in $\mathbf{x}$ and $\mathrm{DF}(\mu_j,y)$ is the number of times μ_j appears in all of the documents in S which are *not* labeled by y. The value ϕ_j grows in proportion to the frequency of μ_j in the document $\mathbf{x}$ but is dampened if μ_j is a frequent word for topics other than y. In the context of this paper, the important point is that each feature is label-dependent.

After making its prediction (a ranking of the labels), the algorithm receives the correct set of relevant labels Y_t. We define the *margin* attained by the algorithm on round t for the example $(\mathbf{x}_t,Y_t)$ as,

$$\gamma\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big) \;=\; \min_{r\in Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) \;-\; \max_{s\notin Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,s).$$

This definition generalizes the definition of margin for binary classification and was employed by both single-label and multilabel learning algorithms for support vector machines (Vapnik, 1998; Weston and Watkins, 1999; Elisseeff and Weston, 2001; Crammer and Singer, 2003a). In words, the margin is the difference between the score of the lowest ranked relevant label and the score of the highest ranked irrelevant label. The margin is positive only if all of the relevant labels are ranked higher than all of the irrelevant labels. However, in the spirit of binary classification, we are not satisfied by a mere positive margin as we require the margin of every prediction to be at least 1. After receiving Y_t, we suffer an instantaneous loss defined by the following hinge-loss function,

$$\ell_{\mathrm{MC}}\big(\mathbf{w};(\mathbf{x},Y)\big) \;=\; \begin{cases} 0 & \gamma\big(\mathbf{w};(\mathbf{x},Y)\big)\geq 1 \\ 1-\gamma\big(\mathbf{w};(\mathbf{x},Y)\big) & \text{otherwise} \end{cases}. \tag{33}$$

As in the previous sections, we use ℓ_t as an abbreviation for $\ell_{\text{MC}}\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big)$. If an irrelevant label is ranked higher than a relevant label, then ℓ_t^2 attains a value greater than 1. Therefore, $\sum_{t=1}^{T}\ell_t^2$ upper bounds the number of multiclass prediction mistakes made on rounds 1 through T.

One way of updating the weight vector $\mathbf{w}_t$ is to mimic the derivation of the PA algorithm for binary classification defined in Sec. 3 and to set

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d}\frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_{\text{MC}}(\mathbf{w};(\mathbf{x}_t,Y_t))=0. \tag{34}$$

Satisfying the single constraint in the optimization problem above is equivalent to satisfying the following set of linear constraints,

$$\forall r\in Y_t \quad \forall s\notin Y_t \qquad \mathbf{w}\cdot\Phi(\mathbf{x}_t,r)-\mathbf{w}\cdot\Phi(\mathbf{x}_t,s)\;\geq\;1. \tag{35}$$

However, instead of attempting to satisfy all of the $|\mathcal{Y}_t|\times(k-|\mathcal{Y}_t|)$ constraints above we focus only on the single constraint which is violated the most by $\mathbf{w}_t$. We show in the sequel that we can still prove a cumulative loss bound for this simplified version of the update. We note that satisfying all of these constraints simultaneously leads to the online algorithm presented in (Crammer and Singer, 2003a). Their online update is more involved and computationally expensive, and moreover, their analysis only covers the realizable case.

Formally, let r_t denote the lowest ranked relevant label and let s_t denote the highest ranked irrelevant label on round t. That is,

$$r_t \;=\; \operatorname*{argmin}_{r\in Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) \qquad \text{and} \qquad s_t \;=\; \operatorname*{argmax}_{s\notin Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,s). \tag{36}$$

The single constraint that we choose to satisfy is $\mathbf{w}\cdot\Phi(\mathbf{x}_t,r_t)-\mathbf{w}\cdot\Phi(\mathbf{x}_t,s_t)\geq 1$ and thus $\mathbf{w}_{t+1}$ is set to be the solution of the following simplified constrained optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}}\frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}\cdot(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t))\;\geq\;1. \tag{37}$$

The apparent benefit of this simplification lies in the fact that Eq. (37) has a closed form solution. To draw the connection between the multilabel setting and binary classification, we can think of the vector $\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)$ as a virtual instance of a binary classification problem with a label of $+1$. With this reduction in mind, Eq. (37) becomes equivalent to Eq. (2). Therefore, the closed form solution of Eq. (37) is

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t+\tau_t(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)). \tag{38}$$

with,

$$\tau_t \;=\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)\|^2}.$$

Although we are essentially neglecting all but two labels on each step of the multiclass update, we can still obtain multiclass cumulative loss bounds. The key observation in our analysis it that,

$$\ell_{\text{MC}}\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big) \;=\; \ell\big(\mathbf{w}_t;(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t),+1)\big).$$

To remind the reader, ℓ on the right-hand side of the above equation is the binary classification loss defined in Eq. (1). Using this equivalence of definitions, we can convert Thm. 2 into a bound for

the multiclass PA algorithm. To do so we need to cast the assumption that for all t it holds that $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\| \leq R$. This bound can immediately be converted into a bound on the norm of the feature set since $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\| \leq \|\Phi(\mathbf{x}_t, r_t)\| + \|\Phi(\mathbf{x}_t, s_t)\|$. Thus, if the norm of the mapping $\Phi(\mathbf{x}_t, r)$ is bounded for all t and r then so is $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|$. In particular, if we assume that $\|\Phi(\mathbf{x}_t, r)\| \leq R/2$ for all t and r we obtain the following corollary.

Corollary 8 *Let $(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_T, Y_T)$ be a sequence of examples with $\mathbf{x}_t \in \mathbb{R}^n$ and $Y_T \subseteq \{1, \ldots, k\}$. Let Φ be a mapping $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ such that $\|\Phi(\mathbf{x}_t, r)\| \leq R/2$ for all t and r. Assume that there exists a vector $\mathbf{u}$ such that $\ell(\mathbf{u}; (\mathbf{x}_t, Y_t)) = 0$ for all t. Then, the cumulative squared loss attained by the multiclass multilabel PA algorithm is bounded from above by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; R^2 \|\mathbf{u}\|^2.$$

Similarly, we can obtain multiclass versions of PA-I and PA-II by using the update rule in Eq. (38) but setting τ_t to be either,

$$\tau_t \;=\; \min\left\{ C \,,\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2} \right\} \qquad \text{or} \qquad \tau_t \;=\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2 + \frac{1}{2C}},$$

respectively. The analysis of PA-I and PA-II in Thms. 4-5 also carries over from the binary case to the multilabel case in the same way.

Multi-prototype Classification In the above discussion we assumed that the feature vector $\Phi(\mathbf{x}, y)$ is label-dependent and used a single weight vector $\mathbf{w}$ to form the ranking function. However, in many applications of multiclass classification this setup is somewhat unnatural. Many times, there is a single natural representation for every instance rather than multiple feature representations for each individual class. For example, in optical character recognition problems (OCR) an instance can be a gray-scale image of the character and the goal is to output the content of this image. In this example, it is difficult to find a good set of label-dependent features.

The common construction in such settings is to assume that each instance is a vector in $\mathbb{R}^n$ and to associate a different weight vector (often referred to as prototype) with each of the k labels (Vapnik, 1998; Weston and Watkins, 1999; Crammer and Singer, 2001). That is, the multiclass predictor is now parameterized by $\mathbf{w}_t^1, \ldots, \mathbf{w}_t^k$, where $\mathbf{w}_t^r \in \mathbb{R}^n$. The output of the predictor is defined to be,

$$\Big((\mathbf{w}_t^1 \cdot \mathbf{x}_t), \ldots, (\mathbf{w}_t^k \cdot \mathbf{x}_t) \Big).$$

To distinguish this setting from the previous one we refer to this setting as the multi-prototype multiclass setting and to the previous one as the single-prototype multiclass setting. We now describe a reduction from the multi-prototype setting to the single-prototype one which enables us to use all of the multiclass algorithms discussed above in the multi-prototype setting as well. To obtain the desired reduction, we must define the feature vector representation $\Phi(\mathbf{x}, y)$ induced by the instance label pair $(\mathbf{x}, y)$. We define $\Phi(\mathbf{x}, y)$ to be a $k \cdot n$ dimensional vector which is composed of k blocks of size n. All blocks but the y'th block of $\Phi(\mathbf{x}, y)$ are set to be the zero vector while the y'th block is set to be $\mathbf{x}$. Applying a single prototype multiclass algorithm to this problem produces a weight vector $\mathbf{w}_t \in \mathbb{R}^{kn}$ on every online round. Analogous to the construction of $\Phi(\mathbf{x}, y)$, the vector $\mathbf{w}_t$ is composed

INPUT: cost function $\rho(y,y')$
INITIALIZE: $\mathbf{w}_1 = (0,\ldots,0)$
For $t = 1,2,\ldots$

- receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
- predict: $\hat{y}_t = \arg\max_{y\in\mathcal{Y}} (\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y))$
- receive correct label: $y_t \in \mathcal{Y}$
- define: $\tilde{y}_t = \arg\max_{r\in\mathcal{Y}} \left(\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, r)}\right)$
- define:
$$q_t = \begin{cases} \hat{y}_t & \text{(PB)} \\ \tilde{y}_t & \text{(ML)} \end{cases}$$
- suffer loss: $\ell_t = \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, q_t) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, q_t)}$
- set: $\tau_t = \frac{\ell_t}{\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, q_t)\|^2}$
- update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \left(\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, q_t)\right)$

Figure 2: The *prediction-based* (PB) and *max-loss* (ML) passive-aggressive updates for cost-sensitive multiclass problems.

of k blocks of size n and denote block r by $\mathbf{w}_t^r$. By construction, we get that $\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) = \mathbf{w}_t^r \cdot \mathbf{x}_t$. Equipped with this construction we can use verbatim any single-prototype algorithm as a proxy for the multi-prototype variant. Namely, on round t we find the pair of indices r_t, s_t which corresponds to the largest violation of the margin constraints,

$$\begin{aligned} r_t &= \operatorname*{argmin}_{r\in Y_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) = \operatorname*{argmin}_{r\in Y_t} \mathbf{w}_t^r \cdot \mathbf{x}_t \ , \\ s_t &= \operatorname*{argmax}_{s\notin Y_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, s) = \operatorname*{argmax}_{s\notin Y_t} \mathbf{w}_t^s \cdot \mathbf{x}_t . \end{aligned} \tag{39}$$

Unraveling the single-prototype notion of margin and casting it as a multi-prototype one we get that the loss in the multi-prototype case amounts to,

$$\ell\left(\mathbf{w}_t^1, \ldots, \mathbf{w}_t^k; (\mathbf{x}_t, Y_t)\right) = \begin{cases} 0 & \mathbf{w}_t^{r_t} \cdot \mathbf{x}_t - \mathbf{w}_t^{s_t} \cdot \mathbf{x}_t \geq 1 \\ 1 - \mathbf{w}_t^{r_t} \cdot \mathbf{x}_t + \mathbf{w}_t^{s_t} \cdot \mathbf{x}_t & \text{otherwise} \end{cases} . \tag{40}$$

Furthermore, applying the same reduction to the update scheme we get that the resulting multi-prototype update is,

$$\mathbf{w}_{t+1}^{r_t} = \mathbf{w}_{t+1}^{r_t} + \tau_t \mathbf{x}_t \quad \text{and} \quad \mathbf{w}_{t+1}^{s_t} = \mathbf{w}_{t+1}^{s_t} - \tau_t \mathbf{x}_t . \tag{41}$$

For the PA algorithm, the value of τ_t is the ratio of the loss, as given by Eq. (40), and the squared norm of $\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)$. By construction, this vector has $k-2$ blocks whose elements are zeros and two blocks that are equal to $\mathbf{x}_t$ and $-\mathbf{x}_t$. Since the two non-zero blocks are non-overlapping we get that,

$$\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2 = \|\mathbf{x}_t\|^2 + \|-\mathbf{x}_t\|^2 = 2\|\mathbf{x}_t\|^2 .$$

Finally, due to our reduction we also get multi-prototype versions of Thm. 4 and Thm. 5.

8. Cost-Sensitive Multiclass Classification

Cost-sensitive multiclass classification is a variant of the multiclass single-label classification setting discussed in the previous section. Namely, each instance $\mathbf{x}_t$ is associated with a single correct label $y_t \in \mathcal{Y}$ and the prediction extended by the online algorithm is simply,

$$\hat{y}_t \;=\; \operatorname*{argmax}_{y \in \mathcal{Y}} \left(\mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y)\right). \tag{42}$$

A prediction mistake occurs if $y_t \neq \hat{y}_t$, however in the cost-sensitive setting different mistakes incur different levels of cost. Specifically, for every pair of labels (y, y') there is a cost $\rho(y, y')$ associated with predicting y' when the correct label is y. The cost function ρ is defined by the user and takes non-negative values. We assume that $\rho(y, y) = 0$ for all $y \in \mathcal{Y}$ and that $\rho(y, y') \geq 0$ whenever $y \neq y'$. The goal of the algorithm is to minimize the *cumulative cost* suffered on a sequence of examples, namely to minimize $\sum \rho(y_t, \hat{y}_t)$.

The multiclass PA algorithms discussed above can be adapted to this task by incorporating the cost function into the online update. Recall that we began the derivation of the multiclass PA update by defining a set of margin constraints in Eq. (35), and on every round we focused our attention on satisfying only one of these constraints. We repeat this idea here while incorporating the cost function into the margin constraints. Specifically, on every online round we would like for the following constraints to hold,

$$\forall r \in \{\mathcal{Y} \setminus y_t\} \quad \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, r) \;\geq\; \sqrt{\rho(y_t, r)}. \tag{43}$$

The reason for using the square root function in the equation above will be justified shortly. As mentioned above, the online update focuses on a single constraint out of the $|\mathcal{Y}| - 1$ constraints in Eq. (43). We will describe and analyze two different ways to choose this single constraint, which lead to two different online updates for cost-sensitive classification. The two update techniques are called the *prediction-based* update and the *max-loss* update. Pseudo-code for these two updates is presented in Fig. 2. They share an almost identical analysis and may seem very similar at first, however each update possesses unique qualities. We discuss the significance of each update at the end of this section.

The prediction-based update focuses on the single constraint in Eq. (43) which corresponds to the predicted label $\hat{y}_t$. Concretely, this update sets $\mathbf{w}_{t+1}$ to be the solution to the following optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, \hat{y}_t) \;\geq\; \sqrt{\rho(y_t, \hat{y}_t)}, \tag{44}$$

where $\hat{y}_t$ is defined in Eq. (42). This update closely resembles the multiclass update given in Eq. (37). Define the cost sensitive loss for the prediction-based update to be,

$$\ell_{\text{PB}}\big(\mathbf{w}; (\mathbf{x}, y)\big) \;=\; \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, \hat{y}) - \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, y) + \sqrt{\rho(y, \hat{y})}. \tag{45}$$

Note that this loss equals zero if and only if a correct prediction was made, namely if $\hat{y}_t = y_t$. On the other hand, if a prediction mistake occurred it means that $\mathbf{w}_t$ ranked $\hat{y}_t$ higher than y_t, thus,

$$\sqrt{\rho(y_t, \hat{y}_t)} \;\leq\; \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, \hat{y}_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, \hat{y}_t)} \;=\; \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)). \tag{46}$$

As in previous sections, we will prove an upper bound on the cumulative squared loss attained by our algorithm, $\sum_t \ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))^2$. The cumulative squared loss in turn bounds $\sum_t \rho(y_t,\hat{y}_t)$ which is the quantity we are trying to minimize. This explains the rationale behind our choice of the margin constraints in Eq. (43). The update in Eq. (44) has the closed form solution,

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t + \tau_t\left(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\right), \tag{47}$$

where,

$$\tau_t \;=\; \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2}. \tag{48}$$

As before, we obtain cost sensitive versions of PA-I and PA-II by setting,

$$\begin{aligned}
\tau_t &= \min\left\{ C\,,\; \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2} \right\} \qquad \text{(PA-I)} \\
\tau_t &= \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2 + \frac{1}{2C}} \qquad \text{(PA-II)},
\end{aligned} \tag{49}$$

where in both cases $C > 0$ is a user-defined parameter.

The second cost sensitive update, the max-loss update, also focuses on satisfying a single constraint from Eq. (43). Let $\tilde{y}_t$ be the label in $\mathcal{Y}$ defined by,

$$\tilde{y}_t \;=\; \operatorname*{argmax}_{r\in\mathcal{Y}} \left(\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) - \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,y_t) + \sqrt{\rho(y_t,r)}\right). \tag{50}$$

$\tilde{y}_t$ is the loss-maximizing label. That is, we would suffer the greatest loss on round t if we were to predict $\tilde{y}_t$. The max-loss update focuses on the single constraint in Eq. (43) which corresponds to $\tilde{y}_t$. Note that the online algorithm continues to predict the label $\hat{y}_t$ as before and that $\tilde{y}_t$ only influences the online update. Concretely, the max-loss update sets $\mathbf{w}_{t+1}$ to be the solution to the following optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,y_t) - \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,\tilde{y}_t) \;\geq\; \sqrt{\rho(y_t,\tilde{y}_t)}, \tag{51}$$

The update in Eq. (51) has the same closed form solution given in Eq. (47) and Eq. (48) with $\hat{y}_t$ replaced by $\tilde{y}_t$. Define the loss for the max-loss update to be,

$$\ell_{\text{ML}}\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \mathbf{w}\cdot\Phi(\mathbf{x},\tilde{y}) - \mathbf{w}\cdot\Phi(\mathbf{x},y) + \sqrt{\rho(y,\tilde{y})}, \tag{52}$$

where $\tilde{y}$ is defined in Eq. (50). Note that since $\tilde{y}$ attains the maximal loss of all other labels, it follows that,

$$\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) \;\leq\; \ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)).$$

From the above inequality and Eq. (46) we conclude that ℓ_{ML} is also an upper bound on $\sqrt{\rho(y_t,\hat{y}_t)}$. A note-worthy difference between ℓ_{PB} and ℓ_{ML} is that $\ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) = 0$ if and only if Eq. (43) holds for all $r \in \{\mathcal{Y}\setminus y_t\}$, whereas this is not the case for ℓ_{PB}.

The prediction-based and max-loss updates were previously discussed in Dekel et al. (2004a), in the context of hierarchical classification. In that paper, a predefined hierarchy over the label set was used to induce the cost function ρ. The basic online algorithm presented there used the prediction-based update, whereas the max-loss update was mentioned in the context of a batch learning setting.

Dekel et al. (2004a) evaluated both techniques empirically and found them to be highly effective on speech recognition and text classification tasks.

Turning to the analysis of our cost sensitive algorithms, we follow the same strategy used in the analysis of the regression and uniclass algorithms. Namely, we begin by proving a cost sensitive version of Lemma 1 for both the prediction-based and the max-loss updates.

Lemma 9 *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be an arbitrary sequence of examples, where $\mathbf{x}_t \in \mathbb{R}$ and $y_t \in \mathcal{Y}$ for all t. Let $\mathbf{u}$ be an arbitrary vector in $\mathbb{R}^n$. If τ_t is defined as in Eq. (48) or Eq. (49) then,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_{PB}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) - \tau_t \|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2 - 2\ell_{ML}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \;\leq\; \|\mathbf{u}\|^2.$$

If τ_t is defined as in Eq. (48) or Eq. (49) with $\hat{y}_t$ replaced by $\tilde{y}_t$ then,

$$\sum_{t=1}^{T} \tau_t \left(2\ell_{ML}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) - \tau_t \|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\tilde{y}_t)\|^2 - 2\ell_{ML}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof We prove the first statement of the lemma, which involves the prediction-based update rule. The proof of the second statement is identical, except that $\hat{y}_t$ is replaced by $\tilde{y}_t$ and $\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))$ is replaced by $\ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))$.

As in the proof of Lemma 1, we use the definition $\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$ and the fact that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2. \tag{53}$$

We focus our attention on bounding Δ_t from below. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned}\Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + \tau_t(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t))\|^2 \\ &= -2\tau_t(\mathbf{w}_t - \mathbf{u})\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) - \tau_t^2\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2. \end{aligned} \tag{54}$$

By definition, $\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ equals,

$$\max_{r\in\mathcal{Y}} \left(\mathbf{u}\cdot(\Phi(\mathbf{x}_t,r) - \Phi(\mathbf{x}_t,y_t)) + \sqrt{\rho(y_t,r)}\right).$$

Since $\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ is the maximum over $\mathcal{Y}$, it is clearly greater than $\mathbf{u}\cdot(\Phi(\mathbf{x}_t,\hat{y}_t) - \Phi(\mathbf{x}_t,y_t)) + \sqrt{\rho(y_t,\hat{y}_t)}$. This can be written as,

$$\mathbf{u}\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) \;\geq\; \sqrt{\rho(y_t,\hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)).$$

Plugging the above back into Eq. (54) we get,

$$\begin{aligned}\Delta_t \;\geq\; &-2\tau_t\mathbf{w}_t\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) + 2\tau_t\left(\sqrt{\rho(y_t,\hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \\ &- \tau_t^2\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2. \end{aligned} \tag{55}$$

Rearranging terms in the definition of ℓ_{PB}, we get that $\mathbf{w}_t \cdot (\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)) = \sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t))$. This enables us to rewrite Eq. (55) as,

$$\begin{aligned}
\Delta_t &\geq -2\tau_t \left(\sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)) \right) + \\
&\qquad 2\tau_t \left(\sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u}; (\mathbf{x}_t, y_t)) \right) - \tau_t^2 \|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2 \\
&= \tau_t \left(2\ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)) - \tau_t \|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2 - 2\ell_{\text{ML}}(\mathbf{u}; (\mathbf{x}_t, y_t)) \right).
\end{aligned}$$

Summing Δ_t over all t and comparing this lower bound with the upper bound provided in Eq. (53) gives the desired bound. ■

This lemma can now be used to obtain cost sensitive versions of Thms. 2,3 and 5 for both prediction-based and max-loss updates. The proof of these theorems remains essentially the same as before, however one cosmetic change is required: $\|\mathbf{x}_t\|^2$ is replaced by either $\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2$ or $\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \tilde{y}_t)\|^2$ in each of the theorems and throughout their proofs. This provides cumulative cost bounds for the PA and PA-II cost-sensitive algorithms.

Analyzing the cost-sensitive version of PA-I requires a slightly more delicate adaptation of Thm. 4. For brevity, we prove the following theorem for the max-loss variant of the algorithm and note that the proof for the prediction-based variant is essentially identical.

We make two simplifying assumptions: first assume that $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|$ is upper bounded by 1. Second, assume that C, the aggressiveness parameter given to the PA-I algorithm, is an upper bound on the square root of the cost function ρ.

Theorem 10 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \mathcal{Y}$ and $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\| \leq 1$ for all t. Let ρ be a cost function from $\mathcal{Y} \times \mathcal{Y}$ to $\mathbb{R}_+$ and let C, the aggressiveness parameter provided to the PA-I algorithm, be such that $\sqrt{\rho(y_t, \hat{y}_t)} \leq C$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$, the cumulative cost obtained by the max-loss cost sensitive version of PA-I on the sequence is bounded from above by,*

$$\sum_{t=1}^{T} \rho(y_t, \hat{y}_t) \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_{ML}(\mathbf{u}; (\mathbf{x}_t, y_t)).$$

Proof We abbreviate $\rho_t = \rho(y_t, \hat{y}_t)$ and $\ell_t = \ell_{\text{ML}}(\mathbf{w}_t; (\mathbf{x}_t, y_t))$ throughout this proof. $\ell_t \geq \sqrt{\rho_t}$ on every round t, as discussed in this section. τ_t is defined as,

$$\tau_t \;=\; \min \left\{ \frac{\ell_t}{\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|^2} \,,\, C \right\},$$

and due to our assumption on $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|^2$ we get that $\tau_t \geq \min\{\ell_t, C\}$. Combining these two facts gives,

$$\min\{\rho_t, C\sqrt{\rho_t}\} \;\leq\; \tau_t \ell_t.$$

Using our assumption on C, we know that $C\sqrt{\rho_t}$ is at least ρ_t and therefore $\rho_t \leq \tau_t \ell_t$. Summing over all t we get the bound,

$$\sum_{t=1}^{T} \rho_t \;\leq\; \sum_{t=1}^{T} \tau_t \ell_t. \tag{56}$$

Again using the definition of τ_t, we know that $\tau_t \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)) \leq C\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ and that $\tau_t \|\phi(\mathbf{x}_t,y_t) - \phi(\mathbf{x}_t,\tilde{y}_t)\|^2 \leq \ell_t$. Plugging these two inequalities into the second statement of Lemma 9 gives,

$$\sum_{t=1}^{T} \tau_t \ell_t \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)). \tag{57}$$

Combining Eq. (57) with Eq. (56) proves the theorem. ■

This concludes our analysis of the cost-sensitive PA algorithms. We wrap up this section with a discussion on some significant differences between the prediction-based and the max-loss variants of our cost-sensitive algorithms. Both variants utilize the same prediction function to output the predicted label $\hat{y}_t$ however each variant follows a different update strategy and is evaluated with respect to a different loss function. The loss function used to evaluate the prediction-based variant is a function of y_t and $\hat{y}_t$, whereas the loss function used to evaluate the max-loss update essentially ignores $\hat{y}_t$. In this respect, the prediction-based loss is more natural.

On the other hand, the analysis of the prediction-based variant lacks the aesthetics of the max-loss analysis. The analysis of the max-loss algorithm uses ℓ_{ML} to evaluate both the performance of the algorithm and the performance of $\mathbf{u}$, while the analysis of the prediction-based algorithm uses ℓ_{PB} to evaluate the algorithm and ℓ_{ML} to evaluate $\mathbf{u}$. The prediction-based relative bound is to some extent like comparing apples and oranges, since the algorithm and $\mathbf{u}$ are not evaluated using the same loss function. In summary, both algorithms suffer from some theoretical disadvantage and neither of them is theoretically superior to the other.

Finally, we turn our attention to an important algorithmic difference between the two update strategies. The prediction-based update has a great advantage over the max-loss update in that the cost function ρ does not play a role in determining the single constraint which the update focuses on. In some cases, this can significantly speed-up the running time required by the online update. For example, in the following section we exploit this property when devising algorithms for the complex problem of sequence prediction. When reading the following section, note that the max-loss update could not have been used for sequence prediction in place of the prediction-based update. This is perhaps the most significant difference between the two cost sensitive updates.

9. Learning with Structured Output

A trendy and useful application of large margin methods is learning with structured output. In this setting, the set of possible labels are endowed with a predefined structure. Typically, the set of labels is very large and the structure plays a key role in constructing efficient learning and inference procedures. Notable examples for structured label sets are graphs (in particular trees) and sequences (Collins, 2000; Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004). We now overview how the cost-sensitive learning algorithms described in the previous section can be adapted to structured output settings. For concreteness, we focus on an adaptation for sequence prediction. Our derivation however can be easily mapped to other settings of learning with structured output. In sequence prediction problems we are provided with a predefined alphabet $\mathcal{Y} = \{1,\ldots,k\}$. Each input instance is associated with a label which is a sequence over $\mathcal{Y}$. For simplicity we assume that the output sequence is of a fixed length m. Thus, on round t, the learning algorithm receives an instance $\mathbf{x}_t$ and then predicts an output sequence $\hat{\mathbf{y}}_t \in \mathcal{Y}^m$. Upon predicting, the algorithm receives

the correct sequence $\mathbf{y}_t$ that is associated with $\mathbf{x}_t$. As in the cost-sensitive case, the learning algorithm is also provided with a cost function $\rho : \mathcal{Y}^m \times \mathcal{Y}^m \to \mathbb{R}_+$. The value of $\rho(\mathbf{y}, \mathbf{y}')$ represents the cost associated with predicting $\mathbf{y}'$ instead of $\mathbf{y}$. As before we assume that $\rho(\mathbf{y}, \mathbf{y}')$ equals zero if $\mathbf{y} = \mathbf{y}'$. Apart from this requirement, ρ may be any computable function. Most sequence prediction algorithms further assume that ρ is decomposable. Specifically, a common construction (Taskar et al., 2003; Tsochantaridis et al., 2004) is achieved by defining, $\rho(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{m} \tilde{\rho}(y_i, y'_i)$ where $\tilde{\rho}$ is any non-negative (local) cost over $\mathcal{Y} \times \mathcal{Y}$. In contrast, we revert to a general cost function over pairs of sequences.

As in the multiclass settings discussed above, we assume that there exists a set of features $\phi_1, \ldots, \phi_d$ each of which takes as its input an instance $\mathbf{x}$ and a sequence $\mathbf{y}$ and outputs a real number. We again denote by $\Phi(\mathbf{x}, \mathbf{y})$ the vector of features evaluated on $\mathbf{x}$ and $\mathbf{y}$. Equipped with Φ and ρ, we are left with the task of finding,

$$\hat{\mathbf{y}}_t \;=\; \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^m} \left(\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y}) \right), \tag{58}$$

on every online round. With $\hat{\mathbf{y}}_t$ on hand, the PA update for string prediction is identical to the prediction-based update described in the previous section. However, obtaining $\hat{\mathbf{y}}_t$ in the general case may require as many as k^m evaluations of $\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y})$. This problem becomes intractable as m becomes large. We must therefore impose some restrictions on the feature representation Φ which will enable us to find $\hat{\mathbf{y}}_t$ efficiently. A possible restriction on the feature representation is to assume that each feature ϕ_j takes the form,

$$\phi_j(\mathbf{x}_t, \mathbf{y}) \;=\; \psi_j^0(y_1, \mathbf{x}_t) + \sum_{i=2}^{m} \psi_j(y_i, y_{i-1}, \mathbf{x}_t), \tag{59}$$

where ψ_j^0 and ψ_j are any computable functions. This construction is analogous to imposing a first order Markovian structure on the output sequence. This form paves the way for an efficient inference, i.e. solving Eq. (58), using a dynamic programming procedure. Similar yet richer structures such as dynamic Bayes nets can be imposed so long as the solution to Eq. (58) can be computed efficiently. We note in passing that similar representation of Φ using efficiently computable feature sets were proposed in (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004).

The analysis of the cost-sensitive PA updates carries over verbatim to the sequence prediction setting. Our algorithm for learning with structured outputs was successfully applied to the task of music to score alignment in (Shalev-Shwartz et al., 2004a).

10. Experiments

In this section we present experimental results that demonstrate different aspects of our PA algorithms and their accompanying analysis. In Sec. 10.1 we start with two experiments with synthetic data which examine the robustness of our algorithms to noise. In Sec. 10.2 we investigate the effect of the aggressiveness parameter C on the performance of the PA-I and PA-II algorithms. Finally, in Sec. 10.3, we compare our multiclass versions of the PA algorithms to other online algorithms for multiclass problems (Crammer and Singer, 2003a) on natural data sets.

The synthetic data set used in our experiments was generated as follows. First a label was chosen uniformly at random from $\{-1, +1\}$. For positive labeled examples, instances were chosen by randomly sampling a two-dimensional Gaussian with mean $(1, 1)$ and a diagonal covariance matrix

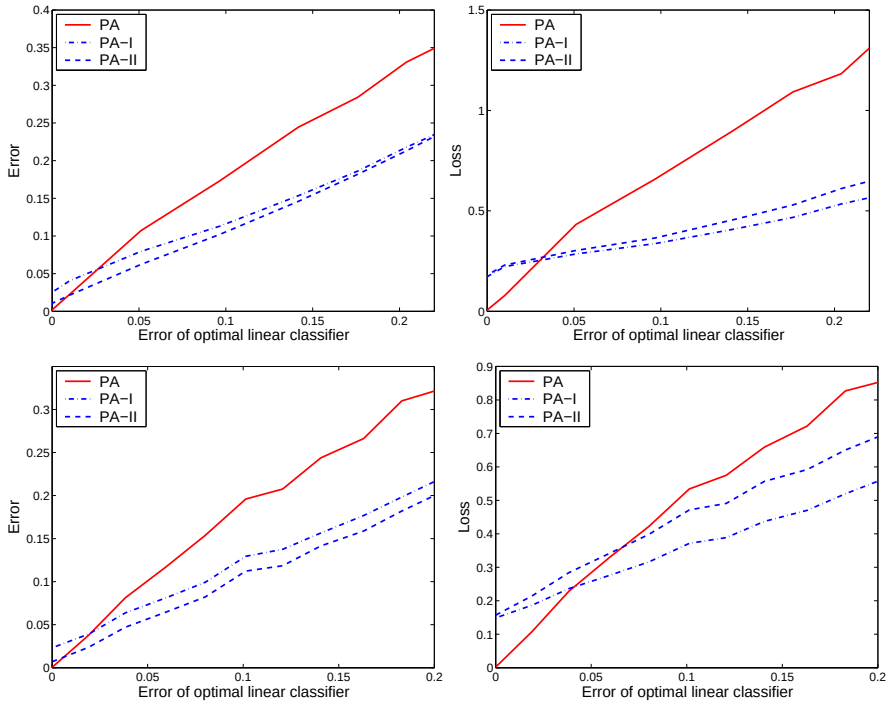

Figure 3: The average error (left) and the average loss (right) of PA, PA-I and PA-II as a function of the error of the optimal fixed linear classifier, in the presence of instance noise (top) and label noise (bottom).

with $(0.2, 2)$ on its diagonal. Similarly, for negative labeled examples, instances were sampled from a Gaussian with a mean of $(-1, -1)$ and the same covariance matrix as for positive labeled examples. To validate our results, we repeated each experiment 10 times where in each repetition we generated 4,000 random examples. The results reported are averaged over the 10 repetitions.

10.1 Robustness to Noise

Our first experiments examine the robustness of our algorithms to both instance noise and label noise. To examine instance noise, we contaminated each instance with a random vector sampled from a zero-mean Gaussian with a covariance matrix σI, where σ varied from 0 to 2. We set the parameter C of PA-I and PA-II to be 0.001. We then ran PA, PA-I and PA-II on the resulting sequence of examples. To evaluate our results, we used a brute-force numerical method to find the optimal fixed linear classifier, that is, the linear classifier that makes the fewest classification mistakes on the entire sequence of examples. We define the *average error* of an online learning algorithm on a given input sequence to be the number of prediction mistakes the algorithm makes on that sequence normalized by the length of the sequence. Similarly, we define the average loss of an online learning algorithm on a given sequence.

In the plots at the top of Fig. 3 we depict the average error and average loss of the three PA variants as a function of the average error of the optimal linear classifier. The plots underscore

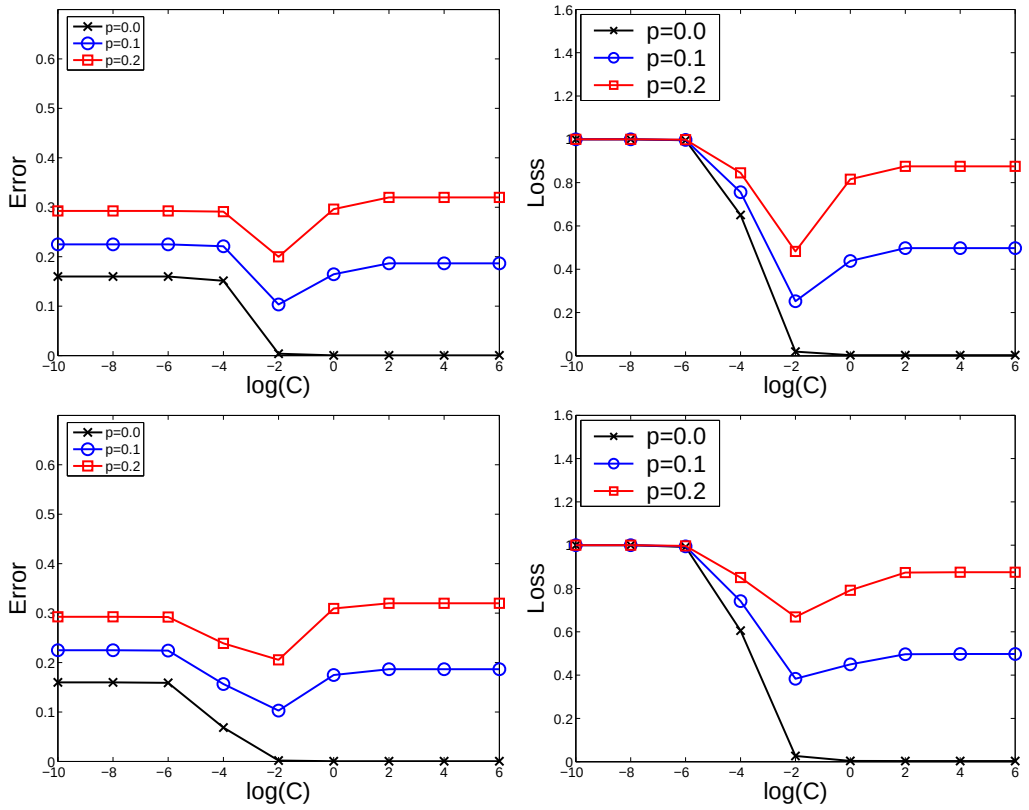

Figure 4: The average error (left) and the average loss (right) of PA-I (top) and PA-II (bottom) as a function of $\log(C)$ with different levels of label noise probability p.

several interesting phenomena. First note that for low levels of noise, all three PA variants make a similar number of errors. Our bounds from Sec. 4 suggest that as the noise level increases, PA-I and PA-II should outperform the basic PA algorithm. It is clear from the graphs that our expectations are met and that PA-I and PA-II outperform the basic PA algorithm when the noise level is high. Finally, in this experiment PA-I and PA-II performed equally well for all levels of noise.

In our second experiment we left the instances intact and instead flipped each label with a probability p, where p was set to different values in $[0, 0.3]$. As in the previous experiment, we set $C = 0.001$ for both PA-I and PA-II . The results are depicted at the bottom of Fig. 3. It is apparent from the graphs that the behavior observed in the previous experiment is repeated here as well.

10.2 The Effect of C

In our second set of experiments, we examine the effect of the aggressiveness parameter C on the performance of PA-I and PA-II . Again we flipped the label of each instance in our synthetic data set with probability p, this time with p set to 0, 0.1 and 0.2. We then ran PA-I and PA-II on the resulting sequence of examples with different values of the parameter C. The average error and average loss of the algorithms as a function of the parameter C are depicted in Fig. 4

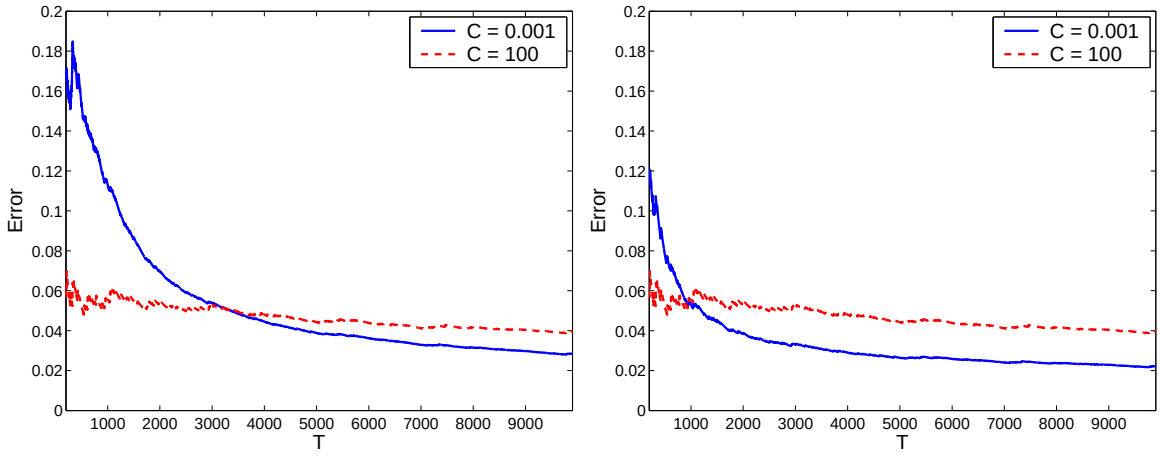

Figure 5: The average error of PA-I (left) and PA-II (right) as a function of the number of online rounds, T, for different values of C.

As can be seen from the graphs, the value of the parameter C significantly effects the results of the algorithms. The graphs can be explained using our loss bounds in Thm. 4 and Thm. 5. For concreteness, let us focus on the loss bound of the PA-II algorithm, given in Thm. 5. The bound on the cumulative loss of the algorithm is comprised of two terms, the first depends on the squared norm of the competitor, $(\|\mathbf{u}\|^2)$, while the second depends on the cumulative (squared) loss of the competitor $(\sum_t (\ell_t^\star)^2)$. The parameter C divides the first term and multiplies the second term. Therefore, when C is small the bound is dominated by the first term $(\|\mathbf{u}\|^2)$ and when C is large the bound is dominated by the second term $(\sum_t (\ell_t^\star)^2)$. Since the label noise applied to the data effects only the second term, we expect that for very small values of C the loss of PA-I and PA-II will be high, regardless of the noise level. On the other hand, as we increase the value of C, the difference between different noise levels becomes apparent. As a general rule-of-thumb, C should be small when the data is noisy.

So far, the length of the sequence of examples presented to the online algorithms was fixed. In the following, we discuss the effect of C on the performance of the algorithms as a function of sequence length (T). We generated a synthetic data set consisting of 10^4 examples with label noise probability $p = 0.02$. We ran the PA-I and PA-II algorithms on the data set, once with $C = 100$ and once with $C = 0.001$. At the end of each online round we calculated the average error attained so far. The results are given in Fig. 5. For both PA-I and PA-II , setting C to be a small number leads to a slow progress rate, since each online update changes the online hypothesis by a small amount. On the other hand, when C is large, the error rate decreases much faster, but at the price of inferior performance later on.

10.3 Multiclass Experiments

Our last experiment demonstrates the efficiency of the PA algorithms on multiclass problems. This experiment was performed with standard multiclass data sets: the USPS and MNIST data sets of handwritten digits. We compared the multiclass versions of PA, PA-I and PA-II to the online

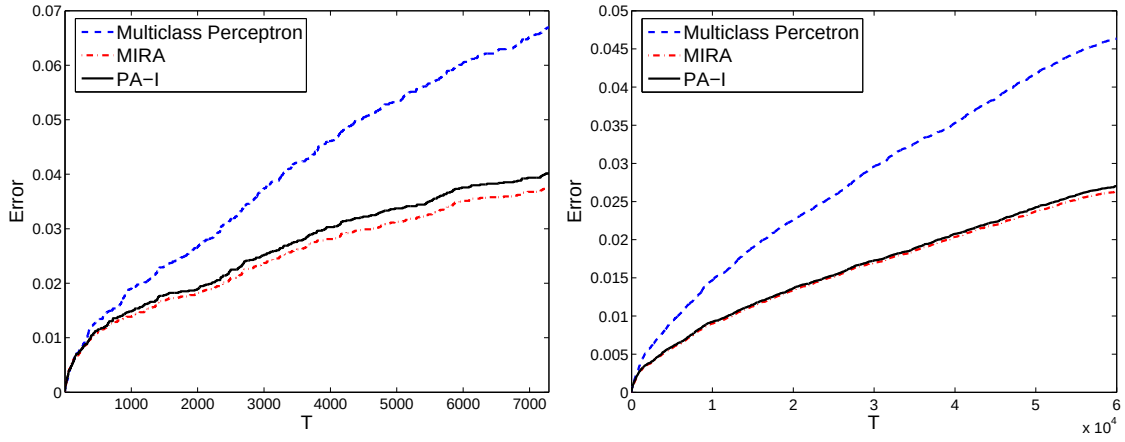

Figure 6: The number of prediction mistakes made by different multiclass online algorithms as a function of the online round index, on the USPS (left) and MNIST (right) data sets.

multiclass algorithms described in (Crammer and Singer, 2003a). Specifically, Crammer and Singer (2003a) present three multiclass versions of the Perceptron algorithm and a new margin based online multiclass algorithm named MIRA. As a preprocessing step, we shifted and scaled the instances of each data set so that its mean equals zero and its average squared Euclidean norm is 1. We used Mercer kernels in all of the algorithms, namely, we replaced the standard dot product with a polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (a + \mathbf{x}_i \cdot \mathbf{x}_j)^d$, where $a = 0$ and $d = 3$ for the USPS data set and $a = 0.5$ and $d = 5$ for the MNIST data set. These kernel parameters were set rather arbitrarily, based on previous experience with these data sets using different algorithms. We set the parameter C of PA-I and PA-II to 100 (we note that similar results hold for any $C > 100$). The parameter β of MIRA was set to 0.01, following (Crammer and Singer, 2003a).

The plots in Fig. 6 depict the number of online prediction mistakes made on the two data sets by three different algorithms: PA-I , the uniform-update version of multiclass Perceptron and MIRA. The performance of PA and PA-II is not presented in this figure, since it is virtually indistinguishable from that of PA-I . For the same reason, only the uniform-update version of the multiclass Perceptron is presented in the figure. It is apparent that both PA-I and MIRA outperform the Perceptron. In addition, the performance of PA-I is comparable to that of MIRA with a slight advantage to the latter. However, while each online update of MIRA requires solving a complex optimization problem, each update of PA has a simple closed-form expression and is thus much faster and easier to implement.

11. Discussion

We described an online algorithmic framework for solving numerous prediction problems ranging from classification to sequence prediction. We derived several loss bounds for our algorithms (Thms. 2-5). The proofs of all of the bounds are based on a single lemma (Lemma 1). There are several possible extensions of the work presented in this paper. We already conducted further research on applications of the PA algorithmic framework for learning margin-based suffix

trees (Dekel et al., 2004b), pseudo-metrics (Shalev-Shwartz et al., 2004b), hierarchical classification (Dekel et al., 2004a), and segmentation of sequences (Shalev-Shwartz et al., 2004a). While the focus of this paper is on online settings, online algorithms can also serve as building blocks in the construction of well performing batch algorithms. Online to batch conversions of the proposed algorithms are yet another important future research direction. The update taken by our algorithms is aggressive in the sense that even a small loss forces an update of the hypothesis. When using kernels, this property often results in the use of many examples for representing the learned predictor. Thus, the memory requirements imposed when using kernels can be quite demanding. We are currently pursuing extensions of the PA framework that operate in the realm of bounded memory constraints.

Acknowledgments

This research was funded by the Israeli Science Foundation under grant number 522/04. Most of this work was carried out at the Hebrew University of Jerusalem.

Appendix A. Derivation of the PA-I and PA-II Updates

As in Sec. 3, whenever $\ell_t = 0$ no update occurs and τ_t equals zero. If $\ell_t > 0$ we derive these updates by defining the Lagrangian of the respective optimization problem and satisfying the KKT conditions. The Lagrangian of the PA-I optimization problem is,

$$\begin{aligned} \mathcal{L}(\mathbf{w},\xi,\tau,\lambda) &= \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi + \tau(1-\xi-y_t(\mathbf{w}\cdot\mathbf{x}_t)) - \lambda\xi \\ &= \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + \xi(C-\tau-\lambda) + \tau(1-y_t(\mathbf{w}\cdot\mathbf{x}_t)), \end{aligned} \tag{60}$$

where $\tau \geq 0$ and $\lambda \geq 0$ are Lagrange multipliers. We now find the minimum of the Lagrangian with respect to the (unconstrained) primal variables $\mathbf{w}$ and ξ. As in the previously discussed PA update, differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero gives Eq. (5) and we can write $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$. Next, note that the minimum of the term $\xi(C-\tau-\lambda)$ with respect to ξ is zero whenever $C-\tau-\lambda = 0$. If however $C-\tau-\lambda \neq 0$ then $\xi(C-\tau-\lambda)$ can be made to approach $-\infty$. Since we need to maximize the dual we can rule out the latter case and pose the following constraint on the dual variables,

$$C-\tau-\lambda = 0. \tag{61}$$

The KKT conditions confine λ to be non-negative so we conclude that $\tau \leq C$. We now discuss two possible cases: if $\ell_t/\|\mathbf{x}_t\|^2 \leq C$ then we can plugging Eq. (61) back into Eq. (60) and we return to the Lagrangian of the original PA algorithm (see Eq. (4)). From this point and on, we can repeat the same derivation as in the original PA update and get $\tau_t = \ell_t/\|\mathbf{x}_t\|^2$. The other case is when $\ell_t/\|\mathbf{x}_t\|^2 > C$. This condition can be rewritten as

$$C\|\mathbf{x}_t\|^2 < 1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t). \tag{62}$$

We also know that the constraint in Eq. (6) must hold at the optimum, so $1-y_t(\mathbf{w}\cdot\mathbf{x}_t) \leq \xi$. Using the explicit form of $\mathbf{w}$ given in Eq. (5), we can rewrite this constraint as $1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)-\tau\|\mathbf{x}_t\|^2 \leq \xi$.

Combining this inequality with the inequality in Eq. (62) gives,

$$C\|\mathbf{x}_t\|^2 - \tau\|\mathbf{x}_t\|^2 \;<\; \xi.$$

We now use our earlier conclusion that $\tau \leq C$ to obtain $0 < \xi$. Turning to the KKT complementarity condition, we know that $\xi\lambda = 0$ at the optimum. Having concluded that ξ is strictly positive, we get that λ must equal zero. Plugging $\lambda = 0$ into Eq. (61) gives $\tau = C$. Summing up, we used the KKT conditions to show that in the case where $\ell_t/\|\mathbf{x}_t\|^2 > C$, it is optimal to select $\tau = C$. Folding all of the possible cases into a single equation, we define τ_t to be,

$$\tau_t = \min\left\{ C \;,\; \ell_t/\|\mathbf{x}_t\|^2 \right\}. \tag{63}$$

The update of PA-I is like the update of PA clipped at C.

Turning to the update of PA-II , we again recall that $\ell_t = 0$ leads to $\tau_t = 0$, and deal with those rounds where $\ell_t > 0$. The Lagrangian of the optimization problem in Eq. (7) equals,

$$\mathcal{L}(\mathbf{w},\xi,\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \;+\; C\xi^2 \;+\; \tau\big(1-\xi-y_t(\mathbf{w}\cdot\mathbf{x}_t)\big), \tag{64}$$

where $\tau \geq 0$ is a Lagrange multiplier. Again, differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero gives Eq. (5) and we can write $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$. Differentiating the Lagrangian with respect to ξ and setting that partial derivative to zero results in,

$$0 \;=\; \frac{\partial\mathcal{L}(\mathbf{w},\xi,\tau)}{\partial\xi} \;=\; 2C\xi - \tau \qquad\Longrightarrow\qquad \xi = \frac{\tau}{2C}.$$

Expressing ξ as above and replacing $\mathbf{w}$ in Eq. (60) with $\mathbf{w}_t + \tau y_t \mathbf{x}_t$, we rewrite the Lagrangian as,

$$\mathcal{L}(\tau) \;=\; -\frac{\tau^2}{2}\left(\|\mathbf{x}_t\|^2 \;+\; \frac{1}{2C}\right) \;+\; \tau\big(1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)\big).$$

Setting the derivative of the above to zero gives,

$$0 \;=\; \frac{\partial\mathcal{L}(\tau)}{\partial\tau} \;=\; -\tau\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) + \big(1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)\big) \qquad\Longrightarrow\qquad \tau = \frac{1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{\|\mathbf{x}_t\|^2 \;+\; \frac{1}{2C}}.$$

As in PA and PA-I , we can give a definition of τ_t which holds in all cases,

$$\tau_t \;=\; \frac{\ell_t}{\|\mathbf{x}\|^2 + \frac{1}{2C}}. \tag{65}$$

References

S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3): 382–392, 1954.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

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

M. Collins. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. A new family of online algorithms for category ranking. *Jornal of Machine Learning Research*, 3:1025–1058, 2003a.

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

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004a.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Advances in Neural Information Processing Systems 17*, 2004b.

A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

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

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

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3), 2002.

D.P Helmbold, J. Kivinen, and M. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.

M. Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 444–460, 2001.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2002.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

N. Klasner and H.U. Simon. From noise-free to noise-tolerant and from on-line to batch learning. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 250–264, 1995.

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

N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, U. C. Santa Cruz, March 1989.

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

J. Rocchio. Relevance feedback information retrieval. In Gerard Salton, editor, *The Smart retrieval system—experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5), 1988.

R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1998. To appear, *Machine Learning*.

R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 32(2/3), 2000.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

S. Shalev-Shwartz, J. Keshet, and Y. Singer. Learning to align polyphonic music. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004a. http://www.cs.huji.ac.il/∼shais/.

S. Shalev-Shwartz, Y. Singer, and A. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004b.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.