Mach Learn (2013) 91:155–187
DOI 10.1007/s10994-013-5327-x

Adaptive regularization of weight vectors

Koby Crammer · Alex Kulesza · Mark Dredze

Received: 11 August 2011 / Accepted: 7 January 2013 / Published online: 22 March 2013

Abstract We present AROW, an online learning algorithm for binary and multiclass problems that combines large margin training, confidence weighting, and the capacity to handle non-separable data. AROW performs adaptive regularization of the prediction function upon seeing each new instance, allowing it to perform especially well in the presence of label noise. We derive mistake bounds for the binary and multiclass settings that are similar in form to the second order perceptron bound. Our bounds do not assume separability. We also relate our algorithm to recent confidence-weighted online learning techniques. Empirical evaluations show that AROW achieves state-of-the-art performance on a wide range of binary and multiclass tasks, as well as robustness in the face of non-separable data.

Keywords Online learning · Supervised learning · Text classification · Adaptive regularization

1 Introduction

Online learning algorithms are fast and simple, make few statistical assumptions, and perform well in a wide variety of settings. The Perceptron algorithm is perhaps the oldest online machine learning algorithm, tracing its origins back to the 1950s. The Perceptron, which

Editor: Shai Shalev-Shwartz.

K. Crammer (✉)
Department of Electrical Engineering, The Technion, Haifa, 32000 Israel
e-mail: koby@ee.technion.ac.il

A. Kulesza
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA
e-mail: kulesza@umich.edu

M. Dredze
Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD 21211, USA
e-mail: mdredze@cs.jhu.edu

uses a gradual additive update based on stochastic gradient ascent, has been supported by numerous mistake bound analyses (Littlestone 1988). Despite its age, it is still widely used for modern problems, including complex structured learning tasks (Collins 2002).

While popular, the Perceptron suffers from several well-known problems. First, while guaranteed to converge on linearly separable data sets, convergence can be slow, often taking many iterations over the data set. This slow convergence is a consequence of non-aggressive updates, which take a fixed step towards a better solution but offer no guarantees about the improvement on the current training example. In many difficult learning settings, even after an update, the Perceptron will still classify an example incorrectly. Similarly, the Perceptron is a strictly mistake driven learning algorithm, meaning that correctly classified training examples with ambiguous scores (the correct label is only slightly preferred) are treated as correct. A common solution to this problem is the Perceptron with margin, in which the size of the learning update is controlled by a learning rate hyperparameter whose increase leads to a more aggressive update (Freund and Schapire 1999).

A second common problem is Perceptron's erratic behavior in high noise settings. Since Perceptron treats every example equally, examples with label noise still receive a full update. In contrast, batch algorithms, such as the Support Vector Machines algorithm, can sacrifice accuracy on noisy examples in favor of improving performance on the majority of training examples through the use of slack variables. The result is that the Perceptron's weights oscillate, sometimes dramatically, as it constantly struggles with noisy labels. A number of strategies exist for addressing this problem, including those proposed by Krogh and Hertz (1992), Krogh (1992), Khardon and Wachman (2007), and the voted or averaged Perceptron proposed by Freund and Schapire (1999). This problem can be particularly dramatic in highly non-separable structured learning tasks, which, as a result, almost universally rely on the averaged Perceptron.

A variety of new algorithms have been proposed to address the shortcomings of the Perceptron. One is the Passive Aggressive (PA) algorithm (Crammer et al. 2003, 2006), also known as MIRA, which is based on the same additive update form as Perceptron. In PA, however, the strength of the update comes from a per example learning rate that is based on the solution to a convex optimization problem; for each example PA enforces a prediction margin based on the hinge loss, updating the algorithm's parameters accordingly. The result is that, after the update, the example is guaranteed to be classified correctly. The algorithm's name comes from this behavior: it aggressively updates each example to enforce a margin and passively ignores examples which are already correctly predicted with a margin. The result is significantly faster convergence. As a result, PA has been widely used in vision (Frome et al. 2007; Jie et al. 2010; Chechik et al. 2010), natural language processing (McDonald et al. 2005; Chiang et al. 2008), and bioinformatics (Bernal et al. 2007). Unfortunately, the aggressive nature of the updates has a significant negative impact on learning in the noisy setting, where incorrectly labeled examples will force large updates to the parameters to achieve the margin requirement. The standard solution relies on slack variables, which effectively clip the update to prevent dramatic parameter change based on a single example.

Recently, research on online learning has returned to this issue of convergence, seeking even more aggressive learning algorithms. One effective source for this aggressiveness has been parameter confidence, which has been shown to effectively guide online learning. Parameter confidence is encoded using an additional set of variables that measure the algorithm's confidence in its current parameter estimates. These algorithms make larger updates to less confident parameters, and then increase the confidence of a parameter each time it is updated. It is also possible to model second order feature interactions (Cesa-Bianchi et al. 2005; Ma et al. 2010). Tracking parameter confidence in this manner generally increases the rate of training convergence.

One implementation of this idea is Confidence Weighted (CW) learning, which is based on the passive-aggressive update. CW maintains parameter confidence through a Gaussian distribution over linear classifier hypotheses, which is then used to control the direction and scale of parameter updates (Dredze et al. 2008; Crammer et al. 2012). Updates not only fix learning mistakes but also increase confidence. In many settings, CW has been shown to be significantly more aggressive than PA, leading to much faster convergence rates. In addition to formal guarantees in the mistake bound model (Littlestone 1988), CW learning has achieved state-of-the-art performance, as well as faster learning rates, on a variety of tasks.

However, the strict update criterion used by CW learning is very aggressive and can overfit (Crammer et al. 2008). As a result, the most popular versions of CW rely on approximate solutions that effectively regularize the update and improve results. However, current analyses of CW learning still assume that the data are separable. It is not immediately clear how to relax this assumption for noisy data.

In this paper, we introduce an algorithm that addresses the need for both faster convergence and resistance to training noise. The core idea is to maintain the formalization of parameter confidence and second order feature interactions introduced by CW, but forego the aggressiveness of both CW and PA. Parameter confidence provides its own form of aggressiveness, so softening the margin requirement allows for robustness to training noise without sacrificing convergence speed. The resulting algorithm adaptively regularizes the prediction function upon seeing each new instance, making it robust to sudden changes in the classification function due to label noise. We call our algorithm AROW: Adaptive Regularization Of Weights. We emphasize that this approach is quite different from simply introducing slack variables, which merely modulate the strength of the update. Instead, we derive a completely new update rule that results in non-matching updates to the model parameters and confidence parameters. If we think of CW as enforcing statements of probability, then AROW can be seen as controlling model expectations. The result is an online learning algorithm that combines the attractive properties of large margin training, confidence weighting, and the capacity to handle non-separable data.

After deriving AROW, we provide a mistake bound analysis, similar in form to the second order Perceptron bound, that does not assume data separability. Our previous work (Crammer et al. 2009b) focused on the binary case of AROW. In this work, we derive an additional binary version, detail a fuller analysis, extend the algorithm to the multi-class setting, and provide new empirical results. We demonstrate that, for clean data, AROW achieves similar performance to CW, already state of the art for many tasks, and that AROW maintains convergence rates and significantly improves performance in the presence of label noise. We believe this second property will be of critical importance in many real world applications.

The paper proceeds as follows. In Sect. 2 we give a brief introduction to confidence weighted online methods. In Sect. 3 we introduce AROW and derive updates for binary and multi-class settings, and in Sect. 4 we provide a theoretical analysis of its behavior. Section 5 contains empirical evaluations of AROW using a variety of binary and multi-class applications. We conclude with a discussion of related work in Sect. 6 and summarize our contributions in Sect. 7.

2 Confidence weighted online learning of linear classifiers

Online algorithms operate in rounds. On round t the algorithm receives an instance $\boldsymbol{x}_t \in \mathbb{R}^d$ and applies its current prediction rule to make a prediction $\hat{y}_t \in \mathcal{Y}$. The true label $y_t \in \mathcal{Y}$

is then revealed, and the classifier suffers a label loss $\ell(y_t, \hat{y}_t)$. In binary classification, for example, we have $\mathcal{Y} = \{-1, +1\}$ and use the zero-one loss

$$\ell_{01}(y_t, \hat{y}_t) = \begin{cases} 0 & \hat{y}_t = y_t, \\ 1 & \hat{y}_t \neq y_t. \end{cases} \tag{1}$$

To complete the round, the algorithm adjusts its prediction rule using the labeled pair $(\boldsymbol{x}_t, y_t)$. In this work we will assume the prediction rule $h_{\boldsymbol{w}}(\boldsymbol{x}_t)$ is parameterized by a weight vector $\boldsymbol{w}$, which is updated on each round. In binary classification a common prediction rule is $h_{\boldsymbol{w}}(\boldsymbol{x}_t) = \text{sign}(\boldsymbol{x}_t \cdot \boldsymbol{w})$. Once the update is complete, learning proceeds to the next round.

Recently Dredze, Crammer, and Pereira (Dredze et al. 2008; Crammer et al. 2008) proposed confidence weighted (CW) learning, an algorithmic framework for online learning of classification problems. CW learning incorporates a notion of *confidence* in the current classifier by maintaining a Gaussian distribution over the weights; its mean is given by $\boldsymbol{\mu} \in \mathbb{R}^d$, and its covariance matrix is given by $\Sigma \in \mathbb{R}^{d\times d}$. Intuitively, μ_p encodes the learner's knowledge of the weight for feature p, and $\Sigma_{p,p}$ encodes its confidence in that weight. Small $\Sigma_{p,p}$ indicates that the learner is certain that the true weight is near μ_p. The off-diagonal covariance terms $\Sigma_{p,q}$ $(p \neq q)$ capture interactions between weights, though they are often unused in practice for reasons of efficiency (Ma et al. 2010).

In theory, a CW classifier labels an instance $\boldsymbol{x}$ by first drawing a parameter vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and then applying the prediction rule $h_{\boldsymbol{w}}$. In practice, however, it can be easier to simply use the average weight vector $\text{E}[\boldsymbol{w}] = \boldsymbol{\mu}$ to make predictions. This is similar to the approach taken by Bayes point machines (Herbrich et al. 2001), where a single weight vector is used to approximate a distribution. Furthermore, for binary classification, the prediction given by the mean weight vector turns out to be Bayes optimal.

CW classifiers are trained according to a passive-aggressive rule (Crammer et al. 2006) that adjusts the distribution at each round to ensure that the probability of a correct prediction is at least $\eta \in (0.5, 1]$. This yields the update constraint

$$\Pr\big[y_t = h_{\boldsymbol{w}}(\boldsymbol{x}_t)\big] \geq \eta. \tag{2}$$

Subject to this constraint, the algorithm makes the smallest possible change to the hypothesis weight distribution, as measured using the KL divergence. For binary classification, this implies the following optimization problem for each round t:

$$\begin{aligned} (\boldsymbol{\mu}_t, \Sigma_t) = \min_{\boldsymbol{\mu}, \Sigma} \quad & \text{D}_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \parallel \mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1})\big) \\ \text{s.t.} \quad & \Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu}, \Sigma)}\big[y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t) \geq 0\big] \geq \eta \end{aligned}$$

Confidence-weighted algorithms have been shown to perform well in practice (Crammer et al. 2008; Dredze et al. 2008), but they suffer from several problems. First, the update is quite aggressive, forcing the probability of predicting each example correctly to be at least $\eta > 1/2$ regardless of the cost to the objective. This may cause severe over-fitting when labels are noisy; indeed, current analyses of the CW algorithm assume that the data are linearly separable (Crammer et al. 2008). Second, CW methods are appropriate only for zero-one loss classification problems due to the form of the constraint in Eq. (2). It is not clear how to usefully generalize the CW approach to alternative loss functions or settings such as regression. In this work we address both shortcomings, developing a CW-like algorithm that copes effectively with label noise and generalizes the advantages of CW learning in an extensible way. We also present an analysis for the general non-separable case.

3 Adaptive regularization of weights

In developing our algorithms, we identify two important properties of the CW update rule that contribute to its strong performance but also make it sensitive to label noise. First, the mean parameters $\boldsymbol{\mu}$ are guaranteed to correctly classify the current training example with margin following each update. This is because the probability constraint $\Pr[y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t) \geq 0] \geq \eta$ can be written explicitly as

$$y_t(\boldsymbol{\mu} \cdot \boldsymbol{x}_t) \geq \phi\sqrt{\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t},$$

where $\phi > 0$ is a positive constant related to η. This aggressiveness yields rapid learning, but when an example is incorrectly labeled it can also force the learner to make a drastic and arbitrary change to its parameters. Second, confidence, as measured by the inverse eigenvalues of Σ, increases monotonically with each update, and the magnitude of the confidence update actually *increases* with the size of the update to the mean parameters. While it is intuitive that our confidence should grow as we see more data, this means that incorrectly labeled examples can cause wild parameter swings and artificially high confidence.

In order to maintain the positives but reduce the negatives of these two properties, we isolate and soften them. As in CW learning, we maintain a Gaussian distribution over weight vectors with mean $\boldsymbol{\mu}$ and covariance Σ; however, we recast the above characteristics of the CW constraint as regularizers, minimizing the following unconstrained objective on each round:

$$\mathcal{C}(\boldsymbol{\mu}, \Sigma) = \mathrm{D}_{\mathrm{KL}}\big(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \parallel \mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1})\big) + \lambda_1 \hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}_t, y_t) + \lambda_2 \boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t, \tag{3}$$

where $\lambda_1, \lambda_2 \geq 0$ are two tradeoff hyperparameters. For simplicity and compactness of notation, we will assume that $\lambda_1 = \lambda_2 = 1/(2r)$ for some $r > 0$. The function $\hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}_t, y_t)$ is the classifier loss suffered when using the weight vector $\boldsymbol{\mu}$ to predict the output for input $\boldsymbol{x}_t$ given that the true output is y_t. We will generally let $\hat{\ell}$ be a squared hinge upper bound on the label loss $\ell(h_{\boldsymbol{\mu}}(\boldsymbol{x}_t), y_t)$, as described in Sect. 3.1 and Sect. 3.3. However, other loss functions, as long as they are convex and differentiable in $\boldsymbol{\mu}$ (at all but a finite set of points), can be used to obtain algorithms for different settings. It can be shown, for example, that the well known recursive least squares (RLS) regression algorithm (Haykin 1996) is a special case of AROW with the squared loss (see also the paper of Vaits and Crammer 2011). In this sense AROW is significantly more general than CW.

The objective in Eq. (3) balances three desires. First, the parameters should not change radically on each round, since the current parameters contain information about previous examples (first term). Second, the new mean parameters should predict the current example with low loss (second term). Finally, as we see more examples, our confidence in the parameters should grow (third term). Note that this objective is not simply the dualization of the CW constraint, but a new formulation inspired by the previous discussion.

To solve for the parameters $\boldsymbol{\mu}, \Sigma$ that minimize Eq. (3), we begin by writing the KL term explicitly:

$$\begin{aligned}\mathcal{C}(\boldsymbol{\mu}, \Sigma) = {}& \frac{1}{2}\log\left(\frac{\det \Sigma_{t-1}}{\det \Sigma}\right) + \frac{1}{2}\mathrm{Tr}\big(\Sigma_{t-1}^{-1}\Sigma\big) + \frac{1}{2}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu})^\top \Sigma_{t-1}^{-1}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}) - \frac{d}{2} \\ & + \frac{1}{2r}\hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}_t, y_t) + \frac{1}{2r}\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t.\end{aligned} \tag{4}$$

Leaving aside the constant term $-d/2$, we can decompose Eq. (4) into two parts—$\mathcal{C}_1(\boldsymbol{\mu})$, depending only on $\boldsymbol{\mu}$, and $\mathcal{C}_2(\Sigma)$, depending only on Σ:

$$\mathcal{C}_1(\boldsymbol{\mu}) = \frac{1}{2}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu})^\top \Sigma_{t-1}^{-1}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}) + \frac{1}{2r}\hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}_t, y_t), \tag{5}$$

$$\mathcal{C}_2(\Sigma) = \frac{1}{2}\log\left(\frac{\det \Sigma_{t-1}}{\det \Sigma}\right) + \frac{1}{2}\operatorname{Tr}\left(\Sigma_{t-1}^{-1}\Sigma\right) + \frac{1}{2r}\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t. \tag{6}$$

The updates to $\boldsymbol{\mu}$ and Σ can therefore be performed independently. The update for $\boldsymbol{\mu}$ is conservative (or passive) since it makes no change unless the classifier loss is non-zero, and we follow CW learning by enforcing a correspondingly conservative update for the confidence parameter Σ, updating it only when $\boldsymbol{\mu}$ changes. This results in fewer updates and is easier to analyze. Our update thus proceeds in two stages.

1. Update the mean parameters: $\qquad \boldsymbol{\mu}_t = \arg\min_{\boldsymbol{\mu}} \mathcal{C}_1(\boldsymbol{\mu}) \qquad (7)$
2. If $\boldsymbol{\mu}_t \neq \boldsymbol{\mu}_{t-1}$, update the confidence parameters: $\qquad \Sigma_t = \arg\min_{\Sigma} \mathcal{C}_2(\Sigma) \qquad (8)$

   otherwise, set: $\qquad \Sigma_t = \Sigma_{t-1}$

Note that many online algorithms perform a gradient step on an objective similar to $\mathcal{C}_1$. Here, in contrast, we optimize $\mathcal{C}_1$ (and $\mathcal{C}_2$) exactly; this leads to strong performance, as we shall see. Furthermore, we show next how these exact updates can be computed efficiently in the binary and multi-class classification settings.

3.1 Binary classification with the squared hinge loss

In the binary case, where $\mathcal{Y} = \{-1, +1\}$, we apply the squared hinge loss, as in previous work (Crammer et al. 2009b). The squared hinge loss is given by

$$\hat{\ell}_{\mathrm{h}^2}(\boldsymbol{\mu}, \boldsymbol{x}, y) = \left(\max\left\{0, 1 - y(\boldsymbol{\mu} \cdot \boldsymbol{x})\right\}\right)^2, \tag{9}$$

which forms a convex, differentiable upper bound on the zero-one loss. We now develop the updates in Eq. (7) and Eq. (8) explicitly, starting with the former. We first observe that if $\hat{\ell}_{\mathrm{h}^2}(\boldsymbol{\mu}_{t-1}, \boldsymbol{x}_t, y_t) = 0$ then no update is required and $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1}$.

We thus assume $1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) \geq 0$. Taking the derivative of $\mathcal{C}_1(\boldsymbol{\mu})$ and setting it to zero, we get

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} - \frac{1}{2r}\left[\frac{d}{d\boldsymbol{\mu}}\hat{\ell}_{\mathrm{h}^2}(\boldsymbol{\mu}, \boldsymbol{x}_t, y_t)\right]_{\mu=\boldsymbol{\mu}_t} \Sigma_{t-1}\boldsymbol{x}_t, \tag{10}$$

assuming Σ_{t-1} is non-singular. Substituting the derivative of the squared hinge loss in Eq. (10) gives

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \frac{y_t}{r}\left(1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)\right)\Sigma_{t-1}\boldsymbol{x}_t. \tag{11}$$

We solve for $\boldsymbol{\mu}_t$ by taking the dot product of each side of the equality with $\boldsymbol{x}_t$, yielding

$$y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) = y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t) + \frac{y_t^2}{r}\left(1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)\right)\boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t.$$

Solving for $y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$ we get

$$y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) = \frac{r y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t) + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}{r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}.$$

Thus,

$$\frac{1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)}{r} = \frac{r - r y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)}{r(r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t)} = \frac{1 - y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)}{r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}.$$

Substituting back in Eq. (11) we obtain the rule

$$\begin{aligned} \boldsymbol{\mu}_t &= \boldsymbol{\mu}_{t-1} + \frac{\max\{0, 1 - y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)\}}{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t + r} \Sigma_{t-1} y_t \boldsymbol{x}_t \\ &= \boldsymbol{\mu}_{t-1} + \frac{\hat{\ell}_{\mathrm{h}}(\boldsymbol{\mu}_{t-1}, \boldsymbol{x}_t, y_t)}{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t + r} \Sigma_{t-1} y_t \boldsymbol{x}_t, \end{aligned} \tag{12}$$

where $\hat{\ell}_{\mathrm{h}}$ is the standard hinge loss. Note that the above update rule includes also the case when no update is performed (i.e., $1 \leq y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)$). It can be easily verified that Eq. (12) satisfies our assumption that $1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) \geq 0$, since

$$\begin{aligned} 1 - y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) &= 1 - y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t) - \frac{1 - y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)}{r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t} \left(\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t\right) \\ &= \left(1 - y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t)\right)\left(1 - \frac{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}{r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}\right) \geq 0. \end{aligned}$$

The update for the confidence parameters is made only if $\boldsymbol{\mu}_t \neq \boldsymbol{\mu}_{t-1}$, that is, if $1 > y_t \boldsymbol{x}_t^\top \boldsymbol{\mu}_{t-1}$. In this case, we compute the update of the confidence parameters by setting the derivative of $\mathcal{C}_2(\Sigma)$ to zero:

$$-\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma_{t-1}^{-1} + \frac{1}{2r}\left[\frac{d}{dz} z|_{z=\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t}\right] \boldsymbol{x}_t \boldsymbol{x}_t^\top = 0. \tag{13}$$

From this we obtain the following update for the confidence parameters:

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \frac{\boldsymbol{x}_t \boldsymbol{x}_t^\top}{r}. \tag{14}$$

Using the Woodbury identity we can also rewrite the update for Σ in non-inverted form:

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1} \boldsymbol{x}_t \boldsymbol{x}_t^\top \Sigma_{t-1}}{r + \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}. \tag{15}$$

Pseudocode for binary AROW appears in Fig. 1. Later we will make use of the following claim:

Claim *The eigenvalues of the confidence parameters obtained during the run of any algorithm derived via Eq.* (14) *and Eq.* (15) *are monotonically decreasing*:

$$\Sigma_t \preceq \Sigma_{t-1}; \qquad \Sigma_t^{-1} \succeq \Sigma_{t-1}^{-1}.$$

```
Input parameters   $r$
Initialize   $\boldsymbol{\mu}_0 = \mathbf{0}$, $\Sigma_0 = I$,
For $t = 1, \dots, T$
– Receive a training example $\boldsymbol{x}_t \in \mathbb{R}^d$
– Compute margin and confidence $m_t = \boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t \quad v_t = \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t$
– Receive true label $y_t$, and suffer label loss $\ell_t = 1$ if $\mathrm{sign}(m_t) \neq y_t$
– If $m_t y_t < 1$, update using Eq. (12) & Eq. (15):
```

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \alpha_t \Sigma_{t-1} y_t \boldsymbol{x}_t \qquad \Sigma_t = \Sigma_{t-1} - \beta_t \Sigma_{t-1} \boldsymbol{x}_t \boldsymbol{x}_t^\top \Sigma_{t-1}, \quad \beta_t = \frac{1}{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t + r}$$

$$\alpha_t = \max\bigl(0, 1 - y_t \boldsymbol{x}_t^\top \boldsymbol{\mu}_{t-1}\bigr) \beta_t \qquad \text{(squared hinge)} \tag{16}$$

$$\alpha_t = \min\left\{\frac{1}{2r}, \max\left\{0, \frac{1 - yit(\boldsymbol{x}_t \cdot \boldsymbol{\mu}_{t-1})}{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t}\right\}\right\} \qquad \text{(hinge)} \tag{17}$$

Output: Weight vector $\boldsymbol{\mu}_T$ and confidence Σ_T.

Fig. 1 The AROW algorithm for online binary classification

Proof The claim follows directly from Eq. (14) (or Eq. (15)) where it is stated that Σ_t^{-1} is a sum of Σ_{t-1}^{-1} and a (rank-1) positive semi-definite matrix. □

3.2 Binary classification with the hinge loss

We can also consider a version of AROW using the standard hinge loss in place of the squared hinge; the hinge loss is simply

$$\hat{\ell}_{\mathrm{h}}(\boldsymbol{\mu}, \boldsymbol{x}, y) = \max\bigl\{0, 1 - y(\boldsymbol{\mu} \cdot \boldsymbol{x})\bigr\}. \tag{18}$$

We can derive the update rules via a reduction to PA-I (Crammer et al. 2006). We first write Eq. (5) for the hinge loss:

$$\min_{\boldsymbol{\mu}} \frac{1}{2} (\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu})^\top \Sigma_{t-1}^{-1} (\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}) + \frac{1}{2r} \max\bigl\{0, 1 - y(\boldsymbol{\mu} \cdot \boldsymbol{x})\bigr\}. \tag{19}$$

Next, we change variables (this will give us a Euclidean distance term):

$$\boldsymbol{\mu} = \Sigma_{t-1}^{1/2} \boldsymbol{v}, \qquad \boldsymbol{\mu}_{t-1} = \Sigma_{t-1}^{1/2} \boldsymbol{v}_{t-1}, \qquad \boldsymbol{x} = \Sigma_{t-1}^{-1/2} \boldsymbol{z}. \tag{20}$$

Substituting in Eq. (19), we get

$$\min_{\boldsymbol{\mu}} \frac{1}{2} (\boldsymbol{v}_{t-1} - \boldsymbol{v})^\top (\boldsymbol{v}_{t-1} - \boldsymbol{v}) + \frac{1}{2r} \max\bigl\{0, 1 - y(\boldsymbol{v} \cdot \boldsymbol{z})\bigr\}. \tag{21}$$

This is exactly the optimization problem of PA-I (Crammer et al. 2006, Sect. 3), for which the solution is given by

$$\boldsymbol{v} = \boldsymbol{v}_{t-1} + \alpha_t y \boldsymbol{z}, \quad \text{where } \alpha_t = \min\left\{\frac{1}{2r}, \max\left\{0, \frac{1 - y(\boldsymbol{z} \cdot \boldsymbol{v}_{t-1})}{\boldsymbol{z}^\top \boldsymbol{z}}\right\}\right\}.$$

Substituting back the original variables from Eq. (20),

$$\Sigma_{t-1}^{-\frac{1}{2}} \boldsymbol{\mu} = \Sigma_{t-1}^{-\frac{1}{2}} \boldsymbol{\mu}_{t-1} + \alpha_t y \Sigma_{t-1}^{\frac{1}{2}} \boldsymbol{x} \quad \Rightarrow \quad \boldsymbol{\mu} = \boldsymbol{\mu}_{t-1} + \alpha_t y \Sigma_{t-1} \boldsymbol{x}, \tag{22}$$

where

$$\alpha_t = \min\left\{\frac{1}{2r}, \max\left\{0, \frac{1 - y(\boldsymbol{x}\cdot\boldsymbol{\mu}_{t-1})}{\boldsymbol{x}^\top \Sigma_{t-1}\boldsymbol{x}}\right\}\right\}. \tag{23}$$

Pseudocode for binary AROW with the hinge loss also appears in Fig. 1. Comparing this update to the squared hinge update, we observe that the update to the mean parameters $\boldsymbol{\mu}$ takes a common additive form of $\boldsymbol{\mu}_{t-1}$ plus a scalar times $\Sigma_{t-1}\boldsymbol{x}$. The difference is the exact value of the scalar (compare Eq. (16) with Eq. (23)).

Finally, we note from Eq. (5) and Eq. (6) that the update for the covariance Σ_t is performed using the same rule as for the squared hinge case, and is given in Eq. (15).

3.3 Multi-class classification

When there are more than two classes, we assume that a feature function f maps the input $\boldsymbol{x}$ and a proposed class y to the weight space $\mathbb{R}^d$ (see Collins 2002). Then the prediction rule is given by

$$h_{\boldsymbol{\mu}}(\boldsymbol{x}) = \arg\max_y\left[\boldsymbol{\mu}\cdot f(\boldsymbol{x}, y)\right], \tag{24}$$

and the squared hinge loss by

$$\hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}, y) = \left(\max\left\{0, 1 + \max_{\hat{y}\neq y}\left\{\boldsymbol{\mu}\cdot f(\boldsymbol{x},\hat{y})\right\} - \boldsymbol{\mu}\cdot f(\boldsymbol{x}, y)\right\}\right)^2. \tag{25}$$

We now present two updates for the multi-class setting. The first is based on directly minimizing the hinge-loss defined above (see, for example, Crammer and Singer 2003); the second update, motivated by the high time complexity of the first update, is based on a top-1 reduction (e.g., see Collins 2002). We start with an update that minimizes the multi-class hinge loss.

To compute the update for $\boldsymbol{\mu}$ (Eq. (7)), we first rewrite the loss using constrained optimization:

$$\begin{aligned}\hat{\ell}(\boldsymbol{\mu}, \boldsymbol{x}, y) = \min_{\gamma} \quad & \gamma^2 \\ \text{s.t.} \quad & \gamma \geq 1 + \boldsymbol{\mu}\cdot f(\boldsymbol{x},\hat{y}) - \boldsymbol{\mu}\cdot f(\boldsymbol{x}, y) \quad \forall\hat{y}\neq y \\ & \gamma \geq 0.\end{aligned} \tag{26}$$

We can now find $\boldsymbol{\mu}_t$ as follows:

$$\begin{aligned}\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{\mu},\gamma} \quad & (\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu})^\top \Sigma_{t-1}^{-1}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}) + \frac{1}{r}\gamma^2 \\ \text{s.t.} \quad & \gamma \geq 1 + \boldsymbol{\mu}\cdot f(\boldsymbol{x},\hat{y}) - \boldsymbol{\mu}\cdot f(\boldsymbol{x}, y) \quad \forall\hat{y}\neq y \\ & \gamma \geq 0.\end{aligned} \tag{27}$$

This is a quadratic programming problem with linear constraints, and can be solved efficiently with standard approaches such as Hildreth's algorithm (Censor and Zenios 1997). Although such algorithms run in polynomial rime, in practice it may still be impractical to include constraints for all $\hat{y}$. In such cases a useful approximation is to use only the top k labels as ranked by the scoring function $\boldsymbol{\mu}_{t-1}\cdot f(\boldsymbol{x},\hat{y})$, for some reasonable choice of k (see

```
Input parameters   $r$
Initialize   $\boldsymbol{\mu}_0 = \mathbf{0}$, $\Sigma_0 = I$,
For $t = 1, \dots, T$
– Receive a training example $\boldsymbol{x}_t$
– Make a prediction $\hat{y}_t = \arg\max_y [\boldsymbol{\mu}_{t-1} \cdot f(\boldsymbol{x}_t, y)]$
– Receive true label $y_t$, and suffer label loss $\ell_t = 1$ if $\hat{y}_t \neq y_t$
– Update $\boldsymbol{\mu}_t$ using Eq. (27)
```

$$\begin{aligned}\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{\mu},\gamma} \quad & (\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu})^\top \Sigma_{t-1}^{-1} (\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}) + \frac{1}{r}\gamma^2 \\ \text{s.t.} \quad & \gamma \geq 1 + \boldsymbol{\mu} \cdot f(\boldsymbol{x}, \hat{y}) - \boldsymbol{\mu} \cdot f(\boldsymbol{x}, y) \quad \forall \hat{y} \neq y \\ & \gamma \geq 0\end{aligned}$$

```
– If an update was performed ($\boldsymbol{\mu}_t \neq \boldsymbol{\mu}_{t-1}$), update:
```

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1} f_t f_t^\top \Sigma_{t-1}}{r + f_t^\top \Sigma_{t-1} f_t} \qquad f_t = \sum_y f(\boldsymbol{x}_t, y) \tag{28}$$

```
Output:   Weight vector $\boldsymbol{\mu}_T$ and confidence $\Sigma_T$.
```

Fig. 2 The full version of AROW algorithm for online multi-class classification

Sect. 5). In the case of $k = 1$, the update of Eq. (27) reduces to an update similar to the update of the binary case, which we discuss in detail below. Since the update for Σ does not depend on the loss function (or in fact the label at all), it is identical to the update of Eq. (15), where the input $\boldsymbol{x}_t$ is replaced with the sum of the features over all labels:

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1} f_t f_t^\top \Sigma_{t-1}}{r + f_t^\top \Sigma_{t-1} f_t}, \tag{29}$$

where

$$f_t = \sum_y f(\boldsymbol{x}_t, y). \tag{30}$$

Pseudo code for the algorithm appears in Fig. 2.

We now move to the second update, which we refer to as the top-1 reduction. Let

$$\tilde{y}_t = \arg\max_{y \neq y_t} \boldsymbol{\mu}_{t-1}^\top f(\boldsymbol{x}_t, y) \tag{31}$$

be the closest competitor at the start of iteration t. Note, if the algorithm makes a prediction mistake and $\hat{y}_t \neq y_t$ then $\tilde{y}_t = \hat{y}_t$ is the label with the largest inner-product value $\boldsymbol{\mu}_{t-1}^\top f(\boldsymbol{x}_t, z)$ over all labels z. Otherwise, if the prediction is correct and $\hat{y}_t = y_t$, then $\tilde{y}_t$ is the label with the second-largest inner-product. Then, ignoring all other labels we perform a the top-1 update using a binary update with features

$$\Delta f_t = f(\boldsymbol{x}_t, y_t) - f(\boldsymbol{x}_t, \tilde{y}_t) \tag{32}$$

assigned with a positive label. The update is summarized in Fig. 3 and analyzed below.

4 Analysis

We start our analysis by showing that AROW can be combined with Mercer kernels (Mercer 1909) using the following representer theorem.

```
Input parameters  $r$
Initialize  $\boldsymbol{\mu}_0 = \mathbf{0}$, $\Sigma_0 = I$,
For $t = 1, \ldots, T$
– Receive a training example $\boldsymbol{x}_t$
– Make a prediction $\hat{y}_t = \arg\max_y [\boldsymbol{\mu}_{t-1} \cdot f(\boldsymbol{x}_t, y)]$
– Receive true label $y_t$, and suffer label loss $\ell_t = 1$ if $\hat{y}_t \neq y_t$
– Compute the competitor label $\tilde{y}_t = \arg\max_{y \neq y_t} \boldsymbol{\mu}_{t-1}^\top f(\boldsymbol{x}_t, y)$
– Define $\Delta f_t = f(\boldsymbol{x}_t, y_t) - f(\boldsymbol{x}_t, \tilde{y}_t)$ using Eq. (32)
– Compute margin and confidence $m_t = \boldsymbol{\mu}_{t-1} \cdot (\Delta f_t)$   $v_t = (\Delta f_t)^\top \Sigma_{t-1} (\Delta f_t)$
– If $m_t < 1$, update using a generalization of Eq. (12) & Eq. (15):
      $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \alpha_t \Sigma_{t-1} (\Delta f_t)$      $\Sigma_t = \Sigma_{t-1} - \beta_t \Sigma_{t-1} (\Delta f_t)(\Delta f_t)^\top \Sigma_{t-1}$
      $\beta_t = \frac{1}{(\Delta f_t)^\top \Sigma_{t-1} (\Delta f_t) + r}$      $\alpha_t = \max(0, 1 - (\Delta f_t)^\top \boldsymbol{\mu}_{t-1}) \beta_t$
Output:  Weight vector $\boldsymbol{\mu}_T$ and confidence $\Sigma_T$.
```

Fig. 3 The top-1 version of AROW algorithm for online multi-class classification

Lemma 1 (Representer Theorem) *Assume that $\Sigma_0 = I$ and $\boldsymbol{\mu}_0 = \mathbf{0}$. The mean parameters $\boldsymbol{\mu}_t$ and confidence parameters Σ_t produced by updating via Eq.* (12) *and Eq.* (15) *can be written as linear combinations of the input vectors* (*resp. outer products of the input vectors with themselves*) *with coefficients depending only on inner products of input vectors.*

Proof Formally, we need to show that the mean $\boldsymbol{\mu}_i$ and covariance Σ_i parameters computed by Fig. 1 can be written as:

$$\Sigma_t = \sum_{p,q=1}^{t} \pi_{p,q}^{(t)} \boldsymbol{x}_p \boldsymbol{x}_q^\top + I, \qquad \boldsymbol{\mu}_t = \sum_{p=1}^{t} \nu_p^{(t)} \boldsymbol{x}_p, \tag{33}$$

where π and ν are scalars that depend only on inner products of input vectors. The proof proceeds by induction. The base case follows from the definitions of $\boldsymbol{\mu}_0$ and Σ_0, and the induction step follows algebraically from the update rules Eq. (12) and Eq. (15).

For the induction step we first substitute Eq. (33) in Eq. (12) to obtain

$$\begin{aligned}\boldsymbol{\mu}_t &= \boldsymbol{\mu}_{t-1} + \alpha_t \Sigma_{t-1} y_t \boldsymbol{x}_t \\ &= \sum_{p=1}^{t-1} \nu_p^{(t-1)} \boldsymbol{x}_p + \alpha_t \left(\sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \boldsymbol{x}_p \boldsymbol{x}_q^\top + I \right) y_t \boldsymbol{x}_t \\ &= \sum_{p=1}^{t-1} \left(\nu_p^{(t)} + \sum_{q=1}^{t-1} \alpha_t \pi_{p,q}^{(t)} y_t \boldsymbol{x}_q^\top \boldsymbol{x}_t \right) \boldsymbol{x}_p + \alpha_t y_t \boldsymbol{x}_t,\end{aligned} \tag{34}$$

which is of the desired form with

$$\nu_p^{(t)} = \nu_p^{(t-1)} + \sum_{q=1}^{t-1} \alpha_t \pi_{p,q}^{(t)} y_t \boldsymbol{x}_q^\top \boldsymbol{x}_t \quad \text{for } p < t, \qquad \nu_t^{(t)} = \alpha_t y_t. \tag{35}$$

Note that $\{\nu_p^{(t)}\}$ can be written in terms of inner products of input vectors as long as α_t can, which follows directly from Eq. (16).

Next, we substitute Eq. (33) into Eq. (15):

$$\begin{aligned}
\Sigma_t &= \Sigma_{t-1} - \beta_t \Sigma_{t-1}\boldsymbol{x}_t\boldsymbol{x}_t^\top \Sigma_{t-1} \\
&= \sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \boldsymbol{x}_p\boldsymbol{x}_q^\top + I - \beta_t \left(\sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \boldsymbol{x}_p\boldsymbol{x}_q^\top + I \right) \boldsymbol{x}_t\boldsymbol{x}_t^\top \left(\sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \boldsymbol{x}_p\boldsymbol{x}_q^\top + I \right) \\
&= \sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \boldsymbol{x}_p\boldsymbol{x}_q^\top + I - \beta_t\boldsymbol{x}_t\boldsymbol{x}_t^\top - \beta_t \sum_{p,q=1}^{t-1} \pi_{p,q}^{(t-1)} \left(\left(\boldsymbol{x}_q^\top\boldsymbol{x}_t\right)\boldsymbol{x}_p\boldsymbol{x}_t^\top + \left(\boldsymbol{x}_p^\top\boldsymbol{x}_t\right)\boldsymbol{x}_t\boldsymbol{x}_q^\top \right) \\
&\quad - \beta_t \sum_{p,q=1}^{t-1} \left(\sum_{r,s=1}^{t-1} \pi_{p,r}^{(t-1)} \pi_{s,q}^{(t-1)} \left(\boldsymbol{x}_r^\top\boldsymbol{x}_t\right)\left(\boldsymbol{x}_t^\top\boldsymbol{x}_s\right) \right) \boldsymbol{x}_p\boldsymbol{x}_q^\top .
\end{aligned} \tag{36}$$

Gathering the appropriate terms in the above calculation, we have that the inductive hypothesis holds with

$$\begin{aligned}
\pi_{p,q}^{(t)} &= \pi_{p,q}^{(t-1)} - \beta_t \sum_{r,s} \pi_{p,r}^{(t-1)} \pi_{s,q}^{(t-1)} \left(\boldsymbol{x}_r^\top\boldsymbol{x}_t\right)\left(\boldsymbol{x}_t^\top\boldsymbol{x}_s\right) \\
\pi_{p,t}^{(t)} = \pi_{t,p}^{(t)} &= -\beta_t \sum_{p,r=1}^{t-1} \pi_{p,r}^{(t-1)} \left(\boldsymbol{x}_r^\top\boldsymbol{x}_t\right) \\
\pi_{t,t}^{(t)} &= -\beta_t .
\end{aligned} \tag{37}$$

From these equations we see that $\{\pi_{p,q}^{(t)}\}$ can be computed via inner product as long as β_t can, which is implied by Eq. (16). □

We now turn to analyzing the number of mistakes AROW makes in the binary case. Denote by $\mathcal{M}$ the set of example indices for which the algorithm makes a mistake (that is, where $y_t(\boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t) \le 0$) and let $M = |\mathcal{M}|$. Similarly, denote by $\mathcal{U}$ the set of example indices for which there is an update but not a mistake ($0 < y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t) \le 1$) and let $U = |\mathcal{U}|$. The remaining examples, for which the algorithm had a margin of at least one ($1 < y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$), do not affect the behavior of the algorithm and can be ignored. We denote the outer product of the mistakes by $\mathbf{X}_{\mathcal{M}} = \sum_{t\in\mathcal{M}} \boldsymbol{x}_i\boldsymbol{x}_i^\top$, the outer product of the errors by $\mathbf{X}_{\mathcal{U}} = \sum_{t\in\mathcal{U}} \boldsymbol{x}_i\boldsymbol{x}_i^\top$, and their sum by $\mathbf{X}_{\mathcal{A}} = \mathbf{X}_{\mathcal{M}} + \mathbf{X}_{\mathcal{U}}$.

Theorem 1 *For any reference weight vector $\boldsymbol{u} \in \mathbb{R}^d$, the number of mistakes made by AROW with squared hinge loss (Fig. 1) is upper bounded by*

$$M \le \sum_{t\in\mathcal{M}\cup\mathcal{U}} g_t + \sqrt{r\|\boldsymbol{u}\|^2 + \boldsymbol{u}^\top\mathbf{X}_{\mathcal{A}}\boldsymbol{u}} \sqrt{\log\left(\det\left(I + \frac{1}{r}\mathbf{X}_{\mathcal{A}}\right)\right) + U} - U, \tag{38}$$

where $g_t = \max(0, 1 - y_t\boldsymbol{u}^\top\boldsymbol{x}_t)$.

Before turning to the proof we highlight a few properties of the bound.

Remark 1 The bound compares the number of *mistakes* the algorithm makes to the *hinge loss* of a reference vector. This asymmetry is typical of mistake bounds, e.g., those for the second-order perceptron (Cesa-Bianchi et al. 2005) and the Perceptron (Gentile 2003).

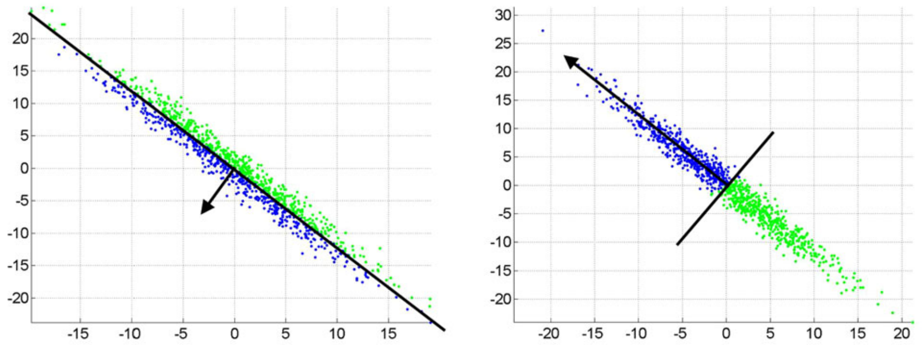


Fig. 4 Illustration of the mistake bound in two different settings. *On the left*, the separating hyperplane is aligned with the primary axis of the data distribution, and most of the input vectors are near the hyperplane. *On the right*, the separating hyperplane is aligned with the secondary axis of the data distribution, and most of the input vectors are far from the hyperplane

Remark 2 The two square root terms of the bound depend on r in opposite ways: the first is monotonically increasing, while the second is monotonically decreasing. One could expect to optimize the bound by minimizing over r. However, the bound also depends on r indirectly via other quantities (e.g., $\mathbf{X}_{\mathcal{A}}$), so there is no direct way to do so.

Remark 3 If all of the updates are associated with errors, that is, $\mathcal{U} = \emptyset$, then the bound reduces to the bound of the second-order perceptron (Cesa-Bianchi et al. 2005). In general, however, the bounds are not comparable since each depends on the actual runtime behavior of its algorithm.

Remark 4 Under the same conditions as the previous remark, $\mathcal{U} = \emptyset$, the second term of the bound is a product of two quantities. The latter is logarithmic in the number of mistakes (or even the total number of examples), while the first is a square root of a constant $\|\boldsymbol{u}\|^2$ and a quantity dependent on the geometry of the problem, $\boldsymbol{u}^\top \mathbf{X}_{\mathcal{A}} \boldsymbol{u}$. If most of the data points with mistakes lie near the hyperplane orthogonal to $\boldsymbol{u}$, as illustrated by the left part of Fig. 4, this term will be small. On the other hand, if most of the data lies far from the hyperplane orthogonal to $\boldsymbol{u}$, as in the right part of Fig. 4, the bound will be larger. Intuitively, data points that lie near the labeling boundary tend to be more helpful for tuning the parameters. (See also the third remark of Cesa-Bianchi et al. (2005, Sect. 3.1).)

Remark 5 The bound has a non-trivial dependency on the number of updates. If $\boldsymbol{u}^\top \mathbf{X}_{\mathcal{A}} \boldsymbol{u}$ is small, then making updates may reduce the bound, since it increases in $\sqrt{U}$ and decreases in U.

Remark 6 The bound of the Perceptron algorithm can be recovered from our bound for large values of r. As r gets large, the bound becomes

$$M \leq \sum_{t \in \mathcal{M} \cup \mathcal{U}} g_t + \sqrt{r \|\boldsymbol{u}\|^2} \sqrt{\operatorname{Tr}\left(\frac{1}{r} \mathbf{X}_{\mathcal{A}}\right) + U} - U, \tag{39}$$

using the inequality $\log(\det(I+\mathbf{A})) \le \mathrm{Tr}(\mathbf{A})$. Next, assume $\|\boldsymbol{x}_t\|^2 \le R^2$ for all t; this yields

$$M \le \sum_{t\in\mathcal{M}\cup\mathcal{U}} g_t + \sqrt{\|\boldsymbol{u}\|^2}\sqrt{(M+U)R^2 + rU} - U. \tag{40}$$

For simplicity, let $\mathcal{L}_{\boldsymbol{u}} = \sum_{t\in\mathcal{M}\cup\mathcal{U}} g_t$. Solving for M we have

$$M \le \mathcal{L}_{\boldsymbol{u}} + \frac{1}{2}\|\boldsymbol{u}\|^2 R^2 + \frac{1}{2}\|\boldsymbol{u}\| R\sqrt{4\mathcal{L}_{\boldsymbol{u}} + \|\boldsymbol{u}\|^2 R^2 + 4\frac{rU}{R^2}} - U. \tag{41}$$

When updates are made only for mistakes, as in the Perceptron algorithm, $U = 0$ and we recover the Perceptron bound.

Remark 7 We do not know of a bound for AROW with standard hinge loss, and leave it as an open problem.

We now prove the theorem. We first prove two auxiliary lemmas.

Lemma 2 *Let* $\hat{\ell}_t = \max(0, 1 - y_t\boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t)$ *and* $\chi_t = \boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t$. *Then, for every* $t \in \mathcal{M}\cup\mathcal{U}$,

$$\boldsymbol{u}^\top \Sigma_t^{-1}\boldsymbol{\mu}_t = \boldsymbol{u}^\top \Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \frac{y_t\boldsymbol{u}^\top\boldsymbol{x}_t}{r}, \tag{42}$$

$$\boldsymbol{\mu}_t^\top \Sigma_t^{-1}\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1}^\top \Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \frac{\chi_t + r - \hat{\ell}_t^2 r}{r(\chi_t + r)}. \tag{43}$$

The proof appears in Appendix.

Lemma 3 *Let T be the number of rounds. Then*

$$\sum_t \frac{\chi_t}{\chi_t + r} \le \log\bigl(\det\bigl(\Sigma_{T+1}^{-1}\bigr)\bigr).$$

Proof We remind the reader of the following definitions from Eq. (16):

$$\beta_t = \frac{1}{\boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t + r},$$
$$\Sigma_t = \Sigma_{t-1} - \beta_t \Sigma_{t-1}\boldsymbol{x}_t\boldsymbol{x}_t^\top \Sigma_{t-1}.$$

Consider the quantity

$$\begin{aligned}
\boldsymbol{x}_t^\top \Sigma_t \boldsymbol{x}_t &= \boldsymbol{x}_t^\top\bigl(\Sigma_{t-1} - \beta_t\Sigma_{t-1}\boldsymbol{x}_t\boldsymbol{x}_t^\top\Sigma_{t-1}\bigr)\boldsymbol{x}_t \\
&= \boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t - \beta_t\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t \\
&= \chi_t - \frac{\chi_t^2}{\chi_t + r} \\
&= \frac{\chi_t(\chi_t + r) - \chi_t^2}{\chi_t + r}
\end{aligned}$$

$$= \frac{\chi_t r}{\chi_t + r}. \tag{44}$$

Using Lemma D.1 from Cesa-Bianchi et al. (2005), we have that

$$\frac{1}{r} \boldsymbol{x}_t^\top \Sigma_t \boldsymbol{x}_t = 1 - \frac{\det(\Sigma_{t-1}^{-1})}{\det(\Sigma_t^{-1})}. \tag{45}$$

Combining Eq. (44) and Eq. (45),

$$\begin{aligned}\sum_t \frac{\chi_t}{\chi_t + r} = \sum_t \frac{1}{r} \boldsymbol{x}_t^\top \Sigma_t \boldsymbol{x}_t = \sum_t \left(1 - \frac{\det(\Sigma_{t-1}^{-1})}{\det(\Sigma_t^{-1})}\right) \\ \le -\sum_t \log\left(\frac{\det(\Sigma_{t-1}^{-1})}{\det(\Sigma_t^{-1})}\right) \\ \le \log\left(\frac{\det(\Sigma_{T+1}^{-1})}{\det(\Sigma_1^{-1})}\right) \\ \le \log\bigl(\det\bigl(\Sigma_{T+1}^{-1}\bigr)\bigr).\end{aligned} \tag{46}$$

□

We are now ready to prove Theorem 1.

Proof Since $1 \le \max\{0, 1 - a\} + a$ for any a, for all examples we have $1 \le g_t + y_t \boldsymbol{x}_t^\top \boldsymbol{u}$. Summing over examples for which an error or update occurs yields

$$M + U \le \sum_{t \in \mathcal{M} \cup \mathcal{U}} g_t + \sum_{t \in \mathcal{M} \cup \mathcal{U}} y_t \boldsymbol{x}_t^\top \boldsymbol{u}. \tag{47}$$

Applying Lemma 2, we replace the second term with $\sum_{t \in \mathcal{M} \cup \mathcal{U}} y_t \boldsymbol{x}_t^\top \boldsymbol{u} = r(\boldsymbol{u}^\top \Sigma_T^{-1} \boldsymbol{\mu}_T)$. Using the Cauchy-Schwartz inequality, we have

$$r\bigl(\boldsymbol{u}^\top \Sigma_T^{-1} \boldsymbol{\mu}_T\bigr) \le r\sqrt{\boldsymbol{u}^\top \Sigma_T^{-1} \boldsymbol{u}}\sqrt{\boldsymbol{\mu}_T^\top \Sigma_T^{-1} \boldsymbol{\mu}_T}. \tag{48}$$

We now bound the two square root terms on the right hand side of Eq. (48), starting with the left term. By definition,

$$\Sigma_T^{-1} = I + \frac{1}{r} \sum_{t \in \mathcal{M} \cup \mathcal{U}} \boldsymbol{x}_i \boldsymbol{x}_i^\top = I + \frac{1}{r}(\mathbf{X}_\mathcal{M} + \mathbf{X}_\mathcal{U}) = I + \frac{1}{r} \mathbf{X}_\mathcal{A},$$

and thus

$$\boldsymbol{u}^\top \Sigma_T^{-1} \boldsymbol{u} = \|\boldsymbol{u}\|^2 + \frac{1}{r} \boldsymbol{u}^\top \mathbf{X}_\mathcal{A} \boldsymbol{u}. \tag{49}$$

For the right term we iterate the second equality to get

$$\boldsymbol{\mu}_T^\top \Sigma_T^{-1} \boldsymbol{\mu}_T = \sum_{t \in \mathcal{M} \cup \mathcal{U}} \frac{\chi_t + r - \hat{\ell}_t^2 r}{r(\chi_t + r)} = \sum_{t \in \mathcal{M} \cup \mathcal{U}} \frac{\chi_t}{r(\chi_t + r)} + \sum_{t \in \mathcal{M} \cup \mathcal{U}} \frac{1 - \hat{\ell}_t^2}{\chi_t + r}. \tag{50}$$

Using Lemma 3, the first term of Eq. (50) is upper bounded by $\frac{1}{r}\log(\det(\Sigma_T^{-1}))$. For the second term in Eq. (50) we consider two cases. First, if a mistake occurred on example t, then $y_t(\boldsymbol{x}_t \cdot \boldsymbol{\mu}_{t-1}) \le 0$ and $\hat{\ell}_t \ge 1$, so $1 - \hat{\ell}_t^2 \le 0$. Second, if the algorithm made an update (but no mistake) on example t, then $0 < y_t(\boldsymbol{x}_t \cdot \boldsymbol{\mu}_{t-1}) \le 1$ and $\hat{\ell}_t \ge 0$, thus $1 - \hat{\ell}_t^2 \le 1$. We therefore have

$$\sum_{t\in\mathcal{M}\cup\mathcal{U}} \frac{1-\hat{\ell}_t^2}{\chi_t + r} \le \sum_{t\in\mathcal{M}} \frac{0}{\chi_t + r} + \sum_{t\in\mathcal{U}} \frac{1}{\chi_t + r} = \sum_{t\in\mathcal{U}} \frac{1}{\chi_t + r} \le \frac{U}{r}, \tag{51}$$

where the last inequality holds since $\chi_t \ge 0$.

Plugging Eq. (48), Eq. (49), Eq. (50) and Eq. (51) into Eq. (47), we get

$$\begin{aligned} M + U &\le \sum_{t\in\mathcal{M}\cup\mathcal{U}} g_t + r\sqrt{\|\boldsymbol{u}\|^2 + \frac{1}{r}\boldsymbol{u}^\top \mathbf{X}_\mathcal{A}\boldsymbol{u}}\sqrt{\frac{1}{r}\log\bigl(\det\bigl(\Sigma_T^{-1}\bigr)\bigr) + \frac{U}{r}} \\ &= \sum_{t\in\mathcal{M}\cup\mathcal{U}} g_t + \sqrt{r\|\boldsymbol{u}\|^2 + \boldsymbol{u}^\top \mathbf{X}_\mathcal{A}\boldsymbol{u}}\sqrt{\log\biggl(\det\biggl(I + \frac{1}{r}\mathbf{X}_\mathcal{A}\biggr)\biggr) + U}, \end{aligned}$$

which concludes the proof. $\square$

4.1 Multi-class problems

In the multi-class setting, where there are $K > 2$ possible labels, we analyze the top-1 version of AROW shown in Fig. 3, which reduces the many-way decision at each iteration to a binary choice between the true label and its current closest competitor.

To remind the reader, we assume that a feature function $f(\boldsymbol{x}_t, y_t) \in \mathbb{R}^d$ is given. The multi-class prediction is $\hat{y}_t = \arg\max_y[\boldsymbol{\mu} \cdot f(\boldsymbol{x}, y)]$, as defined in Eq. (24), and the competitor label is $\tilde{y}_t = \arg\max_{y\ne y_t} \boldsymbol{\mu}_{t-1}^\top f(\boldsymbol{x}_t, y)$. The difference feature vector is $\Delta f_t = f(\boldsymbol{x}_t, y_t) - f(\boldsymbol{x}_t, \tilde{y}_t)$ (Eq. (32)), and the update is defined in Fig. 3. From this construction, the top-1 hinge loss of the algorithm at time t is

$$\hat{\ell}_t = \max\bigl(0, 1 - \boldsymbol{\mu}_{t-1}^\top \Delta f_t\bigr), \tag{52}$$

and the multi-class hinge loss of $\boldsymbol{u}$ is

$$g_t = \max\Bigl(0, 1 + \max_{y\ne y_t}\bigl(\boldsymbol{u}^\top f(\boldsymbol{x}_t, y) - \boldsymbol{u}^\top f(\boldsymbol{x}_t, y_t)\bigr)\Bigr). \tag{53}$$

It follows that $g_t \ge 1 - \boldsymbol{u}^\top \Delta f_t$, and this is enough to ensure that the mistake bound goes through for the reduction. The proof has the same form as that of Theorem 1, but using the definition

$$\mathbf{X}_\mathcal{A} = \sum_{t\in\mathcal{M}\cup\mathcal{U}} \Delta f_t (\Delta f_t)^\top.$$

We first state the analogue of Lemma 2.

Lemma 4 *Let $\chi_t = (\Delta f_t)^\top \Sigma_{t-1}(\Delta f_t)$, and assume the definitions of Eq.* (32) *and Eq.* (52). *Then, for every $t \in \mathcal{M} \cup \mathcal{U}$,*

$$\boldsymbol{u}^\top \Sigma_t^{-1}\boldsymbol{\mu}_t = \boldsymbol{u}^\top \Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \frac{\boldsymbol{u}^\top \Delta f_t}{r}, \tag{54}$$

$$\boldsymbol{\mu}_t^\top \Sigma_t^{-1} \boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1}^\top \Sigma_{t-1}^{-1} \boldsymbol{\mu}_{t-1} + \frac{\chi_t + r - \hat{\ell}_t^2 r}{r(\chi_t + r)}. \tag{55}$$

The proof is identical to the proof of Lemma 2 in Appendix, but with $y_i \boldsymbol{x}_t$ replaced by Δf_t. We also have an analogue of Lemma 3, which is obtained by replacing the update of Σ in Eq. (15) with

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1}(\Delta f_t)(\Delta f_t)^\top \Sigma_{t-1}}{r + (\Delta f_t)^\top \Sigma_{t-1}(\Delta f_t)}. \tag{56}$$

These lemmas suffice to prove the following mistake bound for AROW using the top-1 reduction from multi-class to binary classification. The proof exactly mirrors that of Theorem 1, replacing the vector $y_t \boldsymbol{x}_t$ with the vector Δf_t, and using Lemma 4 instead of Lemma 2.

Theorem 2 *For any reference weight vector $\boldsymbol{u} \in \mathbb{R}^d$, the number of mistakes made by the top-*1 *multiclass version of AROW* (*Fig.* 3) *is upper bounded by*

$$M \leq \sum_{t \in \mathcal{M} \cup \mathcal{U}} g_t + \sqrt{r\|\boldsymbol{u}\|^2 + \boldsymbol{u}^\top \mathbf{X}_{\mathcal{A}} \boldsymbol{u}} \sqrt{\log\left(\det\left(I + \frac{1}{r}\mathbf{X}_{\mathcal{A}}\right)\right) + U} - U, \tag{57}$$

where $g_t = \max(0, 1 - \max_{y \neq y_t}(-\boldsymbol{u}^\top f(\boldsymbol{x}_t, y_t) + \boldsymbol{u}^\top f(\boldsymbol{x}_t, y)))$ *as defined in Eq.* (53).

5 Empirical evaluation

Our empirical evaluation investigates the effectiveness of AROW as both a binary and multi-class classification algorithm. We consider how AROW performs compared with state-of-the-art online classification algorithms in both clean and noisy settings. We also consider several types of data: synthetic, binary document classification and digit recognition (OCR), and multi-class document classification.

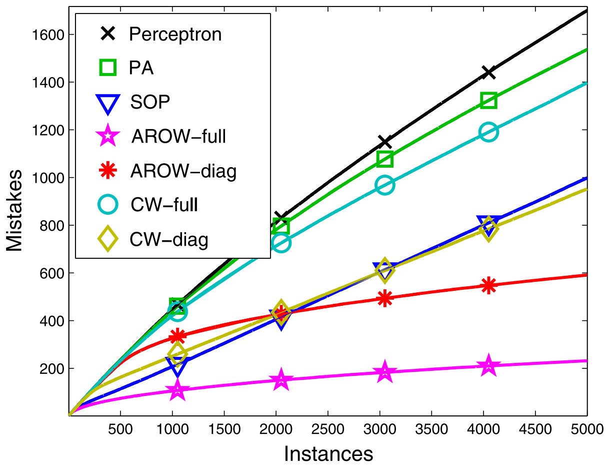
5.1 Setup

We selected three online learning baselines for comparison.

– **Passive-Aggressive (PA)** (Crammer et al. 2006): A large margin based online method that uses additive updates to enforce a fixed margin for each training example. Updates are made on margin violations (aggressive) but not otherwise (passive).
– **Second Order Perceptron (SOP)** (Cesa-Bianchi et al. 2005): An extension of the Perceptron algorithm that captures second order information. It is similar to AROW, with an important distinction being that it only updates on mistakes.
– **Confidence-Weighted (CW) learning** (Dredze et al. 2008): Similar to PA, except that a distribution over weight vectors replaces a single weight vector hypothesis. CW is the inspiration for AROW and is discussed in detail in Sect. 2. We use the "variance" version developed by Dredze et al. (2008).

Since we consider high dimensional datasets, it is computationally infeasible to model all second order feature interactions for SOP, CW and AROW. Instead, we drop cross-feature terms by projecting onto the set of diagonal matrices, following the approach of Dredze

Fig. 5 Learning curves for AROW (full/diagonal) and baseline methods with 5k synthetic binary training examples. 10 % of the labels were flipped at random to create label noise. Results are averaged over 100 runs



et al. (2008). While this may reduce performance, we make the same approximation for all evaluated algorithms. We found this method performed similarly to other projection schemes (Crammer et al. 2008).

All hyper-parameters (including r for AROW) and the number of online iterations (up to 10) were optimized using a single randomized run. We used 2,000 instances from each dataset and report all results over 10-fold cross-validation unless otherwise noted.

5.2 Synthetic data

Our synthetic data experiments follow the setting of Crammer et al. (2008). We generated 5,000 training examples in $\mathbb{R}^{20}$, where the first two coordinates were drawn from a 45° rotated Gaussian distribution with standard deviation 1. The remaining 18 coordinates were drawn from independent Gaussian distributions $\mathcal{N}(0, 2)$. Each point's label depended on the first two coordinates using a separator parallel to the long axis of the ellipsoid, yielding a linearly separable set. (See Fig. 4 (left) for an illustration.) To evaluate performance degradation in noisy label settings, we randomly inverted the labels on 10 % of the training examples. Note that evaluation is still with respect to the correct labels, so we can evaluate the true error rate. Since our synthetic data is low dimensional, we consider the full second order versions of CW, SOP and AROW, as well as the diagonalized versions. Algorithm parameters were tuned and results are reported over 100 runs.

5.2.1 Results

Figure 5 shows online learning curves for both the full and diagonalized versions of the three baseline algorithms on synthetic noisy data. Because of the noisy labels, all algorithms continue to make mistakes as they encounter more data. CW learning improves over previous methods, as reported in previous evaluations of CW (Crammer et al. 2008). We see further improvements with AROW, with the full second order version producing the fewest number of mistakes. For comparison, after 5,000 training examples, AROW-full has made about 75 % fewer mistakes than the next best method (CW). Similar improvements are evident after only 500 training examples. Note that AROW-full outperforms the diagonal version, while CW-full performs worse than CW-diagonal, as has been observed previously for noisy data (Crammer et al. 2008).

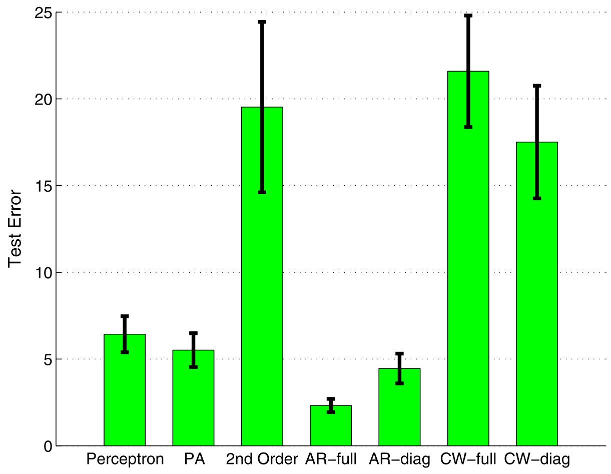


Fig. 6 Test results for AROW (full/diagonal) and baseline methods on 10k synthetic binary test examples trained on 5k examples (Fig. 5). Training label noise was set to 10 % and test labels are unchanged. Results are averaged over 100 runs

Figure 6 reveals a similar trend when the algorithms are evaluated in a batch setting with 10,000 test examples. CW is sensitive to label noise and attains a higher error rate compared with AROW. In fact, CW is worse than both Perceptron and PA due to overfitting to the label noise. This finding suggests that AROW, as a noise sensitive algorithm based on CW, is particularly useful for noisy settings. Finally, both variants of AROW have low variance in performance compared with all other algorithms.

5.3 Binary classification

We selected a variety of binary document classification data sets reflecting different NLP tasks. In total we consider 30 tasks from 4 data sets:

- **Amazon**: This dataset contains product reviews from Amazon.com that are labeled with both a domain (e.g., books or music) and a star rating for the product (Blitzer et al. 2007). This data set is commonly used to evaluate multi-domain sentiment classification. We used this data to create domain classification tasks, in which each task required the classification of a document into one of two domains. We took all pairs of the six domains to yield 15 tasks. Feature extraction follows Blitzer et al. (2007), representing each document as a set of bigram counts.
- **20 Newsgroups**: 20 newsgroups is a commonly used text classification dataset that includes approximately 20,000 newsgroup messages mined from 20 different newsgroups.[1] The dataset is a popular choice for binary and multi-class text classification as well as unsupervised clustering. We used the version of the data with duplicates removed. Following common practice, we created binary problems of choosing between two similar groups:

comp:	`comp.sys.ibm.pc.hardware`	vs.	`comp.sys.mac.hardware`
sci:	`sci.electronics`	vs.	`sci.med`
talk:	`talk.politics.guns`	vs.	`talk.politics.mideast`

These distinctions involve neighboring categories so they are fairly difficult to make. This

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/.

yielded 3 tasks. Each message was represented as a binary bag-of-words and there were between 1850 and 1971 instances per task.

– **Reuters (RCV1-v2/LYRL2004)**: The Reuters Corpus Volume 1 contains over 800,000 manually categorized newswire stories (Lewis et al. 2004). Labels describing the general topic, industry, and region of each article are provided. We created binary decision tasks by deciding between two industry labels, for a total of 3 tasks:

Insurance:	`Life` (I82002)	vs.	`Non-Life` (I82003)
Business Services:	`Banking` (I81000)	vs.	`Financial` (I83000)
Retail Distribution:	`Specialist Stores` (I65400)	vs.	`Mixed Retail` (I65600)

  Like 20 newsgroups, these distinctions involve neighboring categories so they are fairly hard to make. Details on document preparation and feature extraction are given by Lewis et al. (2004). For each problem we selected 2,000 instances using a bag-of-words representation with binary features.

– **Sentiment**: Using the same Amazon product reviews data, the goal is to classify a product review as having either positive or negative sentiment. Feature representations are the same as described above. We created a separate binary task for each of the 6 domains, yielding 6 tasks.

– **Spam**: The ECML/PKDD Spam Challenge[2] provides spam and ham emails for traditional spam classification. Data is provided for different users in two different tasks. We selected three task A users and classify emails as spam or ham (3 tasks). The provided data is already represented as features using bag-of-words.

In addition to binary document classification tasks, we also evaluate on two well known digit recognition tasks (OCR): **MNIST**[3] and **USPS**. For each of the data sets, we created 45 binary all-pairs tasks and an additional 10 one-vs-all tasks from the MNIST data (100 tasks). For these experiments, we report results using standard training and test splits instead of 10-fold cross validation.

For every data set, we introduce noise at various levels by randomly and independently flipping each binary label with a fixed probability (0, 0.05, 0.1, 0.15, 0.2, 0.3).

5.3.1 Results

To capture the large number of results for the four algorithms on multiple tasks at varying noise levels, we summarize our results in two ways. First, we compute the mean rank of each algorithm on all of the tasks. That is, for each task, we rank the four algorithms according to their performance on that task, with a rank of 1 indicating an algorithm outperformed all others and a rank of 4 for the worst algorithm. We then average these ranks over all of the tasks and report the mean rank. While this obscures the raw accuracy differences between each algorithm, it indicates general trends in the results.

Table 1 shows the mean rank for the four algorithms. With no noise in the labels, AROW performs comparably to CW, edging it out slightly with a mean rank of 1.51 versus 1.63. This indicates that across the tasks, AROW and CW consistently outperform the other methods (PA and SOP), which come in 3rd and 4th place respectively. This confirms previous results for CW and demonstrates that AROW gives comparable performance.

[2] http://ecmlpkdd2006.org/challenge.html.

[3] http://yann.lecun.com/exdb/mnist/index.html.

Table 1 Mean rank (out of 4, over all datasets) at different noise levels. A rank of 1 on a task indicates that an algorithm outperformed all the others, while a rank of 2 indicates that it was the second best performing algorithm. These ranks are averaged over all tasks. With no noise, AROW is the best algorithm, outperforming CW on a narrow majority of the tasks. As the noise level increases, the difference between AROW and the other algorithms increase. Additionally, CW does worse and is eventually overtaken by PA

Algorithm	Noise level					
	0.0	0.05	0.1	0.15	0.2	0.3
AROW	**1.51**	**1.44**	**1.38**	**1.42**	**1.25**	**1.25**
CW	1.63	1.87	1.95	2.08	2.42	2.76
PA	2.95	2.83	2.78	2.61	2.33	2.08
SOP	3.91	3.87	3.89	3.89	4.00	3.91

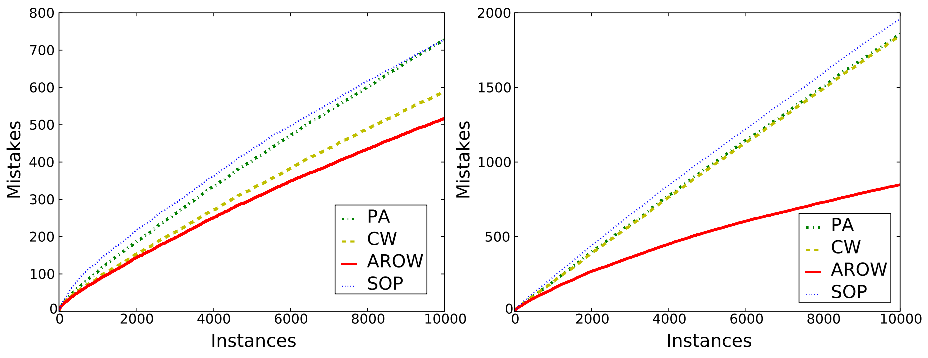


Fig. 7 Learning curves for AROW (diagonal) and baseline methods. MNIST 3 vs. 5 binary classification task for different amounts of label noise (*left*: 0 noise, *right*: 10 %)

Real differences emerge as we consider noisier settings. As the noise level increases, CW gradually worsens relative to PA and AROW. At the 20 % noise level, CW and PA are comparable, and at the 30 % noise level, PA easily outranks CW. Across all of these settings, AROW improves with respect to the other methods.

Our second summarization of the results allows for direct comparison between pairs of algorithms. We present the paired scores for each task as a point in a scatter plot with the x-axis indicating the baseline accuracy and the y-axis indicating AROW accuracy. A line with a slope of 1 allows for a direct comparison; for points above the line, AROW obtains higher accuracy, while points below the line favor the baseline. We include scatter plots at three different noise levels (0 %, 10 % and 30 %); see Fig. 8.

Consider in particular the comparison between AROW and CW, which perform similarly on clean data. For the no noise setting (0 %), most points are close to the line, with the two algorithms performing similarly overall. As noise increases, however, the points move further above the line, indicating the relative improvement of AROW over CW. In almost every high noise evaluation, AROW outperforms CW (as well as the other baselines). Furthermore, Fig. 7 shows the total number of mistakes (with respect to the noise-free labels) made by each algorithm during training on the MNIST dataset for 0 % and 10 % noise. Though absolute performance suffers with noise, the gap between AROW and the baselines increases. These results clearly demonstrate that AROW achieves state-of-the-art perfor-

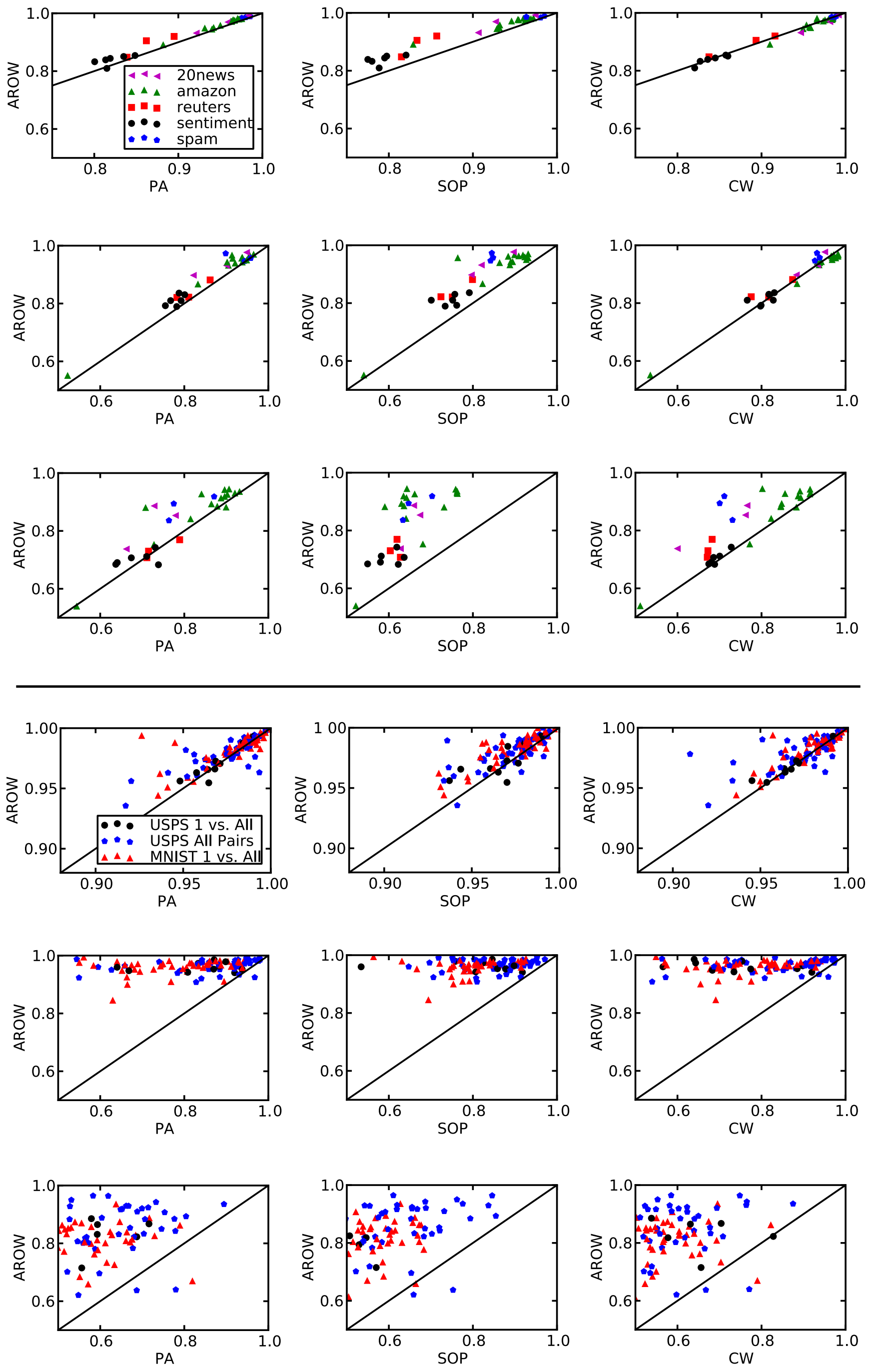


Fig. 8 Accuracy on text (*top*) and OCR (*bottom*) binary classification. Plots compare performance between AROW and a baseline method (*left column*: PA, *center*: SOP, *right*: CW), where markers above the line indicate superior AROW performance. Label noise increases with each row (*top to bottom*: no noise, 10 % noise and 30 % noise)

Table 2 A summary of the nine tasks, including the number of instances, features, and labels, and whether the numbers of examples in each class are balanced

Task	*Instances*	*Features*	*Labels*	*Bal.*
20 News	18,828	252,115	20	Y
Amazon 7	13,580	686,724	7	Y
Amazon 3	7,000	494,481	3	Y
Enron A	3,000	13,559	10	N
Enron B	3,000	18,065	10	N
NYTD	10,000	108,671	26	N
NYTO	10,000	108,671	34	N
NYTS	10,000	114,316	20	N
Reuters	4,000	23,699	4	N

mance in general, and that dramatic improvements over baseline algorithms appear on noisy data.

5.4 Multi-class classification

We next evaluate AROW on multi-class document classification tasks. We selected nine tasks from five different data sets that vary in difficulty, size, and label/feature counts. An overview of the properties of each task is shown in Table 2.

- **Amazon**: Using the Amazon data, we created two domain classification tasks from seven product domains: apparel, books, dvds, electronics, kitchen, music, video. In *Amazon 7*, we include all seven domains and in *Amazon 3* we select the three most common: books, dvds, and music. Feature extraction is the same as above.
- **20 Newsgroups**: We use all messages from the 20 newsgroups, classifying them into the newsgroups from which they originate. Feature extraction is the same as above.
- **Enron**: The Enron email data set contains hundreds of thousands of email messages for over 100 users.[4] The data set has been used for numerous classification tasks. We consider the task of automated sorting of emails into folders (Klimt and Yang 2004; Bekkerman et al. 2004). We selected two users with many email folders and messages: `farmer-d` (*Enron A*) and `kaminski-v` (*Enron B*). We used the ten largest folders for each user, excluding non-archival email folders such as "inbox," "deleted items," and "discussion threads." Emails were represented as binary bags-of-words with stop-words removed.
- **NY Times**: The New York Times Annotated Corpus contains 1.8 million articles that appeared from 1987 to 2007 (Sandhaus 2008). In addition to being one of the largest collections of raw news text, it is possibly the largest collection of publicly released annotated news text. Among other annotations, each article is labeled with the desk that produced the story (Financial, Sports, etc.) (*NYTD*), the online section to which the article was posted (*NYTO*), and the section in which the article was printed (*NYTS*). Articles were represented as bags-of-words with feature counts (stop words removed).
- **Reuters**: We used the Reuters corpus described above along with the four general topic labels for topic classification: corporate, economic, government, and markets. Feature extraction follows the setup above.

[4] http://www.cs.cmu.edu/~enron/.

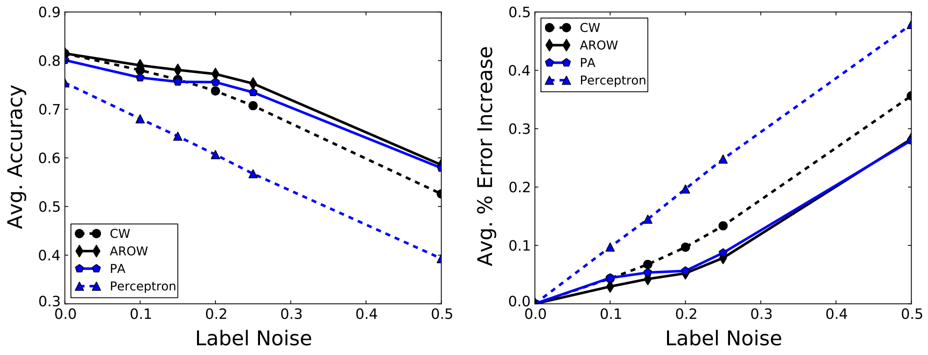


Fig. 9 Results for the seven multi-class classification tasks using AROW (1 constraint) and three first baselines. *Left*: The average accuracy of each method on the seven tasks as a function of label noise. For all noise settings, AROW remains the best performing method. As noise increases, PA overtakes CW, which gets much worse. *Right*: The average increase in error for each method across the seven tasks as a function of label noise

5.4.1 Results

We evaluated four multi-class algorithms on the seven multi-class data sets. First, we considered AROW with different sets of constraints: all (the full multi-class setting), one (the top-1 reduction described in Sect. 3.3), and five (a middle ground). A smaller set of constraints leads to faster rounds during training, but may require more rounds to converge. Second, we tested AROW with hinge loss in place of squared hinge loss, which we call AROW-H. We also included four baselines:

- **Perceptron**: A multi-class Perceptron using a 1-of-k encoding.
- **PA**: A PA classifier with a 1-of-k encoding using a single constraint. Crammer et al. (2009a) found that a single constraint did well on these tasks.
- **CW**: A diagonal CW classifier with a single constraint.
- **AdaGrad**: A diagonally regularized dual averaging (RDA) version with L_1 regularization (Duchi et al. 2011).

All experimental details (number of folds, parameter optimization, etc.) are the same as the binary experiments above. For multi-class data, we randomly select another label to replace the correct label in the case of label noise.

Despite the added complexity of the tasks, the results are qualitatively similar to the binary setting. AROW achieves the best performance overall in the zero noise setting; as noise increases, AROW continues to be robust compared with other methods. To demonstrate this effect, Fig. 9 shows the impact of noise on each of the algorithms (with one constraint in each case.) In Fig. 9 (left) we plot the average accuracy on test data of each method across the seven tasks (using 10-fold cross validation) as label noise increases from no noise to 50 % noise. While the general trend of all curves is downwards, the rate of decline is faster for Perceptron and CW than for AROW and PA. As with binary data, PA eventually overtakes CW with increased noise, while AROW remains the best algorithm. A similar trend can be seen in Fig. 9 (right), which show the average percentage increase in error for each method with increased label noise.

Looking at individual tasks more closely, we observe that the number of constraints has an impact similar to that seen in Crammer et al. (2009a), where a single constraint was found to be most effective. Here, across all tasks, the various constraint settings perform similarly,

Table 3 Accuracy on the multi-class data sets with no label noise

Data	Label Noise = 0.0							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	78.74	88.41	92.76	87.52	90.91	91.85	92.74	93.39
Enrom Kaminski	62.83	67.77	72.27	68.77	71.07	71.73	72.03	71.97
Enron Farmer	75.8	82.83	83.07	79.7	83.53	84.33	84.83	83.6
NYT Desk	77.7	81.22	82.43	74.7	81.19	81.74	81.97	81.98
NYT Online	76.91	81.11	82.62	75.25	81.97	82.7	82.67	82.59
NYT Section	48.53	56.42	54.62	54.69	56.71	55.49	56.05	56.65
Reuters	92.25	93.4	92.4	88.45	93.73	93.35	92.95	92.95
Sentiment 3	92.16	93.46	94.49	93.24	93.93	94.21	94.07	94.07
Sentiment All	74.06	76.31	78.41	74.78	77.64	77.84	78.4	78.49

Table 4 Accuracy on the multi-class data sets with label noise of 0.1

Data	Label Noise = 0.1							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	66.77	85.27	88.77	86.68	88.08	86.31	88.3	89.33
Enrom Kaminski	55.43	68.07	67.03	67.37	68.8	69.07	68.93	69.27
Enron Farmer	67.33	69.73	74.5	78.87	80.07	80.23	80.87	80.93
NYT Desk	69.72	78.43	79.25	75.43	78.58	80.04	79.73	80.21
NYT Online	69.9	78.59	79.46	74.8	79.61	80.35	80.58	80.75
NYT Section	45.41	53.24	51.72	52.98	53.63	54.76	54.93	54.93
Reuters	82.1	88.38	90.63	87.97	91.33	91.63	92.1	92.1
Sentiment 3	86.41	92.19	93.43	93.09	93.87	93.49	93.53	93.53
Sentiment All	69.34	74.71	77.54	74.46	75.42	75.42	75.86	76.19

suggesting that the single constraint algorithm, which is much faster, is sufficient for learning. We include results for all noise settings (0.0, 0.1, 0.15, 0.2, 0.25, 0.5) in Tables 3, 4, 5, 6, 7 and 8 for completeness. Despite our attempts, the performance of AdaGrad was not comparable to AROW or, in some instances, other baselines. This may be due to the fact that it uses L_1 regularization which promotes sparsity, yet may yield inferior performance.

Finally, we consider the impact of task difficulty on performance. Figure 10 shows the average error reduction of AROW from the mean accuracy of the other 3 first baselines for each task (y-axis) as a function of the overall difficulty of the task as measured by the mean accuracy of the 3 first baselines (x-axis). Results are included for 3 different noise levels (0, .1, .25). For each noise level, AROW shows larger improvements for tasks with higher mean accuracy. This trend is reflected in the three lines, which are a best linear fit of each noise level. Additionally, the slope of the lines increase with more noise, showing more significant improvements in higher noise settings.

5.5 Discussion

To help interpret the results, we classify the algorithms evaluated here according to four characteristics: the use of large margin updates, the use of parameter confidence weighting,

Table 5 Accuracy on the multi-class data sets with label noise of 0.15

Data	Label Noise = 0.15							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	63.08	73.65	86.19	85.78	86.81	86.76	87.13	88.27
Enrom Kaminski	52.4	65.87	63.93	65.57	68.13	67.43	67.4	66.53
Enron Farmer	64.07	76.93	70.9	78.4	78.83	79.5	80.23	78.6
NYT Desk	67.09	77.46	77.63	75.59	77.92	78.83	78.8	79.46
NYT Online	66.14	78.16	77.97	76.03	78.61	79.1	79.26	79.8
NYT Section	42.6	54.11	50.12	52.51	54.84	52.98	52.66	53.59
Reuters	75.8	88.33	88.92	88.85	91.02	91.33	91.8	91.8
Sentiment 3	82.86	90.36	92.29	93.5	92.03	92.7	92.49	92.49
Sentiment All	66.07	75.98	77.05	73.88	76.17	74.15	75.9	76.04

Table 6 Accuracy on the multi-class data sets with label noise of 0.2

Data	Label Noise = 0.2							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	56.51	78.02	82.89	84.34	86.1	84.76	83.73	86.45
Enrom Kaminski	48.8	63.6	60.1	64.2	64.07	65.47	65.33	66.3
Enron Farmer	58.53	74.37	67.17	78.23	76.37	78.43	77.63	77.67
NYT Desk	64.11	77.51	74.82	74.77	78.65	77.39	77.76	77.51
NYT Online	63.11	77.9	76.21	75.29	77.33	78.03	76.97	78.09
NYT Section	39.41	53.35	48.41	50.58	52.79	53.37	53.7	53.78
Reuters	72.78	87.92	86.88	88.77	88.5	89.9	90.3	90.3
Sentiment 3	80.39	92.59	91.76	93.29	93.27	92.74	92.8	92.8
Sentiment All	62.45	74.9	75.72	73.61	75.09	75.13	75.65	75.46

Table 7 Accuracy on the multi-class data sets with label noise of 0.25

Data	Label Noise = 0.25							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	51.38	80.16	76.6	82.17	84.95	82.37	82.72	82.36
Enrom Kaminski	48.27	59.17	57.6	61.6	63.67	58.3	59.07	62.13
Enron Farmer	53.57	73.1	61.93	76.07	75.37	77.5	76.67	77.8
NYT Desk	58.28	74.11	74.2	74.44	76.65	77.07	75.87	75.68
NYT Online	57.73	76.69	72.46	73.26	77.29	76.66	75.91	75.78
NYT Section	36.98	47.13	46.63	49.31	50.01	50.84	50.63	50.74
Reuters	68.17	87.38	82.78	88.67	89.0	89.2	88.77	88.77
Sentiment 3	75.97	91.5	89.19	92.41	92.79	91.63	91.24	91.24
Sentiment All	60.27	72.05	75.15	72.19	75.08	74.01	75.1	73.33

a design that accommodates non-separable data, and adaptive per-instance margin (Table 9). While all of these properties can be desirable in different situations, we would like to understand how they interact and achieve high performance while avoiding sensitivity to noise.

Table 8 Accuracy on the multi-class data sets with label noise of 0.5

Data	Label Noise = 0.5							
	Percep.	MIRA	CW	AdaGrad	AROW-H	AROW 1	AROW 5	AROW all
20 News	34.21	50.46	56.0	55.95	64.82	62.01	64.15	65.6
Enrom Kaminski	31.23	46.23	37.4	41.93	46.67	45.7	44.73	43.97
Enron Farmer	38.2	58.27	41.07	54.33	58.83	58.7	58.17	58.53
NYT Desk	37.98	56.41	57.04	56.81	64.69	63.64	59.21	62.28
NYT Online	40.87	64.25	56.66	56.6	61.86	63.47	64.55	63.88
NYT Section	27.27	37.02	34.47	30.17	38.42	37.73	38.89	39.04
Reuters	47.38	75.58	58.93	73.32	75.5	75.2	75.73	75.73
Sentiment 3	53.89	72.0	68.86	72.53	69.3	59.04	79.39	79.39
Sentiment All	42.63	61.09	62.88	54.2	62.84	61.58	62.69	62.66

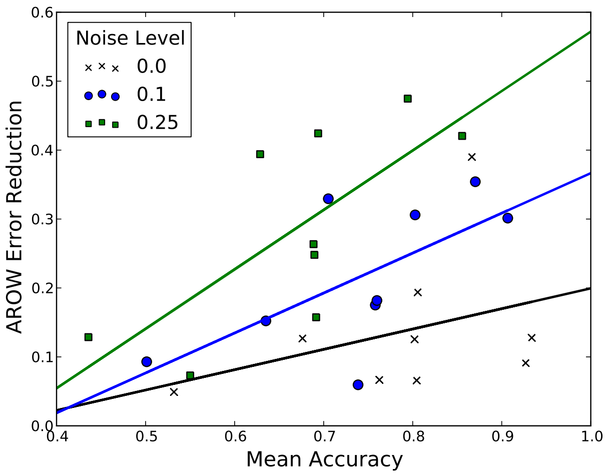


Fig. 10 The overall difficulty of each task as measured by the mean accuracy of the three first baselines (x-axis) versus the average improvement (error reduction) of AROW as compared to this mean (y-axis) for each task at three different noise levels. The three lines are best linear fits for each noise level. Observe that the slope of each line increases with additional noise, showing additional improvements in noisier settings

Table 9 Online algorithm properties overview

Algorithm	*Large Margin*	*Confidence*	*Non-Separable*	*Adaptive Margin*
Perceptron	No	No	**Yes**	No
PA	**Yes**	No	**Yes**	No
SOP	No	**Yes**	**Yes**	No
CW	**Yes**	**Yes**	No	**Yes**
AROW	**Yes**	**Yes**	**Yes**	No
AROW-H	**Yes**	**Yes**	**Yes**	No
AdaGrad	**Yes**	**Yes**	**Yes**	No

Based on the results it is clear that the combination of confidence information and large margin learning is powerful when label noise is low. CW easily outperforms the other baselines in such situations, as it has been shown to do in previous work. However, as noise increases, the separability assumption inherent in CW appears to reduce its performance considerably.

AROW, by combining the large margin and confidence weighting of CW with a soft update rule that accommodates non-separable data, matches CW's performance in general while avoiding degradation under noise. AROW lacks the adaptive margin of CW, suggesting that this characteristic is not crucial to achieving strong performance. We note that AROW-H and AdaGrad have similar properties; however, we leave open for future work the possibility that an algorithm with all four properties might have unique advantages.

6 Related work

Online additive algorithms have a long history, from the Perceptron (Rosenblatt 1958) to more recent methods (Kivinen and Warmuth 1997; Crammer et al. 2006). Our update has a more general form in which the input vector $\boldsymbol{x}_i$ is linearly transformed using the covariance matrix, both rotating the input and assigning weight specific learning rates.

Confidence weighted (CW) (Dredze et al. 2008; Crammer et al. 2008) algorithms, from which AROW was developed, update the mean and confidence parameters simultaneously, while AROW makes a decoupled update and softens the hard constraint of CW. The AROW algorithm can be seen as a variant of the PA-II algorithm by Crammer et al. (2006), where the regularization is modified according to the data. Additionally, future work might include developing a batch version of AROW, for instance, in the way the Gaussian Margin Machines of Crammer et al. (2009c) act as a batch version of CW. It might also be worthwhile to explore the performance of AROW with an (approximated) full covariance matrix, which has been shown to improve performance in some tasks (Ma et al. 2010).

AROW is perhaps most similar to the second order Perceptron (SOP) (Cesa-Bianchi et al. 2005). SOP, CW, and AROW all maintain second-order information. SOP performs the same type of update as AROW, but only in case of a true error. AROW, on the other hand, updates even when its prediction is correct so long as there is insufficient margin. Furthermore, SOP uses the current example in the correlation matrix for prediction, while AROW updates after prediction. Fundamentally, CW and AROW have a probabilistic motivation, while the SOP is geometric, the idea being to replace the ball around an example with a refined ellipsoid. However, a variant of CW similar to SOP follows from our derivation if we set $\alpha_i = 1$ in Eq. (16). Shivaswamy and Jebara (2007) have applied a similar motivation to batch learning.

The idea of using weight-specific variable learning rates has a long history in neural network learning (Sutton 1992), although we do not know of a previous model that specifically models confidence in a way that takes into account the frequency of features.

Ensemble learning shares the idea of combining multiple classifiers. Gaussian process classification (GPC) maintains a Gaussian distribution over weight vectors (primal) or over regressor values (dual). Our algorithm uses a different update criterion than the standard GPC Bayesian updates (Rasmussen and Williams 2006, Chap. 3), avoiding the challenge of approximating posteriors. Bayes point machines (Herbrich et al. 2001) maintain a collection of weight vectors consistent with the training data, and use the single linear classifier which best represents the collection. Conceptually, the collection is a non-parametric distribution over the weight vectors. Its online version (Harrington et al. 2003) maintains a set of weight vectors that are updated simultaneously. The relevance vector machine (Tipping 2001) incorporates probability into the dual formulation of SVMs. As in our work, the dual parameters are random variables distributed according to a diagonal Gaussian with example specific variance. The weighted-majority (Littlestone and Warmuth 1994) algorithm and later improvements (Cesa-Bianchi et al. 1997) combine the output of multiple arbitrary classifiers, maintaining a multinomial distribution over the experts. In this work, we assume linear classifiers as experts and maintain a Gaussian distribution over their weight vectors.

With the growth of available data there is an increasing need for algorithms that process training data very efficiently. A similar approach to ours is to train classifiers incrementally (Bordes and Bottou 2005). The extreme case is to use each example once, without repetitions, as in the multiplicative update method of Carvalho and Cohen (2006).

In Bayesian modeling, there are several existing approaches that use parameterized distributions over weight vectors. Borrowing concepts from support vector machines, Jaakkola et al. (1999) developed maximum entropy discrimination methods, which employ a generative model for each class. The models are specified by distributions over weights as well as margin thresholds, and the weights are learned using the maximum-entropy principle. In a more recent approach, Minka et al. (2009) proposed using additional virtual vectors to allow more expressive power beyond a Gaussian prior and posterior.

Passing the output of a linear model through a logistic function has a long history in the statistical literature, and is extensively covered in many textbooks (e.g. Hastie et al. 2001). Platt (1998) used similar ideas to convert the output of a support vector machine into probabilistic quantities.

Hazan (2006) described a framework for gradient descent algorithms with logarithmic regret in which a quantity similar to Σ_t plays an important role. Our algorithm differs in several ways. First, Hazan (2006) considered gradient algorithms, while we derive and analyze algorithms that directly solve an optimization problem. Second, we bound the loss directly, not the cumulative sum of regularization and loss. Third, the gradient algorithms perform a projection after making an update (not before) since the norm of the weight vector is kept bounded.

Since the conference version of this work was published, several algorithms related to CW and AROW have been proposed. Duchi et al. (2011) and McMahan and Streeter (2010) proposed replacing the standard Euclidean distance in stochastic gradient decent with general Mahalanobis distances defined by second order feature information. Their analysis suggests a logarithmic regret under some conditions, similar to our bounds here. However, the precise forms of the bounds are not comparable in general. Recently, Orabona and Crammer (2010) proposed a framework for online learning that includes an algorithm similar to AROW as a special case. From a different perspective, Crammer and Lee (2010) proposed a “microscopic” view of learning, tracking individual weight vectors as opposed to just macroscopic quantities such as mean and covariance. Their update has similar form to that of AROW (Eq. (16)), but with different rates.

Shivaswamy and Jebara (2010a, 2010b) proposed using second order information in the batch setting where an independent and identically distributed set of training examples is assumed. Their algorithm maximizes the (average) margin while also minimizing its variance. However, they do not maintain a distribution over weight vectors, and the probability space is induced using the distribution over training examples.

Finally, there have been several additional applications of AROW. Mejer and Crammer (2010) formulated a structured prediction learning algorithm based on CW, including different strategies for estimating confidence in a prediction label. These same ideas can be applied to AROW. Crammer (2010) applied CW to the common speech task of phone recognition, which might likewise benefit from AROW due to inherent noise. Saha et al. (2011) developed a multi-task online learning framework based on the AROW objective. Finally, AROW and the idea of confidence have been used for detecting phishing URLs (Le et al. 2010) and for learning language models (Ha-Thuc and Cancedda 2011).

7 Summary

We have presented AROW, an online learning algorithm that improves performance in noisy settings. Building on previous work on Confidence Weighted learning, AROW combines several desirable properties of online learning algorithms: large margin training, confidence weighting, and the capacity to handle non-separable data. The result is an algorithm that outperforms existing online learning algorithms, especially in the presence of label noise. Empirically, these trends hold up on a number of binary and multi-class data sets. Additionally, we derive a mistake bound that does not assume separability. Finally, our results suggest that future research into an algorithm that maintains the benefits of AROW while also using an adaptive margin could lead to a new robust method with potentially even better performance.

Acknowledgements Part of this research was done when all authors were affiliated with the department of information and computer science, the University of Pennsylvania. This work was partly supported by the Israeli Science Foundation grant ISF-1567/10. This publication only reflects the authors' views.

Appendix: Proof of Lemma 2

We present a detailed proof of Lemma 2. The first part follows the following chain of equalities,

$$\begin{aligned}
&\boldsymbol{u}^\top \Sigma_t^{-1}\boldsymbol{\mu}_t - \boldsymbol{u}^\top \Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1}\\
&\quad= \boldsymbol{u}^\top\left(\Sigma_{t-1}^{-1} + \frac{1}{r}\boldsymbol{x}_t\boldsymbol{x}_t^\top\right)\left(\boldsymbol{\mu}_{t-1} + \frac{\ell_t}{\chi_t + r}\Sigma_{t-1}y_t\boldsymbol{x}_t\right) - \boldsymbol{u}^\top\Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1}\\
&\quad= \frac{\ell_t}{\chi_t + r}y_t\boldsymbol{u}^\top\boldsymbol{x}_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\frac{\ell_t}{\chi_t + r}\big(\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t\big) + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\big(y_t\boldsymbol{\mu}_t^\top\boldsymbol{x}_t\big)\\
&\quad= \frac{\ell_t}{\chi_t + r}y_t\boldsymbol{u}^\top\boldsymbol{x}_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\frac{\ell_t}{\chi_t + r}\chi_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\big(y_t\boldsymbol{\mu}_t^\top\boldsymbol{x}_t\big)\\
&\quad= \frac{\ell_t}{\chi_t + r}y_t\boldsymbol{u}^\top\boldsymbol{x}_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\frac{\ell_t}{\chi_t + r}\chi_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\big(-1 + y_t\boldsymbol{\mu}_t^\top\boldsymbol{x}_t + 1\big)\\
&\quad= \frac{\ell_t}{\chi_t + r}y_t\boldsymbol{u}^\top\boldsymbol{x}_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)\frac{\ell_t}{\chi_t + r}\chi_t + \frac{1}{r}\big(y_t\boldsymbol{u}^\top\boldsymbol{x}_t\big)(-\ell_t + 1)\\
&\quad= \frac{y_t\boldsymbol{u}^\top\boldsymbol{x}_t}{r(\chi_t + r)}\big(\ell_t r + \ell_t\chi_t + (-\ell_t + 1)(\chi_t + r)\big)\\
&\quad= \frac{y_t\boldsymbol{u}^\top\boldsymbol{x}_t}{r(\chi_t + r)}\big(\ell_t(r + \chi_t) + (-\ell_t + 1)(\chi_t + r)\big)\\
&\quad= \frac{y_t\boldsymbol{u}^\top\boldsymbol{x}_t}{r(\chi_t + r)}(\chi_t + r)\\
&\quad= \frac{y_t\boldsymbol{u}^\top\boldsymbol{x}_t}{r}.
\end{aligned}$$

For the upper bound we have the following chain of equalities,

$$\boldsymbol{\mu}_t^\top\Sigma_t^{-1}\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}^\top\Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1}$$

$$
\begin{aligned}
&= \left(\boldsymbol{\mu}_{t-1}^\top + \frac{\ell_t}{\chi_t + r} y_t \boldsymbol{x}_t^\top \Sigma_{t-1}\right)\left(\Sigma_{t-1}^{-1} + \frac{1}{r}\boldsymbol{x}_t\boldsymbol{x}_t^\top\right) \times \left(\boldsymbol{\mu}_{t-1} + \frac{\ell_t}{\chi_t + r}\Sigma_{t-1} y_t \boldsymbol{x}_t\right) \\
&\quad - \boldsymbol{\mu}_{t-1}^\top \Sigma_{t-1}^{-1}\boldsymbol{\mu}_{t-1} \\
&= \frac{1}{r}\left(\boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t y_t\right)^2 + \frac{\ell_t^2}{(\chi_t + r)^2}\boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t + \frac{\ell_t^2}{r(\chi_t + r)^2}\left(\boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t\right)^2 \\
&\quad + 2\frac{\ell_t}{\chi_t + r} y_t \boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t + 2\frac{\ell_t}{r(\chi_t + r)}\left(\boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t\right)\left(y_t \boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t\right) \\
&= \frac{1}{r}\left(1 - 1 + \boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t y_t\right)^2 + \frac{\ell_t^2}{(\chi_t + r)^2}\chi_t + \frac{\ell_t^2}{r(\chi_t + r)^2}\chi_t^2 + 2\frac{\ell_t}{\chi_t + r}\left(y_t \boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t - 1 + 1\right) \\
&\quad + 2\frac{\ell_t}{r(\chi_t + r)}\chi_t\left(1 - 1 + y_t \boldsymbol{\mu}_{t-1}^\top \boldsymbol{x}_t\right) \\
&= \frac{1}{r}(1 - \ell_t)^2 + \frac{\ell_t^2}{(\chi_t + r)^2}\chi_t + \frac{\ell_t^2}{r(\chi_t + r)^2}\chi_t^2 + 2\frac{\ell_t}{\chi_t + r}(1 - \ell_t) + 2\frac{\ell_t}{r(\chi_t + r)}\chi_t(1 - \ell_t) \\
&= \frac{1}{r}(1 - \ell_t)^2 + \frac{\chi_t \ell_t^2}{r(\chi_t + r)^2}(r + \chi_t) + 2\frac{\ell_t(1 - \ell_t)}{r(\chi_t + r)}(r + \chi_t) \\
&= \frac{1}{r}\left(1 - \ell_t^2\right) + \frac{\chi_t \ell_t^2}{r(\chi_t + r)} \\
&= \frac{\chi_t \ell_t^2 + \chi_t + r - \ell_t^2 \chi_t - \ell_t^2 r}{r(\chi_t + r)} \\
&= \frac{\chi_t + r - \ell_t^2 r}{r(\chi_t + r)}.
\end{aligned}
$$

References

Bekkerman, R., McCallum, A., & Huang, G. (2004). *Automatic categorization of email into folders: benchmark experiments on Enron and SRI corpora* (Technical Report IR 418:1). Center for Intelligent Information Retrieval.

Bernal, A., Crammer, K., Hatzigeorgiou, A., & Pereira, F. (2007). Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Computational Biology*, *3*(3), e54.

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *ACL*.

Bordes, A., & Bottou, L. (2005). The huller: a simple and efficient online svm. In *LNAI: Vol. 3720. European conference on machine learning (ECML)*.

Carvalho, V. R., & Cohen, W. W. (2006). Single-pass online learning: performance, voting schemes and online feature selection. In *KDD-2006*.

Censor, Y., & Zenios, S. (1997). *Parallel optimization: theory, algorithms, and applications*. New York: Oxford University Press.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the Association for Computing Machinery*, *44*(3), 427–485.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2005). A second-order perceptron algorithm. *SIAM Journal on Computing*, *34*.

Chechik, G., Sharma, V., Shalit, U., & Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, *11*, 1109–1135.

Chiang, D., Marton, Y., & Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 224–233). Stroudsburg: Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on empirical methods in natural language processing* (Vol. 10, pp. 1–8). Stroudsburg: Association for Computational Linguistics.

Crammer, K. (2010). Efficient online learning with individual learning-rates for phoneme sequence recognition. In *IEEE int. conf. on acoustics, speech, and signal processing (ICASSP)*.

Crammer, K., & Lee, D. D. (2010). Learning via Gaussian herding. In *Advances in neural information processing systems 24*.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, *3*, 951–991.

Crammer, K., Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2003). Online passive aggressive algorithms. In *Advances in neural information processing systems 16*.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Crammer, K., Dredze, M., & Pereira, F. (2008). Exact convex confidence-weighted learning. In *Neural information processing systems (NIPS)*.

Crammer, K., Dredze, M., & Kulesza, A. (2009a). Multi-class confidence weighted algorithms. In *Empirical methods in natural language processing (EMNLP)*.

Crammer, K., Kulesza, A., & Dredze, M. (2009b). Adaptive regularization of weight vectors. In *Advances in neural information processing systems 23* (pp. 414–422).

Crammer, K., Mohri, M., & Pereira, F. (2009c). Gaussian margin machines. In *Proceedings of the twelfth intentional conference on artificial intelligence and statistics (AISTATS)*.

Crammer, K., Dredze, M., & Pereira, F. (2012). Confidence-weighted linear classification for text categorization. *Journal of Machine Learning Research*.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-weighted linear classification. In *International conference on machine learning*.

Duchi, J. C., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

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

Frome, A., Singer, Y., Sha, F., & Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE 11th international conference on computer vision*.

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

Ha-Thuc, V., & Cancedda, N. (2011). Confidence-weighted learning of factored discriminative language models. In *Association for computational linguistics (ACL)*.

Harrington, E., Herbrich, R., Kivinen, J., Platt, J., & Williamson, R. (2003). Online Bayes point machines. In *7th pacific-Asia conference on knowledge discovery and data mining (PAKDD)*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: data mining, inference, and prediction*. Berlin: Springer.

Haykin, S. (1996). *Adaptive filter theory*. New York: Prentice Hall.

Hazan, E. (2006). *Efficient algorithms for online convex optimization and their applications*. PhD thesis, Princeton University.

Herbrich, R., Graepel, T., & Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, *1*, 245–279.

Jaakkola, T., Meila, M., & Jebara, T. (1999). Maximum entropy discrimination.

Jie, L., Orabona, F., & Caputo, B. (2010). An online framework for learning novel concepts over multiple cues. In H. Zha, R. Taniguchi, & S. Maybank (Eds.), *Lecture notes in computer science: Vol. 5994. Computer vision – ACCV 2009* (pp. 269–280). Berlin: Springer. doi:10.1007/978-3-642-12307-8_25.

Khardon, R., & Wachman, G. (2007). Noise tolerant variants of the perceptron algorithm. *Journal of Machine Learning Research*, *8*, 227–248.

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

Klimt, B., & Yang, Y. (2004). The enron corpus: a new dataset for email classification research. In *Machine learning: ECML 2004* (pp. 217–226).

Krogh, A. (1992). Learning with noise in a linear perceptron. *Journal of Physics. A, Mathematical and General*, *25*, 1119.

Krogh, A., & Hertz, J. (1992). Generalization in a linear perceptron in the presence of noise. *Journal of Physics. A, Mathematical and General*, *25*, 1135.

Le, A., Markopoulou, A., & Faloutsos, M. (2010). *Phishdef: Url names say it all*. Arxiv preprint arXiv:1009.2275.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Littlestone, N. (1988). Learning when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*, 212–261.

Ma, J., Kulesza, A., Crammer, K., Dredze, M., Saul, L., & Pereira, F. (2010). Exploiting feature covariance in high-dimensional online learning. In: *AIStats*.

McDonald, R., Crammer, K., & Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 91–98). Stroudsburg: Association for Computational Linguistics.

McMahan, H. B., & Streeter, M. (2010). Adaptive bound optimization for online convex optimization. In *Proceedings of the twenty third annual conference on learning theory*.

Mejer, A., & Crammer, K. (2010). Confidence in structured-prediction using confidence-weighted models. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 971–981). EMNLP'10. Stroudsburg: Association for Computational Linguistics. http://portal.acm.org/citation.cfm?id=1870658.1870753.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. A*, *209*, 415–446.

Minka, T. P., Xiang, R., & Qi, Y. A. (2009). Virtual vector machine for Bayesian online classification. In *Proceedings of the twenty fifth conference on uncertainty in artificial intelligence*.

Orabona, F., & Crammer, K. (2010). New adaptive algorithms for online classification. In *Advances in neural information processing systems 24*.

Platt, J. C. (1998). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In P. Bartlett, B. Schölkopf, D. Schuurmans, & A. J. Smola (Eds.), *Advances in large margin classifiers*. Cambridge: MIT Press.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge: MIT Press.

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

Saha, A., Daume, H. III, & Venkatasubramanian, S. (2011). Online learning of multiple tasks and their relationships. In *AISTATS*.

Sandhaus, E. (2008). *The New York Times annotated corpus*. Philadelphia: Linguistic Data Consortium.

Shivaswamy, P., & Jebara, T. (2007). Ellipsoidal kernel machines. In *Artificial intelligence and statistics (AISTATS)*.

Shivaswamy, P., & Jebara, T. (2010a). Empirical Bernstein boosting. In Y. Teh & M. Titterington (Eds.), *JMLR: Vol. 9. Proceedings of the thirteenth international conference on artificial intelligence and statistics (AISTATS) 2010* (pp. 733–740). W&CP.

Shivaswamy, P. K., & Jebara, T. (2010b). Maximum relative margin and data-dependent regularization. *Journal of Machine Learning Research*, *11*, 747–788.

Sutton, R. S. (1992). Adapting bias by gradient descent: an incremental version of delta-bar-delta. In *Proceedings of the tenth national conference on artificial intelligence* (pp. 171–176). Cambridge: MIT Press.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Vaits, N., & Crammer, K. (2011). Re-adapting the regularization of weights for non-stationary regression. In *The 22nd international conference on algorithmic learning theory*. ALT'11.