Leveraging Instance Features for Label Aggregation in Programmatic Weak Supervision

Jieyu Zhang* University of Washington

Linxin Song* Waseda University

Alexander Ratner University of Washington

Abstract

Programmatic Weak Supervision (PWS) has emerged as a widespread paradigm to synthesize training labels efficiently. The core component of PWS is the *label model*, which infers true labels by aggregating the outputs of multiple noisy supervision sources abstracted as *labeling functions (LFs)*. Existing statistical label models typically rely only on the outputs of LF, ignoring the instance features when modeling the underlying generative process. In this paper, we attempt to incorporate the instance features into a statistical label model via the proposed FABLE. In particular, it is built on a mixture of Bayesian label models, each corresponding to a global pattern of correlation, and the coefficients of the mixture components are predicted by a Gaussian Process classifier based on instance features. We adopt an auxiliary variable-based variational inference algorithm to tackle the non-conjugate issue between the Gaussian Process and Bayesian label models. Extensive empirical comparison on eleven benchmark datasets sees FABLE achieving the highest averaged performance across nine baselines. Our implementation of FABLE can be found in `https://github.com/JieyuZ2/wrench/blob/main/wrench/labelmodel/fable.py`.

1 INTRODUCTION

The deployment of machine learning models typically relies on large-scale labeled data to regularly train and evaluate the models. To collect labels, practitioners have increasingly resorted to Programmatic Weak Supervision (PWS) (Ratner et al., 2016; Zhang et al., 2022a; Song et al., 2022), a paradigm in which labels are generated cheaply and efficiently. Specifically, in PWS, users develop weak supervision sources abstracted as simple programs called *labeling functions (LFs)*, rather than make individual annotations. These LFs could efficiently produce noisy votes on the true label or abstain from voting based on external knowledge bases, heuristic rules, *etc.*. To infer the true labels, various *statistical label models* (Ratner et al., 2016, 2019; Fu et al., 2020) are developed to aggregate the labels output by LFs.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

**Signifies equal contribution as co-first authors.*

One of the major technical challenges in PWS is how to infer the true labels given the noisy and potentially conflict labels of multiple LFs. While being diverse in assumptions and modeling techniques, existing statistic label models typically rely solely on the LFs' labels (Ratner et al., 2016; Bach et al., 2017; Varma et al., 2017; Cachay et al., 2021). In this paper, we argue that incorporating instance features into a statistical label model has significant potential to improve the inferred truth. Intuitively, statistical label models aim to recover the pattern of correlation between the LF labels and the ground truth; it is natural to assume that similar instants would share a similar pattern and therefore the instance features could be indicative of the pattern of each instant. When ignoring the instance features, statistical label models have to assume that the patterns or the LF correctness is instance-independent, which is unlikely to be true for real-world dataset.

To attack this problem, we propose FABLE (Feature-Aware laBeL modEl), which exploits the instance features to help identify the correlation pattern of instants. We build FABLE upon a recent model named EBCC (Li et al., 2019), which is a mixture model where each mixture component is a popular Bayesian extension to the DS model (Dawid and Skene, 1979) and aims to capture one sort of LF and true label correlation. To incorporate instance features, we propose to make the mixture coefficients a categorical distribution explicitly depending on instance features. In particular, a predictive Gaussian process (GP) is adopted to learn the distribution of mixture coefficients, connecting the correlation patterns with instance features. However, the categorical distribution of mixture coefficients is non-conjugate to the Gaussian prior, hindering the usage of efficient Bayesian inference algorithm, *e.g.*, variational inference. To overcome this, we introduce a number of auxiliary variables to augment the likelihood function to achieve the desired conjugate representation of our model. Note that there are a couple of recently proposed neural network-based models (Ren et al., 2020; Rühling Cachay et al., 2021) that also leverage instance features, but via neural network. We include them as baselines for comparison and highlight that these neural network-based models typically require gold validation set for hyperparameter tuning and early stopping to be performant with comparison to a statistical model like FABLE.

We conduct extensive experiments on synthetic dataset with varying size and 11 benchmark datasets. Compared with state-of-the-art baselines, FABLE achieves the highest averaged performance and ranking. More importantly, to help understand when FABLE works well and verify our arguments, we measure the correlation of instance features and the LF correctness, *i.e.*, Corr(X, LFs). Then, we calculate the Pearson's correlation coefficient between Corr(X, LFs) and the gain of FABLE over EBCC on synthetic dataset, which is 0.496 with p-value < 0.01, indicating that leveraging instance feature is more beneficial when the LF correctness indeed depends on the features.

2 RELATED WORKS

In PWS, researches have developed a bunch of statistical label model. Ratner et al. (2016) models the joint distribution between LF and ground truth labels to describe the distribution in terms of pre-defined factor functions. Ratner et al. (2019) models the distribution via a Markov network and recover the parameters via a matrix completion-style approach, while Fu et al. (2020) models the distribution via a binary Ising model and recover the parameters by triplet methods. There are other statistical models designed for extended PWS setting (Shin et al., 2021) or for extended definition of LFs, *e.g.*, partial labeling functions (Yu et al., 2022), indirect labeling functions (Zhang et al., 2021a), and positive-only labeling functions (Zhang et al., 2022b). Besides the statistical label models, researchers have recently proposed neural network-based models to leverage instance features (Ren et al., 2020; Rühling Cachay et al., 2021), while in this work, we aim to incorporate instance features into a pure statistical model.

Prior to PWS, statistical models for label aggregation were separately developed in the field of crowdsourcing. Dawid and Skene (1979) used a confusion matrix parameter to generative model LF labels conditioned on the item's true annotation, for clinical diagnostics. Kim and Ghahramani (2012) formulated a Bayesian generalization with Dirichlet priors and inference by Gibbs sampling, while Li et al. (2019) incorporate the subtypes as mixture correlation and decoupled the confusion matrix, which make the Bayesian generation process become a mixture model. Their analysis of inferred worker confusion matrix clustering is a natural precursor to modelling worker correlation.

3 PRELIMINARIES

In this section, we first introduce the setup and notation of the programmatic weak supervision (PWS), then discuss two representative Bayesian models that can be used in PWS. We also discuss the multi-class Gaussian process classification, which is related to our proposed method.

3.1 Notation

Let $X = \{\vec{x}_1, ..., \vec{x}_N\}$ denote a training set with N featured data samples. Assume that there are L labeling functions (LFs) $\vec{y}_i = [y_{i1}, ..., y_{iL}]$ with $j \in [L]$, each of which classifies N each sample into one of K categories or abstain (outputting -1). Let z_i be the latent true label of the sample i, y_{ij} the label that LF j assigns to the item i, Y_i the set of LFs who have labelled the item i.

3.2 Bayesian Classifier Combination (BCC) Models

Independent BCC. The iBCC (Kim and Ghahramani, 2012) model is a directed graphical model and a popular extension to David-Skene (DS) (Dawid and Skene, 1979) model by making a conditional independence assumption between LFs. The iBCC model assumes that given the true label z_i of x_i, LF labels to x_i are generated independently by different LFs,

$$p(y_{i1}, ..., y_{iL} \mid z_i) = \prod_{j=1}^{L} p(y_{ij} \mid z_i). \tag{1}$$

This was referred as the LF's conditional independence assumption. However, the underlying independence assumptions prevent the model from capturing correlations between labels from different LF.

Enhanced BCC. The EBCC model (Li et al., 2019) is an extension of iBCC, which import M subtypes to capture the correlation between LFs and aggregated the captured correlation by tensor rank decomposition.

The joint distribution of observing the outputs of multiple LFs can be approximated by a linear combination of more rank-1 tensors, known also as tensor rank decomposition (Hitchcock (1927)), i.e.,

$$p(y_1, ..., y_L \mid z = k) \approx \sum_{m=1}^{M} \vec{\pi}_{km} \vec{v}_{1km} \otimes \cdots \otimes \vec{v}_{Lkm}, \tag{2}$$

where $\otimes$ is the tensor product. EBCC interpreted the tensor decomposition as a mixture model, where $\vec{v}_{1km} \otimes \cdots \otimes$

$\vec{v}_{Lkm}$ are mixture component shared by all the data samples, and π_{km} is the mixture coefficient. This comes out that

$$p(y_1, ..., y_L \mid z) = \sum_{m=1}^{M} p(g = m \mid z) \prod_{j=1}^{L} p(y_j \mid z, g = m)$$

here g is an auxiliary latent variable used for indexing mixture components. All the mixture components are the result of categorical distribution governed by parameter β_k where $\beta_{kk} = a$ and $\beta_{kk'} = b$, which is equivalent to assuming that every LF has correctly labelled a items under every class, and has to make all kinds of mistakes b times. The M components under class k can be seen as M subtypes, each of which can be used to explain the correlation between LF labels given class k (Li et al., 2019).

3.3 Multi-class Gaussian Process Classification

The multi-class Gaussian process (GP) classification model consists of a latent GP prior for each class $\vec{f} = (f_{i1}, ..., f_{iK})$, where $f_i \sim \text{GP}(m, \Sigma)$, m is the mean over samples, Σ is the kernel function. The conditional distribution is modeled by a categorical likelihood,

$$p(y_i = k \mid x_i, \vec{f}_i) = h^{(k)}(\vec{f}_i(x_i)), \tag{3}$$

where $h^{(k)}(\cdot)$ is a function that maps the real vector of the GP values to a probability vector. For $h(\cdot)$, the most common way to form a categorical likelihood is through the softmax transformation

$$p(y_i = k \mid \vec{f}_i) = \frac{\exp(f_{ik})}{\sum_{k=1}^{K} \exp(f_{ik})} \tag{4}$$

where f_{ik} denotes the $f^k(x_i)$ and for clarity, we omit the conditioning on x_i.

4 METHODS

In this section, we introduce the proposed FABLE model. In a nutshell, it connects the mixture coefficients of the EBCC model with the instance features via a predictive Gaussian process (GP). Then, we introduce a bunch of auxiliary variables to handle the non-conjugation in the model to ensure efficient variational inference. Finally, we present the generative process, joint distribution, and the inference process of the FABLE model.

4.1 Leveraging instance features via mixture coefficient

In this work, we aim to explicitly incorporate instance features into a statistical label model built upon EBCC. To attack this problem, we leverage the Gaussian process (GP) classification. Specifically, we model the mixture coefficients of EBCC as the output of a GP classifier, which inputs the instance features. We generate $N \times K \times M$ functions for each data, class, and subtypes, and take the logistic-softmax distribution for each subtype and class to acquire the mixture coefficient for each data. In particular, we rewrite the Equation 2 as

$$\begin{aligned} & p(y_{i1}, ..., y_{iL} \mid z_i = k) \\ & = \sum_{m=1}^{M} \pi_{ikm} \left[\vec{v}_{1km} \otimes \cdots \otimes \vec{v}_{Lkm}\right] \\ & \approx \sum_{m=1}^{M} h_{\text{softmax}}^{(k,m)}(\sigma(\vec{f}_i)) \left[\vec{v}_{1km} \otimes \cdots \otimes \vec{v}_{Lkm}\right], \end{aligned}$$

where $\sigma(\cdot)$ is sigmoid function and $\pi_{ikm} = p(g_i = m \mid z_i)$. f_i is GP's latent functions for sample x_i with $f_i = f(\vec{x}_i)$ and $f_i \sim \text{GP}(m_i, \Sigma)$. We will soon discuss the details and advantages of our usage of GP classifier in the sequel.

4.2 Handling the Non-conjugate Prior

Given the proposed model, we would like to infer the true labels via the standard mean field variational inference process following prior work (Li et al., 2019). However, a key challenge that prevents us from performing variational inference is that as a categorical likelihood function, softmax is non-conjugate to the Gaussian prior, so the variational posterior $q(f_{ikm})$ cannot be derived analytically. Inspired by Polson et al. (2013); Galy-Fajou et al. (2020), we propose to solve the non-conjugate mapping function in the complete data likelihood by introducing a number of auxiliary latent variables such that the augmented complete data likelihood falls into the exponential family, which is conjugate to the Gaussian prior.

In the following section we (1) decouple the GP latent variables f_{ikm} in the denominator by introducing of a set of auxiliary λ-variables and the logistic-softmax function, (2) simplify the model likelihood by introducing Poisson random variables, and (3) use a Pólya-Gamma representation of the sigmoid function to achieve the desired conjugate representation of our model.

Decouple GP latent variables. Following Galy-Fajou et al. (2020), we first replace the softmax likelihood with the logistic-softmax likelihood,

$$\pi_{ikm} = h_{\text{softmax}}^{(k,m)}(\sigma(\vec{f}_i)) = \frac{\sigma(f_{ikm})}{\sum_{j=1}^{K} \sum_{n=1}^{M} \sigma(f_{ijn})}, \tag{5}$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function. To remedy the intractable normalizer term $\sum_{j=1}^{K} \sum_{n=1}^{M} \sigma(f_{ijn})$, we use the integral identity $\frac{1}{x} =$

$\int_0^\infty e^{-\lambda x} d\lambda$ and express the likelihood (5) as

$$\begin{aligned} & h^{(k,m)}_{\text{softmax}}(\sigma(\vec{f}_i)) \\ =& \sigma(f_{ikm}) \int_0^\infty \exp\left(-\lambda_i \sum_{j=1}^{K}\sum_{n=1}^{M} \sigma(f_{ijn})\right) d\lambda_i. \end{aligned} \quad (6)$$

By interpreting λ_i as an additional latent variable, we obtain the augmented likelihood

$$p(\pi_{ikm} \mid f_{ikm}, \lambda_i) = \sigma(f_{ikm}) \prod_{j=1}^{K}\prod_{n=1}^{M} \exp(-\lambda_i \sigma(f_{ijn})), \quad (7)$$

here we impose the improper prior $p(\lambda_i) \propto \mathbb{1}_{[0,\infty]}, \forall i \in [1, N]$. The improper prior is not problematic since it leads to a proper complete conditional distribution, as we will see at the end of the section.

Poisson augmentation By leveraging the moment generation function of the Poisson distribution $\mathrm{Po}(\lambda)$

$$\exp(\lambda(z-1)) = \sum_{n=0}^{\infty} z^n \, \mathrm{Po}(z \mid \lambda).$$

Using $z = \sigma(-f)$, we rewrite the exponential factors as,

$$\begin{aligned} \exp(\lambda_i \sigma(f_{ikm})) =& \exp(\lambda_i(\sigma(f_{ikm}) - 1)) \\ =& \sum_{j=1}^{K}\sum_{n=1}^{M} (\sigma(-f_{ijn}))^{\upsilon_{ijn}} \, \mathrm{Po}(\upsilon_{ijn} \mid \lambda_i), \end{aligned}$$

which leads to the augmented likelihood

$$\begin{aligned} & p(\pi_{ikm} \mid f_{ikm}, \upsilon_{ikm}, \lambda_i) \\ &= \sigma(f_{ikm}) \cdot \prod_{j=1}^{K}\prod_{n=1}^{M} (\sigma(-f_{ijn}))^{\upsilon_{ijn}}, \end{aligned} \quad (8)$$

where $\upsilon_{ikm} \sim \mathrm{Po}(\lambda_i)$.

Complete with Pólya-Gamma In the last step, we aim for a Gaussian representation of the sigmoid function. The Pólya-Gamma representation allows us for rewriting the sigmoid function as a scale mixture of Gaussian,

$$\sigma(z)^n = \int_0^\infty 2^{-n} \exp\left(\frac{\upsilon z}{2} - \frac{z^2}{2}\omega\right) \mathrm{PG}(\omega \mid \upsilon, 0) \quad (9)$$

where $\mathrm{PG}(\omega \mid \upsilon, b)$ is a Pólya-Gamma distribution. By applying this augmentation to Equation 8 we obtain

$$\begin{aligned} & p(\pi_{ikm} \mid f_{ikm}, \upsilon_{ikm}, \omega_{ikm}) \\ &= \frac{2^{-(\pi_{ikm}+\upsilon_{ikm})} \exp\left\{\frac{(\pi_{ikm}-\upsilon_{ikm}) f_{ikm}}{2}\right\}}{\exp\left\{\frac{(f_{ikm})^2}{2}\omega_{ikm}\right\}}, \end{aligned} \quad (10)$$

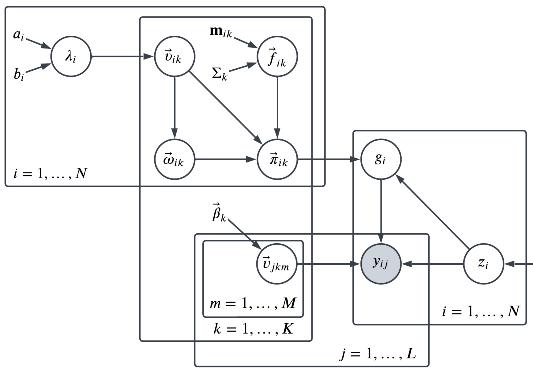

Figure 1: The probabilistic graphical model of FABLE.

where $\omega_{ikm} \sim \mathrm{PG}(\omega_{ikm} \mid \upsilon_{ikm}, 0)$ are Pólya-Gamma variables.

Finally, the complete conditions of the GPs' f_{ikm} are

$$\begin{aligned} & p(f_{ikm} \mid \pi_{ikm}, \omega_{ikm}, \upsilon_{ikm}) \\ =& \mathcal{N}\left(f_{ikm} \mid \frac{1}{2}\hat{\Sigma}_{km}\left(\mathbb{E}[\pi_{ikm}] - \mathbb{E}[\upsilon_{ikm}]\right), \hat{\Sigma}_{km}\right), \end{aligned} \quad (11)$$

where $\hat{\Sigma}_{km} = (\Sigma^{-1}_{km} + \mathrm{diag}(\mathbb{E}[\omega_{ikm}]))^{-1}$. For the conditional distribution, λ_i we have

$$p(\lambda_i \mid \vec{\upsilon}_i) = \mathrm{Ga}\left(1 + \sum_{j=1}^{K}\sum_{n=1}^{M} \gamma_{ijn} + 1, K\right), \quad (12)$$

where $\mathrm{Ga}(\cdot|a, b)$ indicated a gamma distribution with parameter a and b. γ_{ijn} is the parameter of the joint distribution of $p(\omega_{ikm}, \upsilon_{ikm})$, detailed in Appendix B.1.

In summary, by integrating three auxiliary random variable $\lambda_i, \upsilon_{ikm}, \omega_{ikm}$, we successfully turn the posterior of f_{ikm} from non-conjugate softmax to exponential family form, the Gaussian distribution, which is easy to infer by adopting variational inference with $q(f_{ikm}) \sim \mathcal{N}(\hat{m}, \hat{\Sigma})$.

4.3 The Generative Process and Joint Distribution

Here, we summarize the generative process of the proposed model. We use the GP latent functions F and the corresponding auxiliary variables Ω and Υ to generate the mixture coefficient Π. There are $K \times M$ subtypes in total, and we assume the item i belongs to the g_i-th subtype of the class, z_i as in EBCC. The proposed model is shown in Figure 1 and its generative process is:

```
1. for $i$ in 1...$N$
     $\lambda_i \sim \mathrm{Ga}(a_i, b_i)$
     for $k$ in 1...$K$
       $\vec{\upsilon}_{ik} \sim \mathrm{Po}(\lambda_i)$
```

$$\vec{\omega}_{ik} \sim \mathrm{PG}(\vec{v}_{ik}, 0)$$
$$\vec{f}_{ik} \sim \mathcal{N}(\vec{m}_{ik}, \Sigma_k)$$
$$\vec{\pi}_{ik} \sim \mathrm{Ga}(\vec{\phi}_{ik}, \vec{\xi}_{ik})$$

2. for k in $1...K$
   for m in $1...M$, for j in $1...L$
   $\vec{v}_{jkm} \sim \mathrm{Dir}(\vec{\beta}_k)$

3. $\vec{\tau} \sim \mathrm{Dir}(\vec{\alpha})$

4. for i in $1...N$
   $z_i \sim \mathrm{Cat}(\vec{\tau})$
   $g_i \sim \mathrm{Cat}(\vec{\pi}_{z_i})$
   for $j \in \mathcal{L}_i$
   $y_{ij} \sim \mathrm{Cat}(\vec{v}_{j z_i g_i})$

Following the generative process, the joint distribution is

$$\begin{aligned}&p(\boldsymbol{\lambda}, \Omega, \Upsilon, F, \Pi, V, G, Z, T, Y)\\ =&p(\boldsymbol{\lambda})p(\Upsilon \mid \boldsymbol{\lambda})p(\Omega \mid \Upsilon)p(F)p(\Pi \mid \Upsilon, \Omega, F)\cdot\\ &p(V)p(T)p(Z \mid T)p(G \mid \Pi, Z)p(Y \mid Z, G, V). \quad (13)\end{aligned}$$

4.4 The Inference Algorithm

The goal of the inference is to find the most likely Z (true labels) given the LF labels Y, data features X and all hyperparameters,

$$\arg\max_Z p(Z \mid Y, \alpha, a_i, b_i, \vec{m}_{ik}, \Sigma_{km}, \vec{\beta}_k),$$

which is intractable to solve directly. Therefore, we adopt a mean-field variational approach that seeks a distribution q that approximates $p(\boldsymbol{\lambda}, \Omega, \Upsilon, F, \Pi, V, G, Z, T \mid Y, \alpha, a_i, b_i, \vec{m}_{ik}, \Sigma_{km}, \vec{\beta}_k)$, where q is assumed to be factorized as

$$\begin{aligned}&q(\boldsymbol{\lambda}, \Omega, \Upsilon, F, \Pi, V, G, Z, T)\\ =&q(\boldsymbol{\lambda})q(\Omega, \Upsilon)q(F)q(\Pi)q(V)q(G, Z)q(T)\\ =&\prod_i \mathrm{Ga}(\lambda_i \mid a_i, b_i)\cdot\prod_k \mathrm{PG}(\vec{\omega}_{ik} \mid \vec{v}_{ik}, \vec{c}_{ik})\,\mathrm{Po}(\vec{v}_{ik} \mid \vec{\gamma}_{ik})\\ &\mathcal{N}(\hat{m}_{ik}, \hat{\Sigma}_{ik})\,\mathrm{Ga}(\vec{\pi}_{ik} \mid \vec{\phi}_{ik}, \vec{\xi}_{ik})\cdot\prod_k\prod_m\prod_j \mathrm{Dir}(\vec{v}_{kmj} \mid \vec{\mu}_{kmj})\cdot\\ &\prod_i q(g_i, z_i)\cdot\mathrm{Dir}(\vec{\tau} \mid \vec{\nu}).\end{aligned}$$

Since the joint distribution is fully factorized in q, it is easy to solve $\arg\max_Z q(Z)$ by finding k that maximizes every individual $q(z_i = k)$, i.e. $\tilde{z}_i = \arg\max_k q(z_i = k)$.

Let $\rho_{ikm} = q(z_i = k, g_i = m)$, then follow the standard mean-field variational Bayes steps, we can derive the update rules shown below

$$\begin{aligned}\rho_{ikm} &= e^{\mathbb{E}_q[\log \tau_k] + \mathbb{E}_q[\log \pi_{ikm}] + \sum_{j\in\mathcal{L}_i}\mathbb{E}_q[\log v_{kmjy_{ij}}]}\\ q(z_i = k) &= \sum_m \rho_{ikm}\\ \mu_{kmj} &= \beta_{kl} + \sum_{i\in\mathcal{N}_j}\rho_{ikm}\mathbb{1}[y_{ij} = l]\\ \nu_k &= \alpha_k + \sum_i q(z_i = k)\\ \phi_{ikm} &= \rho_{ikm} + 1\\ \xi_{ikm} &= \log 2 - \frac{\hat{m}_{ikm}}{2}\\ \hat{\Sigma}_{km} &= (\Sigma_{km}^{-1} + \mathrm{diag}(\mathbb{E}[\omega_{ikm}]))^{-1}\\ \hat{m}_{ikm} &= \frac{1}{2}\hat{\Sigma}_{km}(\phi_{ikm}/\xi_{ikm} - \mathbb{E}[\upsilon_{ikm}])\\ c_{ikm} &= \sqrt{\hat{m}_{ikm}^2 + \hat{\Sigma}_{km}(i,i)}\\ \gamma_{ikm} &= \frac{\exp(\phi(a_i) - \frac{\hat{m}_{ikm}}{2})}{\beta_i\cosh(\frac{c_{ikm}}{2})}\\ a_i &= \sum_k\sum_m \gamma_{ikm} + 1\\ b_i &= K\end{aligned}$$

The expectations are calculated as follows

$$\begin{aligned}\mathbb{E}_q[\log \tau_k] &= \psi(\nu_k) - \psi(\sum_k \nu_k)\\ \mathbb{E}_q[\log \pi_{ikm}] &= \psi(\phi_{ikm}) - \log(\xi_{ikm})\\ \mathbb{E}_q[\log v_{jkml}] &= \psi(\mu_{jkml}) - \psi(\sum_l \mu_{jkml}).\end{aligned}$$

While updating $\hat{\Sigma}_{ikm}$ through variational inference, one has to calculate the inverse of $(\Sigma_{km}^{-1} + \mathrm{diag}(\mathbb{E}[\omega_{ikm}]))$, a $N \times N$ matrix, $K \times M$ times every inference step, which could be prohibitively slow for large-scale dataset. To address this issue, we adopt the Lanczos algorithm (Golub and Van Loan, 2013) to acquire a low-rank approximation of $(\Sigma_{km}^{-1} + \mathrm{diag}(\mathbb{E}[\omega_{ikm}]))^{-1}$ and achieve at least 10 times acceleration.

Briefly, the Lanczos algorithm factories a symmetric matrix $A \in \mathbb{R}^{n\times n}$ as $QTQ^\top$, where $T \in \mathbb{R}^{n\times n}$ is symmetric tridiagonal and $Q \in \mathbb{R}^{n\times n}$ is orthonormal by using a probe vector $\mathbf{b}$ and computes an orthogonal basis of the Krylov subspace $\mathcal{K}(A, \mathbf{b})$

$$\mathcal{K}(A, \mathbf{b}) = \mathrm{span}\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, ..., A^{n-1}\mathbf{b}\}$$

Applying Gram-Schmidt orthogonalization to these vectors produces the columns of Q, $[\mathbf{b}/\|\mathbf{b}\|, \mathbf{q}_2, \mathbf{q}_3, ..., \mathbf{q}_n]$ (here $\|\mathbf{b}\|$ is the Euclidean norm of $\mathbf{b}$). The orthogonalization coefficients are collected into T. Because A is symmetric, each vector needs only be orthogonalized against the two preceding vectors, which results in the tridigonal structure of T (Golub and Van Loan, 2013). The orthogonalized vectors and coefficients are computed in an iterative manner. k iterations produce the first k orthogonal vectors of $Q_k = [\mathbf{q}_1, ..., \mathbf{q}_k] \in \mathbb{R}^{n\times k}$ and their corresponding coefficients $T_k \in \mathbb{R}^{k\times k}$. These k iterations require only $O(k)$ matrix vector multiplies with the original matrix A.

5 EXPERIMENT

5.1 Implementation Details

Initialization. FABLE has 5 parameters, $q(z_i = k), \vec{\rho}_{ik}, \hat{\Sigma}_{km}, \hat{m}_{ikm}$, and a_i to be initialized. For $q(z_i = k)$, our initialization is similar to EBCC: we first initialize $q(z_i = k)$ by majority voting, i.e. $q(z_i = k) = \frac{1}{|L_i|}\sum_{j\in L_i} 1[y_{ij} = k]$, then multiply it with a random vector drawn from $\text{Dir}(1_M)$ to initialize $\vec{\rho}_{ik}$. We calculate the pair-wise cosine similarity of the input features to initialize $\hat{\Sigma}_{km}$. Finally, we initialize $\hat{m}_{ikm}, a_i$ with an uninformative prior Uniform$(0, 1)$.

Hyperparameter settings. FABLE has 5 hyperparameters, $\beta_{kk}, \beta_{kk'}$ for initializing $\vec{\mu}_{jkm}$, $\vec{\alpha}$ for initializing $\vec{\tau}$ and the number of subtype M. We set $\beta_{kk} = N \times M \times C$, $\beta_{kk'} = 1$, $k \neq k'$, where C is the number of correct labels that LFs gave in each subtype and class. We set $C = 1000$ to encode that we believe LFs are better than random guessing. Following EBCC, we set $\alpha_k = \sum_i q(z_i = k)^{(0)}$ where $q(z_i = k)^{(0)}$ is the MV initialization for $q(z_i = k)$ because MV can provide a reliable estimate of the class portion in the dataset. For the number of subtypes, we set $M = 3$. The key reason of we give a small number of subtypes is that subtypes are learned to capture correlation patterns and a large M increase the number of parameters, increasing the risk of overfitting.

5.2 Compared Methods

We compare our method FABLE against existing label models implemented in the WRENCH benchmark (Zhang et al., 2021b) as well as iBCC and EBCC. For all the baselines, we use the default parameter without hyperparameter tuning because we do not assume a gold validation set. We list the involved baselines as follows:

- **Majority Voting (MV).** The predicted label of each data point is the most common label given by LFs.
- **Data Programming (DP)** (Ratner et al., 2016). DP models the distribution $p(Y, Z)$ as a factor graph. It can describe the distribution in terms of pre-defined factor functions, which reflects the dependency of any subset of random variables. The log-likelihood is optimized by SGD where the gradient is estimated by Gibbs sampling, similarly to contrastive divergence (Salakhutdinov and Larochelle, 2010).
- **MeTaL** (Ratner et al., 2019). MeTal models the distribution via a Markov Network and recover the parameters via a matrix completion-style approach. The latest version of the popular Snorkel system[*] adopts MeTaL as its default label aggregation method.
- **FlyingSquid (FS)** (Fu et al., 2020). FS models the distribution as a binary Ising model, and a Triplet Method is used to recover the parameters. Notably, FS is designed for binary classification and the author suggested applying a one-versus-all reduction repeatedly to apply the core algorithm.
- **Dawid and Skene's model (DS)** (Dawid and Skene, 1979). DS models the confusion matrix of each worker regarding the ground truth labels. This method is widely used in crowdsourcing and is the recommended method for classification tasks in a benchmark on crowdsourcing (Zheng et al., 2017).
- **Independent BCC (iBCC)** (Kim and Ghahramani, 2012). iBCC models the relation between workers' annotation and the ground true label by worker independent assumption, and the relation can be solved by Gibbs sampling, mean-filed variational Bayes and expectation propagation.
- **Enhanced BCC (EBCC)** (Li et al., 2019). Based on iBCC, EBCC models the latent correlation between workers by adding subtypes that have significant potential to improve truth inference.
- **Denoise** (Ren et al., 2020). *Denoise* adopts an attention network to aggregate over weak labels, and use a neural classifier to leverage the data features. These two components are jointly trained in an end-to-end manner.
- **WeaSEL** (Rühling Cachay et al., 2021). WeaSEL shares similar model architecture as Denoise with a new objective to optimize the two components jointly.

5.3 Synthetic Dataset

In this section, we use synthetic datasets to show that leveraging instance features makes a statistical label model robust to the dataset size, and to answer the question of when is leveraging instance features helpful in improving the performance of label aggregation?

Leveraging instance features makes a statistical label model robust to the dataset size. The authors of EBCC showed that the performance of EBCC drops dramatically when the size of synthetic dataset increases, and they hypothesized that it is an optimization problem and EBCC gets stuck on bad local optima (Li et al., 2019). We argue

[*]https://github.com/snorkel-team/snorkel

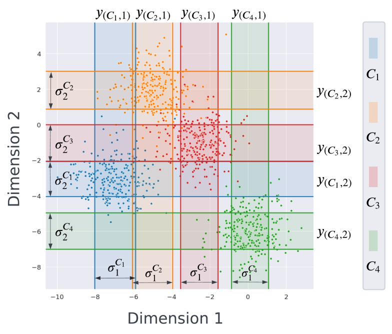

Figure 2: An example of decision space of each LF in the synthetic dataset.

that a statistical label model leveraging instance features like FABLE does might suffer less from bad local optima because it introduces a strong yet realistic inductive bias: similar data tend to have similar correlation pattern, which serves as an implicit constraint and potentially avoid bad local optima.

To confirm our hypothesis, we generated synthetic datasets consisting of 4 classes, *i.e.*, $\{C_1, C_2, C_3, C_4\}$, with different dataset sizes, *i.e.*, $\{1, 5, 10, 15, 20\} \times 10^3$, for evaluation. Data feature $\vec{x}_i$ in each class are sampled from 4 different Gaussian distributions with 2 independent features x_{i1}, x_{i2}, and $\mathcal{X}^{C_j}$ denotes the set of synthetic data belonging to the class C_j. Then, we generate 8 *unipolar* LFs, $\vec{y} = [y_{(C_1,1)}, y_{(C_1,2)}, ..., y_{(C_4,1)}, y_{(C_4,2)}]$ with 2 for each class. Specifically, a LF $y_{(C_j,k)}$ assign label C_j or abstention (-1) to an individual data point based on the k-th dimension of the data feature. For a data point x_i, the output of a LF $y_{(C_j,k)}$ is:

$$y_{(C_j,k)} = \begin{cases} C_j, \text{if } \mu_k^{C_j} - \sigma_k^{C_j} < x_{ik} < \mu_k^{C_j} + \sigma_k^{C_j} \\ -1, \text{otherwise} \end{cases},$$

where -1 means the LF abstaining from voting, and μ^{C_j} and σ^{C_j} indicate the mean and standard deviation respectively of generated data points $x \in \mathcal{X}^{C_j}$. We use subscript k to indicate the value of the k-th dimension value of μ^{C_j} or σ^{C_j}. We provide an example of the generated synthetic data and LFs as in Figure 2.

All results are reported in Figure. 3, solid lines show the averaged accuracy of 100 runs. We compare FABLE against MV, IBCC, and EBCC to show the advantages of leveraging instance features. From the results, we can see that the performance of MV is quite stable because the MV method does not involve any learnable parameter. In addition, the performance of previous BCC models (iBCC, EBCC) drop dramatically when the dataset size increases, and the EBCC decreases much slower than iBCC, which is aligned with the findings of EBCC paper. Finally, our proposed method FABLE is consistently better than MV and more importantly, it is as stable as MV, which confirms our

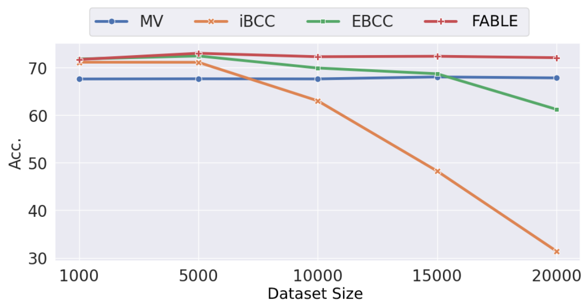

Figure 3: Performance comparison on synthetic dataset, the results are average over 100 runs on each dataset.

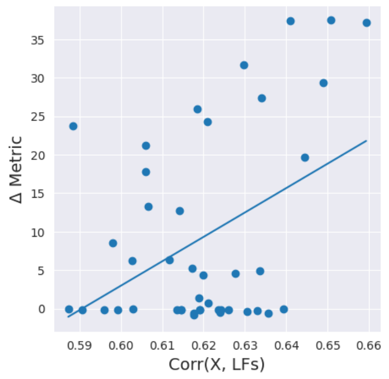

Figure 4: Relation between feature-LF correlation and performance gap between EBCC and FABLE on synthetic dataset.

hypothesis that the regulation effect of leveraging instance features could avoid bad local optima compared to other BCC models without instance features.

When are instance features useful? We are curious about when the instance features are useful and could lead to better performance of label aggregation. We hypothesize that the superiority of leveraging instance features is positively correlated to the correlation between instance features and the correctness of LFs. We formally define such a correlation as follows:

$$\text{Corr}(X, \text{LFs}) = \frac{1}{L}\sum_{j=1}^{L} \text{dCor}(X_j, R_j), \quad (14)$$

where $X_j \in \{\vec{x_i} \mid y_{ij} \neq -1, i \in [N]\}$, $R_j = \{\mathbb{1}[y_{ij} = z_i] \mid y_{ij} \neq -1, i \in [N]\}$, and $\text{dCor}(\cdot,\cdot)$ is the distance correlation. Such a correlation could reflect the level of instance features being indicative of the correlation patterns a BCC model tend to capture because the correlation pattern is indeed an instantiation of confusion matrix of true label z and LF label y and the correctness of LF, *i.e.*, $\mathbb{1}[y_{ij} = z_i]$ can be treated as a simplified version of such a confusion matrix.

To verify the above claim, we fix the dataset size to

Table 1: Dataset statistics

Dataset	IMDB	Youtube	SMS	CDR	Yelp	Commercial	Tennis	TREC	SemEval	ChemProt	AG News
Metric	F1	F1	F1	F1	F1	F1	F1	Acc	Acc	Acc	Acc
#Class	2	2	2	2	2	2	2	6	9	10	4
#LF	5	10	73	33	8	4	4	68	164	26	9
#Data	25,000	1,956	5,571	14,023	38,000	81,105	20,256	2,965	2,641	16,075	120,000

Table 2: Comparison among different methods on benchmark dataset. Red and blue indicate the best and the second-best result of each method.

Dataset	MV	DS	DP	FS	MeTaL	iBCC	EBCC	WeaSEL	Denoise	FABLE
IMDB	72.19	70.32	72.26	72.45	72.10	66.74	74.18	67.99	83.61	73.96
Youtube	80.74	81.62	74.90	78.83	76.63	73.83	86.57	0.00	83.79	88.56
SMS	32.80	43.58	32.79	30.05	32.27	48.41	48.41	0.00	87.40	48.81
CDR	63.16	53.64	53.37	64.81	43.28	9.59	23.89	0.00	61.72	62.15
Yelp	73.42	71.93	73.13	73.95	69.97	68.59	72.87	66.67	66.67	72.50
Commercial	84.23	88.31	76.43	80.86	78.61	76.83	76.43	0.00	87.66	86.62
Tennis	83.82	83.56	84.29	83.31	83.62	83.64	83.67	77.32	19.71	83.63
TREC	52.35	47.18	55.07	48.32	41.94	41.91	46.94	27.60	46.17	53.20
SemEval	78.35	73.53	73.53	11.20	72.69	73.53	73.53	30.19	67.12	74.32
ChemProt	47.96	38.82	45.71	46.25	49.76	31.84	33.80	31.84	45.54	48.35
AG News	63.85	63.95	63.56	63.63	64.15	25.00	55.94	25.00	50.49	62.74
Avg. Perf.	66.63	65.13	64.26	59.42	61.62	54.54	61.48	46.66	63.63	68.55
Avg. Rank	3.36	4.72	4.63	5.09	5.90	7.09	4.72	9.45	5.36	3.09

be $1,000$ and modify the generative process of synthetic dataset by introducing a hyperparameter ψ to generate different LFs. Specifically, each $y_{(C_j,k)}$ now annotates data according to

$$y_{(C_j,k)} = \begin{cases} C_j, \text{ if } \mu_k^{C_j} - \psi\sigma_k^{C_j} < x_{ik} < \mu_k^{C_j} + \psi\sigma_k^{C_j} \\ -1, \text{otherwise} \end{cases}.$$

To generate a synthetic dataset, we randomly sample eight $\psi \sim \text{Uniform}(1, 3)$, one for each LF. We generate 50 synthetic datasets in this way and calculate the performance gain of FABLE over EBCC as Δ Metric because we would like to see how the correlation $\text{Corr}(X, \text{LFs})$ affects the superiority of leveraging instance features (Note that the FABLE can be seen as EBCC with instance features incorporated).

The result in Figure 4 shows that the performance gap between FABLE and EBCC is positively related to the correlation $\text{Corr}(X, \text{LFs})$. The inset line is generated by linear regression over all recorded results. To further prove our hypothesis, we calculate the Pearson's correlation coefficient r between $\text{Corr}(X, \text{LFs})$ and Δ Metric. The $r = 0.469$ with p-value < 0.01, which implies that there exists a positive relationship between the performance gain of leveraging instance features and the correlation $\text{Corr}(X, \text{LFs})$ with high confidence. This finding provides practitioners with some insights of when to incorporate instance features in a statistical model: that is, if the correlation between LFs and true label (in other words, the confusion matrix) is highly dependent on the instance feature, then it is beneficial to incorporate instance features as FABLE does. However, the labeling function will introduce more noise when the center of the randomly generated clusters is close, which is the reason of why the performance gap between EBCC and FABLE sometimes becomes negligible. The performance of EBCC and FABLE will be significantly affected by increasing noise, leading to equivalent performance. Moreover, because LFs can still cover most of the data in the associated class, the distance correlation between the LFs and X in this situation is still high.

Run-time Comparison We analyze the iteration speed with and without the Lanczos algorithm on the synthetic dataset with 10,000 data points produced from two Gaussian distributions to better understand the acceleration after adopting the Lanczos approach. We used the same desired rank of 50 across all benchmark datasets. Without the Lanczos, the run-time is 52.80s/iter, while with the Lanczos, it is 4.18s/iter (12.63 times faster).

5.4 Benchmark Datasets

We conduct experiments on eleven classification datasets across diverse domains (*e.g.*, income/sentiment/span/relation/question/topic classification tasks) from the WRENCH benchmark (Zhang et al., 2021b). The WRENCH benchmark splits each dataset into training/validation/test sets, while we follow Li et al. (2019) to adopt a transductive setting, *i.e.*, we perform model learning and evaluation on the whole dataset without any ground truth label. Details of the benchmark dataset we used are listed in Table. 1. To avoid the impact of the majority class in the imbalanced dataset to the metrics, we use F1-binary metric for binary classification datasets, and accuracy metric for multi-classification datasets. In the case of text dataset, we use RoBERTa (Liu et al., 2019) to extract features following Zhang et al. (2021b), while for other datasets, we use the original features coupled with each dataset.

Main Results. We report the performance comparison over 11 datasets as well as each method's averaged score of evaluation metrics and averaged ranking over datasets in Table 2. FABLE achieves the highest average performance and ranking, outperforming all the baselines. On most datasets, FABLE is either the best or second-best method. And the majority voting is the second-best method in average due to its simplicity and the heterogeneity of the WRENCH benchmark (Zhang et al., 2021b). We identified two failure cases of EBCC, namely, CDR and ChemProt, where EBCC performs much lower than FABLE. After examining the learned parameters of EBCC on CDR, we found that it learns a highly skewed distribution of η_{km}, which is used for generating the mixture coefficient for EBCC. In particular, one of the values of η_{km} is much higher than others, which indicates that EBCC failed to capture the multiple distinct correlation patterns between the LFs and true label. It may be caused by the fact that in CDR, two of the LFs have much higher coverage (portion of non-abstention votes) than the others and such high-coverage LFs might have more complex and instant-dependent correlation patterns than low-coverage ones as they cover more diverse instants, but as an instance-independent model, EBCC may not be able to identify the multi-modal instant-dependent correlation patterns. In contrast, FABLE parameterizes the mixture coefficient using instance features, enabling it to capture the complex instant-dependent patterns. And for the case of ChemProt, we had a similar observation. Additionally, We compare FABLE with two neural network-based methods: WeaSEL and Denoise, which also leverage instance features. The results show that neural network-based methods under-perform FABLE in most cases. The key reason could be that although they use sophisticated neural networks to incorporate instance feature and introduce extra parameters, they highly rely on a gold validation set for hyperparameter tuning and early stopping, which is unavailable in our setup as we do not assume any gold labeled data.

6 CONCLUSION

In this work, we developed a statistical label model for label aggregation in Programmatic Weak Supervision with the goal of leveraging instance features in statistical modeling. Built upon a recent mixture model called EBCC, our model, FABLE (Feature-Aware laBeL modEl), achieves this goal by introducing a predictive Gaussian process to output the mixture coefficient based on instance features. The efficacy of FABLE is demonstrated in extensive experiments on synthetic datasets. We also showed that the performance gain of FABLE over EBCC is positively related to the level of instance features being indicative of correlation patterns between the LFs' votes and the true label. We compared FABLE with 7 baselines on 11 benchmark datasets from various domains, and FABLE achieves the best-averaged performance.

Social impact. All the dataset we used are publicly available and does not involve any human object. We do not foresee any negative social impact of our work. Our study aims to advance the field of Programmatic Weak Supervision, which can reduce human efforts in collecting training labels when developing ML models. Thus, we believe that our work has a positive social impact by making the development of ML model easier.

Limitations. Our proposed method involves a predictive Gaussian process which could be inefficient to scale up for large datasets compared with a simple method like majority voting. However, the rich literature on accelerating the Gaussian process could inspire future improvement in our method regarding scalability.

References

Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282. PMLR, 2017.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. Dependency structure misspecification in multi-source weak supervision models. *arXiv preprint arXiv:2106.10302*, 2021.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pages 3280–3291. PMLR, 2020.

Théo Galy-Fajou, Florian Wenzel, Christian Donner, and Manfred Opper. Multi-class gaussian process classification made conjugate: Efficient inference via data augmentation. In *Uncertainty in Artificial Intelligence*, pages 755–765. PMLR, 2020.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *Artificial Intelligence and Statistics*, pages 619–627. PMLR, 2012.

Yuan Li, Benjamin Rubinstein, and Trevor Cohn. Exploiting worker correlation for label aggregation in crowdsourcing. In *International conference on machine learning*, pages 3886–3895. PMLR, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29, 2016.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. Denoising multi-source weak supervision for neural text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3739–3754, 2020.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. End-to-end weak supervision. *Advances in Neural Information Processing Systems*, 34:1845–1857, 2021.

Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700. JMLR Workshop and Conference Proceedings, 2010.

Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Carl Roberts, and Frederic Sala. Universalizing weak supervision. In *International Conference on Learning Representations*, 2021.

Linxin Song, Jieyu Zhang, Tianxiang Yang, and Masayuki Goto. Adaptive ranking-based sample selection for weakly supervised class-imbalanced text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1641–1655, 2022.

Paroma Varma, Bryan D He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel Rubin, and Christopher Ré. Inferring generative model structure with static analysis. *Advances in neural information processing systems*, 30, 2017.

Peilin Yu, Tiffany Ding, and Stephen H Bach. Learning from multiple noisy partial labelers. In *International Conference on Artificial Intelligence and Statistics*, pages 11072–11095. PMLR, 2022.

Jieyu Zhang, Bohan Wang, Xiangchen Song, Yujing Wang, Yaming Yang, Jing Bai, and Alexander Ratner. Creating training sets via weak indirect supervision. In *International Conference on Learning Representations*, 2021a.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. Wrench: A comprehensive benchmark for weak supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*, 2022a.

Jieyu Zhang, Yujing Wang, Yaming Yang, Yang Luo, and Alexander Ratner. Binary classification with positive labeling sources. In *CIKM*, 2022b.

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.

Leveraging Instance Features for Label Aggregation in Programmatic Weak Supervision: Supplementary Materials

A Inference of EBCC

According to the generative process and graphical model of EBCC, we have:

$$\begin{aligned} p(\pi, V, \tau, Z, G, Y \mid a_\pi, \alpha, \beta) &= p(\pi \mid a_\pi) p(V \mid \beta) \cdot p(\tau \mid \alpha) p(Z \mid \tau) p(G \mid \pi, Z) p(Y \mid Z, G, V) \\ &\propto \prod_{k=1}^{K} \prod_{m=1}^{M} \pi_{km}^{a_\pi - 1} \cdot \prod_{j=1}^{L} \prod_{k=1}^{K} \prod_{m=1}^{M} \prod_{l=1}^{K} v_{jkml}^{\beta_{kl}-1} \cdot \prod_{k=1}^{K} \tau_k^{\alpha_k - 1} \cdot \prod_{i=1}^{N} \tau_{z_i} \cdot \prod_{i=1}^{N} \pi_{z_i g_i} \cdot \prod_{i=1}^{N} \prod_{j \in \mathcal{W}_i} v_{j z_i g_i y_{ij}}. \end{aligned}$$

To find the most likely Z given the worker labels Y and all hyperparameters, EBCC use fully Bayesian inference algorithm, and adopt a mean-field variational approach to find a distribution q that approximates $p(\pi, V, \tau, Z, G, Y \mid a_\pi, \alpha, \beta)$:

$$\begin{aligned} \operatorname{argmax}_Z p(Z \mid Y, a_\pi, \alpha, \beta) &= \operatorname{argmax}_Z \sum_G \int p(Z \mid Y, a_\pi, \alpha, \beta) d\tau d\pi dV \\ &\approx \operatorname{argmax}_Z \sum_G \int q(\tau, Z, G, \pi, V) d\tau d\pi dV \\ &= \operatorname{argmax}_Z q(Z). \end{aligned}$$

And q can be factorized as

$$\begin{aligned} q(\tau, Z, G, \pi, V) &= q(\tau) q(Z, G) q(\pi) q(V) \\ &= \mathrm{Dir}(\tau \mid \nu) \cdot \prod_{i=1}^{N} q(z_i, g_i) \cdot \prod_{k=1}^{K} \mathrm{Dir}(\pi_k \mid \eta_k) \cdot \prod_{k=1}^{K} \prod_{m=1}^{M} \prod_{j=1}^{L} \mathrm{Dir}(v_{kmj} \mid \mu_{kmj}). \end{aligned}$$

A.1 Update rule for $q(Z, G)$ and $q(Z)$

To get the optimal solution for $q(Z, G)$, we need to find the exponential family form of its posterior. According to the PGM, we have

$$\begin{aligned}\log q(z_i = k, g_i = m) &= \mathbb{E}_{\tau,\pi,V}\left[p(z_i \mid \tau) \cdot p(g_i \mid \pi_k, z_i) \cdot p(y_{ij} \mid z_i, g_i, v_{jkm})\right] \\ &\propto \mathbb{E}_{\tau,\pi,V}\left[\log(\tau_k) + \log(\pi_{km}) + \log(v_{jkm})\right],\end{aligned}$$

where the natural parameter $\eta = \mathbb{E}_{\tau,\pi,V}\left[\log(\tau_k) + \log(\pi_{km}) + \log(v_{jkm})\right]$. According to the inverse parameter mapping for categorical distribution with base measure $h(x) = 1$, we have

$$\begin{aligned}\rho_{ikm} &= e^{\eta} \\ &= \exp\Big\{\mathbb{E}_{\tau}[\log \tau_k] + \mathbb{E}_{\pi}[\log \pi_{km}] + \mathbb{E}_{V}[\log v_{jkm}]\Big\} \\ &= \exp\left\{\Psi(\nu_k) - \Psi\left(\sum_{k=1}^{K} \nu_k\right) + \Psi(\eta_{km}) - \Psi\left(\sum_{m=1}^{M} \eta_{km}\right) + \Psi(\mu_{jkml}) - \Psi\left(\sum_{l=1}^{K} \mu_{jkml}\right)\right\}.\end{aligned}$$

Note that for $\theta \sim \mathrm{Dir}(\alpha_1, \cdots, \alpha_k)$, we have

$$\mathbb{E}[\log(\theta_k) \mid \alpha] = \Psi(\alpha_k) - \Psi\left(\sum_{i=1}^{N} \alpha_i\right),$$

where $\Psi(\cdot)$ is the digamma function. Since $q(z_i = k)$ is the marginal distribution of $\rho_{ikm} = q(z_i = k, g_i = m)$, we can also easily get

$$q(z_i = k) = \sum_{m=1}^{M} q(z_i = k, g_i = m) = \sum_{m=1}^{M} \rho_{ikm}$$

A.2 Update rule for $q(T)$

To get the optimal solution for $q(T)$, we need to find the exponential family form of its posterior. According to the PGM, we have

$$\begin{aligned}\log q(\tau_k) &= \mathbb{E}_Z\left[\log\left(p(\tau_k) \cdot \prod_{i=1}^{N} p(z_i = k \mid \tau_k)\right)\right] \\ &= (\alpha_k - 1)\log \tau_k + \sum_{i=1}^{N} \mathbb{E}_Z\left[q(z_i = k)\right] \log \tau_k,\end{aligned}$$

where the natural parameter $\eta = \alpha_k - 1 + \sum_{i=1}^{N} q(z_i = k)$ and $\gamma_{ik} = q(z_i = k)$. According to the inverse parameter mapping for Dirichlet distribution with base measure $h(x) = 1$, we have

$$\begin{aligned}\nu_k &= 1 + (\alpha_k - 1) + \sum_{i=1}^{N} \mathbb{E}_Z\left[q(z_i = k)\right] \\ &= \alpha_k + \sum_{i=1}^{N} \gamma_{ik}\end{aligned}$$

A.3 Update rule for $q(\Pi)$

To get the optimal solution for $q(\Pi)$, we need to find the exponential family form of its posterior. According to the PGM, we have

$$\begin{aligned}\log q(\pi_{km}) &= \mathbb{E}_{Z,G}\left[\log\left(p(\pi_{km}) \cdot \prod_{i=1}^{N} p(g_i = m \mid \pi_{km}, z_i = k)\right)\right] \\ &= (a_{\pi} - 1)\log \pi_{km} + \sum_{i=1}^{N} \mathbb{E}_{Z,G}\left[q(z_i = k, g_i = m)\right] \log \pi_{km},\end{aligned}$$

where the natural parameter $\eta = a_\pi - 1 + \sum_{i=1}^{N} q(z_i = k, g_i = m)$ and $\rho_{ikm} = q(z_i = k, g_i = m)$. According to the inverse parameter mapping for Dirichlet distribution with base measure $h(x) = 1$, we have

$$\begin{aligned}\eta_{km} &= 1 + (a_\pi - 1) + \sum_{i=1}^{N} \mathbb{E}_{Z,G}\left[q(z_i = k, g_i = m)\right] \\ &= a_\pi + \sum_{i=1}^{N} \rho_{ikm}\end{aligned}$$

A.4 Update rule for $q(V)$

To get the optimal solution for $q(V)$, we need to find the exponential family form of its posterior. According to the PGM, we have

$$\begin{aligned}\log q(v_{jkml}) &= \mathbb{E}_{Z,G}\left[\log\left(p(v_{jkml}) \cdot \prod_{i\in\mathcal{N}_j} p(y_{ij} \mid v_{jkml}, g_i = m, z_i = k)\right)\right] \\ &= \mathbb{E}_{Z,G}\left[(\beta_{kl} - 1)\log v_{jkml} + \sum_{i\in\mathcal{N}_j} q(g_i = m, z_i = k)\cdot\delta(y_{ij}, 1)\log v_{jkml}\right]\end{aligned}$$

where the natural parameter $\eta = (\beta_{kl} - 1) + \sum_{i\in\mathcal{N}_j} q(g_i = m, z_i = k)\cdot\delta(y_{ij}, 1)$ and $\rho_{ikm} = q(z_i = k, g_i = m)$. According to the inverse parameter mapping for Dirichlet distribution with base measure $h(x) = 1$, we have

$$\begin{aligned}\mu_{jkml} &= 1 + (\beta_{kl} - 1) + \sum_{i\in\mathcal{N}_j} q(g_i = m, z_i = k)\cdot\delta(y_{ij}, 1) \\ &= \beta_{kl} + \sum_{i\in\mathcal{N}_j} \rho_{ikm}\cdot\delta(y_{ij}, 1)\end{aligned}$$

B Inference of FABLE

We integrate Gaussian process into the EBCC as FABLE with 3 auxiliary variables: $\lambda, \Upsilon, \Omega$. According to the PGM in Fig. 1, we can decompose the prior as

$$\begin{aligned}p(\boldsymbol{\lambda}, \Omega, \Upsilon, F, \Pi, V, G, Z, T, Y) = {}& p(\boldsymbol{\lambda})p(\Upsilon \mid \boldsymbol{\lambda})p(\Omega \mid \Upsilon)p(F)p(\Pi \mid \Upsilon, \Omega, F)\cdot \\ & p(V)p(\tau)p(Z \mid \tau)p(G \mid \Pi, Z)p(Y \mid Z, G, V)\end{aligned} \tag{15}$$

According to Eq. 15, which is based on EBCC, we only need to change the inference process of the variational distribution of mixture coefficient $q(\pi)$ with three more auxiliary variables. The new variational prior q can be factorized as

$$q(\boldsymbol{\lambda}, \Omega, \Upsilon, F, \Pi, V, G, Z, T) = q(\boldsymbol{\lambda})q(\Omega, \Upsilon)q(F)q(\Pi)q(V)q(Z, G)q(T), \tag{16}$$

where the update rule of $q(V), q(T)$ are identical to EBCC. For $q(Z, G)$, we only need to change the $\mathbb{E}_\pi[\log \pi_{km}]$ as $\mathbb{E}_\pi[\log \pi_{ikm}]$, where

$$\mathbb{E}_\pi[\log \pi_{ikm}] = \psi(\phi_{ikm}) - \log \xi_{ikm}. \tag{17}$$

That is because $q(\pi_{ikm}) \sim \mathrm{Ga}(\phi_{ikm}, \xi_{ikm})$, which is detailed in B.3.

B.1 Update rule for $q(\Omega, \Upsilon)$

To get the optimal solution for $q(\Omega, \Upsilon)$, we need to find the exponential family form of its posterior. According to the PGM of FABLE, we have

$$\begin{aligned}
\log q(\Omega,\Upsilon) &= \mathbb{E}_{\Pi,F,\boldsymbol{\lambda}}\left[\log \prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{m=1}^{M} p(\pi_{ikm}\mid f_{ikm},\upsilon_{ikm},\omega_{ikm})p(\omega_{ikm}\mid \upsilon_{ikm})p(\upsilon_{ikm}\mid\lambda_i)\right]\\
&= \mathbb{E}_{\Pi,F,\boldsymbol{\lambda}}\Bigg[\log \prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{m=1}^{M} 2^{-(\pi_{ikm}+\upsilon_{ikm})}\exp\left\{\frac{(\pi_{ikm}-\upsilon_{ikm})\sum_{i=1}^{N} f_{ikm}}{2}-\frac{(\sum_{i=1}^{N} f_{ikm})^2}{2}\omega_{ikm}\right\}\cdot\\
&\qquad \mathrm{PG}(\omega_{ikm}\mid \upsilon_{ikm},0)\frac{\lambda_i^{\upsilon_{ikm}}\exp(-\lambda_i)}{\upsilon_{ikm}!}\Bigg]\\
&= \mathbb{E}_{\Pi,F,\boldsymbol{\lambda}}\Bigg[\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M}\Bigg\{-(\pi_{ikm}+\upsilon_{ikm})\log 2+\frac{(\pi_{ikm}-\upsilon_{ikm})\sum_{i=1}^{N} f_{ikm}}{2}-\frac{(\sum_{i=1}^{N} f_{ikm})^2}{2}\omega_{ikm}\\
&\qquad +\log \mathrm{PG}(\omega_{ikm}\mid \upsilon_{nk},0)+\upsilon_{ikm}[\psi(\alpha_i)-\psi(\beta_i)]-\log \upsilon_{ikm}!\Bigg\}\Bigg]\\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M}\Bigg\{-(\mathbb{E}_{\Pi}[\pi_{ikm}]+\upsilon_{ikm})\log 2-\upsilon_{ikm}\frac{\mathbb{E}_F[\sum_{i=1}^{N} f_{ikm}]}{2}-\omega_{ikm}\frac{\mathbb{E}_F[(\sum_{i=1}^{N} f_{ikm})^2]}{2}\\
&\qquad +\log \mathrm{PG}(\omega_{ikm}\mid \upsilon_{ikm},0)+\upsilon_{ikm}[\psi(\alpha_i)-\log\beta_i]-\log \upsilon_{ikm}!\Bigg\}
\end{aligned}$$

which implies that $q(\omega_{ikm}, \upsilon_{ikm})$ follows the distribution given below:

$$\begin{aligned}
q(\omega_{ikm},\upsilon_{ikm}) &\propto \left(\exp(-\frac{\widehat{m_{ikm}}}{2})\right)^{\upsilon_{ikm}}\exp\left(-\frac{(\bar{f}_{ikm})^2}{2}\omega_{ikm}\right)\mathrm{PG}(\omega_{ikm}\mid\upsilon_{ikm},0)\left(\frac{\exp(\psi(\alpha_i))}{\beta_i}\right)^{\upsilon_{ikm}}\frac{1}{\upsilon_{ikm}!}\\
&\propto \left\{\exp\left(-\frac{(\bar{f}_{ikm})^2}{2}\omega_{ikm}\right)\mathrm{PG}(\omega_{ikm}\mid\upsilon_{ikm},0)\cosh^{\upsilon_{ikm}}\left(-\frac{(\bar{f}_{ikm})^2}{2}\right)\right\}\\
&\propto \mathrm{PG}(\omega_{ikm}\mid\upsilon_{ikm},c_{ikm})\,\mathrm{Po}(\upsilon_{ikm}\mid\gamma_{ikm}),
\end{aligned}$$

where

$$c_{ikm} = \bar{f}_{ikm} = \sqrt{\widehat{m_{ikm}}^2+\widehat{\Sigma_k}(i,i)} \tag{18}$$

$$\gamma_{ikm} = \frac{\exp(\psi(a_i))\exp(-\frac{\widehat{m_{ikm}}}{2})}{\beta_i\cosh(\frac{\bar{f}_{ikm}}{2})} \tag{19}$$

B.2 Update rule for $q(\lambda)$

$$\begin{aligned}
\log q(\boldsymbol{\lambda}) &= \mathbb{E}_{\Upsilon}[\log p(\boldsymbol{\upsilon}\mid\lambda)p(\boldsymbol{\lambda})]\\
&= \sum_{i=1}^{N}\left[\sum_{k=1}^{K}\sum_{m=1}^{M}\gamma_{ikm}\log\lambda_i-K\lambda_i\right]
\end{aligned}$$

where computation of γ_{ikm} is given by Eq. 19. This implies that $q(\lambda_i) \sim \mathrm{Ga}(\lambda_i \mid a_i, b_i)$ where

$$a_i = \sum_{k=1}^{K}\sum_{m=1}^{M}\gamma_{ikm}+1 \tag{20}$$

$$b_i = K \tag{21}$$

B.3 New update rule for $q(\Pi)$

$$\begin{aligned}
\log q(\Pi) &= \mathbb{E}_{F,\Upsilon,\Omega,Z,G}\left[\log \prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{m=1}^{M} p(\pi_{ikm} \mid f_{ikm}, \upsilon_{ikm}, \omega_{ikm}) p(g_i = m \mid \pi_{ikm}, z_i)\right] \\
&= \mathbb{E}_{F,\Upsilon,\Omega,Z,G}\Bigg[\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M} -(\pi_{ikm} + \upsilon_{ikm})\log 2 + \frac{(\pi_{ikm} - \upsilon_{ikm})f_{ikm}}{2} - \frac{(f_{ikm})^2}{2}\omega_{ikm} \\
&\quad + q(z_i = k, g_i = m)\log \pi_{ikm}\Bigg] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M} \mathbb{E}_{Z,G}[q(z_i = k, g_i = m)]\log \pi_{ikm} - (\log 2)\pi_{ikm} + \frac{\mathbb{E}_F[f_{ikm}]}{2}\pi_{ikm} + \text{Const},
\end{aligned}$$

which implies that $q(\pi_{ikm})$ follows the distribution given below:

$$\begin{aligned}
q(\pi_{ikm}) &\propto \pi_{ikm}^{\rho_{ikm}} \cdot e^{-(\log 2 - \frac{\hat{m}_{km}}{2})\pi_{ikm}} \\
&\propto \text{Ga}(\pi_{ikm} \mid \phi_{ikm}, \xi_{ikm}),
\end{aligned}$$

where

$$\begin{aligned}
\phi_{ikm} &= \rho_{ikm} + 1 \\
\xi_{ikm} &= \log 2 - \frac{\hat{m}_{ikm}}{2}.
\end{aligned}$$

B.4 Update rule for $q(F)$

$$\begin{aligned}
\log q(F) &= \mathbb{E}_{\Pi,\Upsilon,\Omega}\left[\log \prod_{i=1}^{N}\prod_{k=1}^{K} p(\pi_{ikm} \mid f_{ikm}, \upsilon_{ikm}, \omega_{ikm}) p(f_{ikm})\right] \\
&= \mathbb{E}_{\Pi,\Upsilon,\Omega}\left[\log \prod_{k=1}^{K} \mathcal{N}(f_{ikm} \mid \frac{\pi_{km} - \boldsymbol{\upsilon}_{km}}{2}, \text{diag}(\boldsymbol{\omega}_{km})^{-1}) \mathcal{N}(f_{ikm} \mid \mathbf{0}, \Sigma_{km})\right]
\end{aligned}$$

which implies that $q(\boldsymbol{f}^{(k,m)}) \sim \mathcal{N}(\boldsymbol{f}^{(k,m)} \mid \hat{\boldsymbol{m}}_k, \hat{\Sigma}_k)$ where

$$\hat{\boldsymbol{m}}_{km} = \frac{1}{2}\hat{\boldsymbol{\Sigma}}_{km}(\boldsymbol{\phi}_{km}/\boldsymbol{\xi}_{km} - \mathbb{E}[\boldsymbol{\upsilon}_{km}]) \tag{22}$$

$$\hat{\boldsymbol{\Sigma}}_{km} = (\boldsymbol{\Sigma}_{km}^{-1} + \text{diag}(\mathbb{E}[\hat{\boldsymbol{\omega}}_{km}]))^{-1}. \tag{23}$$

Note that $\mathbb{E}[\boldsymbol{\upsilon}_{km}] = \boldsymbol{\gamma}_{km}, \mathbb{E}_{q(\omega_{km},\upsilon_{km})}[\omega_{km}] = \frac{\mathbb{E}[\boldsymbol{\pi}_{ikm}] + \boldsymbol{\gamma}_{km}}{2\boldsymbol{c}_{km}} \tanh \frac{\boldsymbol{c}_{km}}{2}$.