
Multiclass Classification with Bandit Feedback using Adaptive Regularization

Koby Crammer

The Technion, Department of Electrical Engineering, Haifa, 32000, Israel

KOBY@EE.TECHNION.AC.IL

Claudio Gentile

Universita' dell'Insubria, DICOM, Via Mazzini 5, 21100 - Varese, Italy

CLAUDIO.GENTILE@UNINSUBRIA.IT

Abstract

We present a new multiclass algorithm in the bandit framework, where after making a prediction, the learning algorithm receives only partial feedback, i.e., a single bit of right-or-wrong, rather than the true label. Our algorithm is based on the 2nd-order Perceptron, and uses upper-confidence bounds to trade off exploration and exploitation. We analyze this algorithm in a partial adversarial setting, where instances are chosen adversarially, while the labels are chosen according to a linear probabilistic model, which is also chosen adversarially. We show a regret of $\mathcal{O}(\sqrt{T} \log T)$, which improves over the current best bounds of $\mathcal{O}(T^{2/3})$ in the fully adversarial setting. We evaluate our algorithm on nine real-world text classification problems, obtaining state-of-the-art results, even compared with non-bandit online algorithms, especially when label noise is introduced.

1. Introduction

Consider a book recommendation system. Given a customer's profile, it recommends a few possible books to the user, with the aim of choosing books that the user will like and eventually purchase. Typical feedback in such a system is the actual action of the user, or specifically what books she bought, if any. The system cannot observe what would have been the user's actions had other books got recommended.

Generally, such problems are referred to as learning with partial feedback. Unlike the full information case,

where the system or the learning algorithm knows the outcome of each possible response, e.g., the user's action for each and every possible book recommended, in the partial setting, the system observes the response only to very limited options and, in particular, the option that was actually recommended.

We consider an instantiation of this problem in the multiclass prediction problem within the online bandit setting. Learning is performed in rounds. On each round the algorithm receives an instance and outputs a label from a finite set of size K . It then receives a single bit indicating whether the predicted label is correct or not, which the algorithm uses to update its internal model and proceed to the next round. Note that for rounds where the feedback indicates wrong prediction, the algorithm's uncertainty about the true label for that instance is almost not reduced, since the number of alternatives is only reduced from K to $K - 1$. Hence the algorithm needs somehow to follow an exploration-exploitation strategy.

Related work. Our algorithm trades off exploration and exploitation via upper-confidence bounds, in a way that is somewhat similar to the work of Auer (2003) and Dani et al. (2008). Yet, the result most closely related to our work is the Banditron algorithm of Kakade et al. (2008), which builds on the immortal Perceptron algorithm. Kakade et al. (2008) investigated this problem in an online adversarial setting, and showed a $\mathcal{O}(T^{2/3})$ bound on the regret compared to the hinge loss of a linear-threshold comparator. Wang et al. (2010) extended their results to a more general potential-based framework for online learning.

Multiclass classification with bandit feedback can be seen as a multi-armed bandit problem with side information. Relevant work within this research thread includes the Greedy-Epoch algorithm analyzed by Langford & Zhang (2007), where a $\mathcal{O}(T^{2/3})$ regret bound

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

has been proven under i.i.d. assumptions, yet covering more general learning tasks than ours. We are aware of at least three more papers that define multi-armed bandit problems with side information, also called bandits with covariates: (Wang et al., 2005; Lu et al., 2010; Rigollet & Zeevi, 2010). However, the models in these paper are very different from ours, and not easily adapted to our multiclass problem. Another paper somewhat related to this work is (Walsh et al., 2009), where the authors adopt a similar linear model as ours (and similar mathematical tools) in a setting where an online prediction algorithm is allowed to sometimes answer "I don't know" (the so-called KWIK setting). A direct adaptation of their results to our setting seems to lead to inferior cumulative regret bounds.

2. Multiclass Bandit Online Learning

Standard online learning with *full* information is performed in rounds. On round t the algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$ to be classified, and predicts a label $\hat{y}_t \in \{1 \dots K\}$. It receives the true label y_t , updates its internal model, and is ready for the next round.

The algorithm we present employs linear models. The algorithm maintains K weight vectors $\mathbf{w}_i \in \mathbb{R}^d$, for $i = 1, \dots, K$. Given an instance \mathbf{x}_t , the algorithm computes a score associated with each of the K classes, defined by $\mathbf{w}_i^\top \mathbf{x}_t$, and outputs a *prediction* to be the label with the highest score, that is,

$$\tilde{y}_t = \arg \max_{i=1 \dots K} \mathbf{w}_i^\top \mathbf{x}_t. \quad (1)$$

We emphasize that two quantities are considered: the label with maximal score \tilde{y}_t defined in Eq. (1) (this quantity is internal to the algorithm), and the label that is actually output by the algorithm, denoted by \hat{y}_t . In the full-information case, most algorithms just output their prediction, that is we have $\hat{y}_t = \tilde{y}_t$. In this paper, we focus on the partial information setting, aka the *bandit* setting. Here, after the algorithm makes a prediction, it does not receive the correct label y_t but only a single bit M_t indicating whether its output \hat{y}_t was correct or not, i.e., $M_t = \{y_t \neq \hat{y}_t\}$, where $\{A\}$ is 1 iff the predicate A is true.

Since learning algorithms receive only very limited feedback there is a natural tradeoff between *exploration* and *exploitation*. On the one hand, the algorithm should output the best scoring label $\tilde{y}_t = \arg \max_{i=1 \dots K} \mathbf{w}_i^\top \mathbf{x}_t$, this step being called exploitation. Yet, it may be the case that the model used at some point will not perform well (for example, the initial model), and thus the algorithm will make many mistakes and most of its feedback will indicate that the output is not correct, that is $M_t = 1$. This feedback is

almost useless, as there is still uncertainty about the true label (one of $K - 1$ options remain). On the other hand, the algorithm may perform *exploration* and output another label, to get useful feedback.

The Banditron algorithm implements one approach to exploration-exploitation tradeoff. From time to time, the algorithm outputs another label than its prediction \tilde{y}_t . The banditron chooses such examples at random with some probability γ , and then chooses a random label uniformly with probability $1/K$. This approach ignores few aspects of the state. First, it ignores the specific input \mathbf{x}_t to be labeled, although some inputs may be classified well by the current model. Second, it ignores the difference in score values $\mathbf{w}_1^\top \mathbf{x}_t, \dots, \mathbf{w}_K^\top \mathbf{x}_t$. For example, it may be the case that two of the score values are very large compared to the others; then it makes sense to output only one of the two, rather than sampling from the entire label set. Third, it ignores the example index t , as it is reasonable to assume that as the algorithm learns more it needs to query less.

An alternative approach, which we employ in this work, is to maintain additional confidence information about the predictions. Specifically, given an input \mathbf{x}_t , the algorithm not only computes score values, but also non-negative *uncertainty* values for these scores, denoted by $\epsilon_{i,t}$. Intuitively, high values of $\epsilon_{i,t}$ indicate that the algorithm is less confident in the value of the score $\mathbf{w}_i^\top \mathbf{x}_t$. Given a new example, the algorithm outputs the label with the highest *upper confidence bound* (UCB), computed as the sum of score and uncertainty, $\hat{y}_t = \arg \max_i (\mathbf{w}_i^\top \mathbf{x}_t + \epsilon_{i,t})$. Intuitively, a label \hat{y}_t is output by the algorithm if either its score is high or the uncertainty in predicting it is high, and there is need to obtain information about it. Specifically, our algorithm maintains a positive semidefinite matrix per label, $A_{i,t} \in \mathbb{R}^{d \times d}$. Given an input instance \mathbf{x}_t to be classified, we define the confidence intervals to be $\epsilon_{i,t}^2 = \eta_t \mathbf{x}_t^\top A_{i,t}^{-1} \mathbf{x}_t$ for some scalar η_t which is used to tradeoff the exploration and exploitation. The matrices $A_{i,t}$ (or their inverses) are used to measure uncertainty in the score, and input examples are used to update them as well as the \mathbf{w}_i 's.

We now describe the specific model we use to motivate our algorithm, and later analyze it. Our setting is slightly less adversarial than the one considered in Kakade et al. (2008); Wang et al. (2010). In particular, we assume the following parametric model for the multiclass labels:¹ We assume that the labels of an example \mathbf{x}_t are generated according to the following

¹This model is a natural extension of the binary label noise model considered elsewhere (Cesa-Bianchi et al., 2009; Dekel et al., 2010).

probabilistic model,

$$\mathbb{P}(y_t = i | \mathbf{x}_t) = \frac{\alpha + \mathbf{u}_i^\top \mathbf{x}_t}{\alpha + 1}, \quad (2)$$

for some K vectors $\mathbf{u}_1, \dots, \mathbf{u}_K \in \mathbb{R}^d$, and a scalar $\alpha \in (-1, 1]$. The model is well defined if, for all $\mathbf{x} \in \mathbb{R}^d$ chosen by the adversary, we have $\sum_{i=1}^K \mathbf{u}_i^\top \mathbf{x} = \alpha + 1 - K\alpha$ and $-\alpha \leq \mathbf{u}_i^\top \mathbf{x}$ for all i . (Notice that this implies $\mathbf{u}_i^\top \mathbf{x} \leq 1$ for all i). For simplicity we also assume $\|\mathbf{x}_t\| = 1$ for all t . Intuitively, α quantifies the closeness of the K tasks according to model (2). On the one hand, the closer α gets to -1 the more the scores $\mathbf{u}_i^\top \mathbf{x}$ are forced to be close to each other (i.e., $\mathbf{u}_i^\top \mathbf{x} \approx 1$ for all i , independent of \mathbf{x}). On the other hand, setting $\alpha = 1$ yields $\mathbb{P}(y_t = i | \mathbf{x}_t) = \frac{1 + \mathbf{u}_i^\top \mathbf{x}_t}{2}$ under the constraints $-1 \leq \mathbf{u}_i^\top \mathbf{x} \leq 1$ and $\sum_{i=1}^K \mathbf{u}_i^\top \mathbf{x} = 2 - K$. This choice allows the probability mass to be concentrated on the i -th label by setting $\mathbf{u}_i^\top \mathbf{x} = 1$ for some i and $\mathbf{u}_i^\top \mathbf{x} = -1$ otherwise. Two natural choices of α are $\alpha = \frac{1}{K-1}$ and $\alpha = 0$. The former yields $\mathbb{P}(y_t = i | \mathbf{x}_t) = (1 + \mathbf{u}_i^\top \mathbf{x}_t(K-1))/K$ under the constraints $-1/(K-1) \leq \mathbf{u}_i^\top \mathbf{x} \leq 1$ and $\sum_{i=1}^K \mathbf{u}_i^\top \mathbf{x} = 0$, which can be satisfied by a constraint that is *independent* of \mathbf{x}_t , namely, $\sum_{i=1}^K \mathbf{u}_i = 0$. The latter choice $\alpha = 0$ forces $(\mathbf{u}_1^\top \mathbf{x}, \dots, \mathbf{u}_K^\top \mathbf{x})$ to be a probability vector. For the sake of our analysis (Section 4), we will restrict our attention to the case $\alpha \geq 0$.

We will bound the extent to which the number of prediction mistakes of our learning algorithms exceeds the number of prediction mistakes of the Bayes optimal predictor $b_t = b(\mathbf{x}_t) = \operatorname{argmax}_{i=1, \dots, K} (\mathbf{u}_i^\top \mathbf{x}_t)$ for this label noise model. In particular, we are aimed at bounding from above the cumulative regret

$$R_T \equiv \sum_{t=1}^T \left(\mathbb{P}_t(y_t \neq \hat{y}_t) - \mathbb{P}_t(y_t \neq b(\mathbf{x}_t)) \right)$$

with high probability over past y 's, possibly taking into account the internal randomization of the algorithms. In the above, \mathbb{P}_t denotes the conditional probability $\mathbb{P}(\cdot | \mathbf{x}_1, \dots, \mathbf{x}_t, y_1, \dots, y_{t-1}, \sigma_{t-1})$, where it is understood that \mathbf{x}_t can also be chosen adversarially as a function of past \mathbf{x} and y , and σ_{t-1} is the (possible) internal randomization of the algorithm under consideration up to round $t-1$. Similarly, we denote by $E_t[\cdot]$ the conditional expectation $E_t[\cdot | \mathbf{x}_1, \dots, \mathbf{x}_t, y_1, \dots, y_{t-1}, \sigma_{t-1}]$.

Our algorithm is a variant of the multiclass second-order Perceptron algorithm that maintains at time t a set of K vectors $\mathbf{w}_{1,t}, \dots, \mathbf{w}_{K,t} \in \mathbb{R}^d$, where $\hat{\Delta}_{i,t} = \mathbf{w}_{i,t}^\top \mathbf{x}_t$ is intended to approximate $\Delta_{i,t} = \mathbf{u}_i^\top \mathbf{x}_t$ for all i and t . The bandit algorithm also maintains a set of K matrices $A_{i,t}$, which are used to compute a

Parameter: $\alpha \in (-1, 1]$
Initialization: $A_0 = (1 + \alpha)^2 I \in \mathbb{R}^{dK \times dK}$,
 $W_0 = (\mathbf{w}_{1,0}, \mathbf{w}_{2,0}, \dots, \mathbf{w}_{K,0}) = 0 \in \mathbb{R}^{dK}$
For $t = 1, 2, \dots, T$:

1. Get instance $\mathbf{x}_t \in \mathbb{R}^d : \|\mathbf{x}_t\| = 1$
2. $W'_{t-1} = \operatorname{argmin}_{W \in C_t} d_{t-1}(W, W_{t-1})$, where C_t is defined in Eq. (4)
3. Set $\hat{\Delta}'_{i,t} = \mathbf{x}_t^\top \mathbf{w}'_{i,t-1}$, $i = 1, \dots, K$
4. Output $\hat{y}_t = \operatorname{argmax}_i (\hat{\Delta}'_{i,t} + \epsilon_{i,t})$, where $\epsilon_{i,t}^2$ is as in Eq. (5)
5. Get feedback $M_t = \{y_t \neq \hat{y}_t\}$
6. If $M_t = 1$ then:
 - 6a. with prob. $(1 - \alpha)/2$ set
 $X_t = (0, \dots, 0, \mathbf{x}_t, 0, \dots, 0)$
 - 6b. with prob. $(1 + \alpha)/2$ set
 $X_t = (0, \dots, 0, -\mathbf{x}_t, 0, \dots, 0)$
7. Else ($M_t = 0$) set
 $X_t = (0, \dots, 0, \mathbf{x}_t, \dots, 0)$
8. Update:

$$A_t = A_{t-1} + X_t X_t^\top,$$

$$W_t = A_t^{-1} (A_{t-1} W'_{t-1} + X_t).$$

Figure 1. The multiclass bandit algorithm.

standard upper confidence scheme of the form

$$\hat{y}_t = \operatorname{argmax}_{i=1 \dots K} \left(\hat{\Delta}_{i,t} + \epsilon_{i,t} \right), \quad (3)$$

where $\epsilon_{i,t}$ is a suitable upper confidence level for class i at time t , which is a function of both \mathbf{x}_t and $A_{i,t}$.

3. The New Bandit Algorithm

Our algorithm, described in Figure 1, is parameterized by the model parameter $\alpha \in (-1, 1]$, assumed to be known. The algorithm maintains, for each class $i = 1, \dots, K$, a weight vector $\mathbf{w}_i \in \mathbb{R}^d$ and a correlation matrix $A_i \in \mathbb{R}^{d \times d}$, and operates similarly to 2nd-order (or ridge regression)-like algorithms (Hoerl & Kennard, 1970; Azoury & Warmuth, 2001; Cesa-Bianchi et al., 2005) (see also, e.g., (Strehl & Littman, 2008; Crammer et al., 2009a; Cesa-Bianchi et al., 2009; Dekel et al., 2010)). The weight vectors are initialized to zero, and the matrices A_i to $(1 + \alpha)^2$ times the identity matrix I of size d . For brevity, we denote by A a single matrix of size $dK \times dK$ defined to be the block-diagonal matrix $A = \operatorname{diag}(A_1, A_2, \dots, A_K)$. We denote by W the dK -dimensional vector which is defined to be the concatenation of the K vectors \mathbf{w}_i . Similarly, $U \in \mathbb{R}^{dK}$ is the concatenation of the K vectors \mathbf{u}_i defined in (2). We use both notations, each in turn to simplify the presentation in place.

Our algorithm works in rounds. On round t , the algorithm receives the (normalized) instance $\mathbf{x}_t \in \mathbb{R}^d$ and

defines the following time- t convex set

$$C_t = \left\{ W = (\mathbf{w}_1, \dots, \mathbf{w}_K) \in \mathbb{R}^{dK} : -\alpha \leq \mathbf{w}_i^\top \mathbf{x}_t \right. \\ \left. i = 1, \dots, K ; \sum_{i=1}^K \mathbf{w}_i^\top \mathbf{x}_t = 1 + \alpha - K\alpha \right\} \quad (4)$$

We remind the reader that for each t , set C_t includes the parameter space where vectors \mathbf{u}_i are assumed to live (see text surrounding Eq. (2)). The algorithm then projects the current vector W_{t-1} onto C_t yielding $W'_{t-1} = (\mathbf{w}'_{1,t-1}, \dots, \mathbf{w}'_{K,t-1})$. The projection is performed using the multiclass Mahalanobis distance $d_{t-1}(U, W) = \frac{1}{2}(U - W)^\top A_{t-1}(U - W)$. This projection can be computed efficiently in time $\mathcal{O}(K \log K)$, the details are omitted due to lack of space.

The algorithm uses $\mathbf{w}'_{i,t-1}$ to estimate the score-values $\hat{\Delta}'_{i,t} = \mathbf{x}_t^\top \mathbf{w}'_{i,t-1}$, and the upper confidence prediction $\hat{y}_t = \arg \max_i (\hat{\Delta}'_{i,t} + \epsilon_{i,t})$ is output. Upon receiving the binary feedback M_t , the algorithm performs either a deterministic or a randomized update, depending on M_t . Specifically, if a mistake has been made ($M_t = 1$) the algorithm flips a coin with bias $\frac{1+\alpha}{2}$ and sets the vector $X_t = (0, \dots, 0, \pm \mathbf{x}_t, 0, \dots, 0)$ accordingly. On the other hand, if no mistake is made ($M_t = 0$), the associated update vector is $X_t = (0, \dots, 0, \mathbf{x}_t, 0, \dots, 0)$, independent of α . In all cases, the nonzero block of X_t is in position \hat{y}_t , i.e., only the \hat{y}_t 's predictor gets directly affected by X_t . The constructed update vectors are used within a standard 2nd-order updating scheme: matrix A_{t-1} undergoes a rank-one update $A_t \leftarrow A_{t-1} + X_t X_t^\top$, and vector W'_{t-1} turns to W_t via an additive update $A_t W_t \leftarrow A_{t-1} W'_{t-1} + X_t$. We call this algorithm **Confidit**, for (upper) confidence based bandit algorithm.

The construction of the update vector X_t essentially determines the algorithm's behavior: The updating sign $\beta_t = \pm 1$ of \mathbf{x}_t within X_t acts either as a *promoter* for class \hat{y}_t or a *demoter*, depending on the sign of β_t . First, observe that, if the algorithm makes a mistake, then β_t is on average equal to $-\alpha$, i.e., $E_t[\beta_t | M_t = 1] = -\alpha$. Hence on a mistaken trial we demote (the mistaken) class \hat{y}_t only if α is positive. On the contrary, if α is negative the algorithm deems all class predictors $\Delta_{i,t}$ to be very close to each other, hence promoting one class is somewhat similar to promoting all the other ones (recall that the projection step forces them to stay very close anyway). Second, it is worth observing that, conditioning only on the past, and setting $p_t = \mathbb{P}_t(M_t = 0) = \frac{\alpha + \Delta_{\hat{y}_t, t}}{1 + \alpha}$, we have $E_t[\beta_t] = E_t[\beta_t | M_t = 1](1 - p_t) + E_t[\beta_t | M_t = 0]p_t = \Delta_{\hat{y}_t, t}$. Hence this expectation is positive if and only if $\Delta_{\hat{y}_t, t} > 0$. One way of stating this is that on any given time step (mistaken or not), $\hat{\Delta}_{\hat{y}_t, t}$ progresses through the updates in Step 8 of the algorithm towards $\Delta_{\hat{y}_t, t}$

by growing more positive or more negative depending on the sign of $\Delta_{\hat{y}_t, t}$, at an average pace of $|\Delta_{\hat{y}_t, t}|$.

The above behavior is similar to the upper-confidence algorithms under bandit feedback of Auer (2003); Dani et al. (2008) for multiarmed bandits, where our update sign β_t plays the role there of a random observation whose (conditional) average is the average payoff $\Delta_{\hat{y}_t, t}$ of the chosen arm. The randomization in the algorithm serves just to "symmetrize" the unbalanced feedback received in this online protocol, to gather information about the true margin $\Delta_{\hat{y}_t, t}$ of the chosen class.

We note in passing that the running time of each round of the algorithm is dominated by the inversion of the matrix $A_{\hat{y}_t, t-1}$, which is $\mathcal{O}(d^2)$ if done incrementally. Moreover, it is easy to see that the algorithm can also be run in dual variables (i.e., in a RKHS). This has a twofold implication: (a) The resulting noise model (2) can be made highly nonlinear in the input space, and (b) the running time per round can be made quadratic in the number of rounds so far, rather than d^2 . In practice, and also in the experiments described in Section 5, we actually used a version of the algorithm which maintains a (*fully*) *diagonal* matrix A instead of a block-diagonal one. All the steps remain the same except step 8 of Alg. 1 where we define the r th diagonal element of the matrix to be $(A_t)_{r,r} = (A_{t-1})_{r,r} + (X_t)_r^2$. The running time is now $\mathcal{O}(Kd + K \log(K))$, as all the operations are linear in the dimensions of X , W and A , except the projection.

4. Analysis

Our analysis, omitted from this version of the paper, allows us to set the upper confidence level $\epsilon_{i,t}$ in Algorithm 1 to be

$$\epsilon_{i,t}^2 = 2\mathbf{x}_t^\top A_{i,t-1}^{-1} \mathbf{x}_t \left(\frac{1}{2}(1 + \alpha)^2 \|U\|_2^2 \right. \\ \left. + \frac{(1 + \alpha)^2}{2} \sum_{s=1}^{t-1} \mathbf{x}_s^\top A_{\hat{y}_s, s}^{-1} \mathbf{x}_s + 9(1 + \alpha)^2 \log \frac{t+4}{\delta} \right), \quad (5)$$

where $\|U\|_2$ is either the actual Euclidean length of vector U or a (known) upper bound thereof. The following is the main theoretical result of this paper, where we emphasize both the data and the time-dependent aspects of the bound.

Theorem 1. *In the adversarial setting described so far with $\alpha \in [0, 1]$, the cumulative regret R_T of Algorithm 1 satisfies*

$$R_T = O\left(\sqrt{B_1 T} \left(\sqrt{B_2 B_3} + B_3\right)\right),$$

where

$$B_1 = 1 + (1 + \alpha)^{-2}, \quad B_2 = \|U\|_2^2 + 18 \log((T + 4)/\delta),$$

$$B_3 = \sum_{i=1}^K \log \frac{|A_{i,T}|}{|A_{i,0}|} \leq dK \log \left(1 + \frac{T}{dK(1 + \alpha)^2} \right)$$

with probability at least $1 - \delta$ uniformly over the time horizon T . In the above $|\cdot|$ denotes the determinant of the matrix at argument.

The bound described in Theorem 1 is essentially a $\sqrt{T} \log T$ regret bound, as both B_2 and B_3 are logarithmic in T . The best previous bound for multiclass prediction in an adversarial bandit setting is $O(T^{2/3})$, which was first shown for the Banditron algorithm in the case when also the labels are adversarial, rather than being stochastically generated.

Low-Noise Assumptions: By making further assumptions on the distribution of the \mathbf{x}_t , such as low noise, one can improve the $\sqrt{T} \log T$ bound and interpolate between $\log^2 T$ (hard-margin separation assumption) and $\sqrt{T} \log T$ (no assumptions on \mathbf{x}_t). In fact, we can proceed by defining the multiclass margin of a label y_t as $\Delta_{b_t,t} - \max_{i \neq b_t} \Delta_{i,t}$. A low-noise assumption places restrictions (lower bounds) on the probability that \mathbf{x}_t is drawn in such a way that this margin is small. For instance, under hard-margin separation assumptions we can easily prove the following logarithmic regret result.

Corollary 1. *In the adversarial setting described so far with $\alpha \in [0, 1]$, if there is an $\epsilon > 0$ such that for any t and any $i \neq b_t$ we have $\Delta_{b_t,t} - \Delta_{i,t} > \epsilon$, then the cumulative regret R_T of Algorithm 1 satisfies*

$$R_T = O\left(\frac{B_1}{\epsilon} \left(B_2 B_3 + B_3^2\right)\right), \quad \text{for } B_1 = 1 + \alpha + \frac{1}{1 + \alpha}$$

where B_2 and B_3 are as in Theorem 1.

This result is essentially a $\log^2 T$ regret bound, which is a log-factor worse than the one achieved by the random projection-based algorithm described by Kakade et al. (2008), working in the linearly separable case. However, unlike ours, their algorithm does not seem to lend itself to an efficient implementation.

5. Experiments

We evaluate our algorithm using five natural language classification tasks over nine datasets, with various size and number of labels. Their characteristics are summarized in Table 1. Below we summarize their main properties, more information can be found in (Cramer et al., 2009b), where they have previously been used.

Table 1. A summary of the nine datasets, including the number of instances, features and labels and whether the number of examples in each class are balanced.

Task	Instances	Features	Labels	Balanced
20 News	18,828	252,115	20	Y
Amazon7	13,580	686,724	7	Y
Amazon3	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	685,071	268,170	4	N

Data: We use two datasets based on Amazon product reviews, which are used for domain classification datasets from seven product types (*Amazon7*) (books, dvds, music, apparel, electronics, kitchen, video) and a smaller subset with reviews on the first three product types. The 20 Newsgroups² is a very popular dataset with about 20,000 newsgroup messages, divided into 20 different newsgroups. Two additional datasets are based on the automatic classification of Enron emails into one of the 10 largest folders³. Two users are used, denoted by *Enron A* and *Enron B*. The NY Times (Sandhaus, 2008) dataset contains 1.8 million articles, published across 20 years starting from 1987. We used a subset with three annotations for each article: 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*). Finally, we also used documents from RCV1v2 dataset, based on newsfeeds from Reuters. We performed topic classification with the four general topics: corporate, economic, government, and markets. In all datasets, we followed the experimental setting described by Cramer et al. (2009b), including data preprocessing.

Algorithms: Five algorithms are evaluated: two of them work in the bandit setting, the other three in the full information setting. The two bandit algorithms are the Banditron algorithm (Kakade et al., 2008), and the following modification to our algorithm. First, as is typical of many upper confidence-based algorithms, the width of the confidence interval is a pessimistic overestimation of the actual uncertainty, which suggests that implementing our algorithm and testing it on *real* data in the exact form given by the theory may not work well in practice. Hence, we replaced the multiplier of $\mathbf{x}_t^\top A_{i,t-1}^{-1} \mathbf{x}_t$ in the definition of $\epsilon_{i,t}^2$ (see Eq. (5)) with some constant η whose value was set by cross validation, that is, we used⁴

²<http://people.csail.mit.edu/jrennie/20Newsgroups/>

³<http://www.cs.cmu.edu/enron/>

⁴On top of this, observe that the multiplier in (5) is dependent on the norm $\|U\|$ of the comparison classifier,

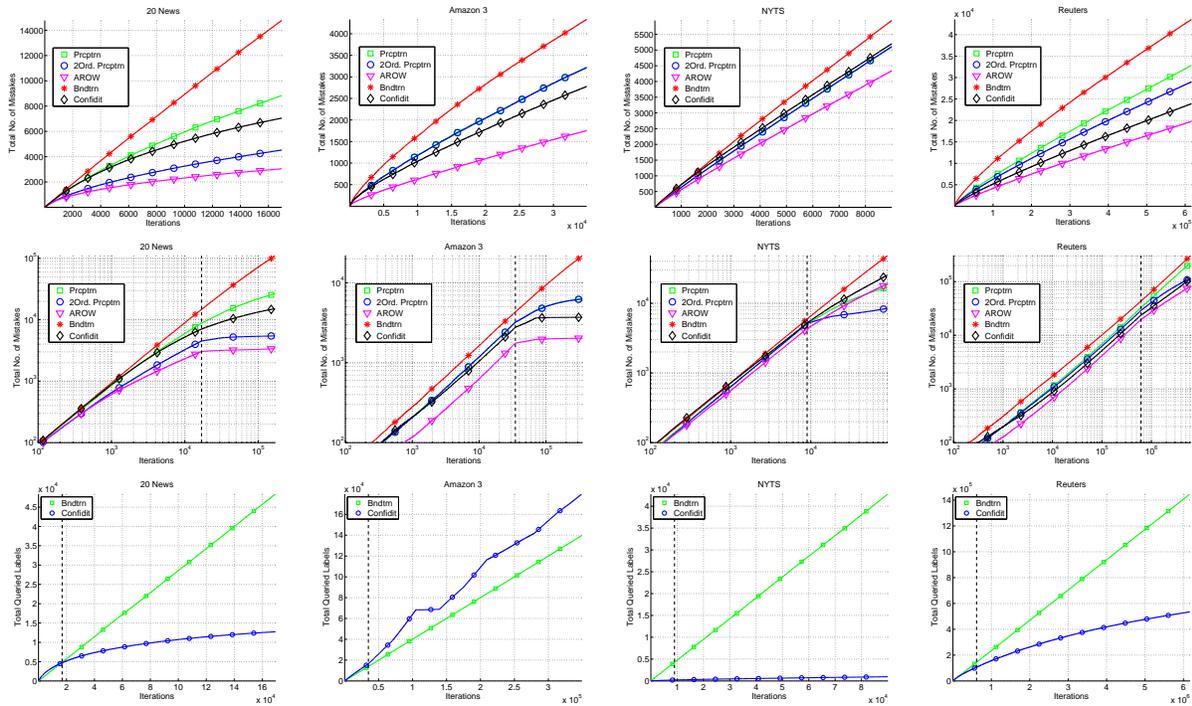


Figure 2. Cumulative no. of prediction mistakes in the first training epoch (top; linear-linear scale), and in all 10 epochs (middle; log-log scale), no label noise. Bottom: Cumulative number of examples for which a prediction \hat{y} of a bandit algorithm is *not* the label output \tilde{y} . Four datasets are used (left to right): 20 Newgroups, Amazon3, NYTS and Reuters.

$\epsilon_{i,t}^2 = \eta \mathbf{x}_t^\top A_{i,t-1}^{-1} \mathbf{x}_t$. Second, the projection step in Algorithm 1 is only needed for technical purposes in the analysis. On preliminary experiments (not reported in this paper) we observed that the actual length of the vectors did not grow large, thereby making the projection immaterial. Thus, we decided to remove this step from our implementation. Third, for computational efficiency reasons, the inverse matrices $A_{i,t-1}^{-1}$ have been replaced by diagonal versions thereof.⁵ The three algorithms that work in the full information setting are the classical (multiclass) Perceptron algorithm (Rosenblatt, 1958), a diagonal multiclass version of the 2nd-order perceptron algorithm (Cesa-Bianchi et al., 2005), and AROW (Crammer et al., 2009a). Only AROW is margin-based among them. All other algorithms are mistake driven (at most).

We performed 10-fold cross validation for all algorithms. Algorithm’s parameters (γ for Banditron, r for AROW, η for Confidit and a for the 2nd order Perceptron) were tuned using a single split of the data. We set $\alpha = 1$ in Confidit, since optimizing this parameter which need not be known to the algorithm.

⁵Note that the multiplier in (5) is logarithmic in t and thus we evaluated another variant where $\eta_t = \eta \log(t)$ which yielded very similar results to the one we report below – we omit the details due to lack of space.

yielded optimal values very close to 1 anyway, with no significant improvement in the results.

Online Results: Evaluation of all algorithms in the online setting is summarized in Fig. 2. We refer to the best scoring label (prediction) as $\tilde{y}_t = \arg \max_i \mathbf{w}_{i,t-1}^\top \mathbf{x}_t$, and to the one actually output as \hat{y}_t . For Perceptron, 2nd-order Perceptron and AROW, these two are always the same. Yet, bandit algorithms need to explore and thus the prediction \tilde{y}_t and output label \hat{y}_t may not be the same. The top row of Fig. 2 summarizes (in linear scale) the cumulative number of *prediction* mistakes each of the five online algorithms makes during its *first* training epoch over the data. The results for four datasets are shown: (left to right): 20 NG, Amazon3, NYTS and Reuters. AROW makes the least number of mistakes, while Banditron makes the most. There is no clear ordering between the other algorithms. This is surprising since Confidit has only partial information, while the Perceptron and the 2nd-order Perceptron do rely on full label information.

The second row of Fig. 2 summarizes (in log scale) the cumulative number of *prediction* mistakes over *ten* training epochs over the data. The vertical dashed black line indicates the end of the first epoch. In general we observe a similar trend to this behavior after the first epoch. The Banditron makes the same rate of

Table 2. Test error with 10-fold cross validation with no label noise. Bold entries indicate lowest error-rate among the Perceptron, 2nd order Perceptron, Banditron and Confidit algorithms (AROW is not included, since it is margin-based, unlike the others), and underlined entries indicates lowest error-rate between the two bandit algorithms. Additional † and * indicate p-value of 0.01 and 0.001 respectively, when comparing all results, while (†) and (*) only include comparison between the two bandit-based algorithms.

Task	Perc.	2Ord. Perc.	AROW	Bndtr.	Confidit
20 News	20.04	13.92	8.14	50.52	*12.41
Amazon7	25.71	25.73	22.30	29.15	(*) 24.73
Amazon3	5.52	5.45	4.47	7.54	(*) <u>6.67</u>
Enron A	21.39	19.57	16.39	27.35	(*) <u>23.01</u>
Enron B	34.02	33.32	27.11	42.49	(*) <u>37.89</u>
NYTD	22.02	20.95	17.61	31.23	(*) <u>27.02</u>
NYTO	21.84	22.69	17.20	31.64	(*) <u>25.20</u>
NYTS	49.16	51.23	43.05	56.06	(†) <u>54.24</u>
Reuters	4.29	3.96	2.92	4.59	*3.55

Table 3. Same as Table 2, with 30% training label noise.

Task	Perc.	2Ord. Perc.	AROW	Bndtr.	Confidit
20 News	45.88	39.92	21.41	65.71	*30.45
Amazon7	42.33	41.41	27.47	45.28	*36.24
Amazon3	28.70	27.38	6.22	31.70	*17.16
Enron A	46.85	46.09	24.04	49.20	*33.01
Enron B	55.52	54.89	37.17	61.10	*50.57
NYTD	43.14	43.36	24.40	50.74	*34.93
NYTO	44.43	42.40	24.78	48.85	*33.85
NYTS	63.39	63.66	48.99	65.63	*55.84
Reuters	35.34	35.70	4.53	36.61	*11.70

mistakes in later epochs as the first one, yet it is not always the case for the other algorithms. For example, on 20 NG (left plots) all other algorithms make noticeably fewer mistakes in the last 9 training epochs (on average) compared to the first one. The parameters of all algorithms were the ones that minimize the error on held-out-data, which may be far from optimal when evaluating just on the number of online mistakes.

Finally, the bottom row of Fig. 2 summarizes the cumulative number of examples for which the prediction and the output of bandit-based algorithms are not the same. This may be thought of as an *exploration rate*. Clearly, this only applies to Banditron and Confidit. The plots are in linear-linear scale. By definition, Banditron has a *fixed* exploration rate (of γ), as opposed to Confidit whose exploration rate varies as more examples are observed. In seven out of the nine datasets (only 4 are plotted) Confidit has *lower* and monotonically decreasing exploration rate. Confidit has more cumulative exploration rate in two of the datasets: Amazon3 and Amazon7. A possible explanation is that the number of features in these datasets is large (the largest among the nine datasets). Since the exploration rate of Confidit is based on a term which is linear in the square of the features, and there are many rare features that are introduced at a high-rate during training, Confidit gets biased towards exploring classes

Table 4. Same as Table 2, with 10% training label noise.

Task	Perc.	2Ord. Perc.	AROW	Bndtr.	Confidit
20 News	31.27	22.03	13.10	53.04	*17.70
Amazon7	30.96	31.53	23.46	34.74	† 29.49
Amazon3	12.51	12.17	5.42	16.12	*8.90
Enron A	32.52	27.65	19.73	36.85	(*) 27.58
Enron B	41.14	40.64	34.05	46.15	(*) <u>41.27</u>
NYTD	28.46	28.15	19.57	37.60	(*) <u>28.94</u>
NYTO	29.35	27.66	20.14	39.00	(*) <u>27.87</u>
NYTS	53.31	54.55	44.26	58.22	(*) <u>54.82</u>
Reuters	16.79	16.86	3.64	18.92	*5.96

with a high number of relatively rare features.

Batch Evaluation: Table 2 summarizes averaged 10-fold cross-validation error rates. Bold entries indicate best results within all algorithms except AROW (which is margin based), while underlined entries indicate superiority when comparing only the two bandit-based algorithms Banditron and Confidit. It is not surprising that AROW outperforms all other algorithms, as it works in the full information case with a margin-sensitive update rule. (This was shown for the binary case by Crammer et al. (2009a)). Thus we consider below only the Perceptron and 2nd-order Perceptron in the full information setting, and their bandit counterparts: Banditron and Confidit. When comparing the two algorithms working in the full information setting, we see that 2nd-order Perceptron outperforms Perceptron in 6 datasets, and is worse in 2 (there is a tie on one dataset). In fact the 2nd-order Perceptron outperform all other three algorithms in 4 datasets. Confidit performs best among all four algorithms (including full information ones) in three datasets, and outperforms the Banditron in all dataset. All results, except one, are statistically significant with p-value of 0.001.

Label Noise: We repeated the above experiments with artificial label noise injected into the training data. Specifically, we picked examples from the training set with probability p , and replaced the true label of these examples with a uniformly distributed random label. This process was performed only for the training subset of the data, the test results were evaluated using uncorrupted data. However, parameter tuning was performed using only corrupted data. The motivation here is that injecting label noise might be more harmful in the bandit setting than the full information setting. This is because the bandit algorithms not only observe partial information, but when they get some, this information may be incorrect.

We collected online results in a similar way to Fig. 2 (the details are again omitted due to lack of space, and will be given in the full version). Generally speaking, we observe that all algorithms make more mistakes, yet the relative ordering remains the same. More in de-

tail, all algorithms continue to make mistakes after the first epoch at a rate that remains similar throughout time. This is in contrast to the evidence we collected on noiseless data, where the mistake rate curves tend to flatten out in later epochs. Finally, compared to the noise-free case, Confidit has more exploration in some datasets, and comparable in others. This is also due to the different value of exploration-exploitation parameter η automatically set by cross-validation.

Finally, averaged 10-fold cross validation error rates on label noise rates 10% and 30% are summarized in Table 4 and Table 3, respectively. Evaluations are performed using uncorrupted data, the noisy data being used only during training and parameter tuning. There are few trends. First, performance degrades as the level of noise increases. E.g., on the Amazon7 dataset the error rate of Banditron increases from 29.15% via 34.74% via 39.46% to 45.28% as the label noise levels increase in 0%, 10%, 20%, 30%. Second, as the level of noise increases, it seems that Confidit suffers the least compared to Perceptron and 2nd-order Perceptron. E.g., on Amazon3 both Perceptron and 2nd-order Perceptron have about 5.50% error when trained with no noise, while having about 12% error with 10% label noise. Confidit, on the other hand, has 6.57% error with no noise, and about 8.90% with noise. Third, while Confidit performs best (among the 4 non margin-based algorithms) on only 3 datasets out of 9 when no noise is induced, it is the best in 4 datasets at a 10% label noise level, and is best on all 9 datasets for 20% and 30% noise. Hence Confidit seems more resistant to label noise compared to Banditron, as well as to the full information algorithms.

Conclusions and Ongoing Research: We presented the Confidit algorithm combining 2nd-order Perceptron for multiclass problems and upper confidence bounds. We proved a regret of $\mathcal{O}(\sqrt{T} \log T)$ in a partial adversarial setting, which improves on the $T^{2/3}$ bound proven for the Banditron. Experiments indicate the superiority of our algorithm. Our current analysis is not capturing the right dependence on the α parameter, as we expect the bounds to improve as α gets closer to -1 . We are developing a margin-based version of the algorithm, as such algorithms often outperform mistake-driven algorithms in practice. It still remains an open problem to show if it is possible to achieve a regret of \sqrt{T} in the full adversarial setting.

Acknowledgments. This research was done while Claudio Gentile was visiting the Technion. This work was partly supported by the PASCAL2 Network of Excellence under EC grant no. 216886 and partly by the Israeli Science Foundation grant ISF-1567/10. This

publication only reflects the authors' views.

References

- Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3, 2003.
- Azoury, K. S. and Warmuth, M. K. Relative loss bounds for online density estimation with the exponential family of distributions. *Machine Learning*, 43, 2001.
- Cesa-Bianchi, N., Conconi, A., and Gentile, C. A second-order perceptron algorithm. *SIAM Journal on Computing*, 43:640–668, 2005.
- Cesa-Bianchi, N., Gentile, C., and Orabona, F. Robust bounds for classification via selective sampling. In *ICML*, 2009.
- Crammer, K., Kulesza, A., and Dredze, M. Adaptive regularization of weighted vectors. In *NIPS*, 2009a.
- Crammer, Koby, Dredze, Mark, and Kulesza, Alex. Multiclass confidence weighted algorithms. In *EMNLP*, 2009b.
- Dani, V., Hayes, T., and Kakade, S. Stochastic linear optimization under bandit feedback. In *COLT*, 2008.
- Dekel, O., Gentile, C., and Sridharan, K. Robust selective sampling from single and multiple teachers. In *COLT*, 2010.
- Hoerl, A. and Kennard, R. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12: 55–67, 1970.
- Kakade, S., Shalev-Shwartz, S., and Tewari, A. Efficient bandit algorithms for online multiclass prediction. In *ICML*, 2008.
- Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, 2007.
- Lu, T., Pal, D., and Pal, M. Showing relevant ads via lipschitz context multi-armed bandits. In *AISTAT*, 2010.
- Rigollet, P. and Zeevi, A. Nonparametric bandits with covariates. In *COLT*, 2010.
- Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Rev.*, 65:386–407, 1958.
- Sandhaus, Evan. The new york times annotated corpus. Linguistic Data Consortium, Philadelphia, 2008.
- Strehl, A. and Littman, M. Online linear regression and its application to model-based reinforcement learning. In *NIPS*, 2008.
- Walsh, T. J., Szita, I., Diuk, C., and Littman, M. L. Exploring compact reinforcement-learning representations with linear regression. In *UAI & Rutgers U. TR.*, 2009.
- Wang, C., Kulkarni, S., and Poor, V. Bandit problems with side observation. *IEEE Trans. Autom. Control*, 50, 2005.
- Wang, S., Jin, R., and Valizadegan, H. A potential-based framework for online multi-class learning with partial feedback. In *AISTAT*, 2010.