Online label aggregation

A variational bayesian approach

Hong, Chi; Ghiassi, Amirmasoud; Zhou, Yichi; Birke, Robert; Chen, Lydia Y.

DOI
10.1145/3442381.3449933

Publication date
2021

Document Version
Final published version

Published in
The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021

Online Label Aggregation: A Variational Bayesian Approach

Chi Hong
Delft University of Technology
Delft, Netherlands
C.Hong@tudelft.nl

Amirmasoud Ghiassi
Delft University of Technology
Delft, Netherlands
S.Ghiassi@tudelft.nl

Yichi Zhou
Tsinghua University
Beijing, China
zhouyc15@mails.tsinghua.edu.cn

Robert Birke
ABB Research
Switzerland
robert.birke@ch.abb.com

Lydia Y. Chen
Delft University of Technology
Delft, Netherlands
lydiaychen@ieee.org

ABSTRACT

Noisy labeled data is more a norm than a rarity for crowd sourced contents. It is effective to distill noise and infer correct labels through aggregating results from crowd workers. To ensure the time relevance and overcome slow responses of workers, online label aggregation is increasingly requested, calling for solutions that can incrementally infer true label distribution via subsets of data items. In this paper, we propose a novel online label aggregation framework, BiLA , which employs variational Bayesian inference method and designs a novel stochastic optimization scheme for incremental training. BiLA is flexible to accommodate any generating distribution of labels by the exact computation of its posterior distribution. We also derive the convergence bound of the proposed optimizer. We compare BiLA with the state of the art based on minimax entropy, neural networks and expectation maximization algorithms, on synthetic and real-world data sets. Our evaluation results on various online scenarios show that BiLA can effectively infer the true labels, with an error rate reduction of at least 10 to 1.5 percent points for synthetic and real-world datasets, respectively.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

online, label aggregation, variational bayesian inference, stochastic optimizer, convergence bound

ACM Reference Format:
Chi Hong, Amirmasoud Ghiassi, Yichi Zhou, Robert Birke, and Lydia Y. Chen. 2021. Online Label Aggregation: A Variational Bayesian Approach. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449933

1 INTRODUCTION

Crowd sourcing platforms provide economic and efficient means to curate datasets which are deemed the new oil for today's artificial intelligence [11]. One of commonly seen crowd tasks is to classify contents, e.g., web pages [27], and images [33], and to provide labels of their respective classes. However, due to differences in the crowd workers' background and experience, the resulting labels of the same content often vary across workers, including missing labels – so called noisy labels. The state of the practise [9, 34, 36] to distill the quality of crowd sourced labels is to aggregate them across all workers and reach consensus for every content. Such a curated dataset can then conveniently power up a wide range of supervised machine learning models for further analysis, e.g., object detection [8], search engine [27], and disease diagnoses [12].

The velocity of knowledge discovery indeed hinges on the speed of data curation [3]. Faster the data is aggregated via crowd sourcing, the more insights can be extracted through machine learning models. For example [20], via instantaneous information from Amazon Mechanical Turk, i.e., in 2200 ms, the accuracy of predicting urban emergencies can be improved by 40%. Moreover, labelling massive datasets come as a daunting tasks requiring months or years of effort. The estimated effort to label ImageNet for a single person working 24/7 is 19 years, but even with crowd sourcing involving 25K workers it still took 21 months [6]. It becomes increasingly imperative that label curation and aggregation can be conducted in online manner [37], i.e., labels can be continuously aggregated over a subset of content, instead of the entire content at once. More, recent privacy and governmental policies [1] regulate the data storage time, asking for prompt action of aggregation.

The key challenge behind online label aggregation is how to utilize a partial label set from workers that only includes a small chunk of content. Existing aggregation methods [4, 36, 39] focus on the quality issues across workers but implicitly overlook the temporal aspect, i.e., timely and accurately label aggregation from online data. In other words, the prior art tailors for offline scenarios, which assumes the availability of all contents at once. As a result, in the online scenario, such approaches end up greedily optimizing for only the available subset, without the global optimization for the entire dataset. The need of online label aggregation thus calls for a novel stochastic optimization scheme which can handle batches of observable data.

Probabilistic graphic models [15] are commonly adopted to aggregate noisy labels from crowd workers without the label ground truths. Their objective is to maximize likelihood of the observed data by capturing the dependency on latent variables, e.g., the true labels and confusion matrix that specifies the generation process

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449933

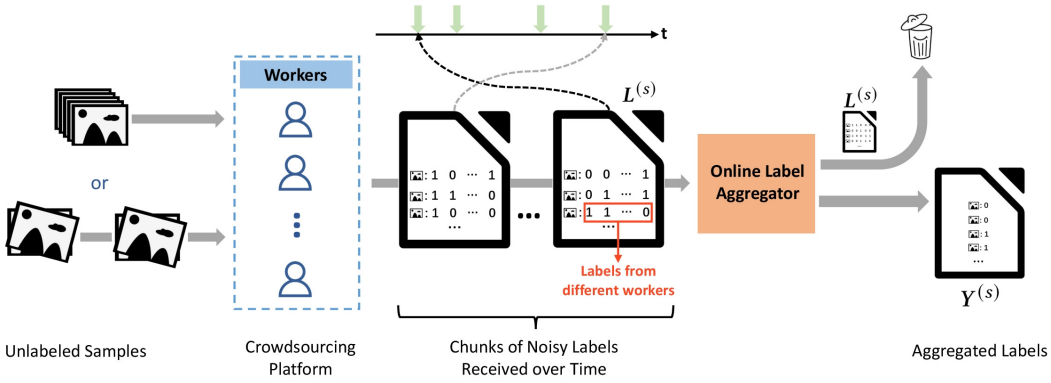


Figure 1: The online label aggregation scenario.

of label noise. Variational Bayesian inference methods [17, 28, 31] can effectively infer the latent features by maximizing the evidence lower bound [2] of the log data likelihood of the observed data. The other popular approach to infer latent variables is Expectation-Maximization (EM) algorithm [4] that has different objectives in expectation and maximization steps - an additional hurdle for stochastic optimization. While variational inference methods have an advantage of single objective for stochastic optimization, the challenges lies in deriving a tracktable posterior distribution of the generation process of label noise.

In this paper, we propose a novel online label aggregation framework, BILA , based on incremental variational Bayesian Inference method. BILA aggregates noisy labels from crowd workers incrementally upon receiving a subset of labeled items via a novel stochastic optimization scheme. To maximize the log likelihood of observed items, BILA minimizes the Kullback-Leibler (K-L) divergence between (i) the noisy label generative distribution p, and (ii) the approximate distribution q. The unique features of BILA are (i) flexibility and extendibility for generative distribution, (ii) exact computation of posterior distribution bypassing the need of the closed from expression, and (iii) the proposed objective function has the exact expression of the expectation term of K-L divergence, avoiding the approximation variance. Using the framework of BILA , we define a label aggregation model for multiple classes, abbreviated as BILA-CM , based on confusion matrix. We employ multi-layer perceptron neural networks for approximate distribution q.

As the data chunks are received in an online fashion, BILA-CM is incrementally trained by data chunks. To such an end, we propose a stochastic optimization scheme - a variant of RMSProp [29]. It enhances RMSProp with a dynamic clip operator, bias-corrected second raw moment estimate and decaying learning rate.

We evaluate BILA on both real-world and synthetic datasets. We compare its aggregation error rates with the state of the art label aggregation algorithms, i.e., Majority Voting, E-M based approaches, neural network based approaches and Minimax Entropy based approaches. BILA is able to achieve significant error reduction in various online scenarios, i.e., different data chunk sizes. Our results also show that BILA is robust against different crowd sourcing scenarios, i.e., different number of workers, noise ratios, and label sparsity. In terms of effectiveness of proposed optimization scheme, we are able to achieve faster convergence than RMSProp, and in par with ADAM [14] but without risks of divergence.

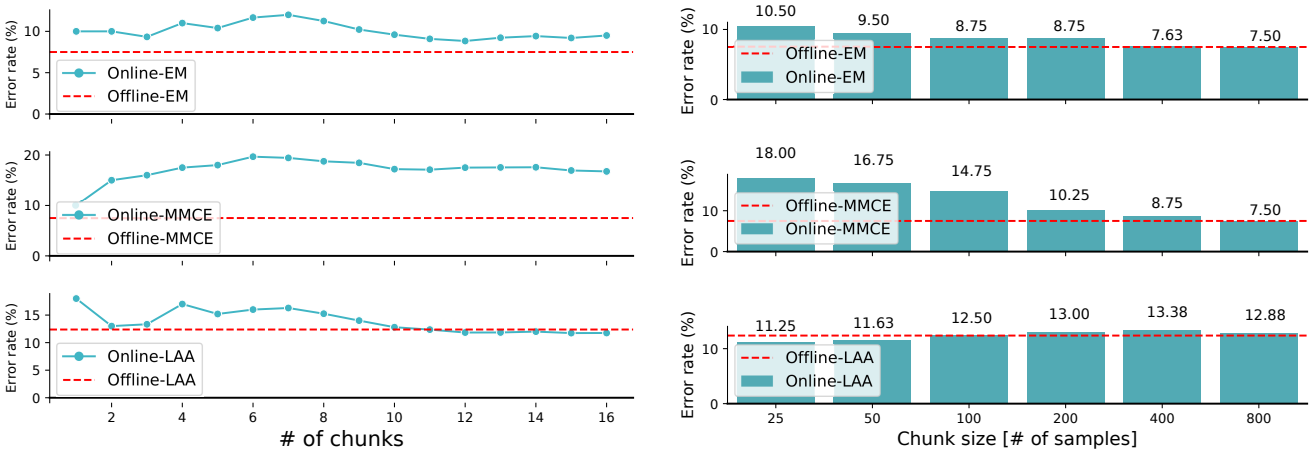
The contributions of this paper are summarized as follows.

- We design a flexible online label aggregation framework, BILA , based on variational Bayesian inference framework (§ 3). BILA uses neural networks for the approximate distribution guided by the generating distribution.
- We provide a confusion matrix based aggregation model, BILA-CM , which outperforms existing algorithms based on EM algorithms, Minimax Entropy and neural networks (§ 5).
- We design a stochastic optimizer and derive its convergence bound (§ 4). Last, we extensively compare BILA against representative label aggregation methods on different online crowd sourcing scenarios (§ 5)

2 SYSTEM SCENARIOS

To overcome the data labelling challenges it is common practise to label datasets via crowd sourcing by non-experts. We can assign unlabeled instances to the workers in two ways: offline and online. In offline mode we publish all unlabeled instances once on a crowdsourcing platform and wait for all workers to complete their assignments before training the label aggregation algorithm. This works well if the data does not change over time and is all available at once. In online mode, shown in Figure 1, we continuously publish single or multiple instances of unlabelled data on the crowdsourcing platform. Then we collect the labelling results and organize them in small chunks of redundant noisy labels. These chunks are fed one-by-one over time to the label aggregation algorithm to update the label aggregator. The processed redundant noisy labels are discarded and only the aggregated, i.e. inferred true, labels are kept. This enables continuous learning but requires the label aggregation algorithm to be be able to (incrementally) learn from small sets of data. This is challenging. Most state-of-the-art label aggregation techniques do not cope well with such a requirement.

We demonstrate this via a motivation example. We run three baseline aggregation algorithms in both online and offline mode



(a) Error rate after aggregating each chunk (chunk size:50 samples)

(b) The effect of the chunk size

Figure 2: Motivation comparison on the RTE dataset: offline v.s. online aggregation.

and compare the achieved error rates. We consider Expectation-Maximization (EM) [4], Min-Max Conditional Entropy (MMCE) [39] and Label Aware Autoencoders (LAA) [36] on the RTE dataset (details given in §5.1). In online mode, we feed each aggregation algorithm with small chunks of 50 redundant noisy labels at a time. Each chunk is used to update the aggregator. We stop at 16 chunks (800 samples). At each step we evaluate the achieved error rate. After each update we use the aggregator to infer the aggregated label for each sample and compute the percentage of samples for which the aggregated label differs from the ground truth label. Note that label aggregation is an unsupervised learning task. The ground truth labels are used only to compute the error rate, not to train the aggregator. Figure 2(a) shows the step-wise error rate for the three methods. Instead, Figure 2(b) shows the sensibility of each method to the chunk size. Each plot reports the achieved error rate when processing 800 samples in chunks of different size. For reference we report the offline performance, i.e. processing all 800 samples at once, as a horizontal line.

EM is commonly used to estimate a confusion matrix for each worker. MMCE is designed to discern the confusion matrices across the workers as well as the instances. Both can not be readily adapted to learn incrementally from small sets of data. EM uses majority voting results to determine a good starting point for the parameter search. Hence the best starting point is different for each chunk. MMCE assigns model parameters to each sample. Since each chunk has different samples, we can not keep the learned parameters. We use this two methods in a sliding window style where each window is a new chunk of data. As a result, EM and MMCE maximize the data likelihood of the current chunk not the full data. Hence, the error rates of these two methods do not converge with time to the offline error rate, i.e. processing the whole data at once. More in detail, the performance of CE (see top plot Figure 2(a)) initially oscillates but then flattens out. After 16 chunks, i.e. all 800 samples, the error rate is still 2 percent points higher. MMCE is worse (see middel plot Figure 2(a)). The error rate first diverges before flattening out leaving a gap of 9.25 percent points after the last chunk. This is because MMCE is a generative model which needs to train a larger number of parameters compared to EM. This makes MMCE more sensible to the chunk size. This is clearly shown in Figure 2(b). The performance of both EM and MMCE (top and middle plot) benefit from processing larger chunks sizes. With chunks of 25 samples, EM is 3 percent point worse than offline, but starting at chunk size 400 EM is able to equal the offline performance. Instead, MMCE is more sensible. At chunk size 25 the gap is 11.5 percent points. The gap diminishes with increasing chunk sizes, but it never reaches the same performance as offline. Note that chunk size 800 is equivalent to offline.

LAA is a neural network based method inspired from autoencoders. This method can be used incrementally so that its optimization goal maximizes the data likelihood of the full dataset, not only the chunk. Consequently, online LAA nicely converges to the offline results over time (bottom plot Figure 2(a)). The small difference between the two is due to the stochasticity of the training. For the same reason the sensibility of this algorithm against different chunk sizes is low (bottom plot Figure 2(b)). After processing 800 samples with different chunk sizes LAA achieves final error rates within ±1 percent points of the offline performance. However this method leads generally to worse results. The best result achieved by LAA is 11.75% error rate compared to 7.5% for EM and MMCE. LAA does not have a probabilistic model to describe the generative process of the observed noisy labels. This harms its performance. Besides LAA needs approximate approaches to calculate the expectation terms in the loss function. Our proposed label aggregation model BILA-CM directly addresses these issues achieving superior performance in both offline and online mode.

3 ONLINE LABEL AGGREGATION

We consider the online learning scenario shown in Figure 1. Each data instance $\boldsymbol{l}_i = \{l_{i1}, ..., l_{iK}\}$ contains redundant noisy labels of sample i. These noisy labels are provided by K workers. $l_{ik} \in \boldsymbol{C}$

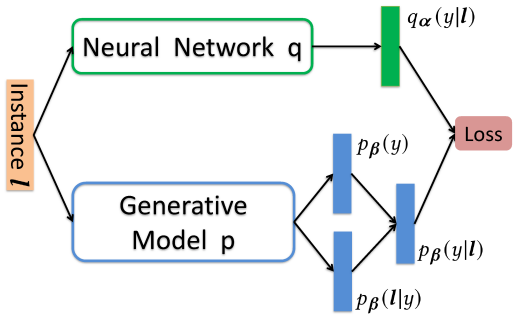


Figure 3: The relationship between q, p and the loss function.

denotes the label of item i given by worker $k \in \{1, ..., K\}$, where $\boldsymbol{C} = \{1, ..., C\}$ is the set of the possible classes. Not every worker might label all samples. If sample i is not labeled by worker k, then the value of l_{ik} is -1. We use $y_i \in \boldsymbol{C}$ to represent the unknown true label of a sample. Data instances stream into the label aggregation model at different times in small sets $L^{(s)}$. We call the small sets as chunks. In our online learning setting, our task is to continuously infer the values of $\{y_i | i \in L^{(s)}\}$ for the current chunk $L^{(s)}$ in real time before receiving the next chunk.

In order to define our optimization goal, we need some additional notations. We use L to represent the collection of all observed noisy labels with N instances, i.e. $\boldsymbol{L} = \{\boldsymbol{l}_1, ..., \boldsymbol{l}_N\}$, and $\boldsymbol{Y} = \{y_1, ..., y_N\}$ for the collection of all the corresponding unknown true labels[1].

3.1 Variational Bayesian Inference Framework (BILA)

In this section, we introduce our label aggregation framework (BILA) and define our optimization goal. The framework aims to predict the unknown true label y_i of each instance i with the sole knowledge of the instance's redundant noisy labels $\boldsymbol{l}_i$. The framework includes two components: a neural network q and a generative model p trained using an optimization goal defined based on the principle of variational inference. From the perspective of variational Bayesian inference, q is an approximate distribution. The choice of q and p is very flexible. q can be a multilayer perceptron (MLP), a convolutional neural network (CNN), or any other neural network. p is the model to define how to generate the observed noisy labels $\boldsymbol{L}$. Since we use a neural network as approximate distribution to learn the label aggregation model, we need stochastic optimization to train the parameters in q and p. This requires that the loss function is differentiable respect to the model parameters in p. This is the only constraint of the definition of p. Besides, the close form of the posterior of model p is needless because we rewrite the expression of the Kullback-Leibler divergence to avoid using the posterior directly. The relationship between q, p and the loss function in BILA is shown in Figure 3.

3.1.1 Definition of q and p. The set of noisy labels $\boldsymbol{L} = \{\boldsymbol{l}_1, ..., \boldsymbol{l}_N\}$ only contains the observed instances $\boldsymbol{l}_i$. The corresponding true labels Y are unknown. The label aggregation task in this paper is to predict the unknown true labels given $\boldsymbol{L}$. So it is an unsupervised learning task.

Given an instance $\boldsymbol{l}_i$ of noisy redundant labels, we use a neural network q with softmax activation on the last layer to predict the corresponding unknown true label y_i. q can be represented as a probability distribution $q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})$, where $\boldsymbol{\alpha}$ denotes the neural network parameters. The output of the network is a C-dimensional vector $[q_{\boldsymbol{\alpha}}(y = c|\boldsymbol{l})]_{c=1}^{C}$, where the c-th element $q_{\boldsymbol{\alpha}}(y = c|\boldsymbol{l})$ is the probability that the true label of the input instance is class c. The predicted label is given by the element with the highest probability.

In order to train q we need an optimization goal. Therefore, we define a generative model p to describe the generative process behind the observed noisy labels $\boldsymbol{L}$. This way we can define a loss function to guide the training according to variational inference rules [2, 31]. p assumes that an instance $\boldsymbol{l}$ is generated from some conditional distributions $p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)$, where y denotes the unknown true label and $\boldsymbol{\beta}$ the parameters of model p. It further assumes that y is generated from a prior distribution $p_{\boldsymbol{\beta}}(y)$. The generative model p potentially defines a posterior distribution:

$$p_{\boldsymbol{\beta}}(y|\boldsymbol{l}) = \frac{p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)p_{\boldsymbol{\beta}}(y)}{p_{\boldsymbol{\beta}}(\boldsymbol{l})}. \tag{1}$$

We do not make many simplifying assumptions about p except that the loss function is differentiable with respect to $\boldsymbol{\beta}$.

3.1.2 Optimization Goal. To solve the unsupervised learning task where we only have the observed noisy labels $\boldsymbol{L}$ a reasonable optimization goal is to maximize the data likelihood of $\boldsymbol{L}$. In particular, we maximize the data log likelihood $\log p_{\boldsymbol{\beta}}(\boldsymbol{L})$ according to the evidence lower bound $\log p_{\boldsymbol{\beta}}(\boldsymbol{L}) = KL(q_{\boldsymbol{\alpha}}(\boldsymbol{Y}|\boldsymbol{L})||p_{\boldsymbol{\beta}}(\boldsymbol{Y}|\boldsymbol{L})) + L(q) \geq L(q)$, where $KL(\cdot)$ denotes the Kullback-Leibler divergence and $L(q) = \mathbf{E}_{q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)}\left[\log \frac{p_{\boldsymbol{\beta}}(\boldsymbol{L},\boldsymbol{Y})}{q_{\boldsymbol{\beta}}(\boldsymbol{Y})}\right]$. We can see that we can maximize the lower bound of $\log p_{\boldsymbol{\beta}}(\boldsymbol{L})$ as we minimize $KL(q_{\boldsymbol{\alpha}}(\boldsymbol{Y}|\boldsymbol{L})||p_{\boldsymbol{\beta}}(\boldsymbol{Y}|\boldsymbol{L}))$. So, we need to find the consensus between the predictions of q and p. Consequently, we use $KL(q_{\boldsymbol{\alpha}}(\boldsymbol{Y}|\boldsymbol{L})||p_{\boldsymbol{\beta}}(\boldsymbol{Y}|\boldsymbol{L}))$ as our loss function and minimize it during the training process.

We assume that each collected label is generated independently, i.e. instances in $\boldsymbol{L}$ are independent from each other. Plugging $q_{\boldsymbol{\alpha}}(\boldsymbol{Y}|\boldsymbol{L}) = \prod_i q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)$ and $p_{\boldsymbol{\beta}}(\boldsymbol{Y}|\boldsymbol{L}) = \prod_i p_{\boldsymbol{\beta}}(y_i|\boldsymbol{l}_i)$ into the loss function, we have:

$$\begin{aligned} KL(q_{\boldsymbol{\alpha}}(\boldsymbol{Y}|\boldsymbol{L})||p_{\boldsymbol{\beta}}(\boldsymbol{Y}|\boldsymbol{L})) &= \sum_{i=1}^{N} -\mathbf{E}_{q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)}\left[\log \frac{p_{\boldsymbol{\beta}}(y_i|\boldsymbol{l}_i)}{q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)}\right] \\ &= \sum_{i=1}^{N} KL(q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)||p_{\boldsymbol{\beta}}(y_i|\boldsymbol{l}_i)) \end{aligned} \tag{2}$$

Equation (2) cannot be directly used to train q and p, because the expression of $p_{\boldsymbol{\beta}}(y|\boldsymbol{l})$ is unknown. The exact expression of the posterior $p_{\boldsymbol{\beta}}(y|\boldsymbol{l})$ may be intractable. So we have to further rewrite

[1] To simplify notation we drop the subscript i when referring to a generic sample.

the loss function. According to (1), we have

$$\begin{aligned} KL(q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})||p_{\boldsymbol{\beta}}(y|\boldsymbol{l})) &= -\mathbf{E}_{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})}\left[\log\frac{p_{\boldsymbol{\beta}}(y|\boldsymbol{l})}{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})}\right] \\ &= -\mathbf{E}_{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})}\left[\log\frac{p_{\boldsymbol{\beta}}(y)}{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})} + \log p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)\right] + const \\ &= KL(q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})||p_{\boldsymbol{\beta}}(y)) - \mathbf{E}_{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})}\left[\log p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)\right] + const \end{aligned} \tag{3}$$

To simplify the notations, we use $\theta = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ to represent the parameters of both models in our framework. According to (2) and (3) the loss function is rewritten as

$$f(\theta; \boldsymbol{L}) = \frac{1}{N}\sum_{i=1}^{N}\{KL(q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)||p_{\boldsymbol{\beta}}(y_i)) - \mathbf{E}_{q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)}\left[\log p_{\boldsymbol{\beta}}(\boldsymbol{l}_i|y_i)\right]\} \tag{4}$$

where we ignore the constant term and the loss function is rescaled by $1/N$. This does not affect the optimization result.

3.1.3 Training. During training we solve the following optimization problem

$$\hat{\theta} = \arg\min_{\theta} f(\theta; \boldsymbol{L}).$$

This optimization problem is solved by stochastic first-order optimization. The update rule of the model parameters is shown in Algorithm 1. Following our discussions in § 2, in order to continuously aggregate small sets of noisy labels, we apply mini-batch training to update the parameters θ. So the loss function for training is

$$f(\theta; \boldsymbol{L}^{(M)}) = \frac{1}{M}\sum_{i=1}^{M}\left\{KL(q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)||p_{\boldsymbol{\beta}}(y_i)) - \mathbf{E}_{q_{\boldsymbol{\alpha}}(y_i|\boldsymbol{l}_i)}\left[\log p_{\boldsymbol{\beta}}(\boldsymbol{l}_i|y_i)\right]\right\}, \tag{5}$$

where $\boldsymbol{L}^{(M)}$ is a mini-batch sampled from current dataset $\boldsymbol{L}^{(s)}$, and M denotes the minibatch size. The gradient of $f(\theta; \boldsymbol{L}^{(M)})$ is required to update the model parameters. Before calculating the gradient, we need to define the expression of the KL divergence and the expectation term. The unobserved variables y_i are discrete variables that take values from 1 to C. Therefore, we have the following expressions

$$KL(q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})||p_{\boldsymbol{\beta}}(y)) = -\sum_{c=1}^{C} q_{\boldsymbol{\alpha}}(c|\boldsymbol{l})\log\frac{p_{\boldsymbol{\beta}}(c)}{q_{\boldsymbol{\alpha}}(c|\boldsymbol{l})}, \tag{6}$$

$$\mathbf{E}_{q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})}\left[\log p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)\right] = \sum_{c=1}^{C} q_{\boldsymbol{\alpha}}(c|\boldsymbol{l})\log p_{\boldsymbol{\beta}}(\boldsymbol{l}|c), \tag{7}$$

where $q_{\boldsymbol{\alpha}}(c|\boldsymbol{l})$ is the c-th element of the neural network output. Note that we do not require any approximation for calculating the expectation terms in our loss function. As such we avoid the problem of high variance of the loss function in stochastic Bayesian inference [22]. According to (6) and (7), the values of $f(\theta; \boldsymbol{L}^{(M)})$ and corresponding stochastic gradient $\nabla_\theta f(\theta; \boldsymbol{L}^{(M)})$ can be easily computed. This completes all necessary blocks in the framework to construct online label aggregation models.

3.2 Label Aggregation Model

In this subsection, we introduce our online label aggregation model, BiLA-CM , based on the BiLA framework. This model can be applied to aggregate discrete labels with noise. We also derive another model for binary label aggregation tasks, BiLA-WA , which is detailed in the Appendix B.

3.2.1 Model Definition. In order to define BiLA-CM and exact loss function f, we need to decide the concrete forms of q and p. We set q to be a fully connected neural network with a softmax activation function on the last layer. q takes an instance $\boldsymbol{l}$ as input and outputs a distribution $q_{\boldsymbol{\alpha}}(y|\boldsymbol{l})$, where $\boldsymbol{\alpha}$ denotes the neural network parameters.

p is a generative model describing the observed noisy labels $\boldsymbol{l}$. From (6) and (7), in order to compute the loss function and its gradient, we need to define the expressions of $p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)$ and $p_{\boldsymbol{\beta}}(y)$. Since every element in an instance is collected independently from different workers, we assume that the k-th element in an instance is generated from an independent distribution $\boldsymbol{\psi}_{ck}$ when the true label of the instance is c. This distribution is defined as

$$\boldsymbol{\psi}_{ck} = softmax(\boldsymbol{\omega}_{ck}), \tag{8}$$

where $\boldsymbol{\omega}_{ck}$ is a C-dimensional vector. Then $p_{\boldsymbol{\beta}}(\boldsymbol{l}|y = c)$ can be defined as

$$p_{\boldsymbol{\beta}}(\boldsymbol{l}_i|y_i = c) = \prod_{k\in S_i}\psi_{ck,l_{ik}}, c \in [C], \tag{9}$$

where $\psi_{ck,l_{ik}}$ is the l_{ik}-th element of $\boldsymbol{\psi}_{ck}$. Since the softmax function is derivable, $\boldsymbol{\omega}_{ck}$ can be updated by stochastic optimization. In this model, the prior distribution $p_{\boldsymbol{\beta}}(y)$ is a multinomial distribution estimated by

$$\hat{p}_{\boldsymbol{\beta}}(y = c) = \frac{\sum_i\sum_k \mathbf{I}(l_{ik} = c)}{\sum_i\sum_k \mathbf{I}(l_{ik} \neq -1)}, c \in [C], \tag{10}$$

where the values of the estimators can be calculated by counting the observed labels. Since $p_{\boldsymbol{\beta}}(y)$ is fixed, we introduce a hyperparameter ζ to constrain the Kullback-Leibler divergence term in the loss function (5). We regard this constrained term as a regularizer. Then, using (6) and (7) the mini-batch loss function used is

$$\begin{aligned} f(\theta; \boldsymbol{L}^{(M)}) = -\frac{1}{M}\sum_{i=1}^{M}\Big\{&\zeta\sum_{c=1}^{C} q_{\boldsymbol{\alpha}}(c|\boldsymbol{l}_i)\log\frac{p_{\boldsymbol{\beta}}(c)}{q_{\boldsymbol{\alpha}}(c|\boldsymbol{l}_i)} \\ &+ \sum_{c=1}^{C} q_{\boldsymbol{\alpha}}(c|\boldsymbol{l}_i)\log p_{\boldsymbol{\beta}}(\boldsymbol{l}_i|c)\Big\}, \end{aligned} \tag{11}$$

3.2.2 Online Model Training. The details of the complete online label aggregation model BiLA-CM are illustrated in Algorithm 1. BiLA-CM continuously receives a new noisy labels chunk $\boldsymbol{L}^{(s)}$ containing multiple redundant noisy label instances $\boldsymbol{l}$. Note that we do not require the size of each set to be equal. This increases the practicality of our algorithm. At the beginning, we accumulate few noisy label sets to construct an initial set $\boldsymbol{L}^*$. This initial set is used to initialize the model parameters $\boldsymbol{\beta} = \{\boldsymbol{\omega}_{ck}\}$ and the prior estimator $\hat{p}_{\boldsymbol{\beta}}(y)$. $\boldsymbol{\beta}$ can be initialized by its definition and majority voting on the noisy redundant labels from the initial set to predict the true labels. After initialization, we start the online aggregation. For each arriving chunk $\boldsymbol{L}^{(s)}$ at time step t, we update the model parameter θ. Then we aggregate each $\boldsymbol{l} \in \boldsymbol{L}^{(s)}$ using the updated

θ. The BILA-CM update and aggregation process is illustrated in function UpdateAndAggregate (lines 10-30). First we retrain the model by computing the terms of the loss function f from Equation (11) on each sampled mini batch (lines 14-21) before updating the model (lines 22-27).

Algorithm 1: Online Label Aggregation Model BILA-CM . The model parameters are $\theta = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, where $\boldsymbol{\alpha} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2\}$ and $\boldsymbol{\beta} = \{\boldsymbol{\omega}_{ck}\}$.

1 **Set:** learning rate $\mu > 0$, exponential decay rate $\gamma \in [0, 1)$, time step $t = 1$
2 **Input:** Continuously receive new noisy labels set $\boldsymbol{L}^{(s)} = \{\boldsymbol{l}\}$
3 Accumulate few sets to construct the initial set $\boldsymbol{L}^*$
4 Initialize θ using $\boldsymbol{L}^*$
5 $\boldsymbol{Y}^*$ = UpdateAndAggregate($\boldsymbol{L}^*$)
6 **Output:** The aggregated labels $\boldsymbol{Y}^*$
7 **for** *each arriving set* $\boldsymbol{L}^{(s)}$ **do**
8 $\quad \boldsymbol{Y}^{(s)}$ = UpdateAndAggregate($\boldsymbol{L}^{(s)}$)
9 $\quad$ **Output:** The aggregated labels $\boldsymbol{Y}^{(s)}$
10 **Function** UpdateAndAggregate($\boldsymbol{L}^{(s)}$):
11 $\quad$ **for** *number of training epochs* **do**
12 $\quad\quad$ **for** *number of minibatchs* **do**
13 $\quad\quad\quad$ Sample a batch $\boldsymbol{L}^{(M)} = \{\boldsymbol{l}_1, ..., \boldsymbol{l}_M\}$ from $\boldsymbol{L}^{(s)}$
$\quad\quad\quad$ /* Calculate each term in f, Eq(11) */
14 $\quad\quad\quad$ **for** $c = 1, ..., C$ **do**
15 $\quad\quad\quad\quad$ **for** $k = 1, ..., K$ **do**
16 $\quad\quad\quad\quad\quad \boldsymbol{\psi}_{ck} = softmax(\boldsymbol{\omega}_{ck})$
17 $\quad\quad\quad$ **for** *instance* $i = 1, ..., M$ **do**
18 $\quad\quad\quad\quad \boldsymbol{h} = \boldsymbol{W}_2\ tanh(\boldsymbol{W}_1 \boldsymbol{l}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2$
19 $\quad\quad\quad\quad [q_{\boldsymbol{\alpha}}(y = c|\boldsymbol{l}_i)]_{c=1}^{C} = softmax(\boldsymbol{h})$
20 $\quad\quad\quad\quad$ **for** $c = 1, ..., C$ **do**
21 $\quad\quad\quad\quad\quad \log g_{\boldsymbol{\beta}}(\boldsymbol{l}_i|y = c) = \sum_{k \in S_i} \log \psi_{ck, l_{ik}}$
$\quad\quad\quad$ /* update the model parameters θ */
22 $\quad\quad\quad g_t \leftarrow \nabla_\theta f_t(\theta_t)$
23 $\quad\quad\quad v_t \leftarrow \gamma \cdot v_{t-1} + (1 - \gamma) \cdot (g_t \odot g_t)$
24 $\quad\quad\quad \mu_t = \mu \cdot \sqrt{1 - \gamma^t}$
25 $\quad\quad\quad \eta_t = Clip(\mu_t / \sqrt{v_t}, \eta_l(t), \eta_u(t))/\sqrt{t}$
26 $\quad\quad\quad \theta_{t+1} \leftarrow \theta_t - \eta_t \odot g_t$
27 $\quad\quad\quad t \leftarrow t + 1$ // count the time step
28 $\quad$ Get new confusion matrices $\boldsymbol{\pi}$ by the updated $\boldsymbol{\beta}$
29 $\quad$ Infer the aggregated labels $\boldsymbol{Y}^{(s)}$ by $\boldsymbol{\pi}$
30 $\quad$ **return** $\boldsymbol{Y}^{(s)}$

3.2.3 Inferring the Aggregated Labels. Before introducing how to infer the aggregated label y, i.e. the predicted true label, for each sample $\boldsymbol{l}$, we discuss the connection between the generative model p of BILA-CM and the confusion matrices of the workers. The confusion matrix $\boldsymbol{\pi}_{c,z}^{(k)}$ of worker k is a matrix for describing the worker's labeling behavior [4]. The matrix element $\pi_{c,z}^{(k)} = p(l_{ik} = z|y_i = c)$ is the probability that worker k assigns the label z to the instance i when the true label y_i is c. According to the definition of p, we have that $\psi_{ck,z} = p_{\boldsymbol{\beta}}(l_{ik} = z|y_i = c)$ which corresponds to $\pi_{c,z}^{(k)}$. Therefore, we can easily construct the confusion matrices of the workers after learning the parameters $\boldsymbol{\beta} = \{\boldsymbol{\omega}_{ck}\}$. Note that this provides insight on the noise process which other methods lack, e.g. LAA. With the confusion matrices the inference problem becomes trivial. After obtaining the values of the confusion matrices, we can infer the aggregated label of an instance by maximizing the data likelihood of the corresponding observed noisy labels, where $p(\boldsymbol{l}_i|y_i = c, \boldsymbol{\pi}) = \prod_{k=1}^{K} \prod_{z=1}^{C} (\pi_{c,z}^{(k)})^{\mathbf{I}(L_{i,k}=z)}$. $\mathbf{I}(\cdot)$ is an indicator function taking the value 1 when the predicate is true, and 0 otherwise,

4 OPTIMIZER AND CONVERGENCE ANALYSIS

We propose a stochastic optimizer to train BILA-CM and summarize key steps in line 22-27 of Algorithm 1. It's a variant of RMSProp [29]. The update of the model parameter θ (line 26) is based on gradient rather than the momentum. Similar to RMSProp, we utilize a second raw moment estimate of the gradient (line 23) to obtain the element-wise adaptive learning rates for every element of θ (line 24-25). The element-wise adaptive learning rates are important because of the observation that in a multilayer neural network, the appropriate learning rates can vary widely between weights [29]. Furthermore, we apply a clip operator applied to avoid gradient explosion (line 25). In order to avoid an abrupt stop in training, we employ decayed learning rate approach. The result of the clip operator is then divided by $\sqrt{t}$ to obtain decayed element-wise learning rates.

Furthermore, we also analyze the convergence property of our stochastic optimization approach in the online convex framework [40]. According to the framework setting, we use an unknown sequence of convex loss functions $f_1(\theta), f_2(\theta), ..., f_T(\theta)$ to represent the loss functions at each iteration time step t. In each time step the training data (mini-batches) are different, so we need different notations to represent the stochasticity of the loss functions. The regret is applied to evaluate the convergence of our label aggregation algorithm. The regret is defined as $R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)]$ where $\theta^* = \arg\min_{\theta \in \chi} \sum_{t=1}^{T} f_t(\theta)$. Actually $R(T)$ represents the sum of the difference between the online prediction θ_t and the best fixed parameter θ^*. We will show that our algorithm has a $O(\sqrt{T})$ regret bound. The details of the derivation process is shown in the Appendix A. To show the bound, we define the notation $\eta_{t,i}$ to be the i^{th} element of η_t.

Theorem 1. *Let $\{\theta_t\}$ be the parameter sequence obtained from our optimizer where $\theta \in R^d$. Suppose $\eta_u(t) \le R_\infty$ and $\frac{t}{\eta_l(t)} - \frac{t-1}{\eta_u(t-1)} \le B$ for all $t \in [T]$. Assume that $\|\theta_n - \theta_m\|_\infty \le D_\infty$ for all $\theta_n, \theta_m \in \chi$ and $\|\nabla f_t(\theta)\|_2 \le G$ for all $t \in [T]$ and $\theta \in \chi$. Our optimizer have the following guarantee of the regret*

$$R(T) \le \frac{1}{2} D_\infty^2 \left[2dB(\sqrt{T} - 1) + \sum_{i=1}^{d} \eta_{1,i}^{-1} \right] + (\sqrt{T} - \frac{1}{2}) R_\infty G^2.$$

Table 1: Datasets overview.

Dataset	Workers	Items	Labels	Classes
Adult	17	263	1370	4
RTE	164	800	8000	2
Heart	12	237	952	2
Age	165	1002	10020	7
CIFAR10^S	10	50K	45K	10
Pendigits^S	10	11K	9.9K	10

^{S} uses synthetic redundant noisy labels.

According to Theorem 1, if we choose the lower and the upper bounds of the clip operator which satisfy $\eta_u(t) \leq R_\infty$ and $\frac{t}{\eta_l(t)} - \frac{t-1}{\eta_u(t-1)} \leq B$ for all $t \in [T]$, the regret bound will be $O(\sqrt{T})$. The lower bound and upper bound with constant values definitely satisfy these conditions. Dynamic clip bounds as shown in the experimental part of the paper [25] also meet the requirement and they have good performance in practise. We choose these bounds to conduct our experiments. Note that because of the clip operator, $\sum_{i=1}^{d} \eta_{1,i}^{-1}$ takes a limited value. Then we have the corresponding convergence rate $\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$ where $\lim_{T\to\infty} \frac{R(T)}{T} = 0$. That shows the average of the difference between the online prediction and the best fixed parameter tend to 0 during the iterations. Thus, the regret bound guarantees the convergence of our algorithm.

5 EVALUATION

5.1 Experimental setup

5.1.1 Datasets. We consider six different datasets in our experiments to evaluate the performance of BiLA-CM comparing to competitors. Table 1 summarizes the characteristics of all datasets. In our experiments, CIFAR-10 and Pendigits are the only synthetic dataset, while the rest of them are real-world datasets.

- **Adult** [23]: It contains data labeled by Amazon Mechanical Turk workers. The labels are categorized into four classes based on the amount of adult content on each web page.
- **RTE** [27]: It includes 164 workers for assigning labels of 800 items into 2 classes of textual entailment.
- **Heart** [12]: It is a dataset provided by 12 medical students categorizing the patients into 2 groups of heart and non-heart diseases based on physical examination. It has 12 workers for 237 samples.
- **Age** [10]: This dataset is the 1001 faces of different people who have been labeled with their age. In our experiments, the labels are discretized into 7 age groups: [0,9], [10,19], [20,29], [30,39], [40,49], [50,59], [60,100].
- **CIFAR-10** [16]: It is a vision dataset including 50K 32×32-pixels training images classified into 10 classes. Here we create synthetic noisy labels. For each image we generate $[6, 8, 10]$ redundant labels, i.e. workers, drawn from a bimodal noise distribution with $[0.4, 0.6, 0.8]$ mislabelling probability and $[0.1, 0.2, 0.3]$ missing label probability. We center the bimodal distribution around classes $\mu_1 = 3.0$, $\mu_2 = 7.0$ with variance $\sigma_1 = 1.0$, $\sigma_2 = 0.5$.
- **Pendigits** [5]: This dataset targets the recognition of 10992 handwritten digits from 44 writers. We create synthetic redundant noisy labels using the same noise model as for CIFAR10.

5.1.2 Baselines. We consider five different baselines to compare BiLA-CM against. The baselines cover both state of the art as well as state of the practise. All algorithms are programmed in Python programming language using Keras version version 2.2.4 and TensorFlow version 1.12.

- **Majority Voting (MV)**: is a basic method which selects from the set of redundant noisy labels $\boldsymbol{l}$ the label with the highest consensus.
- **Expectation Maximization (EM)** [4]: is an iterative method used to estimate each worker's confusion matrix by maximizing the likelihood of observed labels. The off-diagonal elements represent the probability of mislabeling, the diagonal elements of correct labeling.
- **Bayesian Classifier Combination (BCC)** [13]: is an extension of EM. It solves the label aggregation problem by modelling the relationship between the output of multiple classifiers (workers) and the true label.
- **Minimax Entropy (ME)** [38]: assigns a confusion matrix to workers, encoding their labeling ability, and a vector to items, encoding their labeling difficulty. The matrix and vector are estimated jointly using a minimax entropy approach.
- **MiniMax Conditional Entropy (MMCE)** [39]: extends ME by assigning confusion matrices also to items intead of a vector. It uses a minimax conditional entropy approach to jointly estimate both worker and item matrices.
- **Label Aware Autoencoders (LAA)** [36]: represents the labelling problem via an autoencoder model where the encoder acts as classifier inferring the true label, the decoder reconstructs the input and the inferred labels represent the latent space.

5.1.3 BiLA-CM Parameters. As neural network q in BiLA-CM we use a multi layer perceptron with two hidden layers of size 64 and 32, respectively. The sizes of the input and output layers are given by the number of workers and the number of classes of each dataset, respectively. We train the network until convergence using marginal loss as stopping criteria.

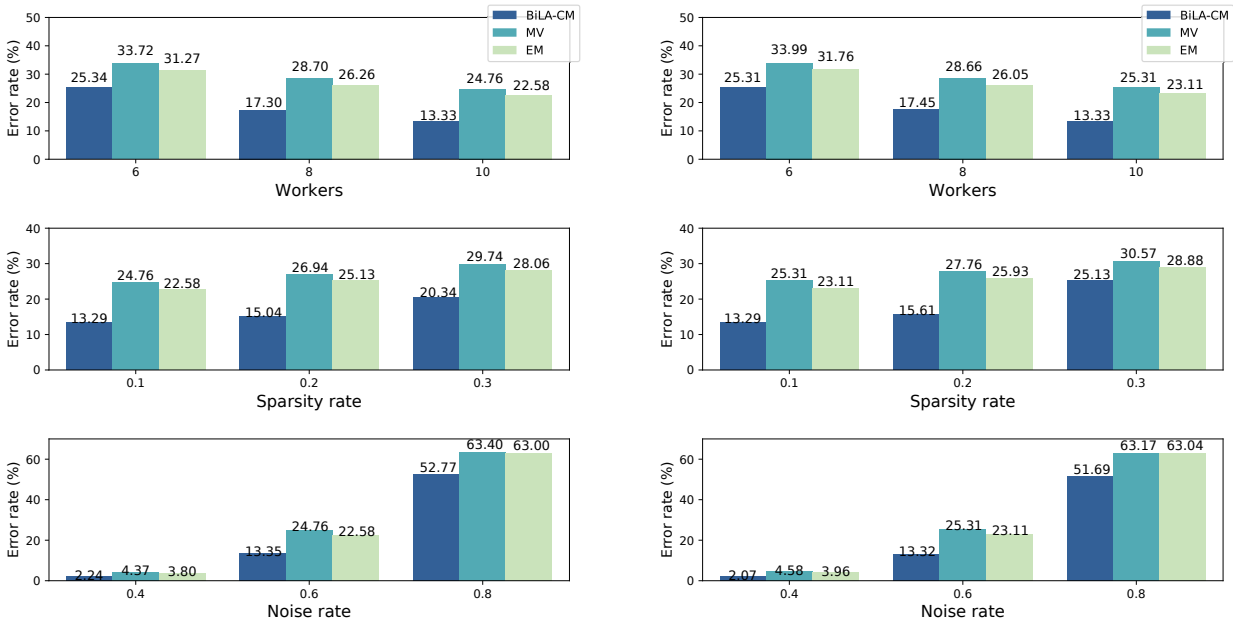
5.1.4 Performance Measure. We use error rate as performance metric in all our experiments. We define the error rate as the percentage of inferred labels which differ from the true label. Note that the true label is only used to compute the error rate but not to train the label aggregators.

5.2 Results

We first provide comparative results for online label aggregation, data processed in chunks, showing the superior performance of BiLA-CM and its robustness to varying chunk sixes. Following we perform a sensitivity analysis of BiLA-CM on dataset parameters, i.e. number of workers, noise rate and label sparsity, and analyze the optimality of BiLA . Finally we conclude with results on offline label aggregation, data processed all at once.

Table 2: Online label aggregation: error-rates (%). The online version of baseline algorithms are appended with prefix of "o".

Dataset	Small chunk size					Big chunk size				
	MV	oEM	oMMCE	oLAA	BILA-CM	MV	oEM	oMMCE	oLAA	BILA-CM
CIFAR10 (200, 500)	24.74	22.58	43.00	29.20	**13.29**	24.74	17.68	14.25	30.29	**13.69**
Pendigits (200, 500)	25.27	23.11	44.77	28.84	**13.34**	25.27	18.15	14.77	30.54	**13.06**
Age (25, 50)	34.73	35.03	41.52	35.53	**33.73**	34.73	34.43	41.92	36.33	**33.63**
RTE (25, 50)	9.88	10.5	18.0	11.25	**7.75**	9.88	9.5	16.75	11.75	**7.5**



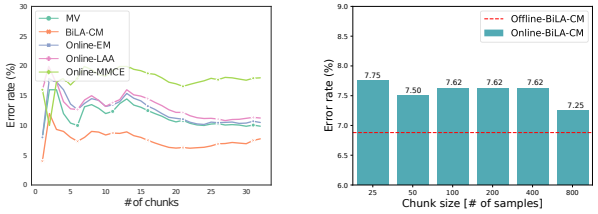
(a) CIFAR-10: chunk size 200, initial set size 1000

(b) Pendigits: chunk size 200, initial set size 1000

Figure 4: The effect of the number of workers, label sparsity and noise ratio on BILA-CM . The default values for the number of workers, sparsity ratio, and noise ratio are 10, 0.1, and 0.6, respectively.

5.2.1 Online Label Aggregation. We summarize error rates of BILA and different baselines, across different combinations of datasets and chunk sizes in Table 2. The small and big chunk size are 200/25 and 500/50 for synthetic/real-world data respectively. Their initial datasets are 1000 and 500 samples for CIFAR-10/Pendigits and Age/RTE, respectively. We initialize all model based approaches by majority voting. Due to the limited number of samples in Adult and Heart dataset, we opt them out from the online evaluation.

When the chunk size is small, labels from workers are received more fluidly. One can see that BILA-CM achieved the lowest error rate. For CIFAR10 and Pendigits, the error rate of BILA-CM is 10 percent points lower than the second best algorithm, i.e., online EM. For the Age and RTE datasets, majority voting is the second best algorithm but still has at least 1.5 percent points higher error rates. As MMCE and LAA both have larger number of parameters than EM, their error rates are remarkably high due to insufficient number of samples per chunk for parameterization, especially for MMCE. We further note that smaller chunks not only affect the error rate for EM and MMCE, but also the convergence speed. Due to the small number of samples, it takes MMCE more iterations to converge, compared to big chunk size.



(a) RTE: small chunk size.

(b) BILA-CM : effect of chunk size.

Figure 5: Online label aggregation of RTE: error rates on BILA-CM and the baselines.

When the chunk size is larger, more items can be aggregated at once, i.e., closer to the offline scenario. BILA-CM is still the best algorithm and MMCE comes second, except for Age. The difference from the small chunk size is that now there are sufficient number of samples in a chunk to parametrize the MMCE model. MMCE captures the confusion matrix at the levels of classes, workers and data items. Regarding EM, the error rate drops significantly for bigger chunk size.

Another observation worth mentioning is the comparison of computational overhead. Due to its simplicity, MV incurs almost no computational overhead. EM algorithm is known to have fast convergence. This is the case observed here. As LAA and BILA-CM both employ neural networks, their computational overheads are in the same order.

We further zoom into the error rates over the online aggregation process of the RTE dataset for small chunks (see Figure 5(a)). When the number of aggregated samples increases, the error rate first increases and then drops because of the large difference between the size of initial set (i.e., 500 samples) and chunk size (i.e., 25 samples). Overall, BILA-CM is able to learn the confusion matrix efficiently from only small chunks of samples and incrementally update the inference model. This is supported by visibly lower error rates across any number of samples processed by the aggregator.

We also demonstrate the robustness of BILA-CM against different chunk sizes or online velocity in Figure 5(b). Recalling the motivation examples in Figure 2, existing label aggregation methods are sensitive to the online velocity, i.e., drastic error rate changes between very big and small chunk sizes. Thanks to incremental updates and stochastic optimization, BILA-CM can keep relatively low and constant error rates when encountering different online velocities.

5.2.2 Sensitivity (robustness) analysis of BILA . We focus on evaluating the robustness of BILA-CM via synthetic redundant noisy labels on two datasets, i.e., CIFAR-10 and Pendigits. Specifically, we evaluate how BILA-CM performs against different types of crowd sourcing scenarios, i.e., number of crowd workers, sparsity of labels, and noise rates. The sparsity of labels defines the percentage of missing labels across all items and workers. The noise rate indicates the percentage of wrong labels of all labels collected.

Figure 4 summarizes such a sensitivity analysis for CIFAR-10 and Pendigits, respectively. We vary one parameter and fix the other two. The default values are 10 workers, sparsity rate of 0.1 and noise rate of 0.6. For the purpose of comparison, we choose the best performing label aggregation methods, i.e., EM and majority voting.

Across all three methods, we can make the following general observations. The error rates decrease with increasing number of workers, and increase with the sparsity and noise rate. In all cases considered, BILA-CM always achieves the lowest error rate, followed by EM and then MV.

Taking a closer look of number of workers, we observe that BILA-CM is able to achieve similar error rates, i.e., 25.34%, as MV but using only 6 instead of 10 workers. As for the robustness against the sparsity rate, EM and MV can better cope with increasing missing labels than BILA-CM . Specifically, when the sparsity increases from 0.1 to 0.3, the error rate of BILA-CM almost doubles, whereas the error rate of EM only increases by less than 30%. It appears that BILA-CM can be more sensitive to the sparsity than other methods, but the absolute performance is still better.

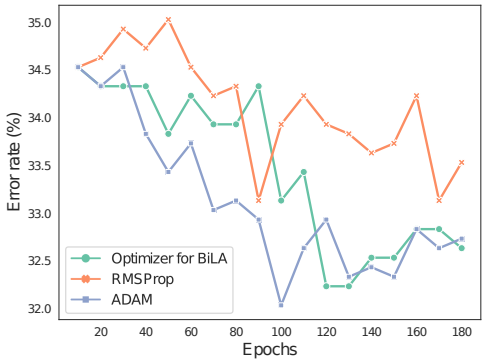


Figure 6: Error rates for different optimizers on Age.

Regarding the impact of noise rate, all methods deteriorate drastically when the noise rate is up to 0.8. Actually none of the methods can reach accuracy above 50%. This is a bottleneck of how label aggregation methods can combat crowd's mistakes. To overcome high percentage of label noise, different solutions may be needed, e.g., a small fraction of ground-truth. When noise ratio is 0.4 and 0.6, we can observe that BILA-CM can achieve half of the error rate of the other two methods.

5.2.3 Optimality of BILA . Next we investigate the optimality of the optimizer used in BILA-CM . We compare our optimizer against RMSProp [29] and ADAM [14]. Our optimizer is a enhanced version of RMSProp, while ADAM is another common choice. Figure 6 shows the evolution of the error rate across training epochs. We use the Age dataset with chunk size 25. We see that our optimizer is faster to converge than RMSProp. Hence we can obtain a higher model performance for the same training effort, i.e. number of epochs. ADAM is initially slightly faster to converge, but starting at epoch 120 the two optimizers achieve similar model performance. RMSProp in our implementation and our optimizer both use a clip operator to avoid the issue of gradient explosion which can lead to divergence. ADAM instead is a momentum-based optimizer. Momentum optimizers can be faster to converge than clip-based optimizers but pose the risk of gradient explosion and divergence. These observations are clearly shown in the results.

5.2.4 Offline label aggregation. Finally, we present offline aggregation results in Table 3. We include two additional baselines, i.e., BCC [13] and ME [38]. BCC combines the probabilistic models and confusion matrix to infer true labels. ME is the predecessor of MMCE [39]. Both jointly estimate worker and item latent variables. Similar to the online results, BILA-CM is able to achieve the lowest error rate in all four datasets. The second best policy depends on the dataset. For Adult and RTE, the second best method is BCC that combines probabilistic models and confusion matrix. As for Heart and Age, the second best method is LAA and ME respectively. Though both LAA and BILA-CM both use neural networks,

Table 3: Offline comparison: error-rates (%) of label aggregation models

Dataset	MV	EM [4]	BCC [13]	ME[38]	MMCE [39]	LAA [36]	BILA-CM
Adult	26.43	25.48	22.81	24.33	24.33	25.86	**21.60**
RTE	9.88	7.5	7.15	7.25	7.50	12.38	**6.88**
Heart	22.36	18.99	18.82	16.03	16.03	13.5	**12.66**
Age	34.73	35.03	33.53	32.63	32.63	34.13	**30.18**

BILA-CM has a more stable performance due to the guidance of the generating distribution.

When contrasting the results of small chunk size in Table 2 with results of Table 3, we can gauge the impact of online data feeding to different aggregation methods. On the one hand BILA-CM has little variation across different online scenarios as it can incrementally update the models chunk by chunk of data. On the other hand, EM and MMCE are observed to have high variability across datasets and online velocity, weakening their applicability for online label aggregation.

6 RELATED WORK

Label aggregation is a well-studied subject in crowd sourcing, especially for offline scenarios. Most of existing label aggregation solutions are unsupervised, except [7]. We summarize the related work in accordance with our contributions: (i) the probabilistic inference framework, (ii) confusion matrix aggregation model, (iii) stochastic models, and (iv) recent advances.

Probabilistic inference models. Probabilistic models are effective to capture how the latent variables, e.g., confusion matrix, affect the likelihood of observed noise labels. BCC [13] is the very first the probabilistic graphical model for label aggregation. It uses confusion matrix to evaluate workers, and uses Gibbs sampling to perform the parameter estimation. CommunityBCC [30] and BCCWords [26] are an extension of BCC. Specifically CommunityBCC divides the workers into worker communities. The workers in the same community have similar confusion matrices. In terms of variational methods, Liu et al.[19] propose a model which uses variational inference to approximate the posterior. Recently, [36] develops LAA, a label-aware autoencoder. LAA is an unsupervised model composed of a classifier and a reconstructor, both of which are neural networks. [18] proposes an offline probabilistic graphical model for label aggregation, including an enhanced variational Bayesian classifier combination with inference based on a mean-field variational method. Orthogonally, Yang et al. [34, 35] apply probabilistic inference models applied to jointly distill noisy labels via experts and learning tasks. Aforementioned studies tailors for offline scenarios where lables of all items is collected at once. And, these methods require to derive a close-form of the generative model's posterior.

Confusion matrix. Confusion matrix specifies how labels are corrupted from their true class to noisy ones. It can be based on the entre dataset, each worker, and even each content, with increasing model complexity. Dawid and Skene [4] uses the confusion matrix to describe the expertise and the bias of a worker. They then design an EM algorithm for label aggregation. Raykar et al. [24] uses noisy labels to train their classification model. Their two-coin model is a variation of the confusion matrix. GLAD [32] is a model that can infer the true labels, the expertise of workers, and the difficulty of items at the same time. However, GLAD is applicable for binary labeling tasks. Furthermore, Zhou et al. [38, 39] propose the minimax entropy estimator and its extensions. In these model, the authors set a separate probabilistic distribution for each worker-item pair. Due to the iterative nature of EM algorithms and minmax entropy, it is not straightforward to extend those methods to construct stochastic optimizer needed for online label aggregation. Different from aformentinoed label aggregation methods, DeepAgg [7] is a supervised model based on a deep neural network. The model is trained by a seed dataset which contains noisy labels and the corresponding ground truth labels. DeepAgg can not aggregate incomplete data, where many annotators only labeled a few items.

Stochastic Optimizer First order stochastic optimization is applied to a wide range of learning problems. RMSProp [29] and Adam [14] are the state-of-the art optimizers. RMSProp updates the model parameters based on the current gradient. It achieves more robust results than stochastic gradient decent because it utilizes element-wise adaptive learning rates to update the model parameters. Adam is a variant of RMSProp. It uses a moving average to estimate the first moment of the gradients and applies the moment to update the model parameters. Often, a clipping operator on the learning rates is used to limit their values during the training and avoid gradient explosion [21]. McMahan et al. [21] provides the theoretical base for deriving the regret bound of optimizers that use clip operator.

7 CONCLUSION

Motivated by the need of timely and accurately data curation and the avoidance of slow response from crowd workers, we design online label aggregation framework, BILA, maximizing the likelihood of noise labels and inferring unobservable true labels. The core components of BILA are variational Bayesian inference model and a stochastic optimizer for incrementally training online data. The general design of BILA is able to model any generating distribution of labels via exact computation of posterior probability distribution and neural networks based approximate distribution. We design a stochastic optimizer that can incrementally minimize the loss function of the variational inference model based on the evidence lower bound. We theoretically prove the convergence bound of the proposed optimizer in terms of parameters of gradient update. We evaluate BILA on both synthetic and real world datasets under various online scenarios. Compared to the state of the art label aggregation algorithms that adopt sliding window update, BILA shows significant and robust error reduction, especially for challenging scenarios with small chunk of dataset.

ACKNOWLEDGMENTS

This work has been partly funded by the Swiss National Science Foundation NRP75 project 407540_167266.

REFERENCES

[1] European Commission 2018. *European Union's General Data Protection Regulation.* European Commission. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en
[2] Christopher M Bishop. 2006. Pattern recognition and machine learning. (2006), 461–517.
[3] José María Cavanillas, Edward Curry, and Wolfgang Wahlster (Eds.). 2016. *New Horizons for a Data-Driven Economy - A Roadmap for Usage and Exploitation of Big Data in Europe.* Springer. https://doi.org/10.1007/978-3-319-21569-3
[4] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979), 20–28.
[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[6] Li Fei-Fei. 2010. ImageNet: crowdsourcing, benchmarking & other cool things. In *CMU VASC Seminar*, Vol. 16. 18–25.
[7] Alex Gaunt, Diana Borsa, and Yoram Bachrach. 2016. Training deep neural nets to aggregate crowdsourced responses. In *UAI*. 242–251.
[8] Amirmasoud Ghiassi, Taraneh Younesian, Robert Birke, and Lydia Y. Chen. 2020. TrustNet: Learning from Trusted Data Against (A)symmetric Label Noise. *CoRR* abs/2007.06324 (2020).
[9] Amirmasoud Ghiassi, Taraneh Younesian, Zilong Zhao, Robert Birke, Valerio Schiavoni, and Lydia Y. Chen. 2019. Robust (Deep) Learning Framework Against Dirty Labels and Beyond. In *TPS-ISA*. 236–244.
[10] Hu Han, Charles Otto, Xiaoming Liu, and Anil K Jain. 2015. Demographic estimation from face images: Human vs. machine performance. *IEEE transactions on pattern analysis and machine intelligence* 37, 6 (2015), 1148–1161.
[11] Muhammad Imran, Carlos Castillo, Ji Lucas, Patrick Meier, and Sarah Vieweg. 2014. AIDR: Artificial intelligence for disaster response. In *WWW*. 159–162.
[12] A Janosi, W Steinbrunn, M Pfisterer, and R Detrano. 1988. Heart disease data set. In *https://archive.ics.uci.edu/ml/datasets/Heart+Disease.*
[13] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *AISTATS*. 619–627.
[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[15] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques.* MIT press.
[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. CIFAR-10 (Canadian Institute for Advanced Research). (2009). http://www.cs.toronto.edu/~kriz/cifar.html
[17] Kenichi Kurihara, Max Welling, and Yee Whye Teh. 2007. Collapsed Variational Dirichlet Process Mixture Models.. In *IJCAI*, Vol. 7. 2796–2801.
[18] Yuan Li, Benjamin Rubinstein, and Trevor Cohn. 2019. Exploiting worker correlation for label aggregation in crowdsourcing. In *ICML*. 3886–3895.
[19] Qiang Liu, Jian Peng, and Alexander T Ihler. 2012. Variational inference for crowdsourcing. In *NeurIPS*. 692–700.
[20] Alan Lundgard, Yiwei Yang, Maya L Foster, and Walter S Lasecki. 2018. Bolt: Instantaneous crowdsourcing via just-in-time training. In *CHI*. 1–7.
[21] H. Brendan McMahan and Matthew J. Streeter. 2010. Adaptive Bound Optimization for Online Convex Optimization. In *COLT*. 244–256.
[22] John W. Paisley, David M. Blei, and Michael I. Jordan. 2012. Variational Bayesian Inference with Stochastic Search. In *ICML*. 1363–1370.
[23] Foster Provost, Wang Jing, and Panagiotis G. Ipeirotis. 2010. Quality management on amazon mechanical turk. In *SIGKDD workshop on human computation*. 64–67.
[24] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research* 11, Apr (2010), 1297–1322.
[25] Pedro Savarese. 2019. On the Convergence of AdaBound and its Connection to SGD. *CoRR* abs/1908.04457 (2019).
[26] Edwin D Simpson, Matteo Venanzi, Steven Reece, Pushmeet Kohli, John Guiver, Stephen J Roberts, and Nicholas R Jennings. 2015. Language understanding in the wild: Combining crowdsourcing and machine learning. In *WWW*. 992–1002.
[27] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*. 254–263.
[28] Yee W Teh, David Newman, and Max Welling. 2007. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *NeurIPS*. 1353–1360.
[29] T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.
[30] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based bayesian aggregation models for crowdsourcing. In *WWW*. 155–164.
[31] Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1, 1–2 (2008), 1–305.
[32] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NeurIPS*. 2035–2043.
[33] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*. 2691–2699.
[34] Jie Yang, Thomas Drake, Andreas Damianou, and Yoelle Maarek. 2018. Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *WWW*. 23–32.
[35] Jie Yang, Alisa Smirnova, Dingqi Yang, Gianluca Demartini, Yuan Lu, and Philippe Cudré-Mauroux. 2019. Scalpel-cd: leveraging crowdsourcing and deep probabilistic modeling for debugging noisy training data. In *WWW*. 2158–2168.
[36] Li'ang Yin, Jianhua Han, Weinan Zhang, and Yong Yu. 2017. Aggregating crowd wisdoms with label-aware autoencoders. In *IJCAI*. 1325–1331.
[37] Taraneh Younesian, Zilong Zhao, Amirmasoud Ghiassi, Robert Birke, and Lydia Y. Chen. 2020. QActor: On-line Active Learning for Noisy Labeled Stream Data. *CoRR* abs/2001.10399 (2020).
[38] Denny Zhou, Sumit Basu, Yi Mao, and John C Platt. 2012. Learning from the wisdom of crowds by minimax entropy. In *NeurIPS*. 2195–2203.
[39] Dengyong Zhou, Qiang Liu, John Platt, and Christopher Meek. 2014. Aggregating ordinal labels from crowds by minimax conditional entropy. In *ICML*. 262–270.
[40] Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*. 928–936.

A APPENDIX: A CONVERGENCE BOUND

A.1 Notations and Lemmas

Notations for the proof. S_+^d is the set of all positive definite $d \times d$ matrix. $\theta^*_{,i}$ is the i^{th} element of θ^*. $\theta_{t,i}$ is the i^{th} element of θ_t. The operator $\odot$ means element-wise product.

Lemma 1. *If a function $f : R^d \to R$ is convex, then for all $x, y \in R^d$, $f(x) - f(y) \leq \nabla f(x)^T (x - y)$.*

Lemma 2 (proposed by [21]). *For any $Q \in S_+^d$ and closed, bounded convex set $\chi \subset R^d$, suppose $u_1 = \min_{\theta \in \chi} \|Q^{1/2}(\theta - z_1)\|_2$ and $u_2 = \min_{\theta \in \chi} \|Q^{1/2}(\theta - z_2)\|_2$ then we have $\|Q^{1/2}(u_1 - u_2)\|_2 \leq \|Q^{1/2}(z_1 - z_2)\|_2$.*

A.2 Proof of Theorem 1

Theorem 1. *Let $\{\theta_t\}$ be the parameter sequence obtained from our optimizer where $\theta \in R^d$. Suppose $\eta_u(t) \leq R_\infty$ and $\frac{t}{\eta_l(t)} - \frac{t-1}{\eta_u(t-1)} \leq B$ for all $t \in [T]$. Assume that $\|\theta_n - \theta_m\|_\infty \leq D_\infty$ for all $\theta_n, \theta_m \in \chi$ and $\|\nabla f_t(\theta)\|_2 \leq G$ for all $t \in [T]$ and $\theta \in \chi$. Our optimizer have the following guarantee of the regret*

$$R(T) \leq \frac{1}{2} D_\infty^2 \left[2dB(\sqrt{T} - 1) + \sum_{i=1}^{d} \eta_{1,i}^{-1} \right] + (\sqrt{T} - \frac{1}{2}) R_\infty G^2.$$

Proof. According to Lemma 1, we have

$$f_t(\theta_t) - f_t(\theta^*) \leq g_t^T (\theta_t - \theta^*) = \left\langle g_t, \theta_t - \theta^* \right\rangle \tag{12}$$

We have the definition $\theta^* = \arg\min_{\theta \in \chi} \sum_{t=1}^{T} f_t(\theta)$ mentioned before. According to the update rule shown in Algorithm 1, we have $\theta_{t+1} = \min_{\theta \in \chi} \|diag(\eta_t^{-1})^{1/2}(\theta - (\theta_t - \eta_t \odot g_t))\|_2$. Applying Lemma 2 and setting $u_1 = \theta_{t+1}$ and $u_2 = \theta^*$, we have

$$\begin{aligned} &\|\eta_t^{-1/2} \odot (\theta_{t+1} - \theta^*)\|_2^2 \\ &\quad \leq \|\eta_t^{-1/2} \odot (\theta_t - \eta_t \odot g_t - \theta^*)\|_2^2 \\ &\quad = \|\eta_t^{-1/2} \odot (\theta_t - \theta^*)\|_2^2 + \|\eta_t^{1/2} \odot g_t\|_2^2 - 2 \left\langle g_t, \theta_t - \theta^* \right\rangle \end{aligned}$$

Rearrange the above inequality, we can have

$$\left\langle g_t, \theta_t - \theta^* \right\rangle \leq \frac{1}{2}\left[\|\eta_t^{-1/2} \odot (\theta_t - \theta^*)\|_2^2 - \|\eta_t^{-1/2} \odot (\theta_{t+1} - \theta^*)\|_2^2\right] + \frac{1}{2}\|\eta_t^{1/2} \odot g_t\|_2^2 \tag{13}$$

According to (12), (13) and the definition of the regret $R(T)$ we have

$$\begin{aligned} R(T) &= \sum_{t=1}^{T}[f_t(\theta_t) - f_t(\theta^*)] \leq \sum_{t=1}^{T}\left\langle g_t, \theta_t - \theta^* \right\rangle \\ &\leq \sum_{t=1}^{T}\frac{1}{2}\left[\|\eta_t^{-1/2} \odot (\theta_t - \theta^*)\|_2^2 - \|\eta_t^{-1/2} \odot (\theta_{t+1} - \theta^*)\|_2^2\right] \\ &+ \sum_{t=1}^{T}\frac{1}{2}\|\eta_t^{1/2} \odot g_t\|_2^2 \end{aligned} \tag{14}$$

We bound $\sum_{t=1}^{T}\frac{1}{2}\|\eta_t^{1/2} \odot g_t\|_2^2$ at first. According to the assumption we have $\|\eta_t\|_\infty \leq R_\infty/\sqrt{t}$ and $\|g_t\|_2^2 \leq G^2$. Thus we can bound the term as

$$\begin{aligned} \sum_{t=1}^{T}\frac{1}{2}\|\eta_t^{1/2} \odot g_t\|_2^2 &\leq \sum_{t=1}^{T}\frac{R_\infty}{2\sqrt{t}}\|g_t\|_2^2 \leq \frac{R_\infty G^2}{2}\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \\ &\leq (\sqrt{T} - \frac{1}{2})R_\infty G^2, \end{aligned} \tag{15}$$

where $\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \leq 2\sqrt{T} - 1$. Now, we bound the another term in (14).

$$\begin{aligned} &\sum_{t=1}^{T}\frac{1}{2}\left[\|\eta_t^{-1/2} \odot (\theta_t - \theta^*)\|_2^2 - \|\eta_t^{-1/2} \odot (\theta_{t+1} - \theta^*)\|_2^2\right] \\ &= \sum_{i=1}^{d}\sum_{t=1}^{T}\frac{1}{2}\left[\eta_{t,i}^{-1}(\theta_{t,i} - \theta_{,i}^*)^2 - \eta_{t,i}^{-1}(\theta_{t+1,i} - \theta_{,i}^*)^2\right] \\ &\leq \sum_{i=1}^{d}\left[\frac{1}{2}\eta_{1,i}^{-1}(\theta_{1,i} - \theta_{,i}^*)^2 + \sum_{t=2}^{T}\frac{1}{2}(\eta_{t,i}^{-1} - \eta_{t-1,i}^{-1})(\theta_{t,i} - \theta_{,i}^*)^2\right] \\ &\leq \sum_{i=1}^{d}\left[\frac{1}{2}\eta_{1,i}^{-1}(\theta_{1,i} - \theta_{,i}^*)^2 + \sum_{t=2}^{T}\frac{1}{2}\left[\frac{\sqrt{t}}{\eta_l(t)} - \frac{\sqrt{t-1}}{\eta_u(t-1)}\right](\theta_{t,i} - \theta_{,i}^*)^2\right] \\ &\leq \frac{1}{2}D_\infty^2\sum_{i=1}^{d}\left[\eta_{1,i}^{-1} + \sum_{t=2}^{T}\frac{1}{\sqrt{t}}\left[\frac{t}{\eta_l(t)} - \frac{t-1}{\eta_u(t-1)}\right]\right] \\ &\leq \frac{1}{2}D_\infty^2\sum_{i=1}^{d}\left[\eta_{1,i}^{-1} + B\sum_{t=2}^{T}\frac{1}{\sqrt{t}}\right] \\ &\leq \frac{1}{2}D_\infty^2\left[2dB(\sqrt{T} - 1) + \sum_{i=1}^{d}\eta_{1,i}^{-1}\right] \end{aligned} \tag{16}$$

In the second inequality we use the inequation $\eta_l(t) \leq \eta_{t,i} \leq \eta_u(t)$ which can be obtained by the clip operator. In the third inequality we applied the bound D_∞. In the fourth we applied the assumption $\frac{t}{\eta_l(t)} - \frac{t-1}{\eta_u(t-1)} \leq B$. Then according to (14), (15) and (16) we have the following regret bound

$$R(T) \leq \frac{1}{2}D_\infty^2\left[2dB(\sqrt{T} - 1) + \sum_{i=1}^{d}\eta_{1,i}^{-1}\right] + (\sqrt{T} - \frac{1}{2})R_\infty G^2.$$

B APPENDIX: ONE BINARY LABEL AGGREGATION MODEL

In this section we show the definition of a binary label aggregation model. This model is called BiLA-WA . In BiLA-WA , q is a MLP. It inputs an instance $\boldsymbol{l}$ and outputs a distribution $q_{\boldsymbol{\alpha}}(t|\boldsymbol{l})$, where $\boldsymbol{\alpha}$ denotes the network parameters.

Next, we define a generative model p to describe the generation of the observed noisy labels. As shown in (6) and (7), in order to compute the loss function and its gradient, we need to define $p_{\boldsymbol{\beta}}(\boldsymbol{l}|y)$ and $p_{\boldsymbol{\beta}}(y)$. In NN-WA, we only consider binary labeling tasks. For each $c \in \{1, 2\}$, the ability of each worker k is represented by a single parameter $\lambda_{ck} \in (-\infty, +\infty)$. We assume that worker k labels each item i correctly with the probability

$$p_{\boldsymbol{\beta}}(l_{ik} = c|y_i = c) = \frac{1}{1 + \mathrm{e}^{-\lambda_{ck}}}, \tag{17}$$

According to this assumption, we have $\lim_{\lambda_{ck}\to+\infty} p_{\boldsymbol{\beta}}(l_{ik} = c|y_i = c) = 1$, $\lim_{\lambda_{ck}\to-\infty} p_{\boldsymbol{\beta}}(l_{ik} = c|y_i = c) = 0$, and $\lim_{\lambda_{ck}\to 0} p_{\boldsymbol{\beta}}(l_{ik} = c|y_i = c) = 0.5$. We can see that the higher the ability of worker k is, the higher the likelihood for him or her to label the item correctly. When $\lambda_k = 0$, he or she just randomly chooses one class. According to (17), the conditional distributions that generated instances are defined as

$$p_{\boldsymbol{\beta}}(\boldsymbol{l}_i|c) = \prod_{k\in S_i}\left(\frac{1}{1 + \mathrm{e}^{-\lambda_{ck}}}\right)^{\mathrm{I}(l_{ik}=c)}\left(\frac{\mathrm{e}^{-\lambda_{ck}}}{1 + \mathrm{e}^{-\lambda_{ck}}}\right)^{\mathrm{I}(l_{ik}\neq c)}, \tag{18}$$

where S_i is a set of workers who have labeled item i. In this model, the prior distribution $p_{\boldsymbol{\beta}}(y)$ is fixed during the training process. It can also be estimated by Equation (10). The optimization goal of BiLA-WA also takes the form as Equation (11). BiLA-WA can also be applied online like BiLA-CM .