Aggregating Crowd Wisdom with Side Information via a Clustering-based Label-aware Autoencoder

Li'ang Yin, Yunfei Liu, Weinan Zhang, Yong Yu
Shanghai Jiao Tong University, No.800 Dongchuan Road, Shanghai, China
{yinla,liuyunfei,wnzhang,yyu}@apex.sjtu.edu.cn

Abstract

Aggregating crowd wisdom infers true labels for objects, from multiple noisy labels provided by various sources. Besides labels from sources, side information such as object features is also introduced to achieve higher inference accuracy. Usually, the learning-from-crowds framework is adopted. However, the framework considers each object in isolation and does not make full use of object features to overcome label noise. In this paper, we propose a clustering-based label-aware autoencoder (CLA) to alleviate label noise. CLA utilizes clusters to gather objects with similar features and exploits clustering to infer true labels, by constructing a novel deep generative process to simultaneously generate object features and source labels from clusters. For model inference, CLA extends the framework of variational autoencoders and utilizes maximizing a posteriori (MAP) estimation, which prevents the model from overfitting and trivial solutions. Experiments on real-world tasks demonstrate the significant improvement of CLA compared with the state-of-the-art aggregation algorithms.

1 Introduction

Aggregating crowd wisdom aims at inferring true labels from multiple noisy labels collected from sources, which is also known as truth inference or label aggregation for crowdsourcing [Li *et al.*, 2016]. As a key technology for massive object labeling, aggregating crowd wisdom is usually considered in the domain of unsupervised learning since labeling tasks do not provide ground-truth labels. A traditional aggregation algorithm takes multiple noisy labels from different sources as input and infers true labels for objects. A simple aggregation algorithm, majority voting, has been widely used in many voting scenarios, which infers the most voted label as the true label. More sophisticated algorithms usually exploit source credibility to model the relationship between true labels and labels from sources (source labels), where high credible sources are supposed more likely to provide correct labels [Zheng *et al.*, 2017]. Besides source labels, recent work introduces side information as object features to achieve higher inference accuracy [Raykar *et al.*, 2010; Rodrigues and Pereira, 2018]. Those algorithms usually adopt the learning-from-crowds framework where aggregated labels from source labels guide the classification of object features. The framework is analogized to supervised learning where aggregated labels are analogized to training labels.

However, *label noise* is still a crucial problem in previous algorithms imitating supervised learning. Traditional supervised learning uses perfect training labels, but in label aggregation, labels from real-world sources are noisy and may lead to noisy aggregated labels. Using aggregated labels to directly guide the classifier may cause imprecise decision boundaries (Figure 1b). Side information such as object features is expected to alleviate label noise by bringing objects with similar features together. However, previous algorithms utilizing object features consider each object in isolation and do not provide explicit solutions to the label noise problem.

To alleviate label noise and take advantage of object features, we propose a clustering-based label-aware autoencoder (CLA). CLA utilizes clusters to gather objects with similar features, where objects in the same cluster are supposed to have similar true labels. A simple and intuitive mechanism is illustrated in Figure 1c. Specifically, CLA assigns each cluster a centroid and a label distribution, and exploits a deep generative process, to simultaneously generate latent embedding from the cluster centroid and an intermediate label from the label distribution. The latent embedding is further used to generate object features, while the intermediate label is used to generate source labels. The deep generative process is inspired by deep Gaussian mixture models in the framework of variational autoencoders (VAE) [Dilokthanakul *et al.*, 2016; Jiang *et al.*, 2016] and label-aware autoencoders [Yin *et al.*, 2017]. CLA combines generating source labels and object features from clusters and adopts cluster labels to infer true labels. Since a cluster label contains labeling information of all objects in the cluster rather than a single object, CLA alleviates the problem of label noise. For model inference, CLA introduces constraints for model parameters by extending the framework of VAE [Kingma and Welling, 2013] and utilizing maximizing a posteriori (MAP) estimation. The MAP estimation can be written into a likelihood term and a prior term over model parameters. The likelihood term is inferred by constructing VAE. The prior term acts as a regularizer that prevents model parameters from overfitting and trivial solutions. Experiments on real-world tasks show that CLA signif-

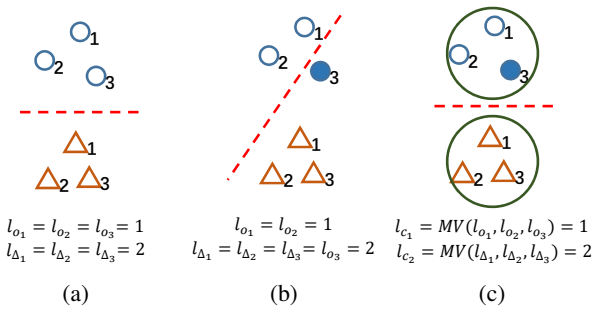


Figure 1: Comparing with the noise-free case (a), label noise causes an imprecise decision boundary (b), and clustering alleviates the problem (c). Consider a binary labeling task where circles denote category 1 and triangles denote category 2. Features of objects are represented by their relative positions in subfigures. We simply assume an object o_1 infers an aggregated label by majority voting from source labels and $l_{o_1} = 1$ indicates the majority voting result is category 1. A dashed line denotes the learned decision boundary. (a) An ideal case where all sources provide correct labels. Therefore aggregated results via majority voting are correct and the learned decision boundary with the guide of aggregated results is precise. (b) If object o_3 has noisy labels from sources which lead to an incorrect majority voting label $l_{o_3} = 2$, then a classifier may mistake o_3 as it is from category 2 and learn an imprecise decision boundary. (c) Clustering is used to alleviate label noise. In the subfigure, objects in relatively short distances form clusters. For a simple illustration, majority voting is applied to inferred labels of objects in the same cluster to achieve a cluster label (l_{c_1} and l_{c_2}). Cluster labels are more likely to be correct by considering all labeling information of objects in the same cluster rather than a single object, that alleviates label noise and guides a model to learn a correct decision boundary.

icantly improves inference accuracy compared with the state-of-the-art aggregation algorithms.

2 Related Work

Traditional aggregation algorithms use source labels to infer true labels. Among them, weighted majority voting [Aydin *et al.*, 2014], trust propagation [Yin *et al.*, 2008], and generative models [Bachrach *et al.*, 2012; Liu *et al.*, 2012; Simpson *et al.*, 2013; Venanzi *et al.*, 2014; Tian and Zhu, 2015] are extensively researched. Label-aware autoencoders (LAA) build a bridge between label aggregation and neural networks which makes aggregation modeling more flexible [Yin *et al.*, 2017]. Features of objects are utilized to further achieve higher aggregation accuracy. Most work exploits the framework named learning from crowds [Raykar *et al.*, 2010]. Yan et al. further model source credibility affected by features based on the framework [Yan *et al.*, 2014]. The learning-from-crowds framework originally uses a linear classifier to classify object features. Recent work utilizes deep neural networks to enhance the learning capacity of the classifier [Albarqouni *et al.*, 2016; Dizaji and Huang, 2018; Rodrigues and Pereira, 2018]. Besides learning from crowds, a generative framework generates both source labels and object features from a true label of the object [Felt *et al.*, 2014]. Among them, some algorithms design a specific model structure for specific data types (e.g., latent Dirichlet allocation for textual data [Rodrigues *et al.*, 2017] and convolutional neural networks for image data [Albarqouni *et al.*, 2016; Cao *et al.*, 2019]). Our model CLA is a general approach and does not assume a specific data type.

The other topic related to CLA is clustering, especially in the framework of VAE. VAE providing effective model inference for generative models, has advantages on unsupervised and semi-supervised learning tasks [Kingma and Welling, 2013; Kingma *et al.*, 2014]. Label aggregation as an unsupervised learning task can be modeled by LAA, which is a variant of VAE. Deep clustering methods, especially deep Gaussian mixture models benefit from VAE by simplifying model inference [Dilokthanakul *et al.*, 2016; Jiang *et al.*, 2016]. Though label aggregation and clustering problems are respectively modeled in the framework of VAE and achieve attractive performance, to the best of our knowledge, there is no pioneering work combining both of them in a unified framework to utilize clustering to infer true labels.

3 Methods

3.1 Problem Definition

Given a label set $\{l_{mn}\}, m \in \{1, ..., M\}$ and $n \in \{1, ..., N\}$, where M is the number of objects and N is the number of sources. l_{mn} denotes a label object m receives from source n. $l_{mn} \in \{1, ..., K\}$ is a categorical label, where K is the number of categories. For the convenience of representation, all labels belonging to object m are collected to form a label vector $\boldsymbol{l}_m$. $\boldsymbol{l}_m$ has N source blocks and contains all labeling information of the object [Yin *et al.*, 2017]. Traditional label aggregation aims to infer a true label $\tilde{t}_m$ for each object m given the label vector. In this paper, feature vector $\boldsymbol{x}_m \in \mathbb{R}^I$ of each object is also utilized to make high-quality inference. I is the dimensionality of the feature vector.

3.2 Clustering-based Label-aware Autoencoder

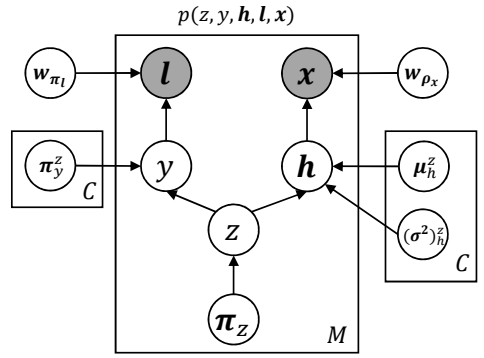
Given label set $\{\boldsymbol{l}_m\}$ and feature set $\{\boldsymbol{x}_m\}$, a clustering-based label-aware autoencoder (CLA) optimizes model parameters $\boldsymbol{\Theta}$ via maximizing a posteriori (MAP) estimation.

$$\begin{aligned}\boldsymbol{\Theta}^* &= \arg\max_{\boldsymbol{\Theta}} p(\boldsymbol{\Theta}|\{\boldsymbol{l}_m\}, \{\boldsymbol{x}_m\}) \\ &= \arg\max_{\boldsymbol{\Theta}} \sum_{m=1}^{M} \log p(\boldsymbol{l}_m, \boldsymbol{x}_m|\boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta}). \end{aligned} \quad (1)$$

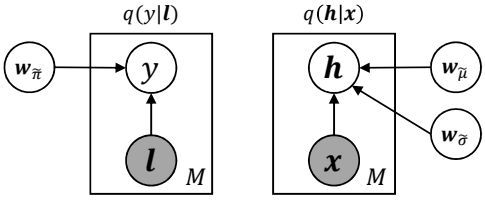
The first term on the right hand is the log-likelihood, and the second term is the log-prior of model parameters. The equation assumes each object has a label vector and a feature vector independent with other objects given model parameters. We also use $p(\boldsymbol{l}, \boldsymbol{x})$ to denote $p(\boldsymbol{l}_m, \boldsymbol{x}_m|\boldsymbol{\Theta})$, for simplicity.

To model the log-likelihood, we introduce a deep generative process to generate label vectors and feature vectors from clusters (Figure 2a). Suppose there are C clusters. For an object, we first draw a cluster index $z \in \{1, ..., C\}$. Given cluster index z, we draw intermediate label y from the chosen cluster, and then draw label vector $\boldsymbol{l}$ from y. Meanwhile, we draw latent embedding $\boldsymbol{h}$ from the cluster, and then draw feature vector $\boldsymbol{x}$ from $\boldsymbol{h}$. A formal description is given below.

For each object:



(a) Generative process (Decoder)



(b) Encoder

Figure 2: Plate notation for CLA. (a) The generative process (decoder) describes joint distribution $p(z, y, \boldsymbol{h}, \boldsymbol{l}, \boldsymbol{x})$. (b) Encoder $q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})$ is decoupled into $q(z|\boldsymbol{l}, \boldsymbol{x})$, $q(y|\boldsymbol{l})$, and $q(\boldsymbol{h}|\boldsymbol{x})$. $q(y|\boldsymbol{l})$ and $q(\boldsymbol{h}|\boldsymbol{x})$ are modeled by neural networks respectively, while $q(z|\boldsymbol{l}, \boldsymbol{x})$ is computed directly. Gray circles illustrate observed data. Prior distributions for model parameters are omitted for simplicity.

1. Draw cluster index z with

$$p(z) = \text{Cat}(\boldsymbol{\pi}_z). \tag{2}$$

$\boldsymbol{\pi}_z$ represents the probability of each cluster being chosen in the categorical distribution. Subscript z in $\boldsymbol{\pi}_z$ is only a name indicating the distribution is used to generate variable z.

2. Draw intermediate label y given cluster index z with

$$p(y|z) = \text{Cat}(\boldsymbol{\pi}_y^z), \tag{3}$$

where $y \in \{1, ..., K\}$. The superscript z in $\boldsymbol{\pi}_y^z$ indicates each cluster has its own distribution $\boldsymbol{\pi}_y^z$, $z \in \{1, ..., C\}$.

3. Draw label vector $\boldsymbol{l}$ from y with

$$p(\boldsymbol{l}|y) = \prod_{n=1}^{N} p(l_n|y),$$
$$p(l_n|y) = \text{Cat}\big(\boldsymbol{\pi}_l^n(y)\big). \tag{4}$$

Since label vector $\boldsymbol{l}$ is a combination of N source blocks, we draw a label for each source block from a categorical distribution independently. Each source n has its own distribution parameter $\boldsymbol{\pi}_l^n(y)$, which is calculated by using y as input, via a neural network with weight matrix $\boldsymbol{w}_{\pi_l}^n$. To make a brief representation, we concatenate all N weight matrices into a weight matrix $\boldsymbol{w}_{\pi_l}$ to form a big network, which computes labels for all source blocks and generates label vector $\boldsymbol{l}$ from the intermediate label y.

4. Draw latent embedding $\boldsymbol{h}$ from cluster z with

$$p(\boldsymbol{h}|z) = \mathcal{N}\bigg(\boldsymbol{\mu}_h^z, diag\big((\boldsymbol{\sigma}^2)_h^z\big)\bigg). \tag{5}$$

$\boldsymbol{h} \in \mathbb{R}^J$ where J is the dimensionality of the embedding feature space. Each cluster has a multivariate normal distribution with mean $\boldsymbol{\mu}_h^z$ and a simplified covariance matrix $diag\big((\boldsymbol{\sigma}^2)_h^z\big)$. $\boldsymbol{\sigma}^2$ is a vector where each element measures the variance of the corresponding element in $\boldsymbol{h}$.

5. Draw feature vector $\boldsymbol{x}$ from $\boldsymbol{h}$ with

$$p(\boldsymbol{x}|\boldsymbol{h}) = \text{Ber}\big(\boldsymbol{\rho}_x(\boldsymbol{h})\big), \tag{6}$$

if each element in $\boldsymbol{x}$ is Bernoulli distributed. $\text{Ber}\big(\boldsymbol{\rho}_x(\boldsymbol{h})\big)$ is a multivariate Bernoulli distribution. Vector $\boldsymbol{\rho}_x(\boldsymbol{h}) \in [0, 1]^I$ is calculated via a multi-layer neural network with weights $\boldsymbol{w}_{\rho_x}$. An alternative is a multivariate normal distribution if $\boldsymbol{x}$ is normally distributed.

By introducing the generative process, we have the joint distribution for each object

$$p(z, y, \boldsymbol{h}, \boldsymbol{l}, \boldsymbol{x}) = p(z)p(y|z)p(\boldsymbol{l}|y)p(\boldsymbol{h}|z)p(\boldsymbol{x}|\boldsymbol{h}). \tag{7}$$

The log-likelihood in Eqn. (1) is then deduced into an ELBO and solved by constructing VAE [Kingma and Welling, 2013]

$$\begin{aligned}
&\log p(\boldsymbol{l}, \boldsymbol{x}) \\
&= ELBO(\boldsymbol{l}, \boldsymbol{x}) + KL\big(q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})||p(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})\big) \\
&\geq ELBO(\boldsymbol{l}, \boldsymbol{x}) \\
&= \sum_{z,y} \int_{\boldsymbol{h}} q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x}) \log \frac{p(z, y, \boldsymbol{h}, \boldsymbol{l}, \boldsymbol{x})}{q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})} d\boldsymbol{h}
\end{aligned} \tag{8}$$

where $p(z, y, \boldsymbol{h}, \boldsymbol{l}, \boldsymbol{x})$ can be regarded as the decoder, and $q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})$ can be regarded as the encoder in CLA.

With the mean-field method, $q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})$ is decoupled

$$\begin{aligned}
q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x}) &= q(z|\boldsymbol{l}, \boldsymbol{x})q(y|\boldsymbol{l}, \boldsymbol{x})q(\boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x}) \\
&= q(z|\boldsymbol{l}, \boldsymbol{x})q(y|\boldsymbol{l})q(\boldsymbol{h}|\boldsymbol{x}).
\end{aligned} \tag{9}$$

y is assumed to be conditionally independent with $\boldsymbol{x}$ given $\boldsymbol{l}$. Similar assumption applies for $\boldsymbol{h}$. In the equation, $q(z|\boldsymbol{l}, \boldsymbol{x})$ is directly computed (see Eqn. (15)).

$q(y|\boldsymbol{l})$ is a categorical distribution

$$q(y|\boldsymbol{l}) = \text{Cat}\big(\tilde{\boldsymbol{\pi}}(\boldsymbol{l})\big), \tag{10}$$

where probability $\tilde{\boldsymbol{\pi}}(\boldsymbol{l})$ is computed via a neural network with weight matrix $\boldsymbol{w}_{\tilde{\pi}}$, given $\boldsymbol{l}$ as input.

$q(\boldsymbol{h}|\boldsymbol{x})$ is a multivariate normal distribution

$$q(\boldsymbol{h}|\boldsymbol{x}) = \mathcal{N}\bigg(\tilde{\boldsymbol{\mu}}(\boldsymbol{x}), diag\big(\tilde{\boldsymbol{\sigma}}^2(\boldsymbol{x})\big)\bigg), \tag{11}$$

where $\tilde{\boldsymbol{\mu}}(\boldsymbol{x})$ and $\tilde{\boldsymbol{\sigma}}^2(\boldsymbol{x})$ are modeled by two multi-layer neural networks with weights $\boldsymbol{w}_{\tilde{\mu}}$ and $\boldsymbol{w}_{\tilde{\sigma}}$ respectively.

By introducing the decoder and the encoder, the framework of CLA is illustrated in Figure 2, with model parameters

$$\boldsymbol{\Theta} \equiv \big\{\boldsymbol{\pi}_z, \{\boldsymbol{\pi}_y^z\}, \boldsymbol{w}_{\pi_l}, \{\boldsymbol{\mu}_h^z\}, \{(\boldsymbol{\sigma}^2)_h^z\}, \boldsymbol{w}_{\rho_x}, \boldsymbol{w}_{\tilde{\pi}}, \boldsymbol{w}_{\tilde{\mu}}, \boldsymbol{w}_{\tilde{\sigma}}\big\}. \tag{12}$$

According to Eqn. (1) and (8), CLA is optimized by maximizing the ELBO and the log-prior term.

After model optimization, cluster labels are utilized to infer a true label for object m. Cluster labels contain labeling

information of all objects belonging to the cluster and are effective to alleviate the problem of label noise.

$$p(\tilde{t}_m) = p(y|\boldsymbol{l}_m, \boldsymbol{x}_m) = \sum_z p(z|\boldsymbol{l}_m, \boldsymbol{x}_m) p(y|z)$$
$$\approx \text{Cat}\big(\sum_{c=1}^{C} q(z_m = c|\boldsymbol{l}_m, \boldsymbol{x}_m)\boldsymbol{\pi}_y^c\big) \quad (13)$$

An inferred true label is then chosen by the category which achieves the maximum probability. We use posterior probability $q(z|\boldsymbol{l}, \boldsymbol{x})$ instead of $p(z|\boldsymbol{l}, \boldsymbol{x})$ since $p(z|\boldsymbol{l}, \boldsymbol{x})$ is not directly modeled in CLA.

3.3 Model Inference: ELBO

We write out the ELBO by using Eqn. (7) and (9).

$$\begin{aligned} &ELBO(\boldsymbol{l}, \boldsymbol{x}) \\ &= \sum_{z,y} \int_{\boldsymbol{h}} q(z, y, \boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x}) \log \frac{p(z)p(y|z)p(\boldsymbol{h}|z)p(\boldsymbol{l}|y)p(\boldsymbol{x}|\boldsymbol{h})}{q(z|\boldsymbol{l}, \boldsymbol{x})q(y|\boldsymbol{l})q(\boldsymbol{h}|\boldsymbol{x})} d\boldsymbol{h} \\ &= \sum_z q(z|\boldsymbol{l}, \boldsymbol{x}) \log \frac{p(z)}{q(z|\boldsymbol{l}, \boldsymbol{x})} + \sum_{z,y} q(z|\boldsymbol{l}, \boldsymbol{x})q(y|\boldsymbol{l}) \log p(y|z) \\ &\quad + \sum_y q(y|\boldsymbol{l}) \log \frac{p(\boldsymbol{l}|y)}{q(y|\boldsymbol{l})} + \int_{\boldsymbol{h}} q(\boldsymbol{h}|\boldsymbol{x}) \log \frac{p(\boldsymbol{x}|\boldsymbol{h})}{q(\boldsymbol{h}|\boldsymbol{x})} d\boldsymbol{h} \\ &\quad + \sum_z \int_{\boldsymbol{h}} q(z|\boldsymbol{l}, \boldsymbol{x})q(\boldsymbol{h}|\boldsymbol{x}) \log p(\boldsymbol{h}|z) d\boldsymbol{h}. \end{aligned} \quad (14)$$

All terms in the equation can be derived by directly using distribution definitions (we do not show detailed solutions), except for $q(z|\boldsymbol{l}, \boldsymbol{x})$. Here we solve $q(z|\boldsymbol{l}, \boldsymbol{x})$ via

$$\begin{aligned} q(z|\boldsymbol{l}, \boldsymbol{x}) &= \sum_y \int_{\boldsymbol{h}} q(y|\boldsymbol{l}, \boldsymbol{x})q(\boldsymbol{h}|\boldsymbol{l}, \boldsymbol{x})q(z|y, \boldsymbol{h}) d\boldsymbol{h} \\ &= \sum_y \int_{\boldsymbol{h}} q(y|\boldsymbol{l})q(\boldsymbol{h}|\boldsymbol{x})p(z|y, \boldsymbol{h}) d\boldsymbol{h} \\ &\approx \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \tilde{\boldsymbol{\pi}}(\boldsymbol{l})|_k \, p(z|y = k, \boldsymbol{h}^t), \end{aligned} \quad (15)$$

where $\boldsymbol{v}|_k$ denotes the k-th element in the vector. The integral over $q(\boldsymbol{h}|\boldsymbol{x})$ is approximated by sampling $\boldsymbol{h}^t$ from distribution $q(\boldsymbol{h}|\boldsymbol{x})$ T times. $p(z|y = k, \boldsymbol{h}^t)$ is a calculable probability by exploiting the generative process

$$\begin{aligned} &p(z = c|y = k, \boldsymbol{h}^t) \\ &= \frac{p(y = k|z = c)p(\boldsymbol{h}^t|z = c)p(z = c)}{\sum_{c'=1}^{C} p(y = k|z = c')p(\boldsymbol{h}^t|z = c')p(z = c')}, \end{aligned} \quad (16)$$

where $p(y = k|z = c)$, $p(\boldsymbol{h}^t|z = c)$, and $p(z = c)$ can be directly computed from corresponding distributions (3), (5), and (2), respectively. Eqn. (16) is an analogy to the 'E-step' in Expectation-Maximization methods which estimates the distribution of latent variables by using current model parameters. We calculate probability $p(z|y, \boldsymbol{h})$ rather than directly modeling $q(z|y, \boldsymbol{h})$, since there is no simple distribution describing $q(z|y, \boldsymbol{h})$, to the best of our knowledge.

3.4 Model Inference: Regularizer

The log-prior term in Eqn. (1) plays an assistant but important role: prevent parameters from overfitting and set prior distributions to avoid trivial solutions. We use the term as a regularizer and assign coefficient λ to control the strength.

For neural network weights, namely, $\boldsymbol{w}_{\pi_l}$, $\boldsymbol{w}_{\rho_x}$, $\boldsymbol{w}_{\tilde{\pi}}$, $\boldsymbol{w}_{\tilde{\mu}}$, and $\boldsymbol{w}_{\tilde{\sigma}}$, a prior is normal distribution $\mathcal{N}(0, 1)$ for each element in weights to prevent weights from overfitting.

For parameters related with clusters, namely $\boldsymbol{\pi}_z$, $\{\boldsymbol{\pi}_y^z\}$, $\{\boldsymbol{\mu}_h^z\}$, and $\{(\boldsymbol{\sigma}^2)_h^z\}$, following prior distributions are used.

$$p(\boldsymbol{\pi}_z) = \text{Dir}(\boldsymbol{\alpha}^m), \quad (17)$$
$$p(\boldsymbol{\pi}_y^z) = \text{Dir}(\boldsymbol{\beta}^z), \quad (18)$$
$$p(\boldsymbol{\mu}_h^z) = \mathcal{N}(\boldsymbol{\mu}^z, \boldsymbol{I}), \quad (19)$$
$$p\big((\boldsymbol{\sigma}^2)_h^z|_j\big) = \text{Inv-Gamma}(a, b). \quad (20)$$

$\text{Dir}(\boldsymbol{\alpha}^m)$ assigns each object a prior Dirichlet distribution. $\text{Dir}(\boldsymbol{\beta}^z)$ assigns each cluster a prior Dirichlet distribution. Mean $\boldsymbol{\mu}_h^z$ of each cluster is assigned a multivariate normal distribution. The variance vector $\boldsymbol{\sigma}^2$ has each element generated from an Inverse-Gamma distribution. We set $a = b = 2$ to make $\mathbb{E}[\boldsymbol{\sigma}^2|_j] = 1$, $j \in \{1, ..., J\}$. We find these prior distributions work well in practice, while a shared distribution (no regularizer or a uniform regularizer) has a force to push clusters together which leads to fewer clusters than expected and inferior results. The coefficient λ of the regularizer is set as the reciprocal of the size of model parameters. This is a balance between ELBO and regularizer, which prevents the optimizing process from favoring the regularizer term even if a model has a million parameters.

We exploit pre-training to find appropriate prior distributions. Specifically, we pre-train $q(\boldsymbol{h}|\boldsymbol{x})$ by constructing a mirror structure to form an autoencoder, which learns embedding $\boldsymbol{h}$ in the middle layer by reconstructing $\boldsymbol{x}$. Then we concatenate $\boldsymbol{h}$ and majority voting label y_{mv} from $\boldsymbol{l}$ and apply the K-means clustering on the concatenated vectors to obtain initial C clusters. The cluster index for each object is encoded into a smoothed one-hot vector to avoid zeros and used as $\boldsymbol{\alpha}^m$ in Eqn. (17). The centroid of each cluster is used as $\boldsymbol{\mu}^z$ in Eqn. (19). Source labels in each cluster are collected by majority voting, which forms vector $\boldsymbol{\beta}^z$ in Eqn. (18).

4 Experiments

4.1 Implementation Details

CLA has only a few hyperparameters and is easy to train. $\boldsymbol{w}_{\rho_x}$ has a fully connected multi-layer structure with node numbers $\{40, 100, I/2, I\}$ from the bottom up. I is the number in the output layer, while 40 is the dimensionality of embedding feature space. $\boldsymbol{w}_{\tilde{\mu}}$, $\boldsymbol{w}_{\tilde{\sigma}}$ have similar structures with VAE and we use node numbers $\{I, I/2, 100, 40\}$ from the bottom up. Sampling time $T = 5$. Cluster number C is discussed in the subsequent section. We implement CLA by TensorFlow and use gradient ascent for training[1]. Hyperparameter search adopts a similar manner with LAA by splitting a dataset into a training set and a validation set [Yin *et al.*, 2017]. Here the learning rate is 0.001. Training is stable and usually achieves desirable inference accuracy after 1,500 epochs.

[1]Demo code is at https://github.com/coverdark/cla_demo

Algorithm	document	bill	head	shape	forehead	throat	underpart	breast
MV	0.7128	0.8168	0.8760	0.8754	0.8651	0.8934	0.9385	0.7608
TF	0.7223	0.8170	0.8760	0.8797	0.8651	0.8934	0.9385	0.7608
DS	0.6937	0.8221	0.8584	0.8777	0.8540	0.8772	0.9415	0.7499
IBCC	0.7088	0.8216	0.8473	0.8944	0.8497	0.8677	0.9370	0.7457
DARE	0.7245	0.8145	0.8652	0.9105	0.8619	0.8913	0.9430	0.7630
LAA	0.7251	0.8175	0.8762	0.8996	0.8651	0.8936	0.9385	0.7610
MomResp†	0.8416	0.8248	0.8810	0.9113	0.8719	0.8964	0.9466	0.7656
LC†	0.8632	0.8263	0.8881	0.9090	0.8662	0.8971	0.9456	0.7661
DLC†	0.8763	0.8381	0.8777	0.9108	0.8674	0.8936	0.9417	0.7684
ML†	-	0.8410	0.8873	0.9211	0.8704	0.8959	0.9468	0.7633
CrowdDeepAE†	0.8781	0.8377	0.8870	0.9135	0.8780	0.8957	0.9388	0.7615
CLA†	**0.8931***	**0.8654***	**0.9112***	**0.9316***	**0.8992***	**0.9174***	**0.9483***	**0.8024***

† Algorithms utilizing object features.
* The bolder values are significant via a t-test: p-values < 0.005 on all tasks.

Table 1: Overall Inference Accuracy Comparison

4.2 Datasets

Reuters contains a **document** categorization task. 1,786 documents from 8 categories receive labels from online users. 38 users contribute labels giving an average of 3 answers per document [Rodrigues *et al.*, 2017]. We apply latent Dirichlet allocation to bag-of-words feature vectors to obtain 200 topics as object features [Blei *et al.*, 2003].

CUB-200-2010 dataset contains tasks to label local characteristics for 6,033 bird images [Welinder *et al.*, 2010]. We use 7 binary labeling tasks, namely **bill** (bill shape is all-purpose or not), **head** (head pattern is plain or not), **shape** (shape is perching-like or not), **forehead** (forehead is black or not), **throat** (throat is black or not), **underpart** (underpart is yellow or not), and **breast** (breast pattern is solid or not). For each task, there are about 500 users contributing labels and each image receives 5 labels. We collect ground truth from whatbird.com for evaluation. For an image, we use 287 local attributes collected from online users as object features.

4.3 Overall Inference Accuracy

We compare inference accuracy among representative aggregation algorithms. Overall inference accuracy is the ratio of the number of correct inferred objects to the overall number of objects. Compared algorithms are: **MV** (majority voting), **TF** (TruthFinder [Yin *et al.*, 2008]), **DS** (Dawid & Skene's model [Dawid and Skene, 1979]), **IBCC** (a bayesian version of the DS model [Kim and Ghahramani, 2012]), **DARE** (a generative model with source credibility and object difficulty [Bachrach *et al.*, 2012]), **LAA** (label-aware autoendcoders [Yin *et al.*, 2017]), **MomResp** (a model generating both object features and source labels from true labels [Felt *et al.*, 2014]), **LC** (learning from crowds [Raykar *et al.*, 2010]), **DLC** (deep learning from crowds [Rodrigues and Pereira, 2018], **ML** (a multiple labelers method [Yan *et al.*, 2014]), **CrowdDeepAE** ([Dizaji and Huang, 2018]), **CLA** (proposed model, a clustering-based label-aware autoencoder). A K-coin model [Raykar *et al.*, 2010] (K is the number of label categories) is exploited to model source credibility for MomResp, LC, DLC, ML, CrowdDeepAE, and CLA for fair comparison. Since there is randomness in the clustering of CLA, we run the algorithm 20 times and report the average. Results are illustrated in Table 1.

From Table 1, TF, DS, IBCC, DARE, and LAA have only a slight or no advantage compared with MV, which indicates the difficulty of modeling source credibility from sparse and conflicting source labels. On the other hand, algorithms utilizing object features usually achieve higher performance. In these algorithms, DLC and CrowdDeepAE are better than LC on some tasks, which indicates the advantage of exploiting deep neural networks. However, on the other tasks, DLC or CrowdDeepAE do not show advantages because of label noise. Deep neural networks enhance learning capability but are easier to be trapped in imprecise decision boundaries in unsupervised learning. ML has a good performance on CUB-200-2010 tasks, but fails on the document task, due to the difficulty of modeling source credibility from noisy features. Among the compared methods, CLA achieves the highest accuracy. The result is significant by conducting a t-test between CLA and the other algorithm achieving the highest accuracy. CLA uses clusters to alleviate label noise and is more effective than the state-of-the-art algorithms.[2]

4.4 Effect of Cluster Number

A key factor making CLA effective is that CLA usually exploits more clusters than label categories. Since CLA assumes objects in the same cluster have similar true labels, the assumption is satisfied when the number of clusters is sufficient. Though ideally, the deep generative process can perfectly cluster objects into corresponding label categories, the ideal case generally does not happen on real-world data. More clusters make it easier to form small clusters that contain similar objects in a relatively short distance in the embedding feature space. In Figure 3, we illustrate the inference

[2]Here we remind the limit of CLA when object features are too noisy to form effective clusters. We conduct experiments on a sentiment polarity dataset [Rodrigues *et al.*, 2013], but CLA (Acc. 0.9078) and other algorithms utilizing object features cannot outperform IBCC (0.9150).

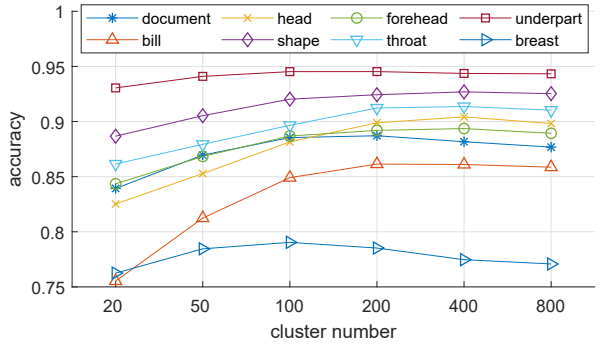


Figure 3: Inference accuracy varies with different cluster numbers.

Figure 4: The clustering result on the bill task ($C = 200$). 20 bird categories are illustrated with different colors.

accuracy along with different numbers of clusters. The accuracy increases when the number of clusters increases, and achieves the highest when the number ranges from 100 to 400. Then the accuracy slightly goes down when the number of clusters is very large. When the number is too small, each cluster contains too many objects even they are not very similar. That breaks the assumption of CLA and results in inaccurate cluster labels. When the number of clusters is too large, a cluster contains too few objects, which is insufficient to infer a reliable cluster label.

Interestingly, the CUB-200-2010 dataset contains images from 200 real-world bird categories and clusters from CLA correspond to these underlying bird categories. In Figure 4, we show the clustering result on the bill task when cluster number $C = 200$. For a clear illustration, we use t-SNE [Maaten and Hinton, 2008] to map embedding $\boldsymbol{h}$ into 2-dimension and plot 20 bird categories with different colors. This figure intuitively shows the learned embedding and clusters, where bird categories roughly accord with learned clusters. Images of the same bird category usually share the same characteristics, that supports the effectiveness of CLA.

4.5 Effect of Regularizer

To show the effect of the regularizer in CLA, we compare results with no regularizer, with a uniform regularizer, and with an appropriate regularizer as described in the Regularizer section. The uniform regularizer uses a uniform distribution in Eqn. (17), (18), and a normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ in Eqn. (19). We experiment on the forehead task, the other tasks show similar trends. In Figure 5, we illustrate the number of objects in each cluster with cluster number $C = 200$. Clusters are sorted by the number of objects in descending order.

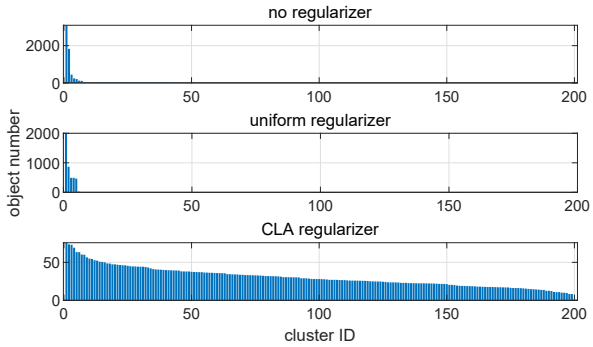


Figure 5: Effect of the regularizer on the forehead task ($C = 200$).

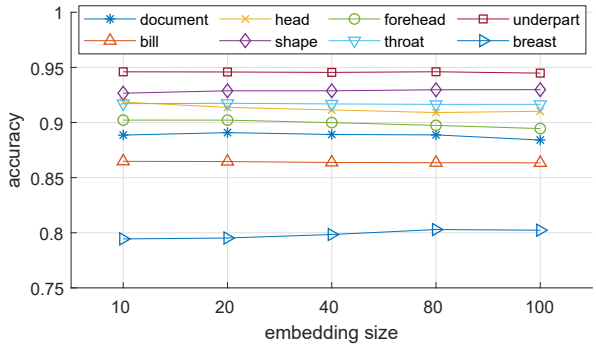


Figure 6: CLA is robust with different settings of embedding size.

A model collapses to a few clusters with no regularizer, or with a uniform regularizer. The collapse leads to a similar inferior result of insufficient cluster numbers, as shown in Figure 3. The collapse does not happen with a CLA regularizer, and the model can give full play to the advantages of CLA.

4.6 Effect of Embedding Size

CLA is robust with different settings of embedding size. We set dimensionality of the embedding feature space ranging from 10 to 100 for CLA and illustrate the corresponding results in Figure 6. The accuracy does not show a significant change with different embedding sizes.

5 Conclusion

This paper proposes a clustering-based label-aware autoencoder (CLA) to utilize object features and alleviate label noise for crowd wisdom aggregation. CLA exploits a deep generative process to generate both source labels and object features from clusters. CLA extends the framework of VAE and utilizes MAP estimation for model inference. Experimental results on real-world tasks show that CLA significantly improves inference accuracy compared with the state-of-the-art aggregation algorithms. Besides high performance and robustness, experiments show the intuition of learned embedding and clusters to support the effectiveness of CLA.

Acknowledgments

The corresponding authors Yong Yu and Weinan Zhang thank the support of NSFC (61702327, 61772333).

References

[Albarqouni *et al.*, 2016] 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.

[Aydin *et al.*, 2014] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[Bachrach *et al.*, 2012] Yoram Bachrach, Thore Graepel, Tom Minka, and John Guiver. How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *ICML-12*, pages 1183–1190, 2012.

[Blei *et al.*, 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[Cao *et al.*, 2019] Peng Cao, Yilun Xu, Yuqing Kong, and Yizhou Wang. Max-mig: an information theoretic approach for joint learning from crowds. *arXiv preprint arXiv:1905.13436*, 2019.

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

[Dilokthanakul *et al.*, 2016] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint*, (arXiv:1611.02648), 2016.

[Dizaji and Huang, 2018] Kamran Ghasedi Dizaji and Heng Huang. Sentiment analysis via deep hybrid textual-crowd learning model. In *AAAI*, 2018.

[Felt *et al.*, 2014] Paul Felt, Robbie Haertel, Eric K. Ringger, and Kevin D. Seppi. Momresp: A bayesian model for multi-annotator document labeling. *LREC*, 2014.

[Jiang *et al.*, 2016] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint*, (arXiv:1611.05148), 2016.

[Kim and Ghahramani, 2012] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *AISTATS*, pages 619–627, 2012.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Kingma *et al.*, 2014] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, pages 3581–3589, 2014.

[Li *et al.*, 2016] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. A survey on truth discovery. *ACM SIGKDD Explorations Newsletter*, 17(2):1–16, 2016.

[Liu *et al.*, 2012] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 692–700, 2012.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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

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

[Rodrigues *et al.*, 2013] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. Learning from multiple annotators: distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436, 2013.

[Rodrigues *et al.*, 2017] Filipe Rodrigues, Mariana Lourenço, Bernardete Ribeiro, and Francisco C Pereira. Learning supervised topic models for classification and regression from crowds. *IEEE PAMI*, 39(12):2409–2422, 2017.

[Simpson *et al.*, 2013] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision Making and Imperfection*, pages 1–35. Springer, 2013.

[Tian and Zhu, 2015] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In *NIPS*, pages 1621–1629, 2015.

[Venanzi *et al.*, 2014] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web*, pages 155–164, 2014.

[Welinder *et al.*, 2010] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. (CNS-TR-2010-001), 2010.

[Yan *et al.*, 2014] Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. Learning from multiple annotators with varying expertise. *Machine learning*, 95(3):291–327, 2014.

[Yin *et al.*, 2008] Xiaoxin Yin, Jiawei Han, and Philip S Yu. Truth discovery with multiple conflicting information providers on the web. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):796–808, 2008.

[Yin *et al.*, 2017] Li'ang Yin, Jianhua Han, Weinan Zhang, and Yong Yu. Aggregating crowd wisdoms with label-aware autoencoders. In *IJCAI*, pages 1325–1331. AAAI Press, 2017.

[Zheng *et al.*, 2017] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.