Data Difficulty Specified Crowdsourcing

Yilun Xu, Tianyuan Zhang
School of EECS
Peking University
{1600011072, 1600012888}@pku.edu.cn

Dinghuai Zhang
School of Mathematical Sciences
Peking University
1600013525@pku.edu.cn

Weiyi Zhang
Yuanpei College
Peking University
1600017749@pku.edu.cn

Abstract

Crowdsourcing has become a popular paradigm for collecting labels of large-scale datasets. However, the collected labels are noisy out of the inaccuracy of experts, and thus we need to aggregate labels to complete a supervised problem. Traditional methods often fail because of the complexity and non-convexity of the log-likelihood function. In this paper, we suppose the experts are people with different ability and the each data point has unique difficulty. Then we tackle the problem via EM algorithm together with our original difficulty processing method. Furthermore, we investigate our algorithms in three aspects and show the superiority of our method through ablation study with synthesized data and real data.[1]

1 Introduction

In this data-driven time, the need for large-scale data has become more urgent than any time in history. Crowdsourcing comes up to solve the problem under such background, collecting labels from crowds to accomplish this mission cosmically. As the name suggests, most crowdsourcing services resort to labeling redundancy, collecting multiple labels from different workers for each item. Though this solution seems appropriate, new problem emerges that the labeled data coming from crowds are always noisy out of non-consistency of workers' ability or the high difficulty of certain task like age estimation or medical images annotation. Under such scenario, several problems are raised:

- how to aggregate and infer the ground truth label from the noisy crowdsourced labels?
- how to learn an accurate classifier based on the imperfect crowdsourced but redundant labels?

Maybe the most naive way to solve this problem is **Majority Vote**, which simply assigns the label returned by the majority of the workers and learns a classifier with the majority answer. But biased answer can be obtained with a difficult task when most experts get wrong while capable experts who give the correct answers are in the minority, causing it to be an error-prone algorithm.

More sophisticated algorithms come with more assumptions about how labels are generated from experts. A classical method proposed by [3] assumes that all samples have the same difficulty for each worker. With this assumption they solve a maximum likelihood problem based on the EM algorithm. Formally, they assume that each worker is associated with a 2×2 confusion matrix, where

[1]We choose Track A for our project.

Statistical Learning course project.

the (c',c)-th entry represents the probability that a random chosen item in class c' is labeled as class c by the worker:

$$\begin{pmatrix} \beta & 1-\beta \\ 1-\alpha & \alpha \end{pmatrix}$$

Hereafter we denote $\alpha = \Pr[y^{\text{worker}} = 1 | y = 1]$ and $\beta = \Pr[y^{\text{worker}} = 0 | y = 0]$. The true labels and worker confusion matrices are jointly estimated by maximizing the likelihood of the observed labels, where the unobserved true labels are treated as latent variables.

Obviously, there does exist data that is hard for people (worker) to deal with, and that is where the above classical algorithms fall (an example is presented in Fig 1). But taking the difficulty of each sample into consideration faces two challenges:

- How to define and measure the difficulty of each sample for every experts?
- How to take difficulty of samples into our algorithm?

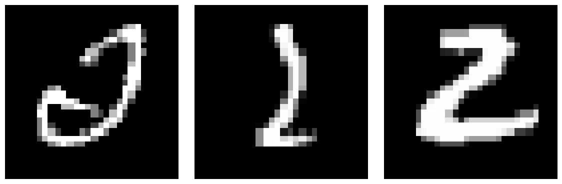

Figure 1: left & middle: images that are hard to classify   right: image that is easy to classify

We tackle the above problem by modeling the labels generation with two assumptions:

- If sample x_i is more difficult than sample x_j, then for every experts, the probability of labeling x_i wrong is bigger than making x_j wrong.
- If expert A is better than expert B, then for every sample, the probability of expert A labeling that sample wrong is smaller than that of expert B.

Under this assumption, we design two algorithms, named **Adaptive Difficulty EM** and **Fixed Difficulty EM**. They mainly varies in the way of measuring the difficulty of each data point. We compare our algorithms with two baselines: **Majority Voting** and **Original EM** [10] both on synthesized and real data, we investigate these algorithms mainly in three aspects:

- How well they can estimate the ground truth
- The generalization of the learnt classifier
- How well they can estimate the expertise of each expert, i.e. sensitivity and specificity.

In all above mentioned three aspects, our algorithms performs better than these two baselines.

The rest of the paper organize as follows: Section 2 will give a brief overview of recently proposed algorithms in this area, most of them are deep free(no deep learning); Section 3 introduce the classic EM approach [10], then describes the details of our algorithms; Section 4 consist of experiment setups and experiment analysis; Section 5 gives the conclusion; Contribution is listed in the last section.

2 Related Works

A series of works consider crowdsourcing problem by mixing the learning process and the aggregation process together. [7] transforms the learning from crowds problem into a maximum likelihood estimation (MLE) problem, then implements an EM algorithm to jointly learn the expertise of different experts and the parameters of a logistic regression classifier. [1] extends this method to combine with the deep learning model. [8] also reduces the learning problem to MLE and assumes that the optimal classifier gives the ground truth labels and the experts make independent mistakes conditioning on the ground truth. Unlike our method, these MLE based algorithms are not robust to correlated mistakes. Recently, [5] and [12] propose methods that model multiple experts individually and explicitly in a neural network. However, their works lack theoretical guarantees and are outperformed by our method in the experiments, especially in the naive majority case. Moreover, unlike our method, their methods cannot be used to employ both the data and the crowdsourced labels to forecast the ground truth.

Another series of methods are based on the generative model proposed by [3]. In particular, [4] proposes a method based on Singular Value Decomposition (SVD) which addresses binary labeling problems under the one-coin model. The analysis in it assumes that the labeling matrix is full, that is, each worker labels all items. To relax this assumption, [2] proposes another SVD-based algorithm which explicitly considers the sparsity of the labeling matrix in both algorithm design and theoretical analysis. It proposed an iterative algorithm for binary labeling problems under the one-coin model and extended it to multi-class labeling tasks by converting a k-class problem into $k-1$ binary problems. This line of work assumes that tasks are assigned to workers according to a random regular graph, thus imposing specific constraints on the number of workers and the number of items.

Several works focus on modeling the experts. [15] modeled both expert competence and image difficulty, but did not consider expert bias. [14] models each expert as a multidimensional classifier in an abstract feature space and considers both the bias of the expert and the difficulty of the image. [11] models the crowds by a Gaussian process. [13] considers the generalized Dawid-Skene model [3] which involves the task difficulty. However, these works are still not robust to correlated mistakes.

3 Our Approach

3.1 Traditional Two-coin Model

Let y be the actual but unobserved true label for the instance and $y^j \in\{0,1\}$. Suppose each annotator (i.e. worker/expert) has sensitivity α^j and specificity β^j. Each annotator provides a version of this hidden true label based on two biased coins. If the true label is one, he flips a coin with bias α^j (sensitivity). If the true label is zero, he flips a coin with bias β^j (specificity).

The sensitivity is defined as the probability that an expert labels a positive sample as one.

$$\alpha^j := \Pr[y^j = 1|y = 1]$$

The specificity is defined as the probability that an expert labels a negative sample as zero.

$$\beta^j := \Pr[y^j = 0|y = 0]$$

If we model the classification problem using logistic regression, that is,

$$\Pr[y = 1|\boldsymbol{x}, \boldsymbol{\omega}] = \sigma(\boldsymbol{\omega}^T \boldsymbol{x})$$

where σ denotes sigmoid function $\sigma(z) = 1/\left(1 + e^{-z}\right)$. With a pre-determined threshold value ξ, we can classify the data to "positive class" when $\Pr[y = 1|\boldsymbol{x}, \boldsymbol{\omega}] > \xi$ and classify it to the other class otherwise.

We wish to maximize the likelihood,

$$\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N}\{\Pr[y_i^1, ..., y_i^R|y_i = 1, \boldsymbol{\alpha}]\Pr[y_i = 1|\boldsymbol{x_i}, \boldsymbol{w}]+$$

$$\Pr[y_i^1, ..., y_i^R|y_i = 0, \boldsymbol{\beta}]\Pr[y_i = 0|\boldsymbol{x_i}, \boldsymbol{w}]\}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\omega}\}$ and $\mathcal{D} = \left\{\boldsymbol{x}_i, y_i^1, \dots, y_i^R\right\}_{i=1}^N$. Besides,

$$\Pr\left[y_i^1, \dots, y_i^R|y_i = 1, \boldsymbol{\alpha}\right] = \prod_{j=1}^{R} \Pr\left[y_i^j|y_i = 1, \alpha^j\right] = \prod_{j=1}^{R} \left[\alpha^j\right]^{y_i^j} \left[1-\alpha^j\right]^{1-y_i^j}$$

Similarly,

$$\Pr\left[y_i^1, \dots, y_i^R|y_i = 0, \beta\right] = \prod_{j=1}^{R} \left[\beta^j\right]^{1-y_i^j} \left[1-\beta^j\right]^{j_i^j}$$

For simplicity, we define

$$p_i := \sigma(\boldsymbol{\omega^T x_i})$$
$$a_i := \prod_{j=1}^{R} [\alpha^j]^{y_i^j} [1-\alpha^j]^{1-y_i^j}]$$
$$b_i := \prod_{j=1}^{R} [\beta^j]^{1-y_i^j} [1-\beta^j]^{y_i^j}$$

The problem could be solved by EM algorithm, first we write out joint distribution between data and hidden variables:

$$\ln \Pr[\mathcal{D}, \boldsymbol{y}|\theta] = \sum_{i=1}^{N} y_i \ln p_i a_i + (1-y_i) \ln (1-p_i) b_i$$

with $y = [y_1, \dots, y_N]$, thus

$$\mathbb{E}\{\ln \Pr[\mathcal{D}, \boldsymbol{y}|\theta]\} = \sum_{i=1}^{N} \mu_i \ln p_i a_i + (1-\mu_i) \ln (1-p_i) b_i$$

then:

- E-step. Given $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\omega}\}$ we compute the posterior probability
$$\mu_i \propto \Pr\left[y_i^1, \dots, y_i^R | y_i = 1, \theta\right] \cdot \Pr\left[y_i = 1 | x_i, \theta\right] = \frac{a_i p_i}{a_i p_i + b_i(1-p_i)}$$
- M-step. By maximizing with regards to all parameters, we have closed form solution for $\boldsymbol{\alpha}$,$\boldsymbol{\beta}$.
$$\alpha_j = \frac{\sum_{i=1}^{N} \mu_i y_i^j}{\sum_{i=1}^{N} \mu_i}, \quad \beta_j = \frac{\sum_{i=1}^{N} (1-\mu_i)(1-y_i^j)}{\sum_{i=1}^{N} (1-\mu_i)}$$
For $\boldsymbol{\omega}$, we use the Newton-Raphson update given by $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t - \eta \boldsymbol{H}^{-1} \boldsymbol{g}$,where
$$\boldsymbol{g(\omega)} = \sum_{i=1}^{N} [\mu_i - \sigma(\boldsymbol{\omega^T x_i})] \boldsymbol{x_i}$$
$$\boldsymbol{H(\omega)} = -\sum_{i=1}^{N} [\sigma(\boldsymbol{\omega^T x_i})][1 - \sigma(\boldsymbol{\omega}^T \boldsymbol{x_i})] \boldsymbol{x_i x_i^T}$$

3.2 Our Modification: Specified Data Difficulty

In the model above, we assume that every instance is the same for a certain expert, which is obviously unrealistic. We assume α is correlated with factors in two dimensions, the ability of an expert and the difficulty of a classification task. In addition, we assume the form of α_i^j and β_i^j as

$$\alpha_i^j = \sigma\{\lambda_j^1 (p_i - 1/2)^2\} \tag{1}$$
$$\beta_i^j = \sigma\{\lambda_j^2 (p_i - 1/2)^2\} \tag{2}$$

where λ_j^1 and λ_j^2 are positive real numbers and $p_i = \Pr[y_i = 1 | x_i, \boldsymbol{\omega}]$, which is given by classifier. $\sigma(\cdot)$ denotes the sigmoid function

To make the structural assumption clear, we claim that:

- In terms of the instance, if p_i is close to 1/2, then the instance is close to decision boundary and thus the instance might well be hard to classify.
- In terms of the expert, he might have different abilities dealing with positive instances and negative ones, which implies difference between λ_j^1 and λ_j^2. For larger λ^1 and λ^2, the expert has higher possibility to do the right classification.

In the meanwhile, we train two types of classifiers, one is logistic regression as the original paper, the other is a neural network, which is believed to perform better in complex classification task.

3.3 Modified Learning Problem

Given the training data $\mathcal{D}$ consisting of N instances with annotations from R annotators, the task is to estimate the classifier parameter $\boldsymbol{\omega}$, which can be used for future prediction, the sensitivity ability $\boldsymbol{\lambda^1} = [\lambda_1^1, \lambda_2^1, ..., \lambda_R^1]$ and the specificity ability $\boldsymbol{\lambda^2} = [\lambda_1^2, \lambda_2^2, ..., \lambda_R^2]$ of R experts, which can help evaluate these experts. The estimated ground truth $\{y_1, ..., y_N\}$ is also of interest.

3.4 Maximum Likelihood Estimator

We introduce the parameters $\boldsymbol{\theta} = \{\boldsymbol{\omega}, \boldsymbol{\lambda}\}$, where $\boldsymbol{\omega}$ is the parameters for the classifier, $\boldsymbol{\lambda = \{\lambda^1, \lambda^2\}}$.

Assuming the training instances are all independent with each other, the likelihood function can be factored as

$$\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N} \Pr[y_i^1, ..., y_i^R|\boldsymbol{x_i}, \boldsymbol{\theta}]$$

Conditioning on the true label y_i and let $p_i = \Pr[y_i = 1|\boldsymbol{x_i}, \boldsymbol{\omega}]$

$$\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N}\{\Pr[y_i^1, y_i^2, ..., y_i^R|y_i = 1, \boldsymbol{\lambda^1}, p_i] \cdot p_i + \Pr[y_i^1, y_i^2, ..., y_i^R|y_i = 0, \boldsymbol{\lambda^2}, p_i] \cdot (1 - p_i)\}$$

Assuming experts make decisions independently

$$\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N}\{\prod_{j=1}^{R} \Pr[y_i^j|y_i = 1, \lambda_j^1, p_i] \cdot p_i + \prod_{j=1}^{R} \Pr[y_i^j|y_i = 0, \lambda_j^2, p_i] \cdot (1 - p_i\}$$

The likelihood can be written as

$$\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N}[a_i p_i + b_i(1 - p_i)]$$

where we define

$$a_i := \prod_{j=1}^{R}[\sigma\{\lambda_j^1(p_i - 1/2)^2\}]^{y_i^j}[1 - \sigma\{\lambda_j^1(p_i - 1/2)\}]^{1-y_i^j}$$

$$b_i := \prod_{j=1}^{R}[\sigma\{\lambda_j^2(p_i - 1/2)^2\}]^{1-y_i^j}[1 - \sigma\{\lambda_j^2(p_i - 1/2)\}]^{y_i^j}$$

The maximum-likelihood estimator is found by maximizing the log-likelihood

$$\hat{\boldsymbol{\theta}} = \{\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\lambda}}\}$$

To tackle this MLE problem, we propose two different methods, named **Adaptive Difficulty EM Algorithm** and **Fixed Difficulty EM Algorithm**.

3.5 Adaptive Difficulty EM Algorithm

We use the EM algorithm to compute the maximum likelihood solution. Suppose we know the missing ground truth $\boldsymbol{y} = [y_1, y_2, ..., y_N]$, then the likelihood can be written as

$$\ln\Pr[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}] = \sum_{i=1}^{N} y_i \ln(p_i a_i) + (1 - y_i)\ln(1 - p_i)b_i$$

Each iteration consists of two steps: E-step and M-step.

- **E-step**. Given $\mathcal{D}$ and current estimation of $\boldsymbol{\theta}$, we can compute posterior probability as above

$$\mu_i = \Pr[y_i = 1|y_i^1, ..., y_i^R, \boldsymbol{x_i}, \boldsymbol{\theta}] = \frac{a_i p_i}{a_i p_i + b_i(1 - p_i)}$$

- **M-step**. Given μ, we can maximize the lower bound on the true likelihood,

$$\mathbf{E}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\} = \sum_{i=1}^{N} \mu_i \text{ln} p_i a_i + (1-\mu_i)\text{ln}(1-p_i)b_i$$

  The maximization step for $\boldsymbol{\lambda^1}$, $\boldsymbol{\lambda^2}$ is through SGD approximation. If the classifier is **Logistic**, we use Newton-Raphson update, while for **neural work**, SGD is applied:

$$\tilde{\omega} = \begin{cases} \boldsymbol{\omega} - \eta_{\boldsymbol{\omega}} \nabla_{\boldsymbol{\omega}} \text{E}^{\text{batch}}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\}, & \text{if we use neural network} \\ \boldsymbol{\omega} - \boldsymbol{H}^{-1}\boldsymbol{g}, & \text{if we use Logistic classifier} \end{cases}$$

$$\tilde{\boldsymbol{\lambda^1}} = \boldsymbol{\lambda^1} - \eta_{\boldsymbol{\lambda^1}} \nabla_{\boldsymbol{\lambda^1}} \text{E}^{\text{batch}}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\}$$

$$\tilde{\boldsymbol{\lambda^2}} = \boldsymbol{\lambda^2} - \eta_{\boldsymbol{\lambda^2}} \nabla_{\boldsymbol{\lambda^2}} \text{E}^{\text{batch}}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\}$$

  where $\boldsymbol{g}$, $\boldsymbol{H}$ is gradient vector and Hessian matrix for $\boldsymbol{\omega}$ in $\mathbf{E}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}\}$, and $\eta_{\boldsymbol{\omega}}, \eta_{\boldsymbol{\lambda^1}}$ and $\eta_{\boldsymbol{\lambda^2}}$ are the learning rate with respect to $\boldsymbol{\omega}$, $\boldsymbol{\lambda^1}$and$\boldsymbol{\lambda^2}$, and

$$\mathbf{E}^{\text{batch}}\{\text{lnPr}[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\} = \frac{N}{\#\text{batch}} \sum_{i \in \text{batch}} (\mu_i \text{ln} p_i a_i + (1-\mu_i)\text{ln}(1-p_i)b_i)$$

  where the "batch" is a fixed size random subset of $\{1, 2, \cdots, N\}$.

```
Algorithm 1: update_classifier_parameter
Input: likelihood, classifier_type, $\boldsymbol{\theta}$, $\boldsymbol{\mu}$
Output: updated $\boldsymbol{\theta}$
1  Set maximal iteration time T.
2  for $t = 1, 2, 3, ..., T$ do
3      $\text{E}^{\text{batch}} = \frac{N}{\#\text{batch}} \sum_{i \in \text{batch}}$ likelihood
4      $\tilde{\boldsymbol{\lambda^1}} = \boldsymbol{\lambda^1} - \eta_{\boldsymbol{\lambda^1}} \nabla_{\boldsymbol{\lambda^1}} \text{E}^{\text{batch}}\{\text{likelihood}\}$
5      $\tilde{\boldsymbol{\lambda^2}} = \boldsymbol{\lambda^2} - \eta_{\boldsymbol{\lambda^2}} \nabla_{\boldsymbol{\lambda^2}} \text{E}^{\text{batch}}\{\text{likelihood}\}$
6      if classifier_type=nn then
7          $\boldsymbol{\omega} = \boldsymbol{\omega} - \eta_{\omega} \nabla_{\boldsymbol{\omega}} \text{E}^{\text{batch}}\{\text{likelihood}\}$
8      end
9      if classifier_type=logistic then
10         $\boldsymbol{\omega} = \boldsymbol{\omega} - \boldsymbol{H}^{-1}\boldsymbol{g}$
11     end
12 end
```

```
Algorithm 2: Adaptive Difficulty EM
1  Initialize the parameters $\boldsymbol{\theta} = \{\boldsymbol{\omega}, \boldsymbol{\lambda^1}, \boldsymbol{\lambda^2}\}$.
2  Set maximal iteration time for gradient descent $T$
3  while not convergent do
4      for $i = 1, 2, 3..., N$ do
5          $a_i = \prod_{j=1}^{R} [\boldsymbol{\sigma}\{\lambda_j^1 (p_i - 1/2)^2\}]^{y_i^j} [1 - \boldsymbol{\sigma}\{\lambda_j^1 (p_i - 1/2)\}]^{1-y_i^j}$
6          $b_i = \prod_{j=1}^{R} [\boldsymbol{\sigma}\{\lambda_j^2 (p_i - 1/2)^2\}]^{1-y_i^j} [1 - \boldsymbol{\sigma}\{\lambda_j^2 (p_i - 1/2)\}]^{y_i^j}$
7          $\mu_i = \text{Pr}[y_i = 1 | y_i^1, ..., y_i^R, \boldsymbol{x_i}, \boldsymbol{\theta}] = \frac{a_i p_i}{a_i p_i + b_i (1-p_i)}$
8      end
9      update_classifier_parameter
10 end
```

3.6 Fixed Difficulty EM Algorithm

We can do a little modification to **Adaptive Difficulty EM**, where we change the "difficulty" of each task during the iteration process making the M step hard to solve. Thus, we propose **Fixed Difficulty EM Algorithm**, where we fix the difficulty of each task by $\hat{p}_i = \sum_{j=1}^{R} y_i^j / R$, which is a constant during EM iteration update.

To clarify the rationality, we point out:

- The classification difficulty for a classifier might sometimes differ from that for human annotators.
- The closer $\hat{p}_i$ is to 0.5, the bigger the divergence is. Thus it works in the same direction as the real probability $\Pr[y_i = 1|\boldsymbol{x_i}]$.
- In terms of algorithm, the easier form of likelihood might provide a better estimation of parameters.

Algorithm 3: Fixed Difficulty EM

```
1  Initialize the parameters $\boldsymbol{\theta} = \{\boldsymbol{\omega}, \boldsymbol{\lambda^1}, \boldsymbol{\lambda^2}\}$.
2  Set maximal iteration time for gradient descent T.
3  for i=1,2,3,...,N do
4      $\hat{p}_i = \sum_{j=1}^{R} y_i^j / R$
5  end
6  while not convergent do
7      for $i = 1, 2, 3..., N$ do
8          $a_i = \prod_{j=1}^{R} [\boldsymbol{\sigma}\{\lambda_j^1 (\hat{p}_i - 1/2)^2\}]^{y_i^j} [1 - \boldsymbol{\sigma}\{\lambda_j^1 (\hat{p}_i - 1/2)\}]^{1-y_i^j}$
9          $b_i = \prod_{j=1}^{R} [\boldsymbol{\sigma}\{\lambda_j^2 (\hat{p}_i - 1/2)^2\}]^{1-y_i^j} [1 - \boldsymbol{\sigma}\{\lambda_j^2 (\hat{p}_i - 1/2)\}]^{y_i^j}$
10         $\mu_i = \Pr[y_i = 1|y_i^1, ..., y_i^R, \boldsymbol{x_i}, \boldsymbol{\theta}] = \frac{a_i \hat{p}_i}{a_i \hat{p}_i + b_i (1 - \hat{p}_i)}$
11     end
12     update_classifier_parameter
13 end
```

The initialization of experts' parameters, i.e. $\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2$ is defered to Appendix. We randomly initialize the classifier parameter ω.

4 Experiments

In this section, we evaluate our methods on classification tasks with both synthesized data and real world data.

4.1 Experiment setup

We compare both of our algorithms: **Adaptive Difficulty EM** and **Fixed Difficulty EM** with two baselines: **Majority Voting** and **Original EM** [10]. We compare their performance mainly in three aspects:

- How well they can estimate the ground truth
- The generalization of the learnt classifier
- How well they can estimate the expertise of each expert, i.e. sensitivity and specificity.

For synthesized data, we sample 2000 two-dimensional data points from a two-class Gaussian mixture with a known Bayesian classification boundary.

For the real data, we tried to collect real crowdsourced labels, but it's difficult and expensive to obtain real crowdsourced labels. We use the **Dogs vs. Cats** [6] dataset, which consists of $25,000$ images from 2 classes, dogs and cats and their golden ground truth label.

Note for the above two data set, the real crowdsourced labels are inaccessible, we use the ground truth label to simulate crowdsourced label in two ways: firstly, we simulate noisy labels with pre-chosen confusion matrix, where the labels are more structural, and coincide to our assumption ; secondly, the probability of worker making wrong decision is sampled from a certain discrete distribution, where the labels are more non-structural.

For the rest of this section, we organize our text as below: We cover the details of the synthesized data set in subsection 4.2, and analyze the performance of each algorithm on this data in subsection 4.3; In subsection 4.3 we cover the performance analysis of the **Dogs vs. Cats** dataset, on both structural (4.3.1) and non-structural (4.3.2) labels.

4.2 Two-dimensional Gaussian Mixture dataset

To validate our methods, we use two-dimensional Gaussian mixture with a known Bayesian classification boundary in order to explicitly incorporate the difficulty of datapoint, i.e. the distance to the decision boundary.

In our experiments, we synthesize a Bayesian classification boundary($x + y = 0$) and two Gaussians(mean= (1,1),variance= 1 & mean= (-1,-1),variance= 1). The Gaussian dataset contains 1000 training datapoints and 1000 test datapoints. The classification method used here is logistic regression. We use the distance of each datapoint to the Bayesian classification boundary to the difficulty of each datapoint.

As for the labels, we use five experts with sensitivity α = [0.8 0.6 0.6 0.7 0.6] and specificity β = [0.6 0.7 0.6 0.7 0.7] to determine the the sensitivity ability $\boldsymbol{\lambda}^{\mathbf{1}}$ and the specificity ability $\boldsymbol{\lambda}^{\mathbf{2}}$ of experts. We used both the sensitivity and specificity of experts and datapoint difficulty to recover the parameters of experts at every specific datapoint to synthesize the crowdsourced labels (refer to equation (1) and (2)). For concision, we defer the experiment details to the Appendix.

Based on the labels from multiple experts, we can simultaneously:
(1) estimate the golden ground truth, **(2)** learn a logistic-regression classifier, and **(3)** estimate the sensitivity and specificity of each expert.

So we compare on three different aspects: **(1)** How good is the estimated ground truth? **(2)** How good is generalization of the learnt classifier? **(3)** How well can we estimate the sensitivity and specificity of each expert?

4.2.1 Results

1. Estimated ground truth

Since the estimates of the actual ground truth are probabilistic scores, we can plot the **ROC curves** of the estimated ground truth. Note the in all EM-based algorithms, we use the output of the converged value in E step (i.e. Pr$[y_i = 1|y_i^1, ..., y_i^R, \boldsymbol{x_i}, \boldsymbol{\theta}]$) as the estimated probability. From Figure 2 we can see that the ROC curve for the proposed methods are superior to the majority voting and the original EM algorithm. The proposed algorithm appropriately weights each expert based on their estimated parameters $\boldsymbol{\lambda}$. The average expertise of expert is low so the learning process with majority voting labels is highly biased.

2. Generalization of the learnt classifier

From Table 1 we can see the generalization property of the learnt classifier is consistently well in all EM-based algorithms. Surprisingly, the proposed **Fixed Difficulty EM** algorithm increases the accuracy by a large margin but the AUC of it on the training set is inferior to **Adaptive Difficulty EM**. Our hypothesis is that the fixed imaged difficulty can play the role of regularization.

Table 1: Guassian dataset: Accuracy of learnt classifier on test set

Method	Majority Voting	Original EM	Adaptive Difficulty EM	Fixed Difficulty EM
Accuracy	93.6 ± 0.05	94.00 ± 0.11	93.9 ± 0.15	$\mathbf{94.4 \pm 0.13}$

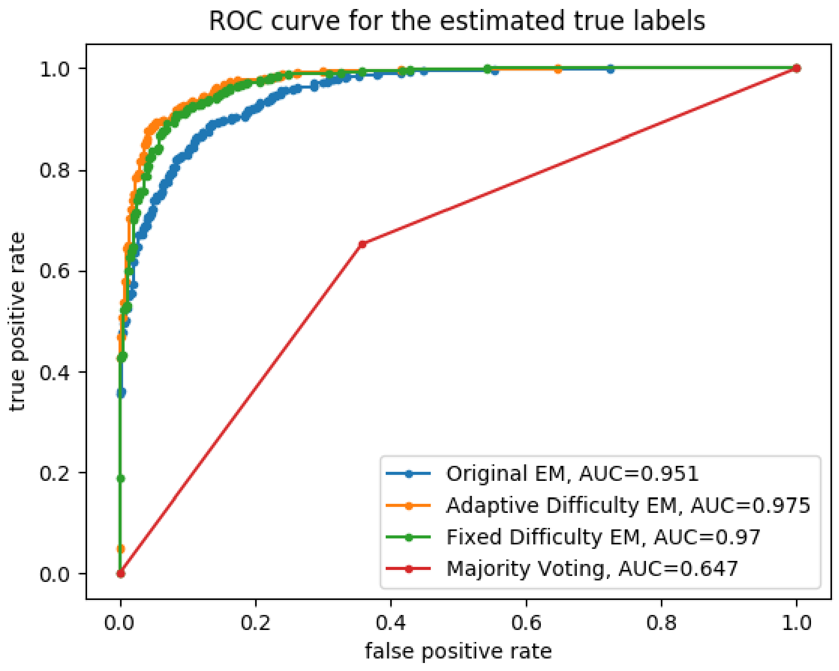

Figure 2: ROC Curve for the estimated true labels. We compare our proposed methods with **Majority Voting** and **Original EM**

3. Estimated expertise

The actual sensitivity and specificity of each expert is marked as blue dots in Figure 3. We can see that the estimated expertise of our proposed EM algorithms is much closer to the actual values of sensitivity and specificity than the original EM algorithm. Moreover, we find that the estimated expertise of the two proposed algorithm is close. So different interpretations of image difficulty do not change the updating trajectory of experts' parameters much. We use $\alpha = \sigma(\frac{\boldsymbol{\lambda}^1}{4})$ and $\beta = \sigma(\frac{\boldsymbol{\lambda}^2}{4})$to recover the sensitivity and specificity of experts.

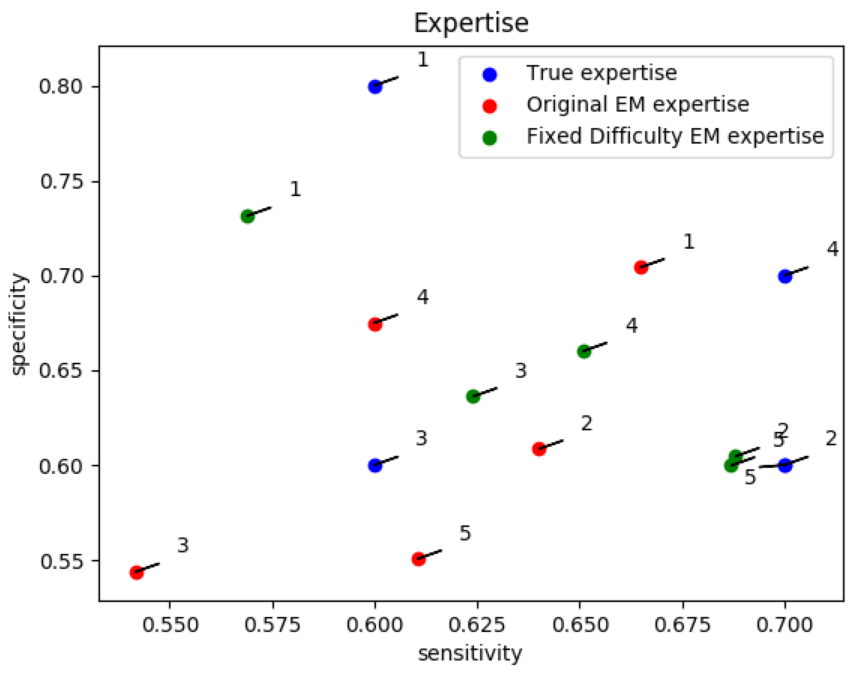

Figure 3: Estimated Expertise on Gaussian dataset. The numbers denote corresponding experts. The expertise of **Adaptive Difficulty EM** and **Fixed Difficulty EM** is very close, so we only plot one of them.

4.3 Dogs vs. Cats dataset

In this experiments, the expert label real-world 2-class dataset. The experiment set up is almost the same as the two-dimension Gaussian mixture dataset except that the classifier used in this experiment is a four-layer convolutional neural network, and we use a pre-trained four-layer convolutional neural network to synthesis image difficulty, i.e the more uncertain the pretrained classifier is the more difficult this data point is. We manually split the dataset into a $12,500$-image training set and a $12,500$-image test set to both test the accuracy of the estimated ground truth on the training set and the generalization performance of the learnt classifier on the test set.

4.3.1 Structural Label

Table 2, Figure 4 and Figure 5 summarize the results. The proposed **Fixed Difficulty EM** algorithm performs well both on estimating true labels and the generalization. The estimated expertise of **Original EM** marginally closer to the ground-truth expertise.

We can see that **Majority Voting** can not converge in this case, and **Original EM** converges poorly. Actually, **Original EM** even can not converge in some experiments.

Table 2: Dogs vs. Cats dataset: Accuracy of learnt classifier on test set

Method	Majority Voting	Original EM	Adaptive Difficulty EM	Fixed Difficulty EM
Accuracy	50.0 ± 0.0	60.2 ± 9.1	70.4 ± 0.9	$\mathbf{74.3 \pm 1.7}$

4.3.2 Non-structural Label

Since in the experiments above, we use our assumption to synthesize the experts' labels, which is very structural. So we would like to compare these methods under settings that may be closer to real

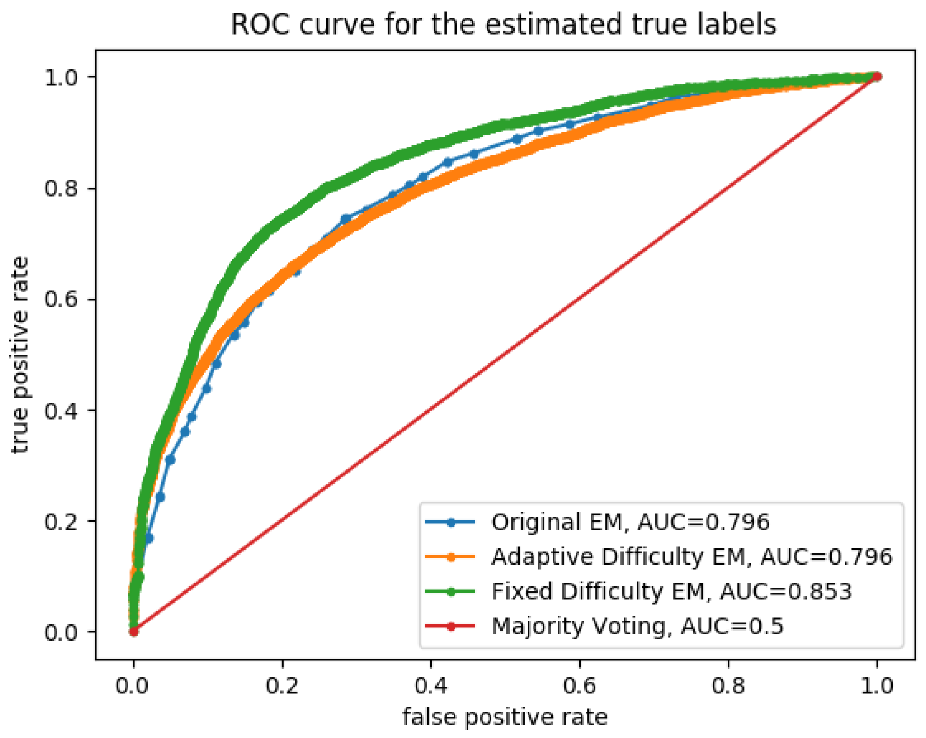

Figure 4: ROC Curve for the estimated true labels. We compare our proposed methods with **Majority Voting** and **Original EM**

world, where the labels tend to be less structural. In such settings, with regard to every datapoint, an expert makes mistakes with probability $\frac{1}{3+k}$, where k is uniformly sampled from $\{0, 1, 2, 3, 4, 5\}$. It should be noted that, to a single expert, the error probabilities corresponding to different images are also different. We only compare the generalization ability of the learnt classifier. The result is shown in Table 3. The proposed algorithms still dominate other algorithms in this setting. The other two algorithms can not converge using non-structural labels.

Table 3: Non-structural label: Accuracy of learnt classifier on test set

Method	Majority Voting	Original EM	Adaptive Difficulty EM	Fixed Difficulty EM
Accuracy	50.0 ± 0.0	50.6 ± 0.6	76.7 ± 0.8	$\mathbf{78.2 \pm 0.3}$

5 Conclusion and Outlook

As an amelioration of [10], we present two EM algorithms for data that has different difficulty, which is a more natural setting in reality. To empirically verify the advantage of our algorithm, we conduct ablation study on synthesized data and real world data. Our methods outperform original method under both situation.

There is still room for improvement. The initialization of EM algorithm is a long-standing problem as EM algorithm only converges to one of its local minima. Some previous work [16] made discussions about this issue, which can be one of our next research target. What's more, [9] tackles crowdsourcing problem with graphical models from a variational inference view, which is quite distinctive and worth further exploration.

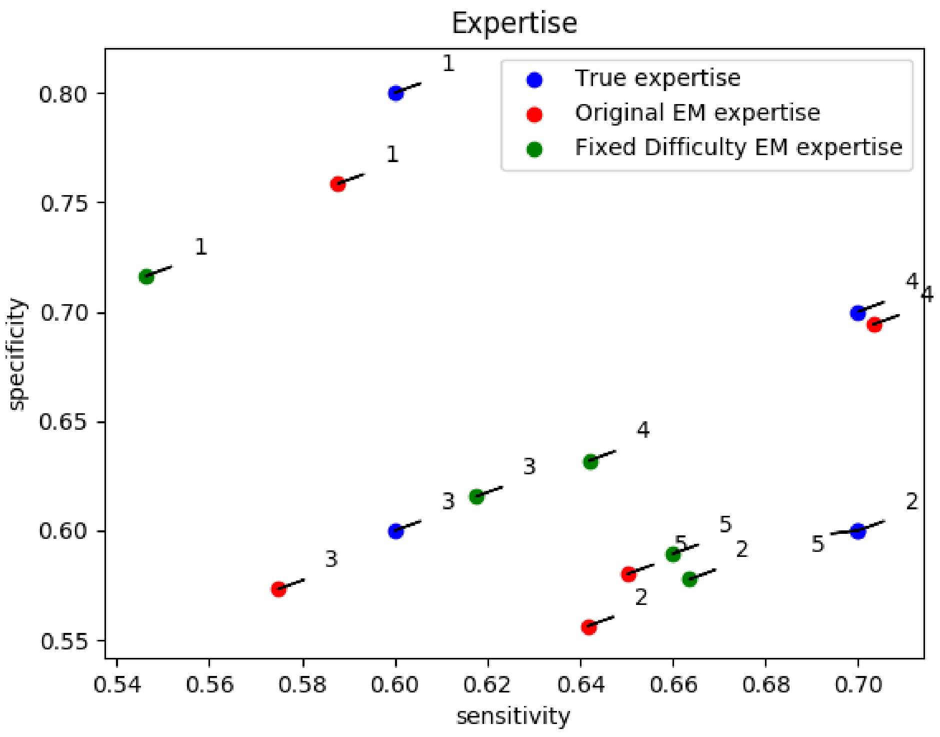

Figure 5: Estimated Expertise on Dogs vs Cats dataset. The numbers denote corresponding experts. The expertise of **Adaptive Difficulty EM** and **Fixed Difficulty EM** is very close, so we only plot one of them.

Contribution

Tianyuan Zhang has tried to use web crawler technique to collect data, but unfortunately failed out of the low qualification of images from the internet. Yilun Xu chose the topic and baseline, and at the meantime he conduct the experiment of Dogs Cats which is the most complicated ones. The two algorithms are put forward by Weiyi Zhang and Tianyuan Zhang. Experiments are designed by Yilun Xu and Tianyuan Zhang. Dinghuai Zhang wrote the article with Weiyi Zhang, besides he conduct the simple experiment of Guassian toy model. The computational resource is supplied by Tianyuan Zhang. The github repo is maintained by Yilun Xu right now.

References

[1] Shadi Albarqouni, Christoph Baur, Felix Achilles, Vasileios Belagiannis, Stefanie Demirci, and Nassir Navab. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE transactions on medical imaging*, 35(5):1313–1321, 2016.

[2] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

[3] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[4] Arpita Ghosh, Satyen Kale, and Preston McAfee. Who moderates the moderators?: crowdsourcing abuse detection in user-generated content. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 167–176. ACM, 2011.

[5] Melody Y Guan, Varun Gulshan, Andrew M Dai, and Geoffrey E Hinton. Who said what: Modeling individual labelers improves classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Kaggle. Dogs vs. cats competition. *https://www.kaggle.com/c/dogs-vs-cats*, 2013.

[7] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

[8] Ashish Khetan, Zachary C Lipton, and Anima Anandkumar. Learning from noisy singly-labeled data. *arXiv preprint arXiv:1712.04577*, 2017.

[9] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *Advances in neural information processing systems*, pages 692–700, 2012.

[10] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

[11] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning*, pages 433–441, 2014.

[12] Filipe Rodrigues and Francisco C Pereira. Deep learning from crowds. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] Nihar B Shah, Sivaraman Balakrishnan, and Martin J Wainwright. A permutation-based model for crowd labeling: Optimal estimation and robustness. *arXiv preprint arXiv:1606.09632*, 2016.

[14] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[15] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[16] Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268, 2014.

A Experiment Details

A.1 The generation of datapoint difficulty and expert parameters

Two-dimensional Gaussian Mixture dataset Denote ground-truth Bayesian classification boundary as $\boldsymbol{w}$ and datapoint i as x_i, and the synthesized datapoint difficulty $\mu_i = \frac{1}{1+w^T x_i}$.

Dogs vs Cats dataset We pre-trained a neural network f and use the ouput of $\mu_i = f(x_i)$ w.r.t datapoint i as the image difficulty.

The expert parameters is calcualted by $\boldsymbol{\lambda^1} = 4 \log \frac{\alpha}{1-\alpha}$ and $\boldsymbol{\lambda^2} = 4 \log \frac{\beta}{1-\beta}$, which is the inverse of $\alpha = \sigma\{\boldsymbol{\lambda}^1 (p - 1/2)^2\}$ and $\beta = \sigma\{\boldsymbol{\lambda}^2 (p - 1/2)^2\}$ where $p = 1$ or $p = 0$.

A.2 Network Architecture

The architecture of the four-layer neural network is:

conv1-relu-maxpooling + conv2-relu-maxpooling + conv3-relu-maxpooling + conv4-relu-maxpooling + batch normalization + fc-relu-dropout + fc + softmax

A.3 Initialization

For all EM based algorithm, we initialize the parameters $\{\alpha, \beta\}$ using the method in [10], then initialize $\{\boldsymbol{\lambda}\}$.

$$\alpha^r = \log \frac{\sum\limits_{i=1}^{N} Q(y_i = c) 1(y_i^r = 1)}{\sum\limits_{i=1}^{N} Q(y_i = 1)} \tag{3}$$

$$\beta^r = \log \frac{\sum\limits_{i=1}^{N} Q(y_i = 0) 1(y_i^r = 0)}{\sum\limits_{i=1}^{N} Q(y_i = 0)} \tag{4}$$

$$\boldsymbol{\lambda_r^1} = 4 * \log \frac{\alpha^r}{1 - \alpha^r} \tag{5}$$

$$\boldsymbol{\lambda_r^2} = 4 * \log \frac{\beta^r}{1 - \beta^r} \tag{6}$$

where $1(y_i^m = c') = 1$ when $y_i^m = c'$ and $1(y_i^m = c') = 0$ when $y_i^m \neq c'$ and N is the total number of datapoints. We average all crowdsourced labels to obtain $Q(y_i = c) := \frac{1}{R} \sum\limits_{r=1}^{R} 1(y_i^r = c)$.