Online Multiclass Boosting with Bandit Feedback

Daniel T. Zhang
Department of Computer Science
University of Michigan
dtzhang@umich.edu

Young Hun Jung
Department of Statistics
University of Michigan
yhjung@umich.edu

Ambuj Tewari
Department of Statistics
University of Michigan
tewaria@umich.edu

Abstract

We present online boosting algorithms for multiclass classification with bandit feedback, where the learner only receives feedback about the correctness of its prediction. We propose an unbiased estimate of the loss using a randomized prediction, allowing the model to update its weak learners with limited information. Using the unbiased estimate, we extend two full information boosting algorithms (Jung et al., 2017) to the bandit setting. We prove that the asymptotic error bounds of the bandit algorithms exactly match their full information counterparts. The cost of restricted feedback is reflected in the larger sample complexity. Experimental results also support our theoretical findings, and performance of the proposed models is comparable to that of an existing bandit boosting algorithm, which is limited to use binary weak learners.

1 INTRODUCTION

We study the online multiclass classification problem with bandit feedback. In this setting, the data instances arrive sequentially, and the learner has to predict the label among a finite, but perhaps large, set of candidates. In certain practical settings, such as when the labels are ads or product recommendations on the web, the learner does not receive the correct label as feedback. Instead, it only receives feedback about whether its predicted label was correct (e.g., the user clicked on the ad or recommendation) or not (e.g., user did not click). However, training machine learning models under such partial feedback is challenging. A common approach is to convert a full information algorithm into a bandit version without incurring too much performance loss (see, for example, Kakade et al. (2008) and Beygelzimer et al. (2017) for work using the perceptron algorithm).

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

In this paper, we design online algorithms for multiclass classification under bandit feedback by building on recent online boosting work in the full-information setting. Online boosting algorithms combine the predictions of multiple online weak learners to improve prediction performance. Classical boosting algorithms were designed for the batch setting. Chen et al. (2012) and Beygelzimer et al. (2015) first developed a theory of online boosting for binary classification. Then Jung et al. (2017) and Jung and Tewari (2018) extended the theory to the multiclass classification and the multilabel ranking problems. These works prove that the boosting algorithm's asymptotic error converges to zero if the number of weak learners, whose predictions are slightly better than random guessing, gets larger.

Designing a boosting algorithm with bandit feedback is particularly difficult as it is not clear how to update the weak learners. For example, suppose that a weak learner WL^1 predicts the label 1, another learner WL^2 predicts the label 2, and the boosting algorithm predicts the label 1, which turns out to be incorrect. We cannot even tell WL^2 whether its prediction is correct. To the best of our knowledge, Chen et al. (2014) are the only ones who have proposed a boosting algorithm in the multiclass bandit setting. However, their algorithm is restricted to use binary weak learners and only updates a subset of them, viz. ones which can get full feedback. In contrast, our algorithms use multiclass weak learners and update every learner at each round.

To derive our algorithms and guarantees, we extend the work of Jung et al. (2017) to the bandit setting. Instead of making a deterministic prediction, our algorithms randomize them. This allows them to estimate the loss using the distribution over labels, and this estimate is used to update the weak learners. Similar to the full information work, we propose a computationally expensive algorithm, BanditBBM, with an optimal error bound and a more practical algorithm, AdaBandit, with a suboptimal bound. AdaBandit is the first adaptive boosting algorithm in the bandit setting that does not assume weak learners' edge over random is known beforehand. Interestingly, our algorithms' asymptotic error bounds match the full information counterparts with increased sample complexity, which can be interpreted as the cost of bandit feedback.

2 PRELIMINARIES

We denote the indicator function by $\mathbb{I}(\cdot)$, the i^{th} standard basis vector by e_i, the vector of ones by $\mathbf{1}$, and the vector of zeros by $\mathbf{0}$. We will use $[n]$ for the set $\{1, \cdots, n\}$, Δ_n for probability distributions over $[n]$.

2.1 Problem Setting

We first describe the online multiclass classification problem with bandit feedback. It is a sequential game between two players: learner and adversary. The set $[k] = \{1, \cdots, k\}$ of k possible labels is known to both players. At each round $t = 1, \cdots, T$, the adversary selects a labeled example $(x_t, y_t) \in \mathcal{X} \times [k]$ (where $\mathcal{X}$ is some domain) and sends only x_t to the learner. The learner then tries to guess its label and sends its prediction $\hat{y}_t$ back to the adversary. As we are in the bandit setting, the adversary only reveals whether the prediction is correct by sending $\mathbb{I}(\hat{y}_t \neq y_t)$ to the learner. The learner's goal is to minimize the number of incorrect predictions. In other words, the learner's performance is evaluated by the zero-one loss (see (1) for definition).

To tackle this problem, we use the online multiclass boosting setup of Jung et al. (2017). In this setting, the learner further splits into N online weak learners, $WL^1, \cdots, WL^N$, as well as a booster that handles the weak learners. When the booster receives an unlabeled instance x_t, it shares this information with the weak learners and then aggregates their predictions to produce the final prediction. Here we assume that each weak learner WL^i predicts a label h_t^i in $[k]$. Once the booster gets the feedback from the adversary, it computes a cost vector $c_t^i \in \mathbb{R}^k$ for WL^i to incur the loss c^i_{t,h_t^i} and update its prediction rule. It should be noted that even though our boosting algorithms are designed for the bandit setting, the weak learners observe full cost vectors $c_t^i \in \mathbb{R}^k$, which are constructed from bandit feedback by the booster.

2.2 Unbiased Estimate of the Zero-One Loss

Even though we have not specified how to compute cost vectors c_t^i, it is naturally expected that they should depend on the final zero-one loss vector:

$$l_t^{0-1} = \mathbf{1} - e_{y_t} \in \mathbb{R}^k. \tag{1}$$

As we are in the bandit setting, the booster only has limited information about this vector. In particular, unless its final prediction is correct, only a single entry of l_t^{0-1} is available.

A popular approach for algorithm design in the partial information setting is to obtain an unbiased estimate of the loss. To do so, many bandit algorithms randomize their prediction. In our setting, instead of making a deterministic prediction $\hat{y}$, the algorithm designs a sampling distribution $p_t \in \Delta_k$ as follows:

$$p_{t,i} = \begin{cases} 1-\rho & \text{if } i = \hat{y}_t \\ \frac{\rho}{k-1} & \text{if } i \neq \hat{y}_t \end{cases}, \tag{2}$$

where ρ is a parameter that controls the exploration rate. This distribution puts a large weight on the label $\hat{y}_t$ and evenly distributes the remaining weight over the rest. The algorithm draws a final prediction $\tilde{y}_t$ based on p_t. In this way, the algorithm can build an estimator using the known sampling distribution. A simplest unbiased estimate of the zero-one loss is

$$\hat{l}_t^{0-1} = \frac{\mathbb{I}(\tilde{y}_t = y_t)}{p_{t,\tilde{y}_t}}(\mathbf{1} - e_{\tilde{y}_t}) \in \mathbb{R}^k. \tag{3}$$

It is easy to check that this is indeed unbiased. However, it is not necessarily the best because it becomes a zero vector when the booster makes a mistake. As the zero loss vector does not provide any useful information, the weak learners cannot update at this round. Therefore, it would be hard for the booster to escape the early training stage using the simple estimate.

As an alternative, we propose a new estimator

$$\begin{aligned}\hat{l}_{t,i}^{0-1} =& \frac{\mathbb{I}(\tilde{y}_t = y_t)}{p_{t,\tilde{y}_t}}\mathbb{I}(y_t \neq i)\mathbb{I}(\hat{y}_t \neq i) \\ &+ \frac{\mathbb{I}(\tilde{y}_t = \hat{y}_t)}{p_{t,\tilde{y}_t}}\mathbb{I}(\hat{y}_t \neq y_t)\mathbb{I}(\hat{y}_t = i).\end{aligned} \tag{4}$$

We first emphasize that this quantity can be computed only using the bandit feedback. The proof that it is actually unbiased appears in Appendix A.1.

This estimator resolves the main issue with the estimator in (3), viz. that the learner cannot update during a mistake round. In fact, it allows the weak learner to update on each instance with probability at least $1-\rho$. Furthermore, the algorithms using this estimator empirically performed much better than ones using the estimator in (3). For these reasons, we will stick to the estimate in (4) from now on.

To apply concentration inequalities, we need to control the variance of estimators. We say a random vector Y

is *b-bounded* if $||Y - \mathbb{E}Y||_\infty \leq b$ almost surely. Note that this definition also applies to random variables (i.e., scalars), in which case the norm above simply becomes the absolute value. It is easy to check our estimator $\hat{l}_t^{0-1}$ is $\frac{k}{\rho}$-bounded.

Now suppose that a cost vector $c_t^i \in \mathbb{R}^k$ (to be fed into weak learner i at time t) requires the knowledge of the true label y_t. Since the label is usually unavailable, we also need to estimate the cost vector. We first compute a matrix $C_t^i \in \mathbb{R}^{k \times k}$, whose j^{th} column is the cost vector c_t^i assuming j is the correct label. Then we will use the following random cost vector:

$$\hat{c}_t^i = C_t^i \cdot (\mathbf{1} - \hat{l}_t^{0-1}). \tag{5}$$

Since C_t^i is a deterministic matrix, we can compute

$$\mathbb{E}_{\tilde{y}_t} \hat{c}_t^i = C_t^i \cdot (\mathbf{1} - l_t^{0-1}) = C_t^i \cdot e_{y_t},$$

which is the y_t^{th} column of C_t^i. This shows that $\hat{c}_t^i$ is an unbiased estimate of c_t^i.

Algorithm 1 Online Bandit Boosting Template

```
1: Input: Exploration rate $\rho$
2: Initialize: Weak learner weights $\alpha_1^i$ for $i \in [N]$
3: for $t = 1, \cdots, T$ do
4:     Receive example $x_t$
5:     Get predictions $h_t^i \in [k]$ from $WL^i$ for $i \in [N]$
6:     Compute expert predictions for $j \in [N]$
         $s_t^j = \sum_{i=1}^j \alpha_t^i e_{h_t^i} \in \mathbb{R}^k$ and $\hat{y}_t^j = \text{argmax}_l\, s_{t,l}^j$
7:     Choose an expert index $i_t \in [N]$
8:     Get an intermediate prediction $\hat{y}_t = \hat{y}_t^{i_t}$
9:     Compute $p_t$ in (2)
10:    Draw $\tilde{y}_t$ using $p_t$ and send to the adversary
11:    Receive feedback $\mathbb{I}(\tilde{y}_t \neq y_t)$
12:    Estimate the loss by $\hat{l}_t^{0-1}$ in (4)
13:    Update weights $\alpha_{t+1}^i$ for $i \in [N]$
14:    Compute cost vectors $\hat{c}_t^i$ for $i \in [N]$
15:    Weak learners suffer the loss $\hat{c}_{t,h_t^i}^i$
16:    Weak learners update the internal parameters
17:    Update the booster's parameters, if any
18: end for
```

3 ALGORITHMS

We introduce two different online boosting algorithms and provide their theoretical error bounds. As the booster's performance obviously depends on the weak learner's predictive power, we need a way to quantify the latter. Firstly, we define the edge of a weak learner over random guessing and assume every weak learner has a positive edge γ. This edge is closely related to the one defined in the full information setting, and hence we can easily compare the error bounds between the two settings. This idea leads to the algorithm BanditBBM, which has a very strong error bound. Secondly, instead of having an additional assumption on the weak learners, we measure empirical edges of the learners and use these quantities to bound the number of mistakes. We call this algorithm AdaBandit, and since it enjoys theoretical guarantees under fewer assumptions, it is more practical.

3.1 Algorithm Template

Our boosting algorithms share a template, which we discuss here. As it adopts the cost vector framework from Jung et al. (2017) and Jung and Tewari (2018), the template is very similar except for the additional step of estimating the loss.

To keep the template in Algorithm 1 general, we do not specify certain steps, which will be finalized later for each algorithm. Also, we do not restrict weak learners in any way except requiring that each WL^i predicts a label $h_t^i \in [k]$, receives a full loss vector $\hat{c}_t^i \in \mathbb{R}^k$, and suffers the loss $\hat{c}_{t,h_t}^i$ according to their prediction.

The booster keeps updating the learner weights α_t^i and constructs experts. There are N experts where the expert j tracks the weighted cumulative votes among the first j weak learners: $s_t^j = \sum_{i=1}^j \alpha_t^i e_{h_t^i} \in \mathbb{R}^k$. We also track $\hat{y}_t^j = \text{argmax}_l\, s_{t,l}^j$, where the tie breaks arbitrarily. Then the booster chooses an expert index $i_t \in [N]$ at each round t and decides the intermediate prediction $\hat{y}_t$ as $\hat{y}_t^{i_t}$. In other words, it takes a weighted majority vote among the first i_t learners. BanditBBM fixes i_t to be N, while AdaBandit draws it randomly using a calibrated distribution. Using $\hat{y}_t$ and the exploration rate ρ, the booster computes the sampling distribution $p_t \in \Delta_k$ as in (2). A random label $\tilde{y}_t$ drawn from p_t is the final prediction, and the booster gets the feedback $\mathbb{I}(\tilde{y}_t \neq y_t)$. Then it constructs the unbiased estimate of the zero-one loss in (4) and updates the learner weights α_t^i. Finally, it computes cost vectors $\hat{c}_t^i \in \mathbb{R}^k$ for WL^i and lets them update their parameters.

3.2 An Optimal Algorithm

The first algorithm, BanditBBM (Bandit Boost-by-Majority), assumes the bandit weak learning condition, which states that weak learners are better than random guessing. The algorithm is optimal: it requires the minimal number of weak learners up to a constant factor to attain a certain accuracy.

3.2.1 Bandit Weak Learning Condition

This section proposes a bandit weak learning condition which requires weak learners to do better than random guessing, having only observed unbiased estimates of cost vectors. At a high level, what the weak learning condition says is that, as far as the random cost vectors provided by the booster satisfy certain conditions, the weak learner can perform better than random guessing when graded against the expected cost vectors. As a baseline, we define $u_\gamma^l \in \Delta_k$ to be almost a uniform distribution that puts γ more weight on the label l. As an example, $u_\gamma^k = (\frac{1-\gamma}{k}, \cdots, \frac{1-\gamma}{k}, \frac{1-\gamma}{k} + \gamma)$. The intuition is that if a learner predicts a label based on $u_\gamma^{y_t}$ at each round, then its accuracy would be better than random guessing by the edge γ.

The booster's goal is to minimize the number of incorrect predictions and hence it wants to put the minimal cost on the correct label. In this regard, Jung et al. (2017) constrain the choice of cost vectors[1] to

$$\mathcal{C}_1^{eor} = \{c \in \mathbb{R}_+^k \mid c_y = 0 \text{ and } ||c||_1 = 1\}, \qquad (6)$$

where y is the correct label. We allow a sample weight that can be multiplied by a cost vector to include scaled cost vectors. One remark is that we can always subtract a common number from every entry of the cost vector as we are interested in the relative loss. This means that as long as the booster puts the minimal cost on the correct label, one can transform it to represent as wc for some weight w and $c \in \mathcal{C}_1^{eor}$. Meanwhile, we are in the bandit setting, where the true label is often unavailable to the booster. Therefore, we allow our booster to compute a random vector $\hat{c}_t^i$, whose expectation lies in $\mathcal{C}_1^{eor}$. Once the exploration rate ρ is specified, we can additionally ensure that the random cost vectors are $\frac{k}{\rho}$-bounded.

It would be theoretically most sound if the bandit weak learning condition can be closely related to its full information counterpart. To do so, we present two weak learning conditions together. The settings are almost identical except the full information version observes a deterministic cost vector $c_t \in \mathcal{C}_1^{eor}$ while the bandit version only observes a randomized vector $\hat{c}_t$ such that $\mathbb{E}_{\tilde{y}_t} \hat{c}_t \in \mathcal{C}_1^{eor}$. Recall that the entire cost vector is shown to the learner even in the bandit setting. For both conditions, the time horizon is T, labeled data are chosen adaptively, and the parameters γ, δ, and the sample weights w_t lie in $[0, 1]$.

Definition 3.1 (OnlineWLC from Jung et al. (2017))**.** *A pair of a learner and an adversary satisfies OnlineWLC(γ, δ, S) if the learner can generate predictions $\hat{y}_t$ such that we have with probability $1 - \delta$,*

$$\sum_{t=1}^T w_t c_{t,\hat{y}_t} \leq \sum_{t=1}^T w_t c_t \cdot u_\gamma^{y_t} + S.$$

[1] In fact, the authors constrain the choice of cost matrices, but it suffices to choose a specific row to get a cost vector.

Definition 3.2 (BanditWLC)**.** *Suppose the random cost vectors $\hat{c}_t$ are b-bounded for some b. A pair of learner and adversary satisfies BanditWLC(γ, δ, S) if the learner can generate predictions $\hat{y}_t$, observing the random cost vectors $w_t\hat{c}_t$, such that we have with probability $1 - \delta$,*

$$\sum_{t=1}^T w_t c_{t,\hat{y}_t} \leq \sum_{t=1}^T w_t c_t \cdot u_\gamma^{y_t} + S,$$

where $c_t = \mathbb{E}\hat{c}_t$ for all t.

Here S is called excess loss. OnlineWLC is a special case of BanditWLC where the bound b is 0. In fact, we can show more intrinsic relations between the two.

Suppose there is a fixed hypothesis class $\mathcal{H}$ and an online learner makes a prediction $h_t(x_t)$ at time t by choosing a hypothesis $h_t \in \mathcal{H}$. Obviously, this setting does not cover all online learners, but the most widely used learners can be interpreted in this manner. Jung et al. (2017) showed that the OnlineWLC can be derived from the following two assumptions:

- (Online Richness Condition) For any sequence of cost vectors $(w_t, c_t) \in [0, 1] \times \mathcal{C}_1^{eor}$, there is a hypothesis $h \in \mathcal{H}$ such that

$$\sum_{t=1}^T w_t c_{t,h(x_t)} \leq \sum_{t=1}^T w_t c_t \cdot u_\gamma^{y_t}.$$

- (Online Agnostic Learnability Condition) For any sequence of (bounded) loss vectors $l_t \in \mathbb{R}^k$, there is an online algorithm which can generate predictions $\hat{y}_t$ such that with probability $1 - \delta$,

$$\sum_{t=1}^T l_{t,\hat{y}_t} \leq \inf_{h \in \mathcal{H}} \sum_{t=1}^T l_{t,h(x_t)} + R_\delta(T),$$

  where $R_\delta(\cdot)$ is a sublinear regret.

Note that the online learnability condition only assumes a bounded loss instead of $\mathcal{C}_1^{eor}$. This condition holds, for example, if the space $\mathcal{H}$ has a finite Littlestone dimension. Interested readers can refer the paper by Daniely et al. (2015). We show that these two conditions also imply the BanditWLC. The proof appears in Appendix A.2.

Theorem 3.1. *Suppose a pair of weak learning space $\mathcal{H}$ and adversary satisfies the richness condition with edge 2γ and the agnostic learnability condition with regret $R_\delta(T)$. Addionally, we assume that $w_t \geq m$ for all t. Then the online learner based on $\mathcal{H}$ satisfies BanditWLC($\gamma, 2\delta, S$) with*

$$S = \sup_T -\frac{\gamma}{k} mT + b\sqrt{2T \log \frac{1}{\delta}} + R_\delta(T),$$

where b is the bound of the random cost vectors.

The extra condition $w_t \geq m$ is acceptable because if $w_t = 0$ for some t, then we can simply ignore this round because any prediction does not incur a loss. The excess loss S is always finite due to the sublinear regret $R_\delta(T)$. Furthermore, a smaller δ would require a larger S. The excess loss in the BanditWLC is larger than the one in the OnlineWLC due to the term $b\sqrt{2T \log \frac{1}{\delta}}$. This is intuitive in that the learner needs more samples if only bandit feedback is available. Finally, the exploration rate ρ also affects S because b is equal to $\frac{k}{\rho}$. This provides the following rough bound:

$$S = \tilde{O}(\frac{k}{\rho}), \tag{7}$$

where $\tilde{O}$ suppresses dependence on $\log \frac{1}{\delta}$.

3.2.2 BanditBBM Details

Throughout the section, we assume the weak learners satisfy BanditWLC(γ, δ, S). BanditBBM is a modification of OnlineMBBM from Jung et al. (2017) by incorporating the unbiased estimate of the loss in (4).

We use the potential function $\phi_i^y(s)$, discussed thoroughly in relation to boosting by Mukherjee and Schapire (2013), to design the cost vectors. The potential function ϕ_i^y takes the current cumulative votes $s \in \mathbb{R}^k$ as an input and estimates the booster's loss when the true label is y and there are i weak learners left until the final prediction. In particular, it can be recursively defined as follows:

$$\phi_0^y(s) = \mathbb{I}(\operatorname*{argmax}_i s_i \neq y)$$
$$\phi_{i+1}^y(s) = \mathbb{E}_{l\sim u_\gamma^y} \phi_i^y(s + e_l).$$

Unfortunately, this potential does not have a closed form. Since potential functions are the main ingredient to design cost vectors, their computation becomes a bottleneck when running the algorithm. This is a weakness of BanditBBM despite its strong mistake bound. However, one can use Monte Carlo simulations to approximate its value.

Returning to our algorithm, we essentially want to set the cost vector to

$$c_{t,l}^i = \phi_{N-i}^{y_t}(s_t^{i-1} + e_l). \tag{8}$$

Jung et al. (2017) prove that this cost vector puts the minimal cost on the correct label and thus it is a valid choice. The booster in our setting, however, cannot compute this vector as it requires the knowledge of the true label y_t. As an alternative, we create the following cost matrix

$$C_t^i[l, r] = \phi_{N-i}^r(s_t^{i-1} + e_l) \tag{9}$$

Algorithm 2 BanditBBM Details

```
 2: **Initialize:** Set $\alpha_1^i = 1$ for $i \in [N]$
 7: Set $i_t = N$
13: Keep weights $\alpha_{t+1}^i = 1$ for $i \in [N]$
14: Compute cost vectors $\hat{c}_t^i$ using (5) and (9)
17: There is no extra parameter
```

and use (5) to compute a random cost vector $\hat{c}_t^i$, which is an unbiased estimate of c_t^i in (8).

The rest of the algorithm is straightforward. We set all weights α_t^i to be one and always choose the last expert: $i_t = N$. This means that the intermediate prediction $\hat{y}_t$ is a simple majority vote among all the weak learners. The reasoning behind this is that the booster wants to include all learners as they are strictly better than random, and all weak learners are equivalent in that they share the same edge γ. Algorithm 2 summarizes the specifications.

3.2.3 Mistake Bound of BanditBBM

We still assume that our weak learners satisfy BanditWLC(γ, δ, S). From observation (7), it is reasonable to assume $S = \tilde{O}(\frac{k}{\rho})$. Upon these assumptions, we can bound the number of mistakes made by BanditBBM. The proof appears in Appendix A.3

Theorem 3.2 (Mistake Bound of BanditBBM)**.** *For any T, N satisfying $\delta \ll \frac{1}{N}$, the number of mistakes made by BanditBBM satisfies the following inequality with probability at least $1 - (N + 1)\delta$:*

$$\sum_{t=1}^{T} \mathbb{I}(\tilde{y}_t \neq y_t) \leq (k-1)e^{-\frac{\gamma^2 N}{2}} T + 2\rho T + \tilde{O}(\frac{k^{7/2}\sqrt{N}}{\rho}),$$

where $\tilde{O}$ suppresses dependence on $\log \frac{1}{\delta}$.

If we set the exploration rate $\rho = \frac{k^{7/4}N^{1/4}}{\sqrt{T}}$, then the bound becomes

$$(k-1)e^{-\frac{\gamma^2 N}{2}} T + \tilde{O}(k^{7/4}N^{1/4}\sqrt{T}).$$

Dividing by T, we can infer that $(k-1)e^{-\frac{\gamma^2 N}{2}}$ is the asymptotic error bound of the algorithm. This bound matches the bound of the full information counterpart, OnlineMBBM. Since it depends exponentially on N, BanditBBM does not require too many weak learners to obtain a desired accuracy. Jung et al. (2017) also provide a lower bound in the full information setting, which shows that the exponential decay is the fastest rate one can expect for the asymptotic error bound. This result applies to our bandit setting as it is harder.

3.3 An Adaptive Algorithm

While BanditBBM is theoretically sound, in real applications it has a number of drawbacks. Firstly, it is hard to identify the edge γ of each weak learner, leading to incorrect computations of the potential function. Also, each learner may have a different edge, and assuming a common edge can underestimate some weak learner's predictive power. Finally, as pointed out in the previous section, evaluating the potential function is computationally expensive, which makes BanditBBM less useful in practice. To address these issues, we propose an adaptive algorithm, AdaBandit, based on the full information adaptive algorithm, Adaboost.OLM by Jung et al. (2017). Using the idea of improper learning, Foster et al. (2018) proposed another adaptive boosting algorithm that has a tighter sample complexity than Adaboost.OLM. However, we stick to Adaboost.OLM as it has the competitive asymptotic error bound, which is of primary interest in this paper.

3.3.1 Logistic Loss and Empirical Edges

Instead of directly minimizing the zero-one loss, the adaptive algorithm tries to minimize a surrogate loss. As in Adaboost.OLM, we choose the following logistic loss $l_y^{\log} : \mathbb{R}^k \to \mathbb{R}$:

$$l_y^{\log}(s) = \sum_{l=1}^{k} \log(1 + \exp(s_l - s_y)),$$

where s is the cumulative votes of a chosen expert.

As for the zero-one loss, computing the loss requires knowledge of the true label, and we again use the idea in (5) to estimate the loss. We want to emphasize that the logistic loss only plays an intermediate role in training, and the learner's predictions are still evaluated by the zero-one loss.

Essentially, we want to set the cost vector $c_t^i = \nabla l_{y_t}^{\log}(s_t^{i-1})$. Since this depends on the true label y_t, we build a cost matrix $C_t^i \in \mathbb{R}^{k \times k}$ as below:

$$C_t^i[l, r] = \begin{cases} \frac{1}{1+\exp(s_{t,r}^{i-1} - s_{t,l}^{i-1})} & \text{if } l \neq r \\ -\sum_{j \neq r} \frac{1}{1+\exp(s_{t,r}^{i-1} - s_{t,j}^{i-1})} & \text{if } l = r \end{cases}. \quad (10)$$

Note that each column also puts the minimal cost on the correct label r. Moreover, the sum of entries equals zero. Using the idea described in (5), we can compute $\hat{c}_t^i$, which is an unbiased estimate of $c_t^i = \nabla l_{y_t}^{\log}(s_t^{i-1})$.

Even though the adaptive algorithm does not assume the BanditWLC, we still need to measure the weak learners' predictive powers to analyze the booster's performance. As in the full information case, we use the following *empirical edge* of WL^i:

$$\gamma_i = \sum_{t=1}^{T} c_{t,h_t^i}^i / \sum_{t=1}^{T} c_{t,y_t}^i.$$

Having the same empirical edge as Adaboost.OLM allows us to precisely evaluate the cost of bandit feedback. Based on our design of cost vector $c_t^i = \nabla l_{y_t}^{\log}(s_t^{i-1})$, we can check that γ_i is in $[-1, 1]$ and a larger value implies a better accuracy. Obviously, the empirical edge is unavailable to the learner as it requires the true cost vector c_t^i. This is fine because we only use this value to provide the mistake bound. Running AdaBandit does not require the knowledge of empirical edges.

3.3.2 AdaBandit Details

Now we describe the details of AdaBandit (see Algorithm 3). The choice of cost vectors $\hat{c}_t^i$ is already discussed in the previous section. As this is an adaptive algorithm, we update the learner weights α_t^i to give more influence to high-performing learners. We also allow negative weights in case a weak learner is worse than random.

As AdaBandit incorporates the logistic loss as a surrogate, we want to pick α_t^i to minimize

$$\sum_{t=1}^{T} f_t^i(\alpha_t^i) \text{ where } f_t^i(\alpha) = l_{y_t}^{\log}(s_t^{i-1} + \alpha e_{h_t^i}),$$

where only the following unbiased estimator $\hat{f}_t^i$ is available to the learner:

$$\hat{f}_t^i(\alpha) = \sum_{j=1}^{k} l_j^{\log}(s_t^{i-1} + \alpha e_{h_t^i}) \cdot (1 - \hat{l}_{t,j}^{0-1}).$$

Since the logistic loss is convex, it is a classical online convex optimization problem, and we can use stochastic gradient descent (see Zinkevich (2003) and Shalev-Shwartz and Ben-David (2014)). Following the convention in Adaboost.OLM, we use the feasible set $F = [-2, 2]$ and the projection function $\Pi(\cdot) = \max\{-2, \min\{2, \cdot\}\}$ to update α_t^i:

$$\alpha_{t+1}^i = \Pi(\alpha_t^i - \eta_t \hat{f}_t^{i\prime}(\alpha_t^i)), \quad (11)$$

where η_t is a learning rate. As the gradient of the logistic loss is universally bounded by k and $\hat{l}_t^{0-1}$ is $\frac{k}{\rho}$-bounded, we can check that $|\hat{f}_t^{i\prime}(\alpha)| \leq \frac{2k^2}{\rho}$ almost surely. From this, if we set $\eta_t = \frac{\rho}{k^2\sqrt{t}}$, then a standard result in online stochastic gradient descent (see Shalev-Shwartz and Ben-David (2014), Chapter 14) provides with probability $1 - \delta$,

$$\sum_{t=1}^{T} f_t^i(\alpha_t^i) \leq \min_{\alpha \in [-2,2]} \sum_{t=1}^{T} f_t^i(\alpha) + \tilde{O}(\frac{k^2}{\rho}\sqrt{T}), \quad (12)$$

Algorithm 3 AdaBandit Details

```
 2: Initialize: Set $\alpha_1^i = 0$ and $v_t^i = 1$ for $i \in [N]$
 7: Randomly draw $i_t$ with $\mathbb{P}(i_t = i) \propto v_t^i$
13: Update weights $\alpha_t^i$ using (11) with $\eta_t = \frac{\rho}{k^2\sqrt{t}}$
14: Compute cost vectors $\hat{c}_t^i$ using (5) and (10)
17: Update $v_{t+1}^i = v_t^i \cdot \exp(-\hat{l}^{0-1}_{t,\hat{y}_t^i})$
```

where $\tilde{O}$ suppresses dependence on $\log \frac{1}{\delta}$.

We cannot prove that the last expert is the best because our weak learners do not adhere to the weak learning condition. Instead, we will show that at least one expert is reliable. To identify this expert, we use the *Hedge algorithm* from Littlestone and Warmuth (1994) and Freund and Schapire (1997). This algorithm generally receives the zero-one loss of each expert. Since that is no longer available, we will feed $\hat{l}^{0-1}$, which in expectation reflects the true zero-one loss. As the exploration rate ρ controls the variance of the loss estimate, we can combine the analysis of the Hedge algorithm with the concentration inequality to obtain a similar result.

3.3.3 Mistake Bound of AdaBandit

As mentioned earlier, we bound the number of mistakes made by the adaptive algorithm using the weak learners' empirical edges. We emphasize again that these empirical edges are defined exactly in the same manner with those used in the full information bound.

Theorem 3.3 (Mistake Bound of AdaBandit)**.** *For any T, N satisfying $\delta \ll \frac{1}{N}$, the number of mistakes made by AdaBandit satisfies the following inequality with probability at least $1-(N+4)\delta$:*

$$\sum_{t=1}^{T} \mathbb{I}(\tilde{y}_t \neq y_t) \leq \frac{8k}{\sum_{i=1}^N \gamma_i^2}T + 2\rho T + \tilde{O}(\frac{k^3N^2}{\rho^2 \sum_{i=1}^N \gamma_i^2}),$$

where $\tilde{O}$ suppresses dependence on $\log \frac{1}{\delta}$.

If we set the exploration rate $\rho = \frac{kN^{2/3}}{(T\sum_{i=1}^N \gamma_i^2)^{1/3}}$, then the bound becomes

$$\frac{8k}{\sum_{i=1}^N \gamma_i^2}T + \tilde{O}(\frac{kN^{\frac{2}{3}}}{(\sum_{i=1}^N \gamma_i^2)^{\frac{1}{3}}}T^{\frac{2}{3}}).$$

This implies that $\frac{8k}{\sum_{i=1}^N \gamma_i^2}$ becomes the asymptotic error bound of AdaBandit, which matches the bound of Adaboost.OLM. Jung et al. (2017) observe that $\gamma_i \geq \gamma$ with high probability if the learner has edge γ. Therefore, if our weak learners satisfy BanditWLC(γ, δ, S) as for BanditBBM, then the asymptotic bound becomes roughly $\frac{8k}{N\gamma^2}$. The bound depends polynomially on N, which is suboptimal. However, AdaBandit resolves the aforementioned issues of BanditBBM and actually shows comparable results on real data sets.

4 EXPERIMENTS

We compare various boosting algorithms on benchmark data sets using publicly available code[2]. The models include our proposed algorithms, BanditBBM and AdaBandit, their full information versions, OnlineMBBM and Adaboost.OLM from Jung et al. (2017), and BanditBoost from Chen et al. (2014). To maximize readability, we will call them by OptBandit, AdaBandit, OptFull, AdaFull, and BinBandit respectively, based on their characteristics. The first four models require multiclass weak learners, whereas BinBandit needs binary learners. For every model, we use online decision trees proposed by Domingos and Hulten (2000) as weak learners.

We examine several data sets from the UCI data repository (Blake and Merz, 1998; Higuera et al., 2015; Ugulino et al., 2012) that are tested by Jung et al. (2017). We follow the authors' data preprocessing to provide a consistent comparison. However, the bandit algorithms need more samples to reach their asymptotic performance. Because these data sets often have insufficient examples to yield this asymptotic performance, we duplicate and shuffle the data sets a number of times before feeding them to the algorithm. The amount of duplication done to each data set is chosen to suggest the asymptotic performance of each algorithm. Table 1 contains a summary of data sets that are examined. The number of actual data points sent to each model is noted under the column StreamCnt.

We optimize the number of weak learners N for each bandit algorithm and data set, with granularity down to multiples of 5. As BinBandit only takes binary weak learners, it needs more of them for data sets with large k. Thus, we use $10k$ weak learners for BinBandit on each data set. Recall that OptBandit and OptFull require the knowledge of the edge γ from their weak learning condition. Since one cannot identify this value in practice, we also do not optimize this value and select $\gamma = 0.1$ to be fixed. Lastly, the three bandit algorithms have the exploration rate ρ, which we optimize through the grid search and record the best results. A more detailed description of the experiment setting appears in Appendix B.

4.1 Asymptotic Performance

Since the theoretical asymptotic error bounds of the proposed algorithms match their full information counterparts, we first compare the models' empirical

[2] `https://github.com/pi224/banditboosting`

Table 1: Bandit and Full Information Asymptotic Performance

Data	k	StreamCnt	OptBandit	AdaBandit	BinBandit	OptFull	AdaFull
Balance	3	6250	0.89	0.97	0.80	0.76	0.93
Car	4	10368	0.88	0.98	0.84	0.84	0.96
Nursery	4	51840	0.93	0.95	0.92	0.94	0.98
Movement	5	165631	0.94	0.97	0.89	0.92	0.97
Mice	8	8640	0.73	0.87	0.81	0.81	0.96
Isolet	26	116955	0.48	0.66	0.66	0.84	0.90

asymptotic performance. To do so, we feed the first 80% of the data without counting mistakes and compute the average accuracy on the last 20% of the data. Table 1 summarizes the results. The accuracy is averaged over 20 rounds for all data sets except Isolet and Movement, which we ran 10 times. These runs were computed with shuffling from 20 random seeds, a predetermined subset of which were used for Isolet and Movement.

The full information algorithms exhibit very strong performance due to data duplication. Despite this, OptBandit and AdaBandit are quite competitive against them across data sets with smaller k. For datasets with larger k, our bandit algorithms do not keep up as well, as they receive less feedback per instance. Our algorithm's perform comparably to BinBandit, showing that our algorithms successfully combine the multiclass weak learners. A noteworthy aspect is how AdaBandit outperforms OptBandit on all the data sets, showing adaptive weighing's power.

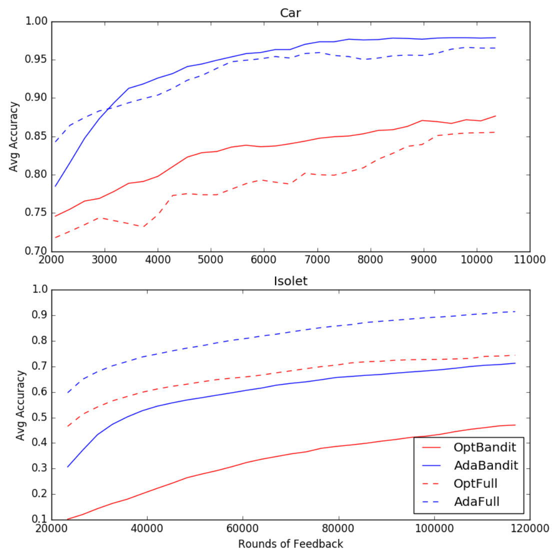

Figure 1: Learning Curves on Car (Top) and Isolet (Bottom); Best Viewed in Color

4.2 Analyzing Learning Curves

Even though our bandit algorithms have the same asymptotic error bounds as their full information counterparts, the cost of bandit feedback is reflected in larger sample complexities. Investigating this, we compute approximate learning curves for these algorithms by recording the moving average accuracy across the latest $0.2\times$*(total rounds to be run)* data instances. For data sets of varying k this illustrates the hardness of the bandit problem: as k increases, learning an appropriately performing hypothesis takes longer, but is achievable nonetheless.

Figure 1 shows the learning curves on Car and Isolet data for our two bandit algorithms as compared with their full information counterparts. The curves for other data sets can be found in the Appendix B. On Car data where k is small, AdaBandit even outperforms OptFull and AdaFull by the end. This competitiveness with full information algorithms is reflected in the other learning curves in the appendix. On Isolet data with large k, our bandit methods lose some of their competitiveness. Given that the exploration rate ρ is set to 0.1 (whereas the theory would have it converge to 0) and that the bandit algorithms have not fully plateaued off, the performance is still reasonable.

Acknowledgements

AT acknowledges the support of the NSF CAREER grant IIS-1452099, a Sloan Research Fellowship, and the LSA Associate Professor Support Fund.

References

Beygelzimer, A., Kale, S., and Luo, H. (2015). Optimal and adaptive algorithms for online boosting. In *Proceedings of the International Coference on Machine Learning*, pages 2323–2331.

Beygelzimer, A., Orabona, F., and Zhang, C. (2017). Efficient online bandit multiclass learning with $\tilde{O}(\sqrt{T})$ regret. In *Proceedings of the International Conference on Machine Learning*, pages 488–497.

Blake, C. and Merz, C. (1998). UCI machine learning repository.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Chen, S.-T., Lin, H.-T., and Lu, C.-J. (2012). An online boosting algorithm with theoretical justifications. In *Proceedings of the International Coference on Machine Learning*, pages 1873–1880. Omnipress.

Chen, S.-T., Lin, H.-T., and Lu, C.-J. (2014). Boosting with online binary learners for the multiclass bandit problem. In *Proceedings of the International Conference on Machine Learning*, pages 342–350.

Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. (2015). Multiclass learnability and the erm principle. *The Journal of Machine Learning Research*, 16(1):2377–2404.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM.

Foster, D. J., Kale, S., Luo, H., Mohri, M., and Sridharan, K. (2018). Logistic regression: The importance of being improper. *Proceedings of Machine Learning Research*, 75:1–42.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Higuera, C., Gardiner, K. J., and Cios, K. J. (2015). Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PloS one*, 10(6):e0129126.

Jung, Y. H., Goetz, J., and Tewari, A. (2017). Online multiclass boosting. In *Advances in Neural Information Processing Systems*, pages 919–928.

Jung, Y. H. and Tewari, A. (2018). Online boosting algorithms for multi-label ranking. In *International Conference on Artificial Intelligence and Statistics*, pages 279–287.

Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2008). Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the International Conference on Machine Learning*, pages 440–447. ACM.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and computation*, 108(2):212–261.

Mukherjee, I. and Schapire, R. E. (2013). A theory of multiclass boosting. *Journal of Machine Learning Research*, 14(Feb):437–497.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Ugulino, W., Cardador, D., Vega, K., Velloso, E., Milidiú, R., and Fuks, H. (2012). Wearable computing: Accelerometers data classification of body postures and movements. In *Advances in Artificial Intelligence-SBIA*, pages 52–61. Springer.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning*, pages 928–936.