Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence

Distribution-Aware Online Classifiers

Tam T. Nguyen, Kuiyu Chang, and Siu Cheung Hui
School of Computer Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798

Abstract

We propose a family of Passive-Aggressive Mahalanobis (PAM) algorithms, which are incremental (online) binary classifiers that consider the distribution of data. PAM is in fact a generalization of the Passive-Aggressive (PA) algorithms to handle data distributions that can be represented by a covariance matrix. The update equations for PAM are derived and theoretical error loss bounds computed. We benchmarked PAM against the original PA-I, PA-II, and Confidence Weighted (CW) learning. Although PAM somewhat resembles CW in its update equations, PA minimizes differences in the weights while CW minimizes differences in weight distributions. Results on 8 classification datasets, which include a real-life micro-blog sentiment classification task, show that PAM consistently outperformed its competitors, most notably CW. This shows that a simple approach like PAM is more practical in real-life classification tasks, compared to more sophisticated approaches like CW.

1 Introduction

In tasks that require huge number of real-time classifiers, online learning is the only viable option. For example, real-time classification of status/micro-blog updates of millions of social network users requires a personalized online classifier to be maintained for every user [Li *et al.*, 2010]. An online learning algorithm updates its decision boundary incrementally after processing each sample. Given a sample, it will first classify it, so called making a prediction. The quantitative difference in the prediction and true label is computed as the loss, which is then used to adjust the classifier weights. The goal is to maximize the correctness of future predictions.

The classical Perceptron [Block, 1962; Novikoff, 1962], Second-order Perceptron (SOP) [Cesa-Bianchi *et al.*, 2005], suite of Passive-Aggressive (PA) algorithms [Crammer *et al.*, 2006] and its second order variants Confidence-Weighted (CW) learning [Dredze *et al.*, 2008] and Adaptive Regularization Of Weight vectors (AROW) [Crammer *et al.*, 2009b], all belong to the same family of online algorithms, which perform well for a variety of real-time applications. However, except for the Second-order Perceptron, these online learning algorithms do not explicitly account for the distribution of the data [Cesa-Bianchi *et al.*, 2005]. PA algorithms that use the squared Euclidean distance work well for data with spherical distributions. However, for hyper-ellipsoidal data distributions, performance can be poor. In fact, PA assumes data to be spherically distribution. To overcome this deficiency, we propose using the Mahalanobis distance measure in place of the Euclidean distance in PA, and call the new approach the Passive Aggressive Mahalanobis (PAM) algorithm. PAM update equations bear a close resemblance to that of the CW, which took a different route by assuming that the weights are normally distributed with a mean vector and covariance matrix. However, PAM is different from CW in its update criterion. CW minimizes the differences between the new and old weight distribution whereas PAM simply maintains the original PA goal by minimizing the weight vector differences adjusted by the weight covariance.

2 Related Work

Online linear classification algorithms have been studied for close to 50 years, starting with the Perceptron [Block, 1962; Novikoff, 1962]. Recently, there has been a renewed interest in Perceptron-like algorithms such as the Second-order Perceptron [Cesa-Bianchi *et al.*, 2005] and the Passive-Aggressive (PA) algorithm [Crammer *et al.*, 2006], with the latter incorporating the margin maximizing criterion of modern machine learning algorithms. Algorithms that improved upon the PA algorithm include the Confidence-Weighted (CW) linear classification [Dredze *et al.*, 2008] and its latest version, the CW algorithm for multi-class classification [Crammer *et al.*, 2009a]. CW assumes that the weight at each time step is Gaussian distributed with a mean vector and covariance matrix. As such, the weight vector is updated by minimizing the Kullback-Leibler divergence between the new and old weight distributions.

There is also a related class of Bandit algorithms [Kakade *et al.*, 2008], whose learning process is similar to the Perceptron algorithm. However, in the prediction phase, the Bandit algorithm does not know the true label of the instance. The Bandit algorithm is actually more realistic with respect to real-world online tasks like micro-blog classification, since in practice the real class label is not known after each prediction unless the user constantly validates every prediction.

Other online learning algorithms use the Newton weight update method, including the LaRank and the OLaRank algorithms [Bordes *et al.*, 2007; 2008]. Some are inspired by support vector machines [Cortes and Vapnik, 1995] and the Huller algorithm [Bordes and Bottou, 2005].

3 Passive-Aggressive Mahalanobis

The online binary classification framework in this section follows the PA algorithm formulation [Crammer *et al.*, 2006].

3.1 Online Learning

Online learning operates on a sequence of data with time stamps. At time step t, the algorithm process an example $\mathbf{x}_t \in \mathbb{R}^n$ by first predicting its label $\hat{y}_t \in \{-1, +1\}$. After prediction, it computes the loss $\ell(y_t, \hat{y}_t)$ which is the difference between its prediction and the revealed true label $y_t \in \{-1, +1\}$. The loss is then used to update the weight with respect to some criterion. The goal is to achieve a margin of at least 1. So on a certain round if the margin is less than 1, the algorithm suffers a loss. The loss can be modeled using the hinge-loss function, which equals to zero when the margin exceeds 1, as shown below.

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

Passive Aggressive Algorithms

The overall objective of online learning is to minimize the cumulative loss over the entire sequence of examples. Crammer [Crammer *et al.*, 2006] formulated it as an optimization problem and derived three versions of the PA algorithms as follows. First, the optimization problem is formulated as follows.

$$\begin{aligned} \mathbf{w}_{t+1} = \; & \underset{\mathbf{w} \in \mathcal{R}^n}{\operatorname{argmin}} \; \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \\ & \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \quad (2)$$

Crammer updates the weight vector $\mathbf{w}_{t+1}$ at each round as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \quad \text{and} \quad \tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2} \quad (3)$$

Second, to allow for incorrect predictions, a slack variable ξ was introduced into the optimization problem (2) with two penalties; linear and quadratic. The weight update equation to the soft-margin problem has the same form as that of (3), but with the weight coefficient τ_t defined as follows.

$$\tau_t = \min\left\{C, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2}\right\} \quad \text{and} \quad \tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}}$$

The three flavors were named PA, PA-I, and PA-II respectively.

Confidence Weighted Learning

Using a probabilistic approach, the confidence-weighted (CW) online learning algorithm learns a Gaussian distribution of weights with mean vector μ and covariance matrix Σ. The weight distribution is updated by minimizing the Kullback-Leibler divergence between the new weight distribution and the old one while ensuring that the probability of correct classification is greater than a threshold as follows.

$$\begin{aligned} (\mu_{t+1}, \Sigma_{t+1}) = \; & \underset{\mu, \Sigma}{\operatorname{argmin}} \, D_{KL}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu_t, \Sigma_t)) \\ \text{s.t.} \; & Pr_{\mathbf{w} \sim \mathcal{N}(\mu, \Sigma)}[y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 0] \geq \eta \end{aligned} \quad (4)$$

where Pr denote the point probability. This optimization problem has a closed form solution:

$$\mu_{t+1} = \mu_t + \alpha_t y_t \Sigma_t \mathbf{x}_t \quad (5)$$

$$\Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t \mathbf{x}_t^T \mathbf{x}_t \Sigma_t \quad (6)$$

where $\alpha_t = \frac{-(1+2\phi M_t) + \sqrt{(1+2\phi M_t)^2 - 8\phi(M_t - \phi V_t)}}{4\phi V_t}$, $\beta_t = \frac{2\alpha_t \phi}{1+2\alpha_t \phi V_t}$, $V_t = \mathbf{x}_t^T \Sigma_t \mathbf{x}_t$, $M_t = y_t(\mu_t \cdot \mathbf{x}_t)$, and ϕ is a confidence parameter depending on η.

3.2 Hard Margin PAM

PAM is similar to PA, except for its use of the Mahalanobis distance measure in place of the Euclidean distance measure. The new optimization problem is defined as follows.

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathcal{R}^n}{\operatorname{argmin}} \; & \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_t) \\ & \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned}$$

where Σ_{t-1} is the covariance matrix of the weight vector distribution at round $t - 1$. Solving the above problem, we have,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \Sigma_{t-1} \mathbf{x}_t \quad \text{and} \quad \tau_t = \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \quad (7)$$

Finally, we obtain the hard margin Mahalanobis Passive-Aggressive (PAM) algorithm as shown in Algorithm 1.

3.3 Soft Margin PAM

Extending PAM to deal with misclassified samples, we introduce the slack variable ξ as follows.

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathcal{R}^n}{\operatorname{argmin}} \; & \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_t) + \\ C\xi \quad & \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \quad \text{and} \quad \xi \geq 0 \end{aligned}$$

where C is the positive aggressiveness constant, which controls the aggressiveness of each update step. The bigger the C, the larger the update. We thus have the following solution.

$$\tau_t = \min\left\{C, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t}\right\}$$

By defining a different objective function that changes quadratically with the slack variable ξ, we have the following optimization problem.

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathcal{R}^n}{\operatorname{argmin}} \; & \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_t) + C\xi^2 \\ & \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \end{aligned} \quad (8)$$

Solving the above problem, we have the following result.

$$\tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t + \frac{1}{2C}}$$

Algorithm 1 Passive-Aggressive Mahalanobis (PAM)

Input:
C = positive aggressiveness parameter
Output:
None
Process:

```
Initialize Σ_0 ← I; w_1 ← 0;
for t = 1, 2, ... do
    Receive instance x_t ∈ R^n
    Predict ŷ_t = sign(w_t · x_t)
    Receive correct label y_t ∈ {−1, +1}
    Suffer loss ℓ_t ← max{0, 1 − y_t(w_t · x_t)}
    if ℓ_t > 0 then
        Set τ_t ← ℓ_t / (x_t^T Σ_{t−1} x_t)                    (PAM)
            τ_t ← min{C, ℓ_t / (x_t^T Σ_{t−1} x_t)}            (PAM-I)
            τ_t ← ℓ_t / (x_t^T Σ_{t−1} x_t + 1/(2C))           (PAM-II)
        Update w_t ← w_{t−1} + τ_t y_t Σ_{t−1} x_t
               Σ_t ← Σ_{t−1} − (Σ_{t−1} x_t x_t^T Σ_{t−1}) / (1 + x_t^T Σ_{t−1} x_t)
    end if
end for
```

The two updates are named PAM-I and PAM-II, respectively. Both share the same general form $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \Sigma_{t-1}\mathbf{x}_t$, with a different update step as follows.

$$\tau_t = \min\left\{C, \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1}\mathbf{x}_t}\right\} \qquad \text{(PAM-I)}$$

and

$$\tau_t = \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1}\mathbf{x}_t + \frac{1}{2C}} \qquad \text{(PAM-II)}$$

3.4 Covariance Matrix Estimation

We describe a way to approximate the covariance matrix Σ over the weight vector $\mathbf{w}$. Consider the evolution of the objective function starting at $t = 0$ and $\Sigma_0 = \mathrm{I}$. At round T, we have the loss function $1 - y_T\mathbf{w}_T \cdot \mathbf{x}_T$ where $\mathbf{w}_T$ is the weight vector at round T. Denoting $\mathbf{y}$ as a vector $[y_1, y_2, \ldots, y_T]$ and $\mathbf{X} = [\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_T]$ as a matrix of column input vectors. We can write the PAM weight update as,

$$1 - \mathbf{y}\mathbf{X}\mathbf{w}_T = 0 \qquad \text{or} \qquad \mathbf{y} - \mathbf{X}\mathbf{w}_T = \mathbf{0}$$

Multiplying the above equality by $\mathbf{X}^T$, we have,

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}_T) = \mathbf{0} \qquad \text{or} \qquad \mathbf{X}^T\mathbf{X}\mathbf{w}_T = \mathbf{X}^T\mathbf{y}$$

Multiplying the above equality by $(\mathbf{X}^T\mathbf{X})^{-1}$, we have,

$$\mathbf{w}_T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Assuming an i.i.d. noise model, we can write $\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{e}$, where $\mathbf{e}$ is an error vector. Substituting $\mathbf{y}$ into the above equality, we have,

$$\mathbf{w}_T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\mathbf{w} + \mathbf{e}) = \mathbf{w} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{e}$$

The covariance matrix of $\mathbf{w}_T$ has the form,

$$\Sigma_T = \mathrm{Cov}(\mathbf{w}_T) = \mathrm{E}[(\mathbf{w}_T - \mathbf{w})(\mathbf{w}_T - \mathbf{w})^T]$$

or

$$\begin{aligned}\Sigma_T &= \mathrm{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{e}\mathbf{e}^T\mathbf{X}(\mathbf{X}\mathbf{X}^T)^{-1}] \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{E}[\mathbf{e}\mathbf{e}^T]\mathbf{X}(\mathbf{X}\mathbf{X}^T)^{-1}\end{aligned}$$

Since, the PAM weights are updated to achieve a margin of at least 1, we can assume that $\mathrm{E}[\mathbf{e}\mathbf{e}^T] = \sigma^2\mathbf{I}$. The covariance matrix can be approximated as follows.

$$\Sigma_T = \sigma^2(\mathbf{X}\mathbf{X}^T)^{-1} \simeq (\mathbf{X}\mathbf{X}^T)^{-1}$$

Following the work of [Cesa-Bianchi *et al.*, 2005], we can write,

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \mathbf{x}_t\mathbf{x}_t^T. \tag{9}$$

Applying the Sherman-Morrison formula to (9), we have,

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1}\mathbf{x}_t\mathbf{x}_t^T\Sigma_{t-1}}{1 + \mathbf{x}_t^T\Sigma_{t-1}\mathbf{x}_t} \tag{10}$$

From Equation (9), we can conclude that $\Sigma_t^{-1} \succeq \Sigma_{t-1}^{-1}$ and $\Sigma_t \preceq \Sigma_{t-1}$. Like the PA family, all three PAM algorithms share the same weight update equation, differing only in the update rate τ_t, as shown in Algorithm 1. In fact, the PAM-II weight update resembles that of Adaptive Weight Regularization [Crammer *et al.*, 2009b] (AROW). The difference between PAM-II and AROW is that PAM-II does not explicitly regulate the update of the covariance matrix Σ_t. PAM was primarily motivated by adding a data noise model to PA, while AROW and CW started out by assuming a distribution of weights. The end results are very similar, differing only in the update rates.

Ma *et al.* [Ma *et al.*, 2010] examined in depth several strategies to estimate the covariance matrix efficiently, along with their practical implications specifically for CW, but which can be used for any second-order learning algorithms, including PAM. In this paper, we will not focus on the practical issue of computing the covariance, but instead measure the classification performances of both CW and PAM assuming that a full covariance matrix is available and feasible.

3.5 PAM Error Analysis

In this section we provide several theoretical results for PAM, omitting the proofs due to lack of space.

Theorem 1 (Relative Loss Bound) *Given a sequence of M examples $[(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)]$, any weight vector $\mathbf{u} \in \mathbb{R}^n$, and loss $\ell_t^* = 0$ for all t, the cumulative relative loss of PAM is upper bounded by*

$$\sum_t \ell_t^2 \le (\| \mathbf{u} \|^2 + \quad \mathbf{u}^T\mathbf{X}_U\mathbf{X}_U^T\mathbf{u}) \max_t \mathbf{x}_t^T\Sigma_t\mathbf{x}_t \tag{11}$$

where U is the set of indices for examples leading to the weight updates and $\mathbf{X}_U\mathbf{X}_U^T = \sum_{t\in U}\mathbf{x}_t\mathbf{x}_t^T$.

Theorem 2 (PAM-I Mistake Bound) *Given a sequence of examples $[(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)]$ and any weight vector $\mathbf{u} \in \mathbb{R}^n$, the number of mistakes made by PAM-I is upper bounded by*

$$\max\{\max_t \mathbf{x}_t^T\Sigma_t\mathbf{x}_t, \frac{1}{C}\}\Big(\| \mathbf{u} \|^2 + \quad \mathbf{u}^T\mathbf{X}_U\mathbf{X}_U^T\mathbf{u} + C\sum_{t=1}^M \ell_t^*\Big) \tag{12}$$

where C is a positive aggressiveness parameter.

Theorem 3 (PAM-II Loss Bound) *Given a sequence of examples* $[(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)]$ *and any weight vector* $\mathbf{u} \in \mathbb{R}^n$*, the cumulative relative loss of PAM-II is upper bounded by*

$$\sum_t \ell_t^2 \leq \Big(\max_t \mathbf{x}_t^T \Sigma_t \mathbf{x}_t + \frac{1}{2C} \Big) \Big(\parallel \mathbf{u} \parallel^2 + \big(\tfrac{2}{1+\frac{1}{2C}}\big) \mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u} + C \textstyle\sum_{t=1}^{M} (\ell_t^*)^2 \Big) \tag{13}$$

In [Crammer *et al.*, 2006], the squared upper loss bound of the PA algorithm was defined as $\|\mathbf{u}\|^2 (\max |\mathbf{x}|)^2$. This bound depends only on the norm of weight vector $\mathbf{u}$. It does not consider the input data distribution while the upper bound of the PAM algorithm depends on both the norm of $\mathbf{u}$ and $\mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u}$, the data spectral term. We know that the second term is finite and bounded by the maximal eigenvalues of the matrix $\mathbf{X}_U \mathbf{X}_U^T$. Another term in the loss bound of the PAM-I and the PAM-II algorithms is $\mathbf{x}_t^T \Sigma_t \mathbf{x}_t$, which is a trade-off factor between the hinge-loss term and the data spectral term. In CW learning, the matrix Σ_t is called the confidence, which decreases monotonically with observed data. We also have $\Sigma_0 = I$ therefore we always have $\mathbf{x}_t^T \Sigma_t \mathbf{x}_t \leq \mathbf{x}_t^T \mathbf{x}_t$, which causes the data spectral term to increase with the hinge-loss quantity. However, it is very difficult to compare the upper loss bounds of the two families because both depends on the input distribution.

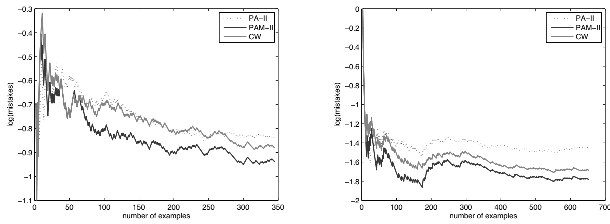
4 Performance Evaluation

A total of 8 datasets were used including two binary classification datasets (CRX and BUPA datasets from UCI [Asuncion and Newman, 2007]), two binary web datasets (WebKB and Twitter Sentiment), and two multi-class datasets (USPS and MNIST). For the multi-class datasets, one random class out of C classes was selected as positive, and a negative class of equal size was generated by sampling (the same number of samples as the positive class) from the remaining $C - 1$ classes. Where applicable, all experiments were repeated 10 times with different randomizations, and the average results shown/plotted. Results on the twitter dataset was single-run, since the data are deterministically ordered and binary.

4.1 Cumulative Error Rate

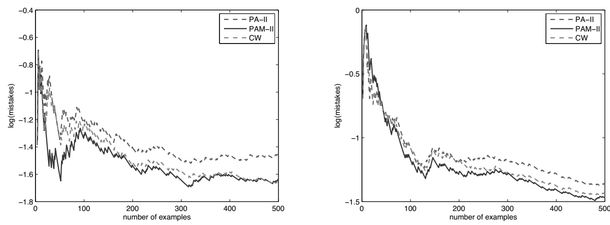
We use the standard cumulative error rate, which is the ratio of mistakes over the total number of examples. To ensure a fair comparison of our proposed algorithms with the original Passive-Aggressive algorithms, we grid-searched the optimal aggressiveness parameter C in all PA-based algorithms. To be fair, we compare our PAM-I and PAM-II with the original PA-I and PA-II, respectively. We excluded the PA results as it performed worse than the PA-I and PA-II [Crammer *et al.*, 2006]. For brevity, we only show the cumulative error rate comparisons between PA-II, PAM-II, and CW here.

The cumulative error rates on the two binary datasets are shown in Figure 1. For BUPA, all three algorithms started off with similar loss, but PAM-II starts to pull away from the pack after 60 examples, and consistently exhibit lower log-mistake rate thereafter. For CRX, PAM-II leads after iteration 30, with an overall lower mistake rate thereafter.



(a) The BUPA dataset (b) The CRX dataset

Figure 1: Cumulative error for BUPA and CRX.



(a) The USPS dataset (b) The MNIST dataset

Figure 2: Cumulative error rate for USPS and MNIST.

Figure 2 shows the cumulative error rates on the two letter recognition datasets. PAM-II performed better right from the start, but not significantly better overall because the negative class in this case is heterogenous (formed by a uniform equal-sized sample of the non-positive class);better results can be expected in a one-of classification.

The WebKB dataset contains 1051 web documents from two classes, each with two views. We tested all algorithms only on the textual view, with results shown in Figure 3. Again, PAM achieved consistently lower log-mistake rate, widening the gap with increasing number of examples.

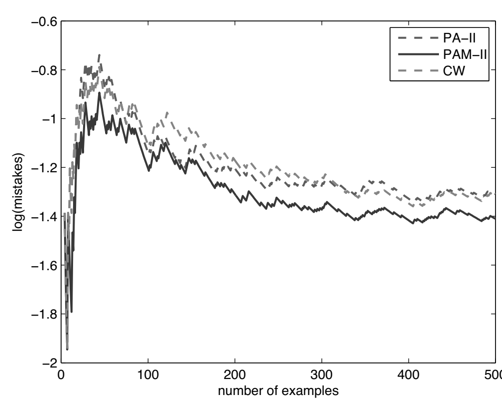


Figure 3: Cumulative error rate for WebKB.

4.2 Classification Accuracy

In practice, classification performance in terms of F-measure is typically more important than cumulative error rates. The positive (+) and negative (-) class F-measures for all 5 datasets are listed in Table 1 with the best results in bold. PAM-II consistently outperform other algorithms, including CW. Although, CW performed better than the original PA-I and PA-II, it still falls a little behind PAM. For BUPA, PAM-II is more than 2% better than CW. PAM-II achieved the largest winning margin against on CRX+, where it is more than 11% better than PA-II, and 2% better than CW. Overall, PAM-II beats CW only by a marginal 1-2%. Again, the positive class improvements over PA-II are significantly better because it is much more homogeneous compared to the artificially consolidated negative class.

Table 1: F1 (%) for positive (+) and negative (-) classes.

Dataset	PA-II	PAM-II	CW
BUPA+	61.21 ± 2.78	$\mathbf{64.28} \pm 1.13$	61.79 ± 2.43
BUPA−	57.85 ± 1.31	$\mathbf{60.20} \pm 2.55$	57.61 ± 2.00
CRX+	68.54 ± 0.40	$\mathbf{80.37} \pm 0.51$	78.28 ± 0.93
CRX−	77.83 ± 0.62	$\mathbf{84.13} \pm 0.62$	82.54 ± 0.60
USPS+	94.08 ± 2.77	$\mathbf{95.26} \pm 2.56$	94.81 ± 2.79
USPS−	93.84 ± 2.78	$\mathbf{95.05} \pm 2.56$	94.57 ± 2.79
MNIST+	57.85 ± 3.43	$\mathbf{59.51} \pm 1.72$	58.64 ± 0.90
MNIST−	58.97 ± 2.47	$\mathbf{60.47} \pm 1.48$	58.52 ± 1.12
WEBKB+	77.02 ± 1.14	$\mathbf{78.90} \pm 1.63$	76.49 ± 1.40
WEBKB−	90.98 ± 0.66	$\mathbf{92.94} \pm 0.84$	90.78 ± 0.74

4.3 Online Microblog Data

To illustrate the utility of online algorithms, we apply them to learn emotions from real-life micro-blogs. The *Twitter*[1] sentiment dataset [Li *et al.*, 2010] is a collection of micro-blogs (tweets) written by 6 users. Each tweet is manually labeled as emotional (positive) or non-emotional (negative). An online model was applied to each user's tweets in chronological sequence. Each model was initialized to some random weights; after it classifies an incoming tweet, the tweet's true label is revealed to update the model weights, and the online classification/learning continues until the last tweet is predicted.

From the individual loss plots in Figure 4, PAM-II again consistently outperformed the other algorithms. However, the advantage of PAM-II depends very much on the dataset. For instance, for user DenyceLawton (c), PAM-II did significantly better than the others but for another user CarlaMedina (b), PAM-II performed only marginally better. On closer examination, we found that CarlaMedina writes equally frequently in Spanish and English. Since our human labeler is not Spanish literate, a large portion of the tweets have been labeled incorrectly. For example, Spanish emotions were not properly labeled, labeled emotional tweets contain a mix of Spanish and English with English terms acting as the decisive factor. As a result, the labeling for user CarlaMedina is very noisy. Another consequence of not knowing the language is the highly imbalanced class distribution, with user CarlaMedina having the smallest raw count of 250 positive (emotional) labeled samples. Specifically, users AudreyWalker, CarlaMedina, DenyceLawton, IheartBrooke, RealMichelleW, and SabrinaBryan have 15.5%, 19.4%, 41.6%, 18.9%, 17.0%, and 17.4% positive tweets respectively. For PAM-II, this means that the model would have very little chance to make a wrong prediction and significantly adjusting its weight; an occasional positive sample would cause the margin to be reduced. For such a case, PAM does not benefit much from considering the sample distribution, since the covariance matrix would account for a far smaller number of samples.

For CarlaMedina, PAM II started to decisively outperform CW only after around 250 samples, by when it should have seen approximately 50 (19.4% of 250) positive samples, assuming a uniform class distribution. For other marginal users like AudreyWalker (483 positive, PAM wins after 40 samples), RealmichelleW (495 positive, PAM wins after 100), and SabrinaBryan (551 positive, PAM wins after 200), who all have around 500 total raw positive tweets, their cumulative PAM loss rates were all able to pull away from the competitor earlier than CarlaMedina (342 positive, PAM wins after 250), simply because they have a larger number of positive tweets.

[1] http://twitter.com

5 Conclusion

We proposed PAM, a generalization of the Passive-Aggressive algorithms [Crammer *et al.*, 2006] that takes into account the data spectral properties. PAM was evaluated on several datasets and found to consistently outperform other online algorithms, including its cousin Confidence Weighted (CW) learning. Results on online classification tasks have shown an average of 4% to 12% improvements in F1-measure. We have also validated the practicality and superiority of PAM on a real-world twitter emotion classification dataset.

Compared to PA, PAM runs slower because it needs to compute the covariance matrix, which scales quadratically with the number of features. To solve this problem, we can deploy the approximate version of the PAM algorithm by calculating the diagonal matrix in the same way as the CW algorithm [Dredze *et al.*, 2008; Ma *et al.*, 2010].

We are currently extending the PAM family of algorithms for multi-class and structural data problems. We are also refining the analytical error loss bounds of PAM, so as to stipulate the data conditions for which PAM will be decisively superior and vice-versa. For future work, we would also want to evaluate extensively the practical performances of AROW versus PAM-II, given their similarities in the weight update equations. In particular, we want to find out if the adaptive update of the covariance matrix in AROW is superior to PAM-II's static covariance update approach.

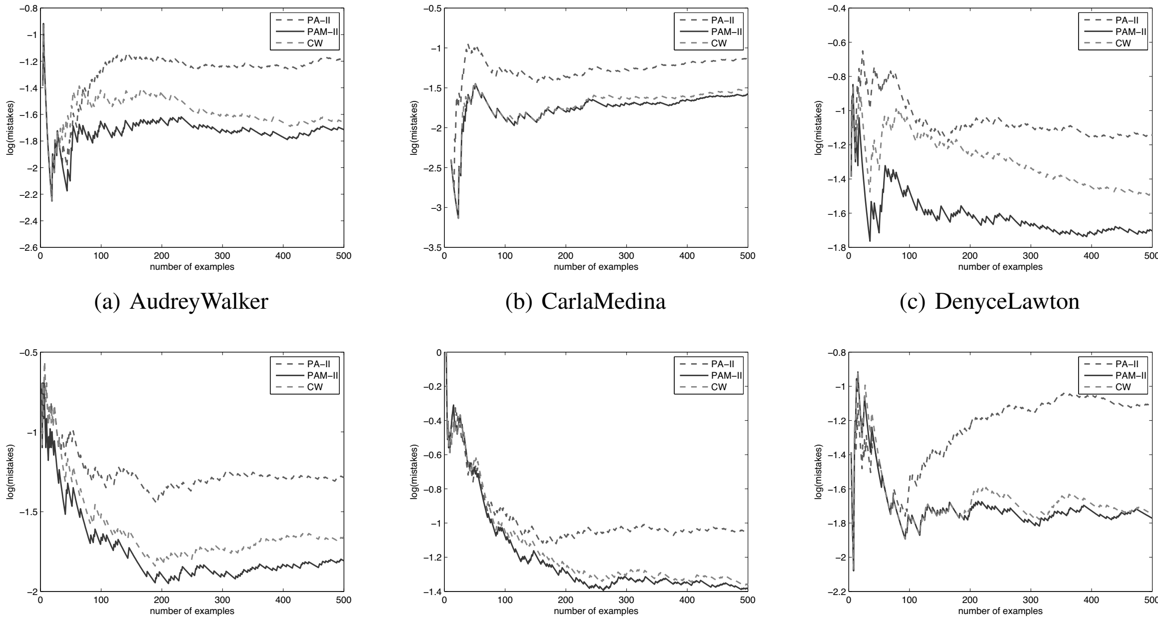
Acknowledgement

This research was supported in part by Singapore Ministry of Education's Academic Research Fund Tier 1 grant RG 30/09.

References

[Asuncion and Newman, 2007] A. Asuncion and D. J. Newman. Uci machine learning repository, 2007.

[Block, 1962] H. Block. The perceptron: A model for brain functioning. *Rev. Modern Phys.*, 34:123–135, 1962.



(a) AudreyWalker (b) CarlaMedina (c) DenyceLawton

(d) IheartBrooke (e) RealMichelleW (f) SabrinaBryan

Figure 4: Cumulative error rate for the Twitter sentiment dataset.

[Bordes and Bottou, 2005] A. Bordes and L. Bottou. The huller: a simple and efficient online svm. *Machine Learning: ECML 2005*, pages 505–512, 2005.

[Bordes *et al.*, 2007] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with larank. In *Proc. ICML*, pages 89–96, New York, NY, USA, 2007. ACM.

[Bordes *et al.*, 2008] A. Bordes, N. Usunier, and L. Bottou. Sequence labelling svms trained in one pass. In *Proc. ECML*, 2008.

[Cesa-Bianchi *et al.*, 2005] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *Siam J. of Comm.*, 34, 2005.

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

[Crammer *et al.*, 2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, pages 551–585, 2006.

[Crammer *et al.*, 2009a] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 496–504, Singapore, August 2009. Association for Computational Linguistics.

[Crammer *et al.*, 2009b] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems*, 2009.

[Dredze *et al.*, 2008] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proc. ICML*, pages 264–271, New York, NY, USA, 2008. ACM.

[Kakade *et al.*, 2008] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 440–447, New York, NY, USA, 2008. ACM.

[Li *et al.*, 2010] G. Li, S. C. H. Hoi, K. Chang, and R. Jain. Micro-blogging sentiment detection by collaborative online learning. In *Proc. IEEE International Conference on Data Mining*, pages 893–898, Sydney, Australia, 2010.

[Ma *et al.*, 2010] J. Ma, A. Kulesza, M. Dredze, K. Crammer, L. K. Saul, and F. Pereira. Exploiting feature covariance in high-dimensional online learning. In *Proc. Int. Conf. on Artificial Intelligence and Statistics*, pages 393–500, Sardinia, Italy, 2010.

[Novikoff, 1962] A. Novikoff. On convergence proofs of perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 7, pages 615–622, 1962.