Learning Multiclass Classifier Under NoisyBandit Feedback

by

Mudit Agrawal, Naresh Manwani

Report No: IIIT/TR/2021/-1

Centre for Others
International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2021

Learning Multiclass Classifier Under Noisy Bandit Feedback

Mudit Agarwal and Naresh Manwani

Machine Learning Lab, KCIS, International Institute of Information Technology
Hyderabad, India
mudit.agarwal@research.iiit.ac.in, naresh.manwani@iiit.ac.in

Abstract. This paper addresses the problem of multiclass classification with corrupted or noisy bandit feedback. In this setting, the learner may not receive true feedback. Instead, it receives feedback that has been flipped with some non-zero probability. We propose a novel approach to deal with noisy bandit feedback based on the unbiased estimator technique. We further offer a method that can efficiently estimate the noise rates, thus providing an end-to-end framework. The proposed algorithm enjoys a mistake bound of the order of $O(\sqrt{T})$ in the high noise case and of the order of $O(T^{2/3})$ in the worst case. We show our approach's effectiveness using extensive experiments on several benchmark datasets.

Keywords: Online Learning · Recommender System· Classification

1 Introduction

In machine learning, multiclass classification is of particular interest due to its widespread application in several domains such as digit-recognition [17], text classification [18] and recommender systems [14]. Some of the well-known batch learning approaches for multiclass classification are discussed in [13,1,5,21]. An extension of Perceptron [23] to the multiclass setting was first proposed in [11], which was later modified by [14] to deal with bandit feedback setting. Unlike the full information setting, the bandit setting's learner receives only partial feedback, indicating whether the predicted label is correct or incorrect, popularly known as bandit feedback. The learner's ability to learn a correct hypothesis under bandit feedback finds several web-based applications, such as sponsored advertising on web pages and recommender systems as mentioned by [14]. In the typical setting of the recommender system, when a user makes a query to the system, then the user is presented with a suggestion based on the past browsing history; finally, the user responds to the suggestion, either positively (clicking it) or negatively (not clicking it). However, the system does not know the behavior of the user if presented with other suggestions.

Banditron [14] uses an exploitation-exploration scheme proposed in [3]. When it updates, it replaces the gradient of the loss function with an unbiased estimator of the gradient. When the data is linearly separable, the expected number of mistakes made by Banditron is shown to be $O(\sqrt{T})$. In the general case, the

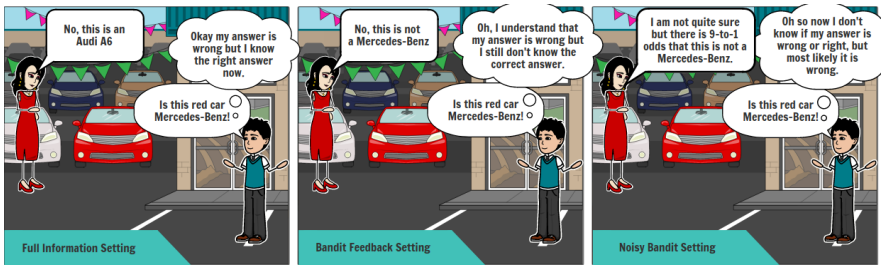

Fig. 1: Three kinds of supervised learning (a) Full Information Setting: In this setting, the learner receives the actual class label. (b) Bandit Feedback Setting: A bandit feedback is revealed to the learner, indicating whether the predicted label is correct or not. (c) Noisy Bandit Setting: The learner receives noisy bandit feedback (noisy feedback is received by flipping the correct feedback with some small probability).

expected number of mistakes of Banditron is $O\left(T^{2/3}\right)$. Another bandit algorithm, named Newtron [12], is based on the online Newton method. It uses a strongly convex objective function (adding regularization term with the loss function) and Follow-The-Regularized-Leader (FTRL) strategy to achieve $O(\log T)$ regret bound in the best case and $O\left(T^{2/3}\right)$ regret bound in the worst case. Second-order Perceptron is also extended in bandit feedback setting by Crammer, and Gentile [6]. It uses upper-confidence bounds (UCB) [2] based approach to handle exploration-exploitation and achieves regret bound of $O\left(\sqrt{T}\log(T)\right)$ Beygelzimer et al. [4] proposed efficient algorithms under bandit feedback when the data is linearly separable by a margin of γ. They show that their algorithm achieves a near-optimal bound of $O\left({}^{K}/_{\gamma}\right)$ under strong linear separability condition [4].

In all the above approaches, it is assumed that the user has provided correct bandit feedback. There are many practical situations where the bandit feedback can become noisy too. In such a scenario, this means that the feedback that indicates that the predicted label is identical to the actual label may be incorrect with some non-zero probability. Consider the following examples of noisy bandit feedback. In the recommendation system, there are few cases in which a user may accidentally click (positive feedback) the recommended ad. In this case, the true feedback should be negative (no clicks). However, instead of negative, the recommender system receives positive feedback. Fake reviews and ratings are also posted using automated bots, which can boost the visibility of those products on recommendation platforms [15].

In this paper, we model the noisy bandit feedback by assuming an adversary between the learner and the environment. Whenever the learner asks a binary query, the environment releases the actual feedback. Then, the adversary flips the actual feedback with probability ρ and releases it to the learner. The problem of multiclass classification under noisy bandit feedback is as follows: on each round, the learner is given an instance vector $\mathbf{x}$; the learner predicts a label $\hat{y}$; then the learner receives the corrupted feedback f_ρ. The noisy version of this problem is

more challenging because, besides bandit feedback, the learner also has to deal with noise or corruption present in the feedback. To learn a robust classifier in the presence of noisy bandit feedback, we propose an unbiased estimator $h(f_\rho)$ of the actual feedback f. The goal is to maximize the sum of $h(f_\rho^t)$, which in expectation, turns out to be the maximizing sum of actual feedbacks. Similar ideas have been explored to handle label noise in classification problems [20] under full information setting. This is the first work proposing a robust multiclass classifier under noisy bandit feedback to the best of our knowledge.

Key Contribution of The Paper:

1. We propose a robust algorithm for learning multiclass classifiers under noisy bandit feedbacks. The proposed algorithm enjoys a mistake bound of $O(\sqrt{T})$ in the high noise case and $O(T^{2/3})$ in the worst case.
2. We also propose an algorithm for noise rate estimation.
3. We validate our algorithms through experiments on benchmark datasets.

2 Multiclass Classification

In the multiclass classification, the goal is to learn a function which maps each example to one of the K categories. Let $g : \mathcal{X} \to [K]$ be the multiclass classifier where $\mathcal{X} \subseteq \mathbb{R}^d$ and $[K] = \{1, \ldots, K\}$. A multiclass classifier can be modeled using a weight matrix $W \in \mathbb{R}^{K \times d}$ as $g(\mathbf{x}) = \arg\max_{j \in [K]} \mathbf{w}_j \cdot \mathbf{x}$, where $\mathbf{w}_j$ is the j^{th} row of matrix W and $\mathbf{x} \in \mathcal{X}$. We need to identify the weight matrix W to find the classifier. In order to identify the parameters in W of the underlying classifier, we use training data of the form $\{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^T, y^T)\}$ where $(\mathbf{x}^t, y^t) \in \mathcal{X} \times \{1, \ldots, K\},\ \forall t \in [T]$. The performance of the classifier f described by parameters W on example $\mathbf{x}^t$ is measured using 0-1 loss as $L_{0-1}(g(\mathbf{x}^t), y^t) = \mathbb{I}[g(\mathbf{x}^t) \neq y^t]$.[1] L_{0-1} is difficult to optimize. In practice, we use convex surrogates of L_{0-1}. L_H is one such surrogate [7] described as follows.

$$L_H(W, (\mathbf{x}^t, y^t)) = \max_{j \neq y^t} [1 - \mathbf{w}_{y^t} \cdot \mathbf{x}^t + \mathbf{w}_j \cdot \mathbf{x}^t]_+ \tag{1}$$

Here $[a]_+ = \max(0, a)$. Loss L_H becomes 0 when $\mathbf{w}_{y^t} \cdot \mathbf{x}^t - \mathbf{w}_j \cdot \mathbf{x}^t \geq 1,\ \forall j \neq y^t$.

Online Multiclass Classification: Full Information Case

In the full information case, the learner receives the actual class label of examples in every trial. A large margin Perceptron algorithm for multiclass classification using L_H is proposed in [8]. The algorithm works as follows. The algorithm starts with W^1 as a zero matrix. Let W^t be the weight matrix, and $\mathbf{x}^t$ be the example presented at trial t, to algorithm. Then the algorithm predicts the labels $\hat{y}^t$ as

[1] Here, $\mathbb{I}[A] = 1$ when the predicate A is true and 0 otherwise.

$\hat{y}^t = \arg\max_{j\in[K]} \mathbf{w}_j^t \cdot \mathbf{x}^t$. Now it receives the true class label y^t of $\mathbf{x}^t$. Algorithm incurs a loss $L_H(W^t, (\mathbf{x}^t, y^t))$ and updates the parameters as $W^{t+1} = W^t + U^t$.

$$U_{r,j}^t = \Big[\mathbb{I}[y^t = r] - \mathbb{I}[\hat{y}^t = r]\Big] x_{t,j}. \tag{2}$$

This algorithm converges in finite iterations if the data is linearly separable [8].

Online Multiclass Classification: Bandit Feedback Case

In the bandit feedback setting [14], the learner can only know whether the predicted label is correct or not. Banditron [14] modifies the Perceptron algorithm to deal with the bandit feedback. Let W^t be the weight matrix in the beginning of trial t and $\mathbf{x}^t$ be the example presented at trial t. Let $\hat{y}^t = \arg\max_{j\in[K]} \mathbf{w}_j^t \cdot \mathbf{x}^t$. Banditron defines a probability distribution p^t on class labels as follows.

$$p^t(i) = (1-\gamma)\mathbb{I}\left[i = \hat{y}^t\right] + \frac{\gamma}{K} \tag{3}$$

Here, $\gamma \in [0, 1)$ is the probability of exploration. The algorithm predicts the label $\tilde{y}^t$, which is randomly drawn from the distribution p^t. The algorithm then receives a feedback $f^t = \mathbb{I}[\tilde{y}^t = y^t]$. Banditron updates the weight matrix as $W^{t+1} = W^t + \tilde{U}^t$ where $\tilde{U}_{r,j}^t = x_{t,j}\left(\frac{\mathbb{I}[y^t = \tilde{y}^t]\mathbb{I}[\tilde{y}^t = r]}{p^t(r)} - \mathbb{I}[\hat{y}^t = r]\right)$.

3 Learning Using Noisy Bandit Feedback

In the noisy feedback setting, an adversary is present between the learner and the feedback, which manipulates the feedback to confuse the learner. It is hypothetical to assume noise-free data [15] in the real world. So, one can find many real-world applications which are more appropriately modeled using a noisy feedback setting. For example, in a click-based recommendation system, we try to model the user behavior based on the clicks. These clicks are nothing but the bandit feedbacks, which are assumed to describe whether the user liked the recommended ad/product. Indeed, a user clicking the ad (or like the product) and likes it are two correlated events. However, the user may like the ad and does not click on it. On the other hand, the user may not like the ad but clicks on it (accidentally or in the absence of other exciting ads). These clicks are noisy as each user click does not necessarily mean that they agree with the recommended ad/product.

In this paper, we model the noisy bandit feedback as follows. Let there be an adversary which flips the true feedback, f, with a non-zero probability and generates noisy feedback. We denote the noisy bandit feedback by f_ρ. Let $P(f_\rho = 1|f = 0) = \rho_0$, $P(f_\rho = 0|f = 1) = \rho_1$ be the noise rates ($\rho_1 + \rho_0 < 1$).

Proposed Approach

Here, we propose a robust algorithm that can learn the true underlying classifier given noisy bandit feedback. To deal with the noisy or corrupted feedback, we

propose a modified or proxy feedback $h(f_\rho)$, which is an unbiased estimator of true feedback f, as follows. Given the noisy feedback f_ρ, Lemma 1 shows how to construct an unbiased estimator of the true feedback f.[2]

Lemma 1. *Let $f^t = \mathbb{I}[\tilde{y}^t = y^t]$ be the true feedback. Let $h(f_\rho^t)$ be defined as,*

$$h(f_\rho) = \frac{(1-\rho_{f_\rho'})f_\rho - \rho_{f_\rho} f_\rho^{'}}{1-\rho_0-\rho_1} \tag{4}$$

where $f_\rho^{'} = 1 - f_\rho$. Then, $\mathbb{E}_{f_\rho^t}[h(f_\rho^t)] = \mathbb{I}[\tilde{y}^t = y^t] = f^t$.

Instead of noisy feedback f_ρ, we use $h(f_\rho)$ (see eq (4)) which is an unbiased estimator of the true feedback f (Lemma 1). Similar ideas have been used to deal with the label noise in full information case [20]. We are now in a position to state a robust classifier for noisy bandit feedback. When there is no noise (*i.e*, $\rho_0 = \rho_1 = 0$), we see that $h(f_\rho) = f_\rho = f$. Thus, under noise-free case, $h(f_\rho)$ becomes same as the noise-free bandit feedback f. At each round, the learner finds $\hat{y}^t = \arg\max_{j\in[K]} (\mathbf{w}_j^t \mathbf{x}^t)$ and defines a distribution P^t over the class labels as described in eq (3). Now, it samples a label $\tilde{y}^t$ randomly from P^t. It receives noisy bandit feedback f_ρ^t. We find $h(f_\rho^t)$ and update as $W^{t+1} = W^t + H^t$, where

$$H_{r,j}^t = x_j^t \left(\frac{h(f_\rho^t)\mathbb{I}[\tilde{y}^t = r]}{P^t(r)} - \mathbb{I}[\hat{y}^t = r] \right). \tag{5}$$

H^t has two sources of randomness, namely, $\tilde{y}^t$ (randomness used in the RCNBF algorithm) and f_ρ^t (randomness due to noise). Lemma 2 shows that the update matrix H^t used in RCNBF is an unbiased estimator of the matrix U^t (used in multiclass Perceptron), described in eq (2).

Lemma 2. *Suppose H^t be the update matrix as defined in eq (5) and let U^t be the matrix as defined in eq (2). Then, $\mathbb{E}_{\tilde{y}^t, f_\rho^t}[H^t] = U^t$, where $\mathbb{E}_{\tilde{y}^t, f_\rho^t}[H^t]$ is the expected value conditioned on $y^1, \cdots, y^{t-1}$.*

We keep repeating these steps for T trials. Complete details of the approach are given in Algorithm 1.

Mistake Bound Analysis of RCNBF

In this section, we derive the mistake bound for the RCNBF (Algorithm 1). To do that, we first show that the expected value of the norm of H^t is bounded.

Lemma 3. *Let H^t be defined as in eq (5) and $\beta = 1 - \rho_0 - \rho_1$. Then,*

$$\mathbb{E}_{\tilde{y}^t, f_\rho^t}[\left\|H^t\right\|^2] \leq \left\|\mathbf{x}^t\right\|^2 \left(A_1 \mathbb{I}[y^t \neq \hat{y}^t] + A_2 \mathbb{I}[y^t = \hat{y}^t] \right)$$

where $A_1 = \frac{2K}{\gamma} + \frac{2\rho_0(1-\rho_0)K}{\beta\gamma} + \frac{K\rho_1}{\beta^2\gamma} + \frac{\rho_0(1-\rho_0)K^2}{\beta^2\gamma^2}$, $A_2 = 2\gamma + \frac{\rho_1}{\beta^2(1-\gamma)} + \frac{\rho_0(1-\rho_0)K^2}{\beta^2\gamma}$.

[2] All the omitted proofs can be found in the supplementary material.

Algorithm 1 Robust Classifier for Noisy Bandit Feedback (RCNBF)

Input: $\gamma \in (0, 0.5)$, $\rho_0, \rho_1 : \rho_0 + \rho_1 < 1$
Initialize: Set $W^1 = 0 \in \mathbb{R}^{K \times d}$

for $t = 1, 2, \cdots, T$ **do**
Receive $\mathbf{x}^t \in \mathbb{R}^d$.
Set $\hat{y}^t = \arg\max_{r \in [K]} (\mathbf{w}_r^t \cdot \mathbf{x}^t)$
Set $P^t(r) = (1-\gamma)\mathbb{I}[r = \hat{y}^t] + \frac{\gamma}{K}$, $\forall r$
Randomly sample $\tilde{y}^t$ according to P^t.

Predict $\tilde{y}^t$ and receive feedback f_ρ^t
Calculate $h(f_\rho^t)$ using

$$h(f_\rho^t) = \frac{(1-\rho_{f_\rho^{t'}}) f_\rho^t - \rho_{f_\rho^t} f_\rho^{t'}}{1-\rho_0-\rho_1}$$

Compute $H^t \in \mathbb{R}^{K \times d}$ such that

$$H_{r,j}^t = x_j^t \left(\frac{h(f_\rho^t)\mathbb{I}[\tilde{y}^t = r]}{P^t(r)} - \mathbb{I}[\hat{y}^t = r] \right)$$

Update: $W^{t+1} = W^t + H^t$
end for

Algorithm 2 RCNBF with Implicit Noise Estimation (RCINE)

Input: $\gamma \in (0, 0.5)$, N_s
Initialize: $W^1 = 0 \in \mathbb{R}^{K \times d}, \hat{\rho}_0 = \hat{\rho}_1 = 0, \mathcal{S}$

for $t = 1, 2, \cdots, T$ **do**
Receive $\mathbf{x}^t \in \mathbb{R}^d$.
Set $\hat{y}^t = \arg\max_{r \in [K]} (\mathbf{w}_r^t \cdot \mathbf{x}^t)$
Set $P^t(r) = (1-\gamma)\mathbb{I}[r = \hat{y}^t] + \frac{\gamma}{K}$, $\forall r$
Randomly sample $\tilde{y}^t$ according to P^t.
Predict $\tilde{y}^t$ and receive feedback f_ρ^t
Calulate $h(f_\rho^t)$ using

$$h(f_\rho^t) = \frac{(1-\hat{\rho}_{f_\rho^{t'}}) f_\rho^t - \hat{\rho}_{f_\rho^t} f_\rho^{t'}}{1-\hat{\rho}_0-\hat{\rho}_1}$$

Define $H^t \in \mathbb{R}^{K \times d}$ such that

$$H_{r,j}^t = x_j^t \left(\frac{h(f_\rho^t)\mathbb{I}[\tilde{y}^t = r]}{P^t(r)} - \mathbb{I}[\hat{y}^t = r] \right)$$

Update: $W^{t+1} = W^t + H^t$
Data: Push $\{(\mathbf{x}^t, \tilde{y}^t), f_\rho^t\}$ in $\mathcal{S}$
if $t \% N_s == 0$ **then**
$\hat{\rho}_0, \hat{\rho}_1 = \text{NREst}(\mathcal{S})$, Clear $\mathcal{S}$
end if
end for

Note that the norm of the matrix H^t is inversely proportional to $\beta = 1-\rho_0-\rho_1$. Thus, if the noise rate increases, the upper bound on the norm of H^t will increase. We now find the expected mistake bound of the RCNBF algorithm.

Theorem 1 (Mistake Bound). *Let $\mathbf{x}^1, \cdots, \mathbf{x}^T$ be the sequence of examples presented to the RCNBF in T trials. Let, $\|\mathbf{x}^t\| \leq 1, \forall t \in [T]$ and $y^t \in [K]$. Let $R_H = \sum_{t=1}^T L_H(W^*; (\mathbf{x}^t, y^t))$ and $D = \|W^*\|_F^2 = \sum_{r=1}^K \sum_{j=1}^d (W_{i,j}^*)^2$ be the cumulative hinge loss and the complexity of any matrix, W^*. Let ρ_0 and ρ_1 be the noise parameters. Then the expected number of mistakes made by RCNBF is upper bounded as $\mathbb{E}[M] \leq R_H + \sqrt{A_1 D R_H} + 3\max\left\{A_1 D, \sqrt{A_2 D T}\right\} + \gamma T$. Here, expectation is with respect to all the randomness of the algorithm.*

Before moving, let us find the optimal value for the exploration-exploitation parameter γ and the corresponding mistake bound.

Corollary 1. *(Zero Noise Case, $\rho_0 = \rho_1 = 0$) In this case the mistake bound of* RCNBF *is of the order $O(\sqrt{T})$ which can be obtained by setting $\gamma = O(T^{-1/2})$.*

Algorithm 3 Noise Rate Estimator (NREst)

Input: $\mathcal{S} = \{(\mathbf{x}^t, \tilde{y}^t), f_\rho^t) \ : \ t = 1 \dots T\}$

Train a network using $\mathcal{S}$ which approximates $q(\mathbf{x}, \tilde{y}) = \hat{p}(f_\rho = 1|\mathbf{x}, \tilde{y})$

Find $\mathbf{x}^j = \arg\max_{\mathbf{x} \in \mathcal{X}} \ \hat{p}(f_\rho = 1|\mathbf{x}, \tilde{y} = j), \ j \in [K]$

Set $1 - \rho_1 = \hat{p}(f_\rho = 1|\mathbf{x}^l, \tilde{y} = l)$ and $\rho_0 = \hat{p}(f_\rho = 1|\mathbf{x}^k, \tilde{y} = l)$

Output: ρ_0, ρ_1

Corollary 2. *(High Noise Case, $\rho_0, \rho_1 \leq \min\left\{0.5, O(\sqrt{\frac{D}{T}})\right\}$) In this case, we obtain the bound $\mathbb{E}[M] \leq O(\sqrt{DT}\beta^{-1})$ for $\gamma = O(\sqrt{\frac{D}{\beta^2 T}})$.*

Corollary 3. *(Very High Noise Case, $\rho_0, \rho_1 \leq 1$) In this case the mistake bound of is $O(T^{2/3}\beta^{-1})$ for $\gamma = O(T^{-1/3}\beta^{-1})$.*

We see that the above mistake bound is inversely proportional to β, *i.e.*, as we increase the noise rate, the mistake bound will increase, which is as expected and also aligns with the batch mode algorithm in the presence of label noise [20].

Noise Rate Estimation

Here, we propose an approach for estimating ρ_0 and ρ_1 which uses ideas presented in [22,16]. The proposed approach is based on the following Theorem.

Theorem 2. *Assume that*

1. *There exist at least one "perfect example" for every class $j \in [K]$. Which means, there exists $\mathbf{x}_j^* \in \mathcal{X}$ (prefect example for class j) such that $p(\mathbf{x}_j^*) > 0$ and $p(y = \tilde{y}|\mathbf{x}_j^*, \tilde{y} = j) = p(y = j|\mathbf{x}_j^*) = 1$.*
2. *There exist sufficient corrupted examples to estimate $p(f_\rho|\mathbf{x}, \tilde{y} = l)$ accurately.*

Then it follows that $1 - \rho_1 = p(f_\rho = 1|\mathbf{x}_l^*, \tilde{y} = l), \ l \in [K]$ and $\rho_0 = p(f_\rho = 1|\mathbf{x}_k^*, \tilde{y} = l), \ l \neq k$, where $\mathbf{x}_l^*$ and $\mathbf{x}_k^*$ are perfect examples of class l and k.

Theorem 2 assumes that for every class $j \in [K]$, there exists a perfect example $\mathbf{x}_j^*$ such that $p(f = 1|\mathbf{x}_j^*, \tilde{y} = j) = p(y = j|\mathbf{x}_j^*) = 1$. We use this idea to estimate the noise rates as follows. We use the data generated by RCNBF under noisy bandit feedback setting. Using this, we create a training set $\mathcal{S}$ with following sequence of examples $\{(\mathbf{x}^t, \tilde{y}^t), f_\rho^t\}$ for $t = 1 \dots N_s$. Note that the input to the network is $\mathbf{x}^t$ concatenated with $\tilde{y}^t$. This is the major difference with the noise rate estimation presented in [22]. We use $\mathcal{S}$ to train a neural network with a output layer of size 2 and softmax as the activation function of the output layer. Our classification problem is binary however following [24], we prefer to use softmax with one-hot output instead of sigmoid as it allows the network to learn non-convex boundaries. This network approximates $q(\mathbf{x}, \tilde{y}) = \hat{p}(f_\rho = 1|\mathbf{x}, \tilde{y})$. Now we find perfect example for each class. A perfect example $\mathbf{x}_j^*$ for class j is the one for which $\hat{p}(y = j|\mathbf{x}_j^*) = \hat{p}(f_\rho = 1|\mathbf{x}_j^*, \tilde{y} = j) = 1$. We can find $\mathbf{x}_j^*$ as

$$\mathbf{x}_j^* = \arg\max_{\mathbf{x} \in \mathcal{S}} \ \hat{p}(f_\rho = 1|\mathbf{x}, \tilde{y} = j), \ j \in [K] \tag{6}$$

Table 1: Estimated noise rates (rounded to 3 decimal digits)

Actual Noise Rates		Estimated Noise Rates					
		MNIST		USPS		Fashion-MNIST	
ρ_0	ρ_1	$\hat{\rho}_0$	$\hat{\rho}_1$	$\hat{\rho}_0$	$\hat{\rho}_1$	$\hat{\rho}_0$	$\hat{\rho}_1$
0.000	0.000	0.063	0.029	0.017	0.000	0.090	0.004
0.150	0.150	0.172	0.147	0.181	0.153	0.189	0.140
0.250	0.250	0.248	0.264	0.258	0.257	0.264	0.259
0.200	0.400	0.211	0.439	0.194	0.419	0.215	0.393
0.400	0.200	0.400	0.260	0.393	0.229	0.404	0.222
0.400	0.400	0.403	0.508	0.402	0.515	0.397	0.502

Now, we can approximate $\hat{\rho}_0$ and $\hat{\rho}_1$ as $1 - \hat{\rho}_1 = \hat{p}(f_\rho = 1|\mathbf{x}_l^*, \tilde{y} = l)$ and $\hat{\rho}_0 = \hat{p}(f_\rho = 1|\mathbf{x}_k^*, \tilde{y} = l)$. The noise estimation approach is described in Algorithm 3.

Learning using Noisy Bandit Feedback with Implicit Noise Rate Estimation

RCNBF (Algorithm 1) runs under the online setting while NREst (Algorithm 3) is a batch algorithm. With the help of the above two algorithms, we are proposing a pseudo online mode algorithm, RCNBF with Implicit Noise Estimation(Algorithm 2), which runs under the online setting. The RCINE Algorithm[3] uses RCNBF to make predictions and generate dataset $\mathcal{S}$ for Noise Estimation. After every N_s trails, the algorithm updates the estimated noise rate parameters by running the NREst algorithm on the collected dataset $\mathcal{S}$. The crux of this setup is that the RCNBF will run in the online mode, while NREst, which is running parallelly at the same time, will estimate the noise rates parameter $\hat{\rho}_0$ and $\hat{\rho}_1$ and update them repetitively after a small interval of time.

4 Experimentation

We do experiments on various real-world as well as synthetic datasets. The synthetic dataset is called SynSep. SynSep is a 9-class, 400-dimensional synthetic data set of size 10^5. While constructing SynSep, we ensure that the dataset is linearly separable. For more detail about the dataset, one can refer to [14]. We also perform experiments on MNIST and Iris datasets from UCI repository [9], USPS dataset[4] and Fashion-MNIST for image classification [25].[5].

Feature Extraction for Fashion-MNIST dataset: We first randomly sampled $35,000$ images from the dataset for feature extraction and trained a four-layer convolutional neural network. The first layer is a convolutional layer with 32 feature maps having a size of 3x3 and a stride of 1. It takes an input of 28 x 28 grayscale images. The convolutional layer is followed by a max-pooling layer

[3] The complete code for all the experiments can be found here.

[4] https://www.kaggle.com/bistaumanga/usps-dataset

[5] The results and further discussion for SynSep and IRIS dataset are included in the supplementary file due to the space restrictions

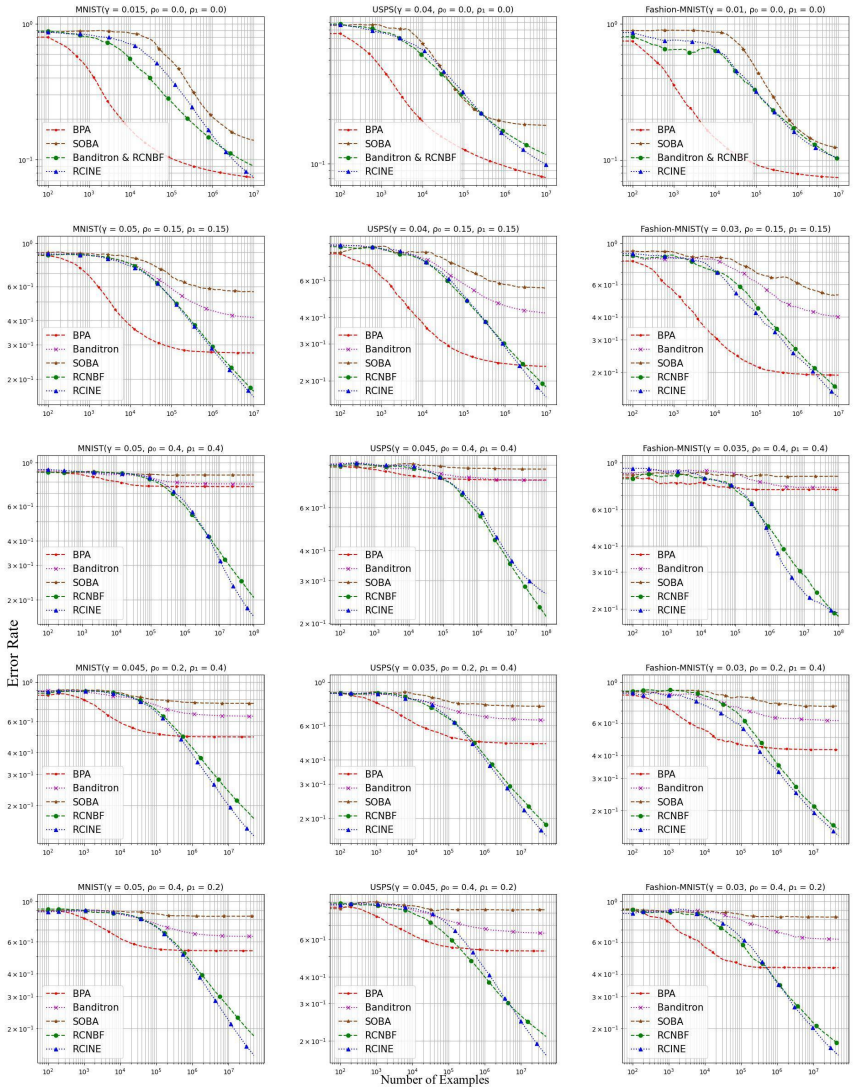

Fig. 2: Average error rates of RCNBF, RCINE and other benchmarking algorithms under noise-free case (first row; $\rho_0 = \rho_1 = 0$), low noise case (second row; $\rho_0 = \rho_1 = 0.15$), high noise case (third row; $\rho_0 = \rho_1 = 0.40$) and mixed noise case (fourth row; $\rho_0 = 0.2, \rho_1 = 0.4$ and fifth row; $\rho_0 = 0.4, \rho_1 = 0.2$). Three datasets are used (left to right): MNIST, USPS and Fashion-MNIST.

having 2x2 as pool size. The next layer is a fully-connected layer with 100 units and a dropout of the probability of 0.2. The last layer is a fully connected softmax layer. To extract features, we took the output of the fully connected layer of size

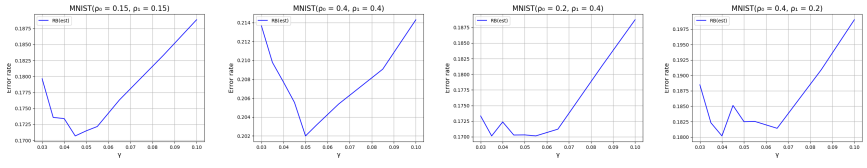

Fig. 3: Average error rates of RCINE against parameter's value γ under different noise rate setting on MNIST.

100. By experimenting on this dataset, we show that our approach can also be used for learning classifiers for complex datasets.

Benchmark Algorithms and Noise Rate Setting: We present experimental comparisons of our proposed algorithms (RCNBF and RCINE) with Banditron [14], Bandit Passive Aggressive [26] and Second Order Banditron Algorithm [4]. Five different settings of noise rate are used. These are (a) $\rho_0 = \rho_1 = 0.0$, (b) $\rho_0 = \rho_1 = 0.15$, (c) $\rho_0 = \rho_1 = 0.4$, (d) $\rho_0 = 0.2, \rho_1 = 0.4$ and (e) $\rho_0 = 0.4, \rho_1 = 0.2$. On each of the different noise setting, we ran our proposed algorithm, RCNBF (using original noise rates) and RCINE (with initial value of $\hat{\rho}_0 = \hat{\rho}_1 = 0$). For updating the noise rates parameter, the RCINE algorithm, runs the NREst algorithm after N_s trails on the collected dataset $\mathcal{S}$. NREst algorithm uses a neural network to estimate the noise rates. Table 1 shows the results of estimation of noise rates at an intermediate instance of RCINE algorithm.

In NREst algorithm, train-test ratio of 90:10 is taken. Cross-entropy loss is chosen for comparison. 10% of the training set is used for validation. The mini-batch size used for training is 128. The activation function for all the network is ReLU and optimizer is AdaGrad [10] with initial learning rate 0.01 and $\delta = 10^{-6}$. After training, we apply the estimator to find $\hat{\rho}_0, 1-\hat{\rho}_0, \hat{\rho}_1$ and $1-\hat{\rho}_1$ on $\mathcal{S}$. Then we normalize the values of $\hat{\rho}_0, 1-\hat{\rho}_0$ and $\hat{\rho}_1, 1-\hat{\rho}_1$ such that they sum up to 1. From [19,22] we know that the sample maximum is susceptible to the outliers, so instead of argmax eq (6), we take 89%-percentile.

For *MNIST dataset*, the architecture consists of two dense hidden layers of size 128 with a dropout of the probability of 0.2. We train the network for 70 epochs. For the next set of experiments, we consider the *USPS dataset*. We trained an architecture with three dense hidden layers of $32, 256$, and 32 respectively, with a dropout of probability 0.2 for 70 epochs. Lastly, for *Fashion-MNIST dataset*, the architecture consists of three dense layers of size $32, 128$ and 32 respectively with a dropout of probability 0.2 and is trained for 70 epochs.

Parameter Selection: For each dataset and each different noise setting, simulations for RCINE are run for a wide range of values of the exploration parameter, γ. [6] For MNIST dataset, γ exploration results are shown in Figure 3. We choose the γ value for which the minimum error rate is achieved.

[6] The value of γ as shown in the figure are for RCINE. For other algorithms, the optimal value of γ is chosen.

Results: We ran our proposed algorithms (RCNBF and RCINE) and compared the average [7] error rate with other benchmark algorithms as shown in Fig 2. For better visualization of the asymptotic bounds, we plotted the result on a log-log scale. It shows that in the presence of noise, the final error rate of RCINE and RCNBF is significantly better than SOBA, BPA, and Banditron. While all other algorithms converge, RCNBF and RCINE are still learning and yet to converge.

Analysis of Fig. 2 shows that as the number of examples grows, the slope of the error rate of RCNBF and RCINE under all different settings of noise rate is comparable to that of SOBA, BPA, and Banditron for the noise-free (0%) setting. The final error rate of RCNBF and RCINE under all different noise rate settings is also close to SOBA, BPA, and Banditron under the noise-free setting. RCINE performs comparably to RCNBF for all the datasets and noise settings. This happens as we can efficiently estimate the noise rates.

5 Conclusion and Future Work

In this paper, we proposed a noisy bandit feedback setting in online multiclass classification, which can effectively incorporate the noise present in real-world data. We proposed a novel algorithm based on the unbiased estimation technique, which enjoys a favorable bound (both theoretically and practically) under the proposed noisy bandit feedback setting. The proposed algorithm is robust to the noisy bandit feedback and can learn the true hypothesis in the presence of noise. We also propose a technique to estimate the noise rate, thus providing an end-to-end framework. Experimental comparisons on various datasets with benchmarking algorithms show that RCNBF and RCINE are comparable to other algorithms under noise-free bandit feedback settings but far better than others under noisy bandit feedback settings.

References

1. Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.
2. Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.
3. Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, January 2003.
4. Alina Beygelzimer, David Pal, Balazs Szorenyi, Devanathan Thiruvenkatachari, Chen-Yu Wei, and Chicheng Zhang, editors. *Bandit Multiclass Linear Classification: Efficient Algorithms for the Separable Case*, Proceedings of the 36th International Conference on Machine Learning ICML, 02 2019.
5. Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
6. Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. volume 90, pages 273–280, 01 2011.

[7] Note that here averaging is done over ten independent simulations of the algorithm

7. Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.
8. Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, March 2003.
9. Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
10. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.
11. Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.
12. Elad Hazan and Satyen Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24, pages 891–899. Curran Associates, Inc., 2011.
13. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.
14. Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pages 440–447, 2008.
15. Sayash Kapoor, Kumar Kshitij Patel, and Purushottam Kar. Corruption-tolerant bandit learning. *Machine Learning*, 108(4):687–715, 2019.
16. Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.
17. Caiyun Ma and Hong Zhang. Effective handwritten digit recognition based on multi-feature extraction and deep analysis. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 297–301. IEEE, 2015.
18. Andrew McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI workshop on Text Learning*, pages 1–7, 1999.
19. Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International Conference on Machine Learning*, pages 125–134, 2015.
20. Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.
21. Guobin Ou and Yi Lu Murphey. Multi-class pattern classification using neural networks. *Pattern Recognition*, 40(1):4–18, 2007.
22. Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.
23. Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
24. Sarath Sivaprasad, Naresh Manwani, and Vineet Gandhi. The curious case of convex networks. *arXiv preprint arXiv:2006.05103*, 2020.
25. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
26. Hongliang Zhong and Emmanuel Daucé. Passive-aggressive bounds in bandit feedback classification. *Proceedings of the ECMLPKDD*, pages 255–264, 2015.