Machine Learning (2023) 112:4693–4721
https://doi.org/10.1007/s10994-023-06388-9

Ranking-preserved generative label enhancement

Yunan Lu[1] · Weiwei Li[2] · Huaxiong Li[3] · Xiuyi Jia[1]

Received: 20 January 2023 / Revised: 11 May 2023 / Accepted: 13 August 2023 /
Published online: 20 September 2023
© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2023

Abstract
Label distribution learning (LDL) is effective for addressing label ambiguity. In LDL, ground-truth label distributions are hardly available due to the high annotation cost, whereas it is relatively easy to obtain examples with logical labels. Hence, label enhancement (LE) is proposed to automatically transform logical labels into label distributions. Most existing LE methods employ discriminative approaches. However, discriminative approaches specialize in obtaining better predictive performance under supervised learning, and their capability is limited in LE that lacks supervisory information. Therefore, we propose a generative LE model, and infer label distributions by the variational Bayes capable of preserving the label ranking within the logical label vector. Our method consists of a generation process and an inference process. In the generation process, we treat label distributions as latent variables, and assume that label distributions generate logical labels and feature values of the instance itself and logical labels of the neighbors of this instance. In the inference process, we design a function, which mines the label correlation and preserves the label ranking within the logical label vector, to parameterize the variational posterior. Finally, we conduct extensive experiments to validate our proposal.

Keywords Label enhancement · Label distribution learning · Learning with ambiguity · Generative model

Editor: Eyke Hüllermeier.

✉ Xiuyi Jia
jiaxy@njust.edu.cn

Yunan Lu
luyn@njust.edu.cn

Weiwei Li
liweiwei@nuaa.edu.cn

Huaxiong Li
huaxiongli@nju.edu.cn

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

[3] Department of Control Science and Intelligence Engineering, Nanjing University, Nanjing 210093, China

1 Introduction

Label ambiguity is ubiquitous in real-world tasks. Multi-label learning (Tsoumakas & Katakis, 2007; Zhang & Zhou, 2014) is an effective learning paradigm that deals with label ambiguity by assigning multiple labels to an instance. However, it can only answer the question "*which labels can describe the instance?*", but not "*how much does each label describe the instance?*". Hence, a learning paradigm called label distribution learning (LDL) is proposed to solve this problem (Geng, 2016). In LDL, an instance is annotated by a real-valued vector with probabilistic form. This real-valued vector is also called label distribution. LDL has been applied in many fields, such as age estimation (Gao et al., 2018; Hou et al., 2017; Wen et al., 2020; Geng et al., 2013; Deng et al., 2020; Chen et al., 2016), head pose estimation (Geng & Xia, 2014), crowd counting (Ling & Geng, 2019; Zhang et al., 2015), visual sentiment distribution prediction (Yang et al., 2017), emotion analysis from texts (Zhou et al., 2016), and so on.

A critical bottleneck limiting LDL is the availability of the ground-truth label distributions. Most existing datasets only contain the logical labels, and manually annotating instances with label distributions is costly. To alleviate these problems, a process called label enhancement (LE) (Xu et al., 2018) is proposed to transform logical labels into label distributions automatically.

Generally speaking, LE studies how to recover the label distributions with the help of the features and logical labels of the instances. This problem can be addressed in discriminative or generative approaches. Most existing studies employ the discriminative approach. They directly treat the latent label distribution values as conditional probabilities of logical labels conditioned on features, and then exploit some additional information to "guess" (due to the lack of ground-truth label distributions) the conditional probabilities. Typically, they leverage the information from two main aspects. One is the neighborhood relation of instances (relation between the instance and its neighbors), such as the low-rank representation (Tang et al., 2020), feature manifold reconstruction (Hou et al., 2016), and K nearest neighbors (Xu et al., 2018). The other is the co-occurrence relation of labels, such as the local and global correlations between labels (Jia et al., 2023). The discriminative approach specializes in obtaining better predictive performance under supervised learning (Jebara, 2012), while the label enhancement task lacks the ground-truth label distribution as supervisory information, which limits the capability of the discriminative model. Compared with the discriminative approach, generative approach is concerned with describing the generation process of observations in a principled way and mining the underlying patterns of observations, which is more essential for unsupervised learning (Jebara, 2012), and thus is more suitable for label enhancement (an unsupervised task). Variational label enhancement (LEVI) (Xu et al., 2020) for the first time demonstrates the effectiveness of the generative approach for the LE task. It treats the label confidence[1] as the latent representation of the feature vector and the logical label vector, and uses the auto-encoding variational Bayes (AEVB) (Kingma & Welling, 2014) framework to learn the representation.

However, as is illustrated in Fig. 1, due to the assumption that the logical labels are generated by a Bernoulli distribution $\mathrm{Ber}(\boldsymbol{y}|\mathrm{MLP}(\boldsymbol{z}))$, where MLP is a multilayer perceptron, LEVI fails to preserve the label ranking within the logical label vector, while the

[1] The LE methods usually normalize a real-valued vector, whose elements take values between 0 and 1 and may not sum to 1, to the form of label distribution. We call this unnormalized real-valued vector the label confidence.

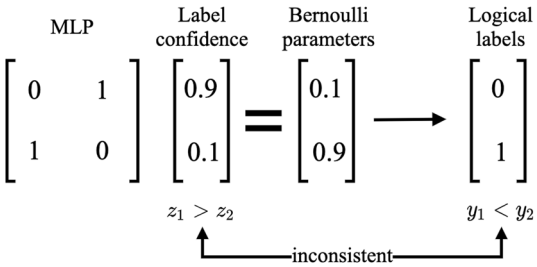


Fig. 1 An example of ranking inconsistency between the label confidence and the logical labels caused by MLP. Typically, we expect that the label confidence of $[0.9, 0.1]^\top$ tends to generate logical labels of $[1, 0]^\top$, whereas after an MLP transformation, the label confidence tends to generate logical labels of $[0, 1]^\top$

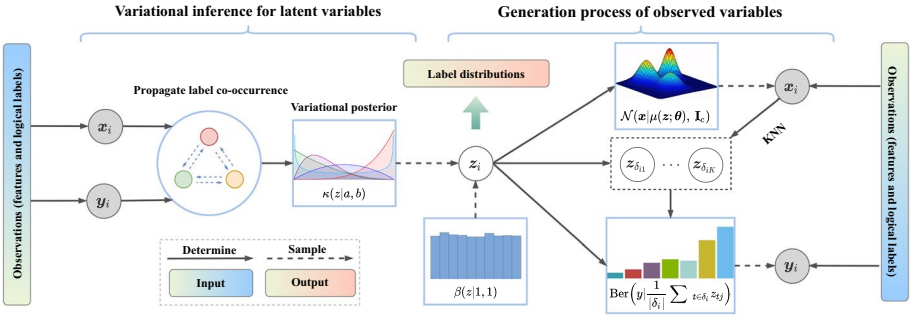


Fig. 2 Overview of our method. The input is the observation of feature variables $\boldsymbol{x}$ and logical labels $\boldsymbol{y}$. The output is the label distribution calculated by the posterior of label confidence. The prior and variational posterior of label confidence z are Beta distribution $\text{Beta}(z|1,1)$ and Kumaraswamy distribution $\kappa(z|a,b)$, respectively. During the variational inference, a and b are modeled by a function of features and logical labels, which encodes the label co-occurrence relation by message passing. During the generation process, a function of label confidence $\mu(\boldsymbol{z}_i;\boldsymbol{\theta})$ and an identity matrix $\mathbf{I}_c$ are treated as the parameters of the Gaussian for generating feature vectors. Based on the generated feature vectors, we can obtain the neighbors of the instance $\boldsymbol{x}_i$ (including $\boldsymbol{x}_i$ itself), i.e., $|\delta_i|$, and use their label confidence to generate the logical labels for $\boldsymbol{x}_i$

correct label ranking is a prerequisite for label distribution to induce optimal decisions in subsequent tasks (Jia et al., 2023; Wang & Geng, 2021; Wang et al., 2022; Wang & Geng, 2019, 2021). Besides, LEVI ignores the neighborhood relation and the label relation, which are also essential to LE.

In this paper, we propose GLERB, a novel **G**enerative **L**abel **E**nhancement model, in which the latent label confidence is inferred by a **R**anking-preserved variational **B**ayes. As shown in Fig. 2, our method consists of a generation process and an inference process.

Generation process. The generation process describes how the observed variables (features and logical labels) are generated. First, the label confidence $\boldsymbol{z}$ is generated from the Beta prior. Next, we take the linear transformation of the label confidence as the mean of a Gaussian distribution and use this Gaussian to generate the feature values of the instance, which follows a similar assumption to LEVI that label confidence is the low-dimensional representation of the feature vector. Besides, inspired by the assumption that the labels of neighboring instances are sampled from a common distribution (Liu et al., 2019; Weerasooriya et al., 2020), we make the label confidence of instance $\boldsymbol{x}$ to generate not only the logical labels of itself but also those of its neighbors, i.e., the logical labels of an instance is generated jointly by its own label confidence and that of

its neighbors. Accordingly, we mix the label confidence of the neighbors of an instance to model the parameters of a Bernoulli distribution, and use this Bernoulli to generate the logical labels of this instance.

Inference process. To infer the label confidence, we modify the traditional AEVB framework by imposing the restriction about label co-occurrence and label ranking on it. Specifically, we model the varitional family as Kumaraswamy distribution (Kumaraswamy, 1980). The reasons for this choice are twofold: (1) Kumaraswamy, which has closed-form inverse cumulative density function, make the reparameterization easy; (2) Kumaraswamy's Kullback–Leibler divergence from the Beta prior can be obtained in closed-form. These two aspects, i.e., reparameterization and closed-form KL divergence from the prior, are necessary for AEVB. Other candidate probability distributions with a support of (0, 1) either cannot apply the reparameterization (e.g., Beta distribution) or have no closed-form KL divergence from the prior (e.g., logistic-normal distribution).[2] Then we design a function based on label co-occurrence to model the parameters of the Kumaraswamy, rather than a multilayer perceptron. This function, which we call the label propagation function, is derived from the message propagation on the graph whose nodes and edges represent labels and label co-occurrence probabilities, respectively. Furthermore, we conceptually and theoretically illustrate the reason why label propagation function can preserve the ranking consistency between label confidence and logical labels.

Based on the above generation and inference processes, we derive the Monte Carlo approximate of the ELBO (evidence lower bound), and derive the lower bound of ELBO to avoid the troublesome gradients caused by the log of summation expression. Finally, we compare our method with several state-of-the-art algorithms on 14 real-world datasets. The experimental results validate the effectiveness of our proposal. The main contributions of our work are summarized as follows:

- We propose a novel generative model for label enhancement task, which encodes the relationships among logical labels, features, label confidence, and neighbors in a principled way.
- We adapt the traditional auto-encoding variational Bayes to our generative label enhancement model, which makes the label confidence incorporated with the label co-occurrence information and theoretically guarantees the label ranking consistency between the label confidence and logical label vector.
- We derive the Monte Carlo approximate of the ELBO for our generative label enhancement model, which allows any gradient-based optimization algorithm to be applied; we also derive the lower bound of ELBO, which avoids the troublesome gradients caused by the log of summation.

2 Related work

The current works on LE can be roughly divided into two groups: one is the discriminative LE methods, and the other is the generative LE methods.

[2] In Appendix, we discuss the demerits of more distributions w.r.t. modelling the variational posterior for the variable with Beta prior.

The discriminative LE methods aim to learn the mapping from features to labels directly. Typically, they study how to enhance the labels using the information from three main aspects: the instance relation, the label relation, and the label embedding. To leverage the instance relation, some LE algorithms, e.g., ML^2 (Hou et al., 2016), L^2 (Liu et al., 2021), WSMLLE (Lv et al., 2019), LEMLL (Shao et al., 2018), LESC (Tang et al., 2020), SLE-SSE (Wen et al., 2022), MLFE (Zhang et al., 2018), and gLESC (Zheng et al., 2023), reconstruct the feature vector of each instance by the linear combination of the feature vector of its neighbors, and assume that the label distribution for each instance can also be reconstructed by the same linear combination. More specifically, ML^2, LEMLL, and L^2 only require that the coefficients of the linear combination sum to one; SLE-SSE, LSEC, and gLESC assume that the coefficient matrix of linear combinations of all instances should be of low-rank; MLFE and WSMLLE assume the coefficient vectors of the linear combination to be sparse. There are also some algorithms, e.g., GLLE (Xu et al., 2018), PLLE (Xu et al., 2019), BDLE (Liu et al., 2021), RELIAB (Li et al., 2015), that achieve the consistency of instance relations between feature space and label distribution space leveraging the Laplacian regularization of instance relation graph. Besides, PLEML (Zhu et al., 2020) captures the complex structural relation of instance through a multi-label learning model, which is an implicit way to mine the information of instance relation, and it avoids the noise introduced by manual graph construction; To leverage the label relation, WSMLLE (Lv et al., 2019) and the extend GLLE (Xu et al., 2021) regularize the recovered label distribution by the Graph Laplacian of label correlation graph; LEPNLR (Jia et al., 2023) enhances the labels by maintaining the rank relation of logical labels. The discriminative LE methods based on label embedding recovers the label distribution by mining the vector representation of label semantics. For example, FCMLE (Gayar et al., 2006), KMLE (Jiang et al., 2006), and FLE (Wang et al., 2023) find the representative point in the feature space for each label, and then calculate the similarity between the feature vector of the instance and the representative point of the label, which in turn is used to construct the label distribution; LCM (Guo et al., 2021), LEBS (Liu et al., 2021), and EWLLE (Zeng et al., 2021) embed the feature vector and logical label vector of an instance into a common space, and calculate the similarity between the embeddings of features and labels in the common space, which in turn is used to construct the label distribution.

The above LE methods all use discriminative approach. Discriminative approach specializes at obtaining better predictive performance under supervised task, and this predictive performance relys on accurate supervisory information. However, since LE is unsupervised, the above methods all use logical labels as inaccurate supervisory information, which introduces additional error. Compared with the discriminative approach, generative approach is good at describing the generation process of observations in a principled way and mining the underlying patterns of observations, which is more essential for LE task. To the best of our knowledge, LEVI (Xu et al., 2020) is currently the only generative LE method. It treats the label confidence as the latent variable which generate features and logical labels, and uses the variational approach to infer the posterior of label confidence. Although LEVI is a successful application of generative approach to LE task, it fails to guarantee the label ranking within the logical label vector.

3 Preliminary

In this section, we first introduce some notations and formulate LDL and LE problems. Then, we introduce the background of variational inference and AEVB framework. Finally, we introduce the Kumaraswamy distribution.

3.1 Notations and problem formulation

We denote the matrix of observed feature variables by $\mathbf{X} \in \mathbb{R}^{n\times c}$, where n and c denote the number of observations and features, respectively; the i-th feature vector is denoted by $\boldsymbol{x}_i$. Let $\mathbf{Y} \in \{0,1\}^{n\times m}$ denote the matrix of observed logical label variables, where m is the number of labels; the i-th logical label vector is denoted by $\boldsymbol{y}_i$; the j-th logical value of $\boldsymbol{y}_i$ is denoted by y_{ij}. The matrix of latent label confidence is denoted by $\mathbf{Z} \in \mathbb{R}_+^{n\times m}$; the i-th label confidence vector is denoted by $\boldsymbol{z}_i$; the j-th label confidence of $\boldsymbol{z}_i$ is denoted by z_{ij}. The matrix of latent label distributions is denoted by $\mathbf{D} \in \mathbb{R}_+^{n\times m}$; the i-th label distribution is $\boldsymbol{d}_i$; the j-th element of $\boldsymbol{d}_i$ is d_{ij}, which satisfies $d_{ij} > 0$ and $\sum_{j=1}^{m} d_{ij} = 1$. Actually, d_{ij} is equal to the normalized label confidence $(\sum_{t=1}^{m} z_{it})^{-1} z_{ij}$.

The goal of label distribution learning is to learn a mapping from the feature vector to the label distribution according to the dataset $(\mathbf{X}, \mathbf{D})$ in which each instance $\boldsymbol{x}_i$ is associated with a label distribution $\boldsymbol{d}_i$; the learned mapping can in turn be used to predict label distributions for unseen instances.

Since the ground-truth label distributions are usually unavailable due to the expensive annotation cost, the goal of label enhancement (i.e., the goal of this paper) is to recover the label confidence $\mathbf{Z}$ (or label distribution $\mathbf{D}$) from the logical labels $\mathbf{Y}$, and thus transforming the dataset $(\mathbf{X}, \mathbf{Y})$ into the LDL-applicable dataset $(\mathbf{X}, \mathbf{D})$.

3.2 Variational inference and AEVB framework

Here, we introduce the background of variational inference and AEVB framework.

Given the observed variables $\boldsymbol{o}$ and the corresponding latent variables $\boldsymbol{z}$, we are interested in inferring the posterior $p(\boldsymbol{z}|\boldsymbol{o})$. In practice, the exact inference of $p(\boldsymbol{z}|\boldsymbol{o})$ is typically intractable due to the computationally intractable integral. Therefore, variational inference is introduced to provide an approximate inference of $p(\boldsymbol{z}|\boldsymbol{o})$. Variational inference tries to find a simple distribution $q(\boldsymbol{z}|\boldsymbol{o})$ to approximate the intractable posterior $p(\boldsymbol{z}|\boldsymbol{o})$:

$$q^{\star}(\boldsymbol{z}|\boldsymbol{o}) = \underset{q(\boldsymbol{z}|\boldsymbol{o})\in Q}{\arg\min}\ D_{KL}(q(\boldsymbol{z}|\boldsymbol{o})\|p(\boldsymbol{z}|\boldsymbol{o})), \tag{1}$$

where D_{KL} denotes the Kullback–Leibler divergence, Q denotes a family of variational posterior. Since the posterior $p(\boldsymbol{z}|\boldsymbol{o})$ is intractable, and the following equation holds

$$\log p(\boldsymbol{o}) = \int q(\boldsymbol{z}|\boldsymbol{o}) \log \frac{p(\boldsymbol{z},\boldsymbol{o})}{q(\boldsymbol{z}|\boldsymbol{o})} \mathrm{d}\boldsymbol{z} + D_{KL}(q(\boldsymbol{z}|\boldsymbol{o})\|p(\boldsymbol{z}|\boldsymbol{o})). \tag{2}$$

Therefore, Eq. (1) is equivalent to:

$$q^{\star}(\boldsymbol{z}|\boldsymbol{o}) \quad = \underset{q(\boldsymbol{z}|\boldsymbol{o})\in Q}{\arg\max} \int q(\boldsymbol{z}|\boldsymbol{o}) \log \frac{p(\boldsymbol{z},\boldsymbol{o})}{q(\boldsymbol{z}|\boldsymbol{o})} \mathrm{d}\boldsymbol{z} = \underset{q(\boldsymbol{z}|\boldsymbol{o})\in Q}{\arg\max}\ \mathcal{L}_{ELBO}, \tag{3}$$

where the integral term is called evidence lower bound (ELBO), abbreviated as $\mathcal{L}_{ELBO}$. AEVB is a popular framework for solving the optimization problem in Eq. (3). First, AEVB rewrites the $\mathcal{L}_{ELBO}$ as the reconstruction term minus prior regularization term:

$$\mathcal{L}_{ELBO} = \underbrace{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{o})}\left[\log p(\boldsymbol{o}|\boldsymbol{z})\right]}_{\text{reconstruction}} - \underbrace{D_{KL}(q(\boldsymbol{z}|\boldsymbol{o})\|p(\boldsymbol{z}))}_{\text{prior regularization}}. \tag{4}$$

Second, AEVB reparameterize the latent variable $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{o})$ by a differentiable transformation f_q of observed variables $\boldsymbol{o}$ and an auxiliary noise variable ϵ:

$$\tilde{\boldsymbol{z}} = f_q(\boldsymbol{o}, \epsilon), \quad \epsilon \sim p(\epsilon). \tag{5}$$

Finally, the reconstruction term can be estimated by Monte Carlo samples, and the prior regularization term can be derived analytically:

$$\mathcal{L}_{ELBO} \approx \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{o}|f_q(\boldsymbol{o}, \epsilon^{(t)})) - D_{KL}(q(\boldsymbol{z}|\boldsymbol{o})\|p(\boldsymbol{z})), \tag{6}$$

where $\epsilon^{(t)} \sim p(\epsilon)$ and T is the number of Monte Carlo samples. Equation (6) can be maximized by the gradient-based optimization algorithms, and the posterior of the latent variable can be obtained.

3.3 Kumaraswamy distribution

The density function of the Kumaraswamy distribution with two shape parameters a and b is:

$$\kappa(v|a, b) = abv^{a-1}(1 - v^a)^{b-1}, \tag{7}$$

where $v \in (0, 1)$ is a random variable. The median of Kumaraswamy distribution is:

$$\mathbb{M}_{\kappa(v|a,b)}[v] = (1 - 2^{-\frac{1}{b}})^{\frac{1}{a}}, \text{ i.e., } a = \frac{\log(1 - 2^{-\frac{1}{b}})}{\log \mathbb{M}_{\kappa(v|a,b)}[v]}. \tag{8}$$

If a random variable $v \sim \kappa(v|a, b)$, then the reparameterization trick can be applied by

$$v = (1 - \epsilon^{\frac{1}{b}})^{\frac{1}{a}}, \tag{9}$$

where $\epsilon \sim \text{Uniform}(0, 1)$. The KL divergence of Kumaraswamy distribution from Beta distribution is given as follows:

$$\begin{aligned} D_{KL}(\kappa(v|a_\kappa, b_\kappa)\|\text{Beta}(v|a_B, b_B)) = \log(B(a_B, b_B)a_\kappa b_\kappa) - \frac{b_\kappa - 1}{b_\kappa} + \\ \frac{a_\kappa - a_B}{a_\kappa}\left(-\gamma_e - \digamma(b_\kappa) - \frac{1}{b_\kappa}\right) + (b_B - 1)b_\kappa \sum_{i=1}^{\infty} \frac{1}{i + a_\kappa b_\kappa} B\left(\frac{i}{a_\kappa}, b_\kappa\right), \end{aligned} \tag{10}$$

where γ_e is Euler's constant; $\digamma$ is the digamma function; B is the Beta function. In particular, if $a_B = 1$ and $b_B = 1$, the KL divergence of $\kappa(v|a_\kappa, b_\kappa)$ from $\text{Beta}(v|a_B, b_B)$ can be simplified as

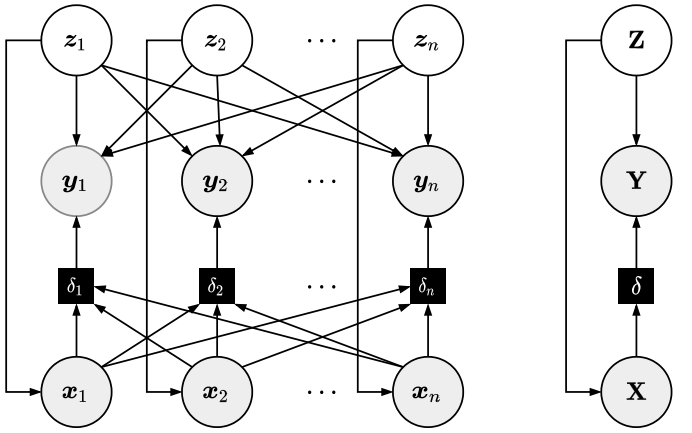


Fig. 3 Grapical representation of the generative model

$$D_{KL}(\kappa(v|a_\kappa, b_\kappa)\|\text{Beta}(v|1,1)) = \log(a_\kappa b_\kappa) + a_\kappa^{-1}\left((1-a_\kappa)F(b_\kappa) + b_\kappa^{-1} + \gamma_e\right) + \text{const.} \tag{11}$$

4 Generation process

In this section, we formally describe the generation process of the observed variables. We assume that the observations of logical label vector $\boldsymbol{y}$ and feature vector $\boldsymbol{x}$ of an instance arise from the following process whose plate notation is shown as Fig. 3:

1. Generate label confidence $\mathbf{Z}$ from a Beta prior:

$$p(\mathbf{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{m}\text{Beta}(z_{ij}|1,1). \tag{12}$$

2. Generate feature matrix $\mathbf{X}$ from a Gaussian distribution:

$$p(\mathbf{X}|\mathbf{Z}) = \prod_{i=1}^{n}\mathcal{N}(\boldsymbol{x}_i|\mu(\boldsymbol{z}_i;\boldsymbol{\theta}), \mathbf{I}_c), \tag{13}$$

where $\mu(\boldsymbol{z}_i;\boldsymbol{\theta})$ is a linear function of $\boldsymbol{z}_i$ with the learnable parameter $\boldsymbol{\theta}$.

3. Generate logical labels. We assume that the logical label vector of an instance is generated from its own label confidence and that of its neighbors:

$$p(\mathbf{Y}|\mathbf{Z},\mathbf{X}) = \prod_{i=1}^{n}\prod_{j=1}^{m}\text{Ber}\left(y_{ij}\Big|\frac{1}{|\delta_i|}\sum_{t\in\delta_i} z_{tj}\right), \tag{14}$$

where $\delta_i \triangleq \{j \mid \boldsymbol{x}_i \in K_{\mathrm{NN}}(\boldsymbol{x}_j)\}$, $K_{\mathrm{NN}}(\boldsymbol{x}_j)$ is the set of indices of the K-nearest neighbors (including itself) of instance $\boldsymbol{x}_j$, K is the neighborhood size.

According to the above generation process, the joint density of the complete-data can be factorized as:

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = p(\mathbf{Y}|\mathbf{Z}, \mathbf{X})p(\mathbf{X}|\mathbf{Z})p(\mathbf{Z}). \tag{15}$$

5 Inference of label confidence

In this section, we use the AEVB framework to infer the posterior of label confidence. Specifically, we assume the mean-field variational posterior $q(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ which factorizes as $q(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \prod_{i=1}^{n} q(\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{y}_i)$, and use a differentiable function of observations $\boldsymbol{x}_i, \boldsymbol{y}_i$ to model $q(\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{y}_i)$. Then we maximize the $\mathcal{L}_{ELBO}$ of our model to obtain the approximate posterior of label confidence:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\left[\log p(\mathbf{X}, \mathbf{Y}|\mathbf{Z})\right] - D_{KL}(q(\mathbf{Z}|\mathbf{X}, \mathbf{Y})\|p(\mathbf{Z})). \tag{16}$$

In order to estimate and maximize Eq. (16), we need to specify three elements: (1) the variational family, (2) the reconstruction term $\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\left[\log p(\mathbf{X}, \mathbf{Y}|\mathbf{Z})\right]$, and (3) the prior regularization term $D_{KL}(q(\mathbf{Z}|\mathbf{X}, \mathbf{Y})\|p(\mathbf{Z}))$. Therefore, we derive these elements in the following three subsections and give the detailed algorithm in the last subsection.

5.1 Variational family

We represent the posterior of the label confidence as the Kumaraswamy distribution whose median is given by a function of feature vector and logical label vector, i.e.,

$$\begin{aligned} q(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) &= \prod_{i=1}^{n}\prod_{j=1}^{m} \kappa(z_{ij}|a_{ij}, b_{ij}), \\ a_{ij} = \frac{\log(1 - 2^{-\frac{1}{b_{ij}}})}{\log\left(f_j(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\omega})\right)}&, b_{ij} = \exp(\psi_j(\boldsymbol{x}_i; \boldsymbol{\eta})), \end{aligned} \tag{17}$$

where f is a deterministic function on $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, $\boldsymbol{\omega}$ is the learnable parameter of f, $f_j(\cdot;\boldsymbol{\omega})$ denotes the j-th component of $f(\cdot;\boldsymbol{\omega})$, and $\psi(\cdot;\boldsymbol{\eta})$ is given by a linear model with the learnable parameter of $\boldsymbol{\eta}$. It should be noted that $f(\cdot;\boldsymbol{\omega})$ can also be used as the expectation of the Kumaraswamy distribution; however, the calculation of Kumaraswamy's expectation involves the integration, which makes it difficult to express the shape parameters as a simple function of $f(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\omega})$ by solving equation $\mathbb{E}\left[z_{ij}\right] = f_j(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\omega})$. In contrast, the median has a simple formula, so we turn to use $f(\cdot;\boldsymbol{\omega})$ as the median. Furthermore, instead of specifying $f(\cdot;\boldsymbol{\omega})$ as the MLP, we design a function which encodes the label co-occurrence. Specifically, we propagate the label co-occurrence message among all labels on a graph whose nodes and edges encode the labels and label co-occurrence probabilities, respectively.

Next, we first derive the label propagation function, then illustrate conceptually and theoretically the effectiveness of this function w.r.t. preserving the label ranking consistency between label confidence and logical labels, and finally give the variational family formally.

5.1.1 Label propagation function

First, we give the co-occurrence probability of i-th and j-th labels:

$$g_{ij} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_{ni} = 1 \text{ and } y_{nj} = 1))}{\sum_{n=1}^{N} \mathbb{I}(y_{ni} = 1 \text{ or } y_{nj} = 1)}, \tag{18}$$

where $\mathbb{I}(\cdot)$ is the indicator function; the co-occurrence probabilities of all pairs of labels form the co-occurrence matrix $\mathbf{G}$ whose (i, j) entry is g_{ij}. Then, the spectral graph convolution (Kipf & Welling, 2017) of $\mathbf{G}$ can be obtained by $\mathbf{P} = \text{diag}(\mathbf{G}\mathbf{1}_m)^{-\frac{1}{2}}\mathbf{G}\text{diag}(\mathbf{G}\mathbf{1}_m)^{-\frac{1}{2}}$, where $\mathbf{1}_m$ is an m-dimensional all-one vector. Let $\boldsymbol{v}^{(0)} = \boldsymbol{y}$ be the initial vector, the message passing on the label co-occurrence graph can be formulated as:

$$\boldsymbol{v}^{(t+1)} = \alpha\mathbf{P}\boldsymbol{v}^{(t)} + (1 - \alpha)\boldsymbol{y}, \tag{19}$$

where $\boldsymbol{v}^{(t)}$ denotes the result of the t-th iteration, $\alpha \in (0, 1)$ controls the extent to which label co-occurrence information affects the results. Running the Eq. (19) iteratively yields the result of the message passing process:

$$\boldsymbol{v}^{(t+1)} = (\alpha\mathbf{P})^{t+1}\boldsymbol{y} + (1 - \alpha)\sum_{i=0}^{t}(\alpha\mathbf{P})^{i}\boldsymbol{y}. \tag{20}$$

Since $\mathbf{P}$ can be eigendecomposed as $\mathbf{U}\boldsymbol{\Gamma}\mathbf{U}^{\top}$, where $\boldsymbol{\Gamma} = \text{diag}([\gamma_1, \cdots, \gamma_m]^{\top})$ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{P}$, $\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_m]$ is an orthogonal matrix, $\boldsymbol{u}_i$ is the (column) eigenvector corresponding to the eigenvalue γ_i. Without loss of generality, we assume that $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_m$. Since $\mathbf{P}$ is the symmetrically normalized Laplacian for an undirected graph (where each node has a self-loop), the spectral radius (i.e., the maximum of the absolute values of eigenvalues) of $\mathbf{P}$ is 1, and any eigenvalue is positive. Then the limits of the two terms in Eq. (20) become:

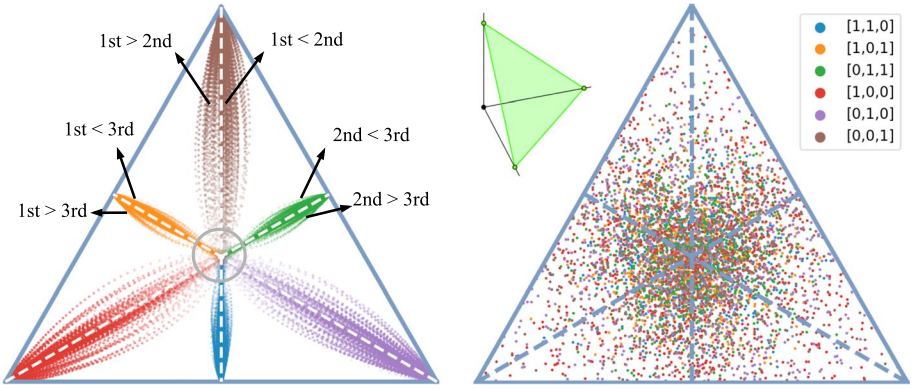
$$\lim_{t\to\infty}(\alpha\mathbf{P})^{t+1}\boldsymbol{y} = \lim_{t\to\infty}\mathbf{U}(\alpha\boldsymbol{\Gamma})^{t}\mathbf{U}^{\top}\boldsymbol{y} = \mathbf{0}, \quad \lim_{t\to\infty}\sum_{j=0}^{t}(\alpha\mathbf{P})^{j}\boldsymbol{y} = (\mathbf{I}_m - \alpha\mathbf{P})^{-1}\boldsymbol{y}. \tag{21}$$

Therefore, the convergence result of Eq. (19) is

$$\boldsymbol{v}^{\star} = (1 - \alpha)(\mathbf{I}_m - \alpha\mathbf{P})^{-1}\boldsymbol{y} = \sum_{i=1}^{m}\frac{1 - \alpha}{1 - \alpha\gamma_i}\boldsymbol{u}_i\boldsymbol{u}_i^{\top}\boldsymbol{y}. \tag{22}$$

Finally, we can define the label propagation function as:

$$\varphi(\boldsymbol{y}, \alpha) = \sum_{i=1}^{m}\frac{1 - \alpha}{1 - \alpha\gamma_i}\boldsymbol{u}_i\boldsymbol{u}_i^{\top}\boldsymbol{y}. \tag{23}$$



(a) Label propagation function (b) MLP

Fig. 4 Output comparison between label propagation function and MLP

5.1.2 Understanding the label propagation function

Here we clarify intuitively and theoretically why label propagation function can preserve the label ranking within the logical label vector.

Intuitive explaination. Let us consider the example of three-dimensional logical label vectors. We transform logical label vectors into real-valued vectors by the label propagation function and multilayer perceptron, respectively, and normalize them to label distributions; the parameter values of both label propagation function and multilayer perceptron are arbitrary. As shown in Fig. 4, these three-dimensional label distributions are visualized on the 2-simplex (triangle). Figure 4a shows the normalized outputs (i.e., label distributions) of the label propagation function, where parameter α takes random values between 0 and 1, and Laplacian matrix $\mathbf{P}$ is derived from the label graph having random co-occurrence probabilities. Figure 4b shows the normalized outputs of the multilayer perceptron having a sigmoid activation function. Each dashed line in the two simplexes divides each simplex into two parts, which indicate the areas where the outputs corresponding to one label are smaller or larger than the outputs corresponding to another label, respectively. For example, the left (or right) side of the vertical dashed line indicates that the outputs corresponding to the 1st label are larger (or smaller) than the outputs corresponding to the 2nd label. As shown in the legend of Fig. 4, points with different colors indicate different logical label vector. In Fig. 4a, the higher transparency of a point indicates a larger value of α.

Intuitively, points of the same color in Fig. 4b (i.e., the multilayer perceptron output of the same logical label vector) may be distributed anywhere. In contrast, the point cluster with the same color in Fig. 4a (i.e., the label propagation function output of the same logical label vector) roughly crosses only one dashed line. More specifically, for the logical label vector with the same logical value at the i-th and j-th labels, the output area of label propagation function roughly crosses only the dashed line segmenting the i-th and j-th labels. This intuitively shows that the output of the label propagation function can roughly maintain the ranking consistency with the logical label vector. In addition, it can be found that the points with higher transparency (i.e., the larger α) in Fig. 4a are closer to the triangle center and appear to be more likely to cross more dashed lines (i.e., violating the label ranking).

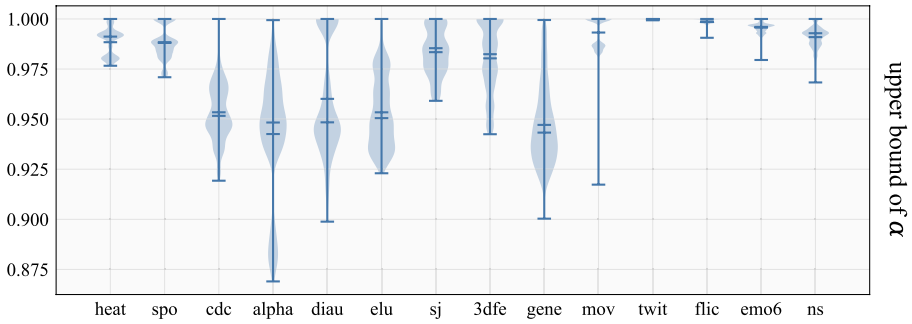


Fig. 5 Distributions of $\alpha_y^\star$ in 14 datasets shown in Table 1

Theoretical analysis. Motivated by the above observation, we next theoretically answer "what range of α always preserves label ranking?". We first give a formal definition:

Definition 1 Given a label confidence vector $\boldsymbol{v}$ and a logical label vector $\boldsymbol{y}$, we say that $\boldsymbol{v}$ preserves the label ranking underlying $\boldsymbol{y}$, denoted by $\boldsymbol{v} \simeq \boldsymbol{y}$, if $v_i < v_j$ holds for any i and j satisfying $y_i = 0$ and $y_j = 1$.

Then we give a theorem to answer "what range of α will always preserve the label ranking?".

Theorem 1 *Suppose that $\boldsymbol{y}$ is any logical label vector, $\mathbf{P}$ is the symmetrically normalized Laplacian of any label co-occurrence graph, $\boldsymbol{v} = \varphi(\boldsymbol{y}, \alpha)$ (as defined in Eq. (23)). Then we have $\boldsymbol{v} \simeq \boldsymbol{y}$ if $\alpha \in (0, \alpha_y^\star)$, where $\alpha_y^\star = \inf(\mathbb{A} \cup \{1\})$,*

Table 1 Datasets statistics

No.	Dataset	# Instances	# Features	# Labels
1	SJAFFE (sj)	213	243	6
2	SBU-3DFE (3dfe)	2500	243	6
3	Twitter-LDL (twit)	11150	168	8
4	Flickr-LDL (flic)	10045	168	8
5	Emotion6 (emo6)	1980	168	7
6	Natural Scene (ns)	2000	294	9
7	Movie (mov)	7755	1869	5
8	Human Gene (gene)	30542	36	68
9	Yeast-alpha (alpha)	2465	24	18
10	Yeast-heat (heat)	2465	24	6
11	Yeast-spo (spo)	2465	24	6
12	Yeast-diau (diau)	2465	24	7
13	Yeast-cdc (cdc)	2465	24	15
14	Yeast-elu (elu)	2465	24	14

$$\mathbb{A} = \bigcup_{\substack{i \in \{i \mid y_i = 1\} \\ j \in \{j \mid y_j = 0\}}} \left\{ \frac{1}{2} \leq \alpha < 1 \mid \varphi_i(\boldsymbol{y}, \alpha) = \varphi_j(\boldsymbol{y}, \alpha) \right\}. \tag{24}$$

The proof can be found in Appendix. We visualizes the distributions of the upper bound of α computed from the 14 datasets (shown in Table 1) in Fig. 5. It is worth noting that the equation $\varphi_i(\boldsymbol{y}, \alpha) = \varphi_j(\boldsymbol{y}, \alpha)$ in Eq. (24) is essentially polynomial, and there are many mathematical tools that can be used to efficiently obtain the numerical solution of α. To shed light on the calculation of $\alpha_{\boldsymbol{y}}^{\star}$, we give an example in Appendix.

5.1.3 Modeling variational family

Based on the above analysis, we use the following logistic function to model α in Eq. (23):

$$\tilde{\alpha}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \left(1 + e^{-h(\boldsymbol{x}; \boldsymbol{\omega})}\right)^{-1} \alpha_{\boldsymbol{y}}^{\star}, \tag{25}$$

where $\alpha_{\boldsymbol{y}}^{\star}$ is defined in Eq. (24), $h(\cdot)$ is a linear function with learnable parameter $\boldsymbol{\omega}$. Obviously, $0 < \tilde{\alpha}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) < \alpha_{\boldsymbol{y}}^{\star}$. Further, we can use $\varphi(\boldsymbol{y}, \tilde{\alpha}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}))$ to represent $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$, which can preserve the label ranking. According to Eq. (17), the variational family can be specified as:

$$\begin{aligned} q(\mathbf{Z} | \mathbf{X}, \mathbf{Y}) &= \prod_{i=1}^{n} \prod_{j=1}^{m} \kappa(z_{ij} | a_{ij}, b_{ij}), \\ a_{ij} = \frac{\log(1 - 2^{-\frac{1}{b_{ij}}})}{\log\left(\varphi_j(\boldsymbol{y}_i, \tilde{\alpha}(\boldsymbol{x}_i; \boldsymbol{\omega}))\right)}, &\quad b_{ij} = \exp(\psi_j(\boldsymbol{x}_i; \boldsymbol{\eta})), \end{aligned} \tag{26}$$

where the functions $\varphi(\cdot)$ and $\tilde{\alpha}(\cdot)$ are defined in Eqs. (23) and (25), respectively.

5.2 Reconstruction term

Here, we use the reparameterization trick to estimate the reconstruction term $\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\left[\log p(\mathbf{X}, \mathbf{Y} | \mathbf{Z})\right]$. According to the generation process, the log likelihood $\log p(\mathbf{X}, \mathbf{Y} | \mathbf{Z})$ is:

$$\log p(\mathbf{X}, \mathbf{Y} | \mathbf{Z}) = \sum_{i=1}^{n} \log \mathcal{N}(\boldsymbol{x}_i | \mu(\boldsymbol{z}_i; \boldsymbol{\theta}), \mathbf{I}_c) + \sum_{j=1}^{m} \log \mathrm{Ber}\left(y_{ij} \middle| \frac{1}{|\delta_i|} \sum_{t \in \delta_i} z_{tj}\right). \tag{27}$$

It is worth noting that the Gaussian likelihood and Bernoulli likelihood may differ considerably w.r.t. the order of magnitude; in this case, the likelihood function of the feature vector and the likelihood function of the logical label vector may differ considerably in their contributions to ELBO, which leads to poor reconstruction of some variables (Kendall et al., 2018). Therefore, we use a hyperparameter λ to reweight the likelihood for logical labels. Formally, the reconstruction term can be rewritten as:

$$\log p(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \text{const} + \sum_{i=1}^{n} -\frac{1}{2}\|\boldsymbol{x}_i - \mu(\boldsymbol{z}_i;\boldsymbol{\theta})\|_2^2 + \lambda \sum_{j=1}^{m} y_{ij} \log\left(\sum_{t\in\delta_i} \frac{z_{tj}}{|\delta_i|}\right) + (1-y_{ij}) \log\left(1 - \sum_{t\in\delta_i} \frac{z_{tj}}{|\delta_i|}\right). \tag{28}$$

In addition, it is difficult to compute derivatives of the log of summation expression concerning model parameters; therefore, we consider the lower bound of Eq. (28). By applying the Jensen's inequality, we have $\log p(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \geq \log \check{p}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$, where

$$\log \check{p}(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \text{const} + \sum_{i=1}^{n} -\frac{1}{2}\|\boldsymbol{x}_i - \mu(\boldsymbol{z}_i;\boldsymbol{\theta})\|_2^2 + \frac{\lambda}{|\delta_i|} \sum_{j=1}^{m} \sum_{t\in\delta_i} y_{ij} \log(z_{tj}) + (1-y_{ij}) \log(1-z_{tj}). \tag{29}$$

To ensure the full differentiability of our model, according to Eqs. (9) and (26), we have $z_{ij} = (1-\epsilon^{\frac{1}{b_{ij}}})^{\frac{1}{a_{ij}}}$, where $\epsilon \sim \text{Uniform}(\epsilon|0,1)$, a_{ij} and b_{ij} are defined by Eq. (26). Then the expectation of log likelihood can be approximated by:

$$\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\left[\log \check{p}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})\right] \approx \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} -\frac{1}{2}\|\boldsymbol{x}_i - \mu(\boldsymbol{z}_i^{(t)};\boldsymbol{\theta})\|_2^2 + \frac{\lambda}{|\delta_i|} \sum_{j=1}^{m} \sum_{t\in\delta_i} y_{ij} \log(z_{tj}^{(t)}) + (1-y_{ij}) \log(1-z_{tj}^{(t)}) + \text{const}, \tag{30}$$

where $z_{ij}^{(t)} = (1-(\epsilon^{(t)})^{\frac{1}{b_{ij}}})^{\frac{1}{a_{ij}}}$, $\epsilon^{(t)} \sim \text{Uniform}(\epsilon|0,1)$, T is the number of Monte Carlo samples.

5.3 Prior regularization term

As shown in Eq. (12), we assume a non-informative prior for latent variable $\boldsymbol{z}$, i.e., $p(\mathbf{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{m} \text{Beta}(z_{ij}|1,1)$. Then we can analytically derive the KL divergence of the variational posterior $q(\mathbf{Z}|\mathbf{X},\mathbf{Y})$ from the prior $p(\mathbf{Z})$ according to Eq. (11):

$$D_{KL}(q(\mathbf{Z}|\mathbf{X},\mathbf{Y})\|p(\mathbf{Z})) = \int \prod_{i=1}^{n}\prod_{j=1}^{m} q(z_{ij}) \log\left(\frac{\prod_{i=1}^{n}\prod_{j=1}^{m} q(z_{ij})}{\prod_{i=1}^{n}\prod_{j=1}^{m} p(z_{ij})}\right) \mathrm{d}\mathbf{Z} = \sum_{i=1}^{n}\sum_{j=1}^{m} \log(a_{ij}b_{ij}) + \frac{\left((1-a_{ij})\digamma(b_{ij}) + b_{ij}^{-1} + \gamma_e\right)}{a_{ij}}, \tag{31}$$

where $\digamma$ is the digamma function, γ_e is the Euler's constant, a_{ij} and b_{ij} are defined in Eq. (26).

5.4 Variational learning objective

According to Eqs. (26), (30) and (31), we obtain the optimization objective, i.e., the lower bound of $\mathcal{L}_{ELBO}$:

$$\begin{aligned}\arg\max_{\boldsymbol{\theta},\boldsymbol{\omega},\boldsymbol{\eta}} \quad & \sum_{i=1}^{n}\frac{1}{T}\Bigg(\sum_{t=1}^{T}-\frac{1}{2}\|\boldsymbol{x}_i-\mu(\boldsymbol{z}_i^{(t)};\boldsymbol{\theta})\|_2^2+\frac{\lambda}{|\delta_i|}\sum_{j=1}^{m}\sum_{k\in\delta_i}y_{ij}\log(z_{kj}^{(t)})+\\ & (1-y_{ij})\log(1-z_{kj}^{(t)})\Bigg)-\sum_{j=1}^{m}\log(a_{ij}b_{ij})+\frac{\left((1-a_{ij})\digamma(b_{ij})+b_{ij}^{-1}+\gamma_e\right)}{a_{ij}},\end{aligned} \tag{32}$$

where $\mu(\boldsymbol{z}_i^{(t)};\boldsymbol{\theta})$ is a linear function of $\boldsymbol{z}_i^{(t)}$ with the parameter of $\boldsymbol{\theta}$, $z_{ij}^{(t)}=(1-(\epsilon^{(t)})^{\frac{1}{b_{ij}}})^{\frac{1}{a_{ij}}}$; a_{ij} and b_{ij} are defined in Eq. (26); $\epsilon^{(t)}\sim\text{Uniform}(\epsilon|0,1)$, T is the number of Monte Carlo samples; $\digamma$ is the digamma function; γ_e is the Euler's constant; $\delta_i=\{j\mid\boldsymbol{x}_i\in K_{\text{NN}}(\boldsymbol{x}_j)\}$; K is the neighborhood size; $\lambda>0$ is a hyperparameter. Equation (32) can be maximized by Adam algorithm (Kingma & Ba, 2015).

Once obtaining the optimal parameters, we use the median of variational posterior as the label confidence and normalize it to recover the label distributions:

$$d_{ij}=\frac{\varphi_j(\boldsymbol{y}_i,\tilde{\boldsymbol{\alpha}}(\boldsymbol{x}_i;\boldsymbol{\omega}))}{\sum_{k=1}^{m}\varphi_k(\boldsymbol{y}_i,\tilde{\boldsymbol{\alpha}}(\boldsymbol{x}_i;\boldsymbol{\omega}))}. \tag{33}$$

6 Experiment

6.1 Experimental configuration

Algorithm 1 Binarization (reducing the label distributions to logical labels)

Input: the matrix of ground-truth label distributions $\mathbf{D}$;
Output: the matrix of logical labels $\mathbf{Y}$;
$\mathbf{Y}\leftarrow\mathbf{0}_{n\times m}$;
for $i=1,2,\cdots,n$ **do**
  $t\leftarrow 0$;
  while $t<0.5$ **do**
    $d_{ij}\leftarrow$ Find the largest value in $\boldsymbol{d}_i$;
    $t\leftarrow t+d_{ij}$, $y_{ij}\leftarrow 1$, $d_{ij}\leftarrow 0$;
  end while
end for
return $\mathbf{Y}$

Algorithm 2 Recovery and predictive experiments

Input: the label distribution dataset $(\mathbf{X}, \mathbf{D})$;
// Procedure of recovery experiment
$\mathbf{Y} \leftarrow$ Binarizing $\mathbf{D}$ by Algorithm 1;
$\hat{\mathbf{D}} \leftarrow$ Training an LE model on $(\mathbf{X}, \mathbf{Y})$ and recovering the label distributions;
Computing LDL measures according to $\hat{\mathbf{D}}$ and $\mathbf{D}$;
// Procedure of predictive experiment
$\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}}, \mathbf{D}_{\text{train}}, \mathbf{D}_{\text{test}} \leftarrow$ Splitting the dataset into training sets and testing sets;
$\mathbf{Y}_{\text{train}} \leftarrow$ Binarizing $\mathbf{D}_{\text{train}}$ by Algorithm 1;
$\hat{\mathbf{D}}_{\text{train}} \leftarrow$ Training an LE model and recovering the label distributions for all training instances $\mathbf{X}_{\text{train}}$;
$\hat{\mathbf{D}}_{\text{test}} \leftarrow$ Training an LDL model f_{LDL} on $(\mathbf{X}_{\text{train}}, \hat{\mathbf{D}}_{\text{train}})$, and using f_{LDL} to predict label distributions of $\mathbf{X}_{\text{test}}$;
Computing LDL measures based on $\hat{\mathbf{D}}_{\text{test}}$ and $\mathbf{D}_{\text{test}}$;

6.1.1 Datasets and evaluation measures

The datasets used in our experiments are summarized in Table 1. These datasets involve areas such as sentiment analysis (No. 1-5), natural landscape recognition (No. 6), crowd opinion on movies (No. 7), biological research (No. 8-14).[3] Notably, for the "twit", "flic" (Yang et al., 2017) and "emo6" (Peng et al., 2015) datasets, we use the preprocessed versions (Ren et al., 2019). Besides, to accelerate the convergence, we use min-max normalization to preprocess the feature matrices of all datasets.

We consider six distance or similarity-based measures (i.e., Chebyshev distance, Clark distance, Canberra metric, Kullback-Leibler divergence, cosine coefficient and intersection similarity) suggested by literature (Geng, 2016) and one ranking-based measure (i.e., Spearman's Rho) suggested by literature (Jia et al., 2023). Cheb (Chebyshev distance), Clark (Clark distance), Canber (Canberra metric) and KL (Kullback–Leibler divergence) assess the distance between the estimated label distributions and ground-truth; Cosine (cosine coefficient) and Intersec (intersection similarity) assess the similarity between the estimated label distributions and ground-truth; Rho (Spearman's Rho) assess the similarity between the label ranking w.r.t. the estimated label distribution and that w.r.t. the ground-truth. The lower values of Cheb, Canber, Clark, and KL indicate better performances. The higher values of Cosine, Intersec and Rho indicate better performances. The formulas of these measures can be found in their respective literature (Geng, 2016; Jia et al., 2023). Due to page limitation, we do not show the performance on the Canber and Clark measures, which are similar to the results on Cheb.

[3] Since label relation plays a vital role in this paper, we ignore the Yeast datasets which contain less than five label variables.

6.1.2 Experimental procedure

We evaluate the performance of LE methods using the experimental procedure suggested by Xu et al. (2018). Briefly speaking, the experiments consists of two parts. The first part is the recovery experiment. We run different LE algorithms with the same random seed to recover the label distributions, compare the recovered label distributions with the ground-truth label distributions, and record the performances on multiple LDL measures. The above process is repeated ten times, and the averaged performances are reported. The second part is the predictive experiment. We use different LE methods to recover the label distributions of training instances and use these recovered distributions to train an LDL model, whose performance on test instances will be reported. AAkNN (Geng, 2016) with parameter $K = 7$ is used as the LDL model in this paper. We randomly partition the whole dataset into two equal chunks. One chunk is used as the training set and the other is used as the test set. We repeat the above process 10 times and report the average and variance of the results. The experimental procedures of recovery experiment and predictive experiment are shown in Algorithm 2.

6.1.3 Comparison algorithms

We compare our approach with six recently proposed LE algorithms, i.e., LESC (Tang et al., 2020), LEVI (Xu et al., 2020), BDLE (Liu et al., 2021), FLE (Wang et al., 2023), PLEA (Tan et al., 2022), and gLESC (Zheng et al., 2023). For each comparison algorithm, we adopt the hyperparameter configuration or hyperparameter tuning scheme suggested in their literature. For LESC, the parameters λ_1 and λ_2 are selected among $\{10^{-4}, 10^{-3}, ..., 10\}$. For LEVI, the MLPs were constructed with two hidden layers, each with 500 hidden units and softplus activation functions. For BDLE, the parameters α and λ are both set to 10^{-2}. For FLE, the parameters $\alpha, \beta, \lambda, \gamma$ are selected among $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, \cdots, 5 \times 10^{2}, 10^{3}\}$. For gLESC, the parameters λ_1 and λ_2 are selected among $\{10^{-4}, 10^{-3}, \cdots, 10^{1}\}$. For our method GLERB, the hyperparameters are set as $\lambda = 50, K = 8$, the learning rate and the number of iterations (of the Adam optimizer) are set as 10^{-3} and 200, respectively, other configurations of Adam follows the default settings given by pytorch.[4]

6.2 Performance of recovery and prediction

Here, we show the performance of different LE methods on the recovery experiment and the predictive experiment. The recovery performance is shown in Tables 2 and 3. The predictive performance is shown in Fig. 6. The average ranks of each LE algorithms on the recovery experiment and predictive experiment are shown in Tables 4 and 5, respectively. Besides, for predictive experiment, we conduct a pairwise two-tailed t-test with 0.05 significance level, whose results are summarized in Table 6. Overall, our method achieves competitive performance. It is worth highlighting the outstanding performance of our algorithm on the Rho metric. In terms of the Rho metric, not only does our algorithm achieve the first place on all datasets, but it also outperforms the second place by more than 10% on some datasets (e.g., 38% on emo6 and 13% on flic, ns, and mov). This strongly supports the effectiveness of our designed label propagation

[4] https://pytorch.org/docs/1.13/generated/torch.optim.Adam.html.

Table 2 Recovery performance (value (rank)) on 7 datasets (No. 1-7)

	sj	3dfe	emo6	twit	flic	ns	mov
Cheb (↓)							
GLERB	**0.070 (1)**	0.097 (3)	**0.225 (1)**	**0.209 (1)**	**0.175 (1)**	0.272 (2)	**0.105 (1)**
gLESC	0.070 (2)	0.123 (4)	0.332 (6)	0.505 (6)	0.419 (5)	0.344 (6)	0.133 (3)
PLEA	0.122 (7)	0.145 (7)	0.320 (3)	0.308 (2)	0.331 (2)	0.363 (7)	0.162 (6)
LESC	0.075 (4)	0.124 (5)	0.329 (5)	0.508 (7)	0.423 (6)	0.342 (5)	0.140 (4)
LEVI	0.073 (3)	**0.092 (1)**	0.271 (2)	0.468 (4)	0.374 (3)	0.324 (3)	0.109 (2)
BDLE	0.078 (5)	0.130 (6)	0.324 (4)	0.478 (5)	0.405 (4)	0.339 (4)	0.167 (7)
FLE	0.083 (6)	**0.092 (1)**	0.360 (7)	0.325 (3)	0.447 (7)	**0.268 (1)**	0.153 (5)
KL (↓)							
GLERB	**0.025 (1)**	0.054 (3)	**0.351 (1)**	**0.445 (1)**	**0.363 (1)**	0.651 (2)	**0.061 (1)**
gLESC	0.028 (2)	0.068 (4)	0.638 (6)	1.128 (6)	0.924 (6)	1.011 (7)	0.111 (3)
PLEA	0.083 (7)	0.115 (7)	0.624 (4)	0.719 (3)	0.753 (3)	1.008 (6)	0.204 (7)
LESC	0.030 (3)	0.070 (5)	0.628 (5)	1.142 (7)	0.942 (7)	1.005 (5)	0.123 (4)
LEVI	0.032 (4)	0.041 (2)	0.474 (2)	0.987 (4)	0.783 (4)	0.924 (3)	0.081 (2)
BDLE	0.033 (5)	0.085 (6)	0.618 (3)	1.047 (5)	0.903 (5)	0.971 (4)	0.171 (6)
FLE	0.033 (5)	**0.039 (1)**	0.995 (7)	0.662 (2)	0.612 (2)	**0.648 (1)**	0.143 (5)
Cosine (↑)							
GLERB	**0.974 (1)**	0.951 (3)	**0.885 (1)**	**0.933 (1)**	**0.939 (1)**	0.848 (2)	**0.965 (1)**
gLESC	0.973 (2)	0.933 (4)	0.695 (6)	0.598 (6)	0.632 (6)	0.656 (6)	0.929 (3)
PLEA	0.923 (7)	0.895 (7)	0.696 (5)	0.802 (3)	0.733 (3)	0.632 (7)	0.886 (6)
LESC	0.971 (3)	0.931 (5)	0.703 (4)	0.590 (7)	0.621 (7)	0.661 (5)	0.919 (4)
LEVI	0.969 (4)	0.958 (2)	0.814 (2)	0.683 (4)	0.725 (4)	0.713 (3)	0.954 (2)
BDLE	0.968 (5)	0.919 (6)	0.708 (3)	0.642 (5)	0.644 (5)	0.676 (4)	0.883 (7)
FLE	0.967 (6)	**0.959 (1)**	0.601 (7)	0.875 (2)	0.821 (2)	**0.849 (1)**	0.905 (5)
Intersec (↑)							
GLERB	**0.912 (1)**	0.865 (3)	**0.685 (1)**	**0.716 (1)**	**0.735 (1)**	0.562 (2)	**0.873 (1)**
gLESC	0.905 (2)	0.856 (4)	0.550 (7)	0.381 (6)	0.450 (6)	0.409 (7)	0.818 (3)
PLEA	0.845 (7)	0.827 (7)	0.564 (3)	0.612 (2)	0.561 (2)	0.438 (4)	0.784 (6)
LESC	0.904 (3)	0.854 (5)	0.554 (6)	0.377 (7)	0.444 (7)	0.411 (6)	0.807 (4)
LEVI	0.896 (6)	0.884 (2)	0.621 (2)	0.416 (4)	0.494 (4)	0.442 (3)	0.850 (2)
BDLE	0.898 (5)	0.840 (6)	0.559 (4)	0.404 (5)	0.464 (5)	0.426 (5)	0.774 (7)
FLE	0.900 (4)	**0.892 (1)**	0.558 (5)	0.546 (3)	0.553 (3)	**0.573 (1)**	0.786 (5)
Rho (↑)							
GLERB	**0.856 (1)**	**0.782 (1)**	**0.751 (1)**	**0.569 (1)**	**0.602 (1)**	**0.698 (1)**	**0.861 (1)**
gLESC	0.814 (2)	0.394 (4)	0.353 (5)	0.466 (5)	0.429 (5)	0.457 (6)	0.732 (4)
PLEA	0.167 (7)	0.153 (7)	0.298 (7)	0.499 (4)	0.408 (6)	0.363 (7)	0.644 (6)
LESC	0.800 (3)	0.364 (5)	0.399 (4)	0.456 (6)	0.400 (7)	0.491 (4)	0.659 (5)
LEVI	0.755 (4)	0.736 (2)	0.422 (3)	0.423 (7)	0.452 (3)	0.550 (3)	0.755 (3)
BDLE	0.376 (6)	0.318 (6)	0.341 (6)	0.500 (3)	0.435 (4)	0.480 (5)	0.644 (7)
FLE	0.755 (5)	0.733 (3)	0.541 (2)	0.561 (2)	0.534 (2)	0.626 (2)	0.760 (2)

Bold values indicate the best performance

Table 3 Recovery performance (value (rank)) on 7 datasets (No. 8-14)

	gene	alpha	heat	spo	diau	cdc	elu
Cheb (↓)							
GLERB	**0.052 (1)**	0.014 (2)	0.040 (2)	0.054 (3)	0.033 (2)	**0.013 (1)**	**0.013 (1)**
gLESC	0.053 (3)	0.018 (5)	0.047 (3)	0.061 (4)	0.040 (3)	0.017 (4)	0.019 (4)
PLEA	0.054 (5)	0.033 (6)	0.052 (4)	0.065 (5)	0.083 (6)	0.026 (5)	0.027 (5)
LESC	0.789 (7)	0.047 (7)	0.257 (7)	0.243 (7)	0.104 (7)	0.039 (7)	0.054 (7)
LEVI	0.052 (2)	0.018 (3)	0.053 (5)	0.054 (2)	0.044 (4)	0.016 (3)	0.018 (3)
BDLE	0.059 (6)	0.018 (3)	0.073 (6)	0.081 (6)	0.050 (5)	0.034 (6)	0.037 (6)
FLE	0.053 (4)	**0.014 (1)**	**0.032 (1)**	**0.048 (1)**	**0.032 (1)**	0.015 (2)	0.017 (2)
KL (↓)							
GLERB	**0.205 (1)**	**0.005 (1)**	0.011 (2)	0.022 (2)	0.011 (2)	**0.004 (1)**	**0.004 (1)**
gLESC	0.223 (3)	0.011 (4)	0.016 (3)	0.027 (4)	0.015 (3)	0.008 (3)	0.008 (3)
PLEA	0.244 (5)	0.031 (6)	0.017 (4)	0.032 (5)	0.071 (6)	0.019 (5)	0.019 (5)
LESC	25.626 (7)	0.064 (7)	0.435 (7)	0.377 (7)	0.093 (7)	0.042 (7)	0.069 (7)
LEVI	0.207 (2)	0.011 (3)	0.028 (5)	0.025 (3)	0.024 (4)	0.013 (4)	0.014 (4)
BDLE	0.316 (6)	0.012 (5)	0.037 (6)	0.047 (6)	0.024 (4)	0.031 (6)	0.033 (6)
FLE	0.236 (4)	0.006 (2)	**0.007 (1)**	**0.017 (1)**	**0.009 (1)**	0.005 (2)	0.006 (2)
Cosine (↑)							
GLERB	**0.859 (1)**	**0.995 (1)**	0.989 (2)	0.980 (2)	0.990 (2)	**0.996 (1)**	**0.996 (1)**
gLESC	0.845 (3)	0.989 (4)	0.985 (3)	0.975 (4)	0.986 (3)	0.992 (3)	0.992 (3)
PLEA	0.829 (5)	0.977 (6)	0.983 (4)	0.970 (5)	0.942 (6)	0.982 (5)	0.982 (5)
LESC	0.119 (7)	0.940 (7)	0.774 (7)	0.782 (7)	0.923 (7)	0.961 (7)	0.939 (7)
LEVI	0.857 (2)	0.990 (3)	0.976 (5)	0.978 (3)	0.981 (4)	0.988 (4)	0.987 (4)
BDLE	0.778 (6)	0.989 (5)	0.966 (6)	0.957 (6)	0.977 (5)	0.970 (6)	0.969 (6)
FLE	0.835 (4)	0.995 (2)	**0.993 (1)**	**0.984 (1)**	**0.992 (1)**	0.995 (2)	0.994 (2)
Intersec (↑)							
GLERB	**0.813 (1)**	**0.964 (1)**	0.946 (2)	0.926 (2)	0.946 (2)	**0.967 (1)**	**0.969 (1)**
gLESC	0.797 (3)	0.944 (3)	0.932 (3)	0.912 (3)	0.936 (3)	0.954 (3)	0.951 (3)
PLEA	0.778 (5)	0.922 (6)	0.930 (4)	0.900 (5)	0.844 (6)	0.921 (5)	0.924 (5)
LESC	0.045 (7)	0.867 (7)	0.663 (7)	0.680 (7)	0.840 (7)	0.891 (7)	0.864 (7)
LEVI	0.810 (2)	0.933 (5)	0.895 (5)	0.904 (4)	0.909 (4)	0.927 (4)	0.925 (4)
BDLE	0.734 (6)	0.941 (4)	0.895 (5)	0.882 (6)	0.909 (4)	0.904 (6)	0.902 (6)
FLE	0.786 (4)	0.962 (2)	**0.957 (1)**	**0.937 (1)**	**0.953 (1)**	0.967 (2)	0.958 (2)
Rho (↑)							
GLERB	**0.657 (1)**	**0.777 (1)**	**0.780 (1)**	**0.776 (1)**	**0.772 (1)**	**0.781 (1)**	**0.786 (1)**
gLESC	0.628 (3)	0.195 (6)	0.190 (5)	0.183 (6)	0.349 (7)	0.175 (6)	0.205 (5)
PLEA	0.145 (5)	0.200 (5)	0.098 (7)	0.200 (5)	0.382 (5)	0.191 (5)	0.202 (7)
LESC	0.004 (7)	0.182 (7)	0.179 (6)	0.167 (7)	0.367 (6)	0.173 (7)	0.203 (6)
LEVI	0.628 (3)	0.754 (3)	0.761 (3)	0.739 (3)	0.760 (3)	0.767 (2)	0.753 (3)
BDLE	0.097 (6)	0.236 (4)	0.199 (4)	0.213 (4)	0.418 (4)	0.217 (4)	0.244 (4)
FLE	0.654 (2)	0.777 (2)	0.771 (2)	0.771 (2)	0.762 (2)	0.747 (3)	0.771 (2)

Bold values indicate the best performance

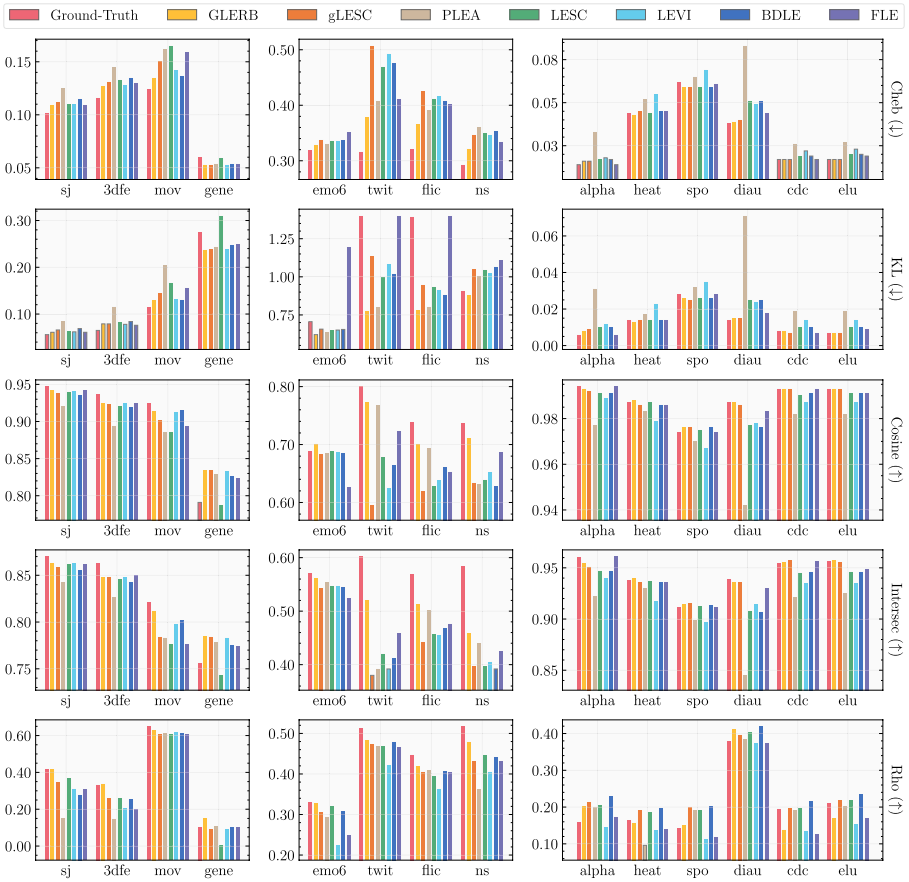


Fig. 6 Predictive performance on 14 datasets

Table 4 Average ranks of different LE algorithms on the recovery experiment. The boldface denotes the best performance

	GLERB	gLESC	PLEA	LESC	LEVI	BDLE	FLE
Cheb	**1.57**	4.14	5.00	6.07	2.93	5.25	3.04
KL	**1.43**	4.07	5.21	6.07	3.32	5.29	2.61
Cosine	**1.43**	4.00	5.29	6.00	3.29	5.36	2.64
Intersec	**1.43**	4.00	4.79	6.21	3.71	5.36	2.50
Rho	**1.00**	4.96	5.93	5.71	3.25	4.79	2.36

Bold values indicate the best performance

function in preserving label rankings. In addition, it can be seen that the performance on "Rho" in the predictive experiment seems to be unsatisfactory. This is caused by the LDL algorithm (AAkNN) itself. For example, given a test instance $\boldsymbol{x}_i$ with ground-truth $\boldsymbol{d}_i = [0.5, 0.5]^\top$ and neighbors $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}$, then the predicted label distribution $\hat{\boldsymbol{d}}_i$ for $\boldsymbol{x}_i$ based on the ground-truth ($\boldsymbol{d}_{i_1}$ and $\boldsymbol{d}_{i_2}$) and recovered label distributions ($\tilde{\boldsymbol{d}}_{i_1}$ and $\tilde{\boldsymbol{d}}_{i_2}$) of these neighbors may be as follows:

Table 5 Average ranks of different LE algorithms on the predictive experiment. The boldface denotes the best performance

	GLERB	Ground-Truth	gLESC	LESC	LEVI	PLEA	BDLE	FLE
Cheb	**2.14**	2.18	4.61	5.36	5.57	6.50	5.46	4.18
Cosine	**2.07**	2.21	4.64	5.39	5.50	6.54	5.29	4.36
Intersec	**2.14**	2.18	4.54	5.46	5.61	6.25	5.50	4.32
KL	**2.07**	3.61	4.07	5.14	5.18	5.93	5.07	4.93
Rho	**2.96**	3.07	4.07	4.11	6.93	5.46	3.04	6.36

Bold values indicate the best performance

Table 6 Counts of win/tie/loss for predictive experiments under pairwise two-tailed t-test with 0.05 significance level

	Ground-Truth	gLESC	LESC	LEVI	PLEA	BDLE	FLE
Cheb	2/3/9	8/6/0	11/3/0	11/3/0	13/1/0	13/1/0	5/8/1
KL	6/4/4	8/5/1	12/2/0	11/3/0	14/0/0	12/2/0	5/7/2
Cosine	4/4/6	9/5/0	12/2/0	11/3/0	13/1/0	12/2/0	6/7/1
Intersec	3/2/9	9/4/1	13/1/0	12/2/0	14/0/0	13/1/0	6/6/2
Rho	3/7/4	8/3/3	8/4/2	14/0/0	11/2/1	7/2/5	14/0/0

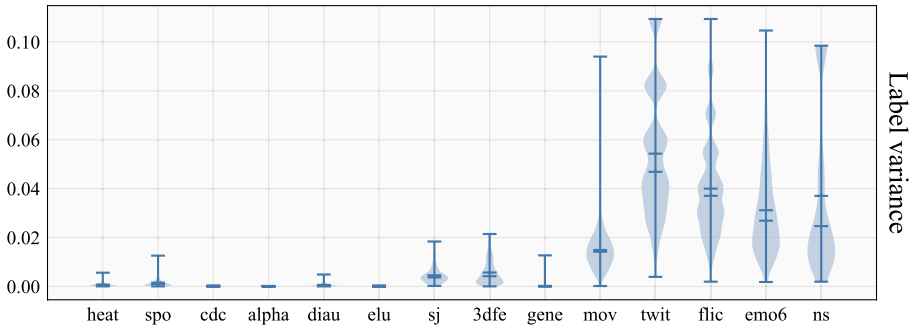


Fig. 7 Distributions over the label variance of the instances. The first nine are low-variance datasets and the last five are high-variance datasets

$$\boldsymbol{d}_{i_1} = [0.2, 0.8]^\top, \quad \boldsymbol{d}_{i_2} = [0.4, 0.6]^\top \implies \hat{\boldsymbol{d}}_i = [0.3, 0.7]^\top,$$
$$\tilde{\boldsymbol{d}}_{i_1} = [0.7, 0.3]^\top, \quad \tilde{\boldsymbol{d}}_{i_2} = [0.3, 0.7]^\top \implies \hat{\boldsymbol{d}}_i = [0.5, 0.5]^\top.$$

It can be seen that the ground-truth derives a wrong label distribution, which explains why in some datasets our method achieves good recovery performance but mediocre predictive performance. Likewise, this also explains why the performance of AAkNN trained on the ground-truth label distributions is worse on some datasets than that trained on the recovered label distributions. According to the above example, the main reason is that the predictive performance of AAkNN depends not only on the quality of the label distributions in the training set, but also on whether the k-nearest neighbors of the test instance $\boldsymbol{x}_i$ in the

feature space can well reflect $\boldsymbol{x}_i$. If the k-nearest neighbors of the test instance $\boldsymbol{x}_i$ can well reflect $\boldsymbol{x}_i$, the LE algorithm with better recovery quality will strongly lead to good predictive performance; otherwise, the quality of LE algorithms may not definitively affect on the good prediction performance.

With a further observation, compared to other methods, our method slightly improves performance on the Yeast datasets (No. 9-14 in Table 1), "sj", "3dfe", and "gene" datasets (low-variance group), but outperforms the other methods by a considerable advantage on the "emo6", "twit", "flic", "ns" and "mov" datasets (high-variance group). This is due to the different distributions over the label variance of each dataset; the label variance of an instance is the variance of the elements in this instance's label distribution. As shown in Fig. 7, in the datasets in low-variance group, instances tend to be concentrated in the region with very low label variance. In contrast, in the datasets in high-variance group, instances tend to spread over the region with a wide range of label variance. Obviously, when dealing with a new task (dataset), our method can stably recover trustworthy label distributions, regardless of whether this dataset has a distribution of high or low label variances.

6.3 Parameter analysis and ablation study

Here we study the effectiveness of our proposed modules (i.e., the label propagation function and neighbor generation assumption) via parameter analysis and ablation experiment.

6.3.1 Parameter analysis

Here, we analyze the effects of hyperparameters on the recovery performance on six datasets (including four datasets in low-variance group and two datasets in high-variance group).

Neighborhood size. Figure 8 shows the effect of increasing the neighborhood size to the recovery performance. The values of K are taken in $\{2^0, 2^1, \cdots, 2^7\}$, where $K = 1$

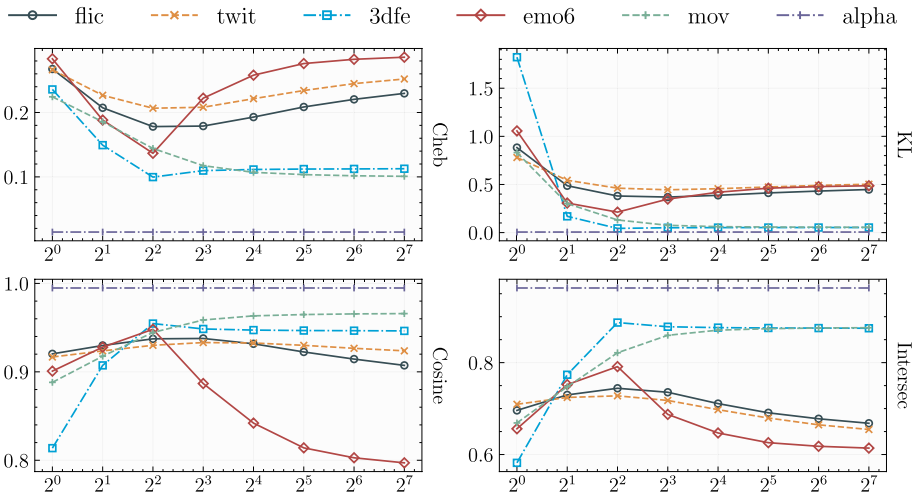


Fig. 8 Recovery performance on varying K

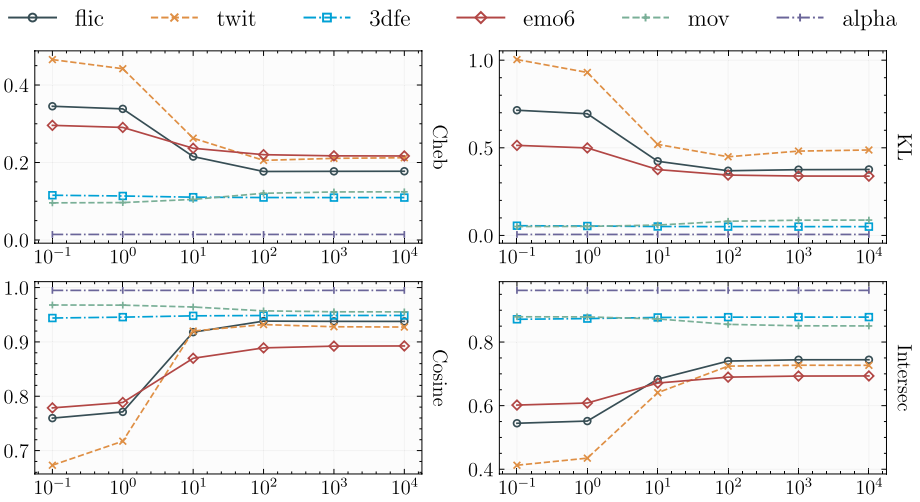


Fig. 9 Recovery performance on varying λ

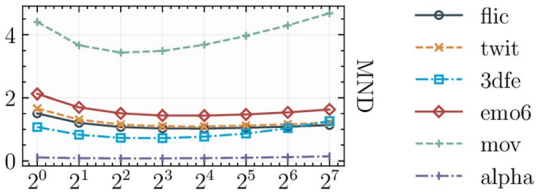


Fig. 10 MND values of different K

denotes the case without using the knowledge of instance neighbors. Overall, an appropriate increase on the number of nearest neighbors can always improve the recovery performance, which verifies that considering the instance neighbor relation is beneficial to recover the ground-truth label distributions. With a further observation, the recovery performance measured by Cheb, Canber, KL, Cosine and Intersec gets better first and then gets worse or stays unchanged when K increases, which is in line with our expectations. The reason why recovery performance is poor when K is large (beyond a certain threshold) is that the large K causes many non-neighboring instances to be treated as neighbors, thus making the neighborhood information very noisy.

As can be seen from Fig. 8, an appropriate K plays an important role in the recovery performance. Hence, we next discuss the selection criterion of K. We first define a metric called mean neighborhood divergence:

$$\mathrm{MND}(\mathbf{X}, K) = \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{x}_i - \frac{1}{K}\sum_{j\in\mathbb{B}_i}\boldsymbol{x}_j\right\|_2^2, \tag{34}$$

where $\mathbb{B}_i$ denotes the K-nearest neighbors of the instance $\boldsymbol{x}_i$ (excluding $\boldsymbol{x}_i$ itself). Roughly speaking, MND can measure the quality of K-nearest neighbors, whose lower value denotes the higher quality (Fig. 9). Further, we compute $\{\mathrm{MND}(\mathbf{X}, 2^k)\}_{k=0}^{7}$ for multiple datasets. The results are shown in Fig. 10. It can be seen that the optimal MND for each dataset appears roughly at $K = 2$, $K = 3$, or $K = 4$, and the MND values corresponding to

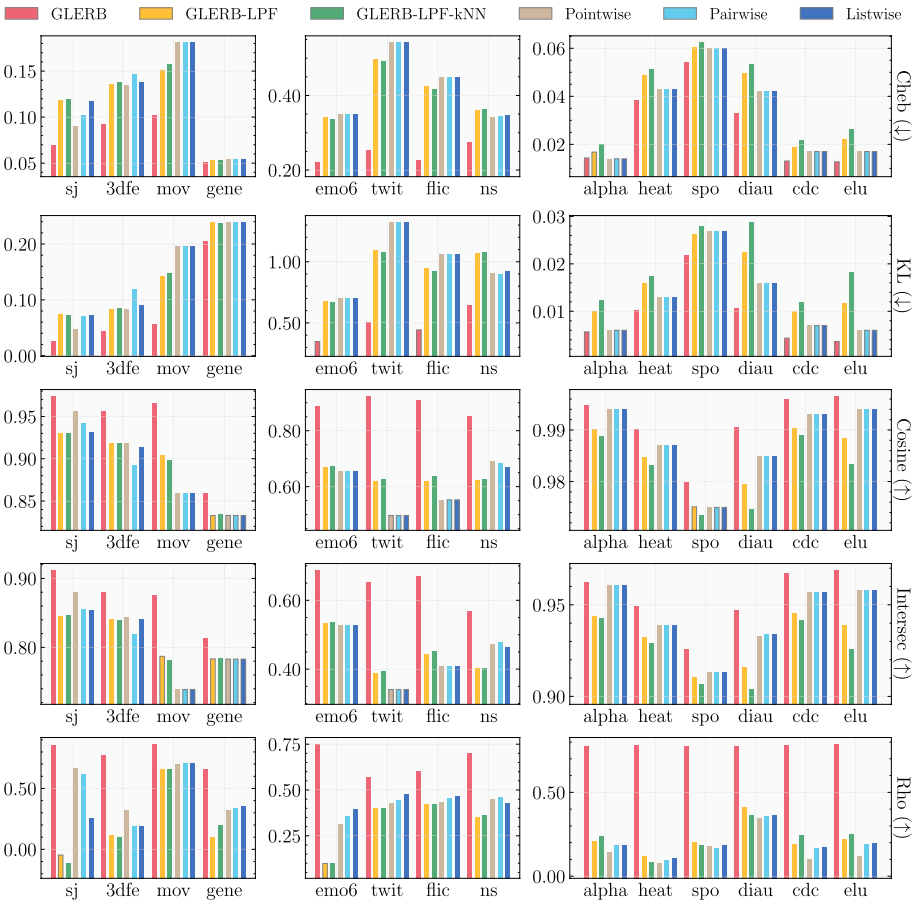


Fig. 11 Ablation study on 14 datasets

these three cases are almost identical. Besides, according to Fig. 8, the optimal recovery performance also falls in these three cases, and the trend of recovery performance is consistent with MND. Therefore, we can empirically choose the hyperparameter K according to the value of MND.

Weight of logical labels. Figure 9 shows the effect of increasing the weight of logical labels to the recovery performance. The values of λ are taken in $\{10^{-1}, 10^{0}, \cdots, 10^{4}\}$. Overall, the recovery performance gets better or stays unchanged when λ increases. This suggests that the larger contribution of neighborhood knowledge to model learning can lead to the better recovery performance.

6.3.2 Ablation study

To test the effectiveness our designed label propagation function and k-nearest neighbors generation, we use five reduced versions of our proposed method as comparison, which are denoted as GLERB-LPF, GLERB-LPF-kNN, Pointwise, Pairwise, and Listwise. "GLERB-LPF" uses an MLP to represent $f(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\omega})$ rather than $\varphi(\boldsymbol{y}_i, \tilde{\alpha}(\boldsymbol{x}_i; \boldsymbol{\omega}))$, i.e.,

the label co-occurrence information is not used and the label ranking is not preserved; this MLP has one hidden layer (with 128 hidden units), and its activation function is sigmoid function; Besides $f(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\omega})$, all other configurations of GLERB-LPF are the same as GLERB. "GLERB-LPF-KNN" is the version of GLERB-LPF with $K = 1$, i.e., the neighbor generation assumption is also ignored.

Actually, there are many off-the-shelf ranking loss functions in the field of LTR (learning to rank) that can be used to maintain label ranking. Therefore, we integrate different ranking loss functions with GLERB-LPF. The loss functions in LTR can be divided into three categories in general, i.e., pointwise loss, pairwise loss, and listwise loss. Hence, we choose a representative from each of these categories. Specifically, we choose RankNet loss (Burges et al., 2005) as pairwise loss, ordinal regression loss (Crammer & Singer, 2001) as pointwise loss, and negative log-likelihood of Plackett-Luce distribution (Guiver & Snelson, 2009) as listwise loss. The trade-off hyperparameter of each ranking loss term is selected from $\{10^{-5}, 10^{-4}, \cdots, 10^{5}\}$. Other configurations are the same as Ours-LPF. Ours-LPF that have been modified by these three ranking loss terms are denoted by "Pointwise", "Pairwise" and "Listwise", respectively.

Figure 11 shows the results. Overall, our designed label propagation function can effectively improves the recovery performance. Besides, it is worth mentioning that although label propagation function has little effect on the Cheb, KL, Cosine, and Intersec measures w.r.t. the Yeast datasets, it can significantly improve the Rho measure. In general, the models with ranking loss terms are better than both Ours-LPF and Ours-LPF-kNN on the Rho measure due to their focus on ranking, but still inferior to GLERB. It can be seen that our proposed GLERB algorithm significantly outperforms the models with ranking loss on all metrics.

7 Conclusion

In this paper, we propose a novel generative label enhancement model, and infer the latent label distribution by a restricted auto-encoding variational Bayes. Our method consists of a generation process and an inference process. The generation process not only mines the relationship between label confidence and observed data (i.e., features and logical labels), but also captures the neighborhood relation of instances in the feature space. The inference process is restricted by label co-occurrence relation thereby capturing label correlation, which can preserve the label ranking. Besides, we derive the evidence lower bound for the model, and introduce the Kumaraswamy distribution (which can be easily reparameterized) to ensure full differentiability of the optimization objective. Extensive experiments on the real-world datasets show the effectiveness of our method.

Appendix

Why choose Kumaraswamy distribution

Here we discuss in more detail the rationale for modeling the variational posterior as Kumaraswamy distribution. We briefly show several probability distributions with a support of (0, 1) as follow:

$$
\begin{aligned}
\text{Beta}(x|\alpha,\beta) &= \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\text{Beta}(\alpha,\beta)} \\
\text{Arcsine}(x) &= \frac{1}{\pi\sqrt{x(1-x)}} \\
\text{PERT}(x) &= \frac{(x-a)^{\alpha-1}(c-x)^{\beta-1}}{\text{B}(\alpha,\beta)(c-a)^{\alpha+\beta-1}} \\
\text{Logit-normal}(x) &= \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(\text{logit}(x)-\mu)^2}{2\sigma^2}} \frac{1}{x(1-x)} \\
\text{Kumaraswamy}(x|a,b) &= abx^{a-1}(1-x^a)^{b-1}
\end{aligned}
\tag{35}
$$

In addition to the probability distributions listed in Eq. (35), there are also some probability distribution with complicated form, such as truncated normal distribution, U-quadratic distribution, triangular distribution and trapezoidal distribution. Using these probability distributions to model variational posteriors may result in some problems. For example, the shape of "Arcsine" and "Uniform" cannot be controlled by the learnable function; "PERT" is difficult to apply the reparameterization trick; "Logit-normal" and truncated normal distribution do not have closed-form KL divergence w.r.t. the prior distribution (Beta distribution); Triangular distribution and trapezoidal distribution have segmented density functions, which are not conducive to gradient-based optimization; U-quadratic distribution is difficult to express the case that mean value is around 0.5. Among all these probability distributions, only the Kumaraswamy distribution is the most suitable for modeling the variational posterior of the variable with Beta prior because it not only allows easy implementation of the reparameterization trick but also has closed-form KL divergence w.r.t. the Beta prior.

Proof of Theorem 1

Suppose that $\boldsymbol{y}$ is any logical label vector, $\mathbf{P}$ is the symmetrically normalized Laplacian of any label co-occurrence graph, $\boldsymbol{v} = \varphi(\boldsymbol{y}, \alpha)$. Then we have $\boldsymbol{v} \simeq \boldsymbol{y}$ if $\alpha \in (0, \alpha_{\boldsymbol{y}}^{\star})$, where $\alpha_{\boldsymbol{y}}^{\star} = \inf(\mathbb{A} \cup \{1\})$,

$$
\mathbb{A} = \bigcup_{\substack{i \in \{i \mid y_i = 1\} \\ j \in \{j \mid y_j = 0\}}} \left\{ \frac{1}{2} \le \alpha < 1 \mid \varphi_i(\boldsymbol{y}, \alpha) = \varphi_j(\boldsymbol{y}, \alpha) \right\}.
$$

Proof Let $[\boldsymbol{u}]^{+} \triangleq \inf\{u_i \mid y_i = 1\}$ and $[\boldsymbol{u}]^{-} \triangleq \sup\{u_i \mid y_i = 0\}$, where $\boldsymbol{u}$ is a m-dimensional vector.

1) Proving that $\boldsymbol{v} \simeq \boldsymbol{y}$ holds for any $\alpha \in (0, 2^{-1})$. We consider the source of $\boldsymbol{v} = \varphi(\boldsymbol{y}, \alpha)$, i.e., $\boldsymbol{v}^{(t+1)} = \alpha\mathbf{P}\boldsymbol{v}^{(t)} + (1-\alpha)\boldsymbol{y}$. Suppose that $\boldsymbol{v}^{(t)} \in [0,1]^m$, we have $\mathbf{P}\boldsymbol{v}^{(t)} \in [0,1]^m$, and $\boldsymbol{v}^{(t+1)} = \alpha\mathbf{P}\boldsymbol{v}^{(t)} + (1-\alpha)\boldsymbol{y} \in [0,1]^m$. Since $\boldsymbol{v}^{(0)} = \boldsymbol{y}$, we have $\boldsymbol{v}^{(t)} \in [0,1]^m$ holds for any $t \ge 0$. Since $\boldsymbol{v}^{(t)} \in [0,1]^m$ implies $[\alpha\mathbf{P}\boldsymbol{v}^{(t)}]^{+} = 0$ and $[\alpha\mathbf{P}\boldsymbol{v}^{(t)}]^{-} = \alpha$, we can obtain $[\boldsymbol{v}^{(t+1)}]^{+} = 1-\alpha$ and $[\boldsymbol{v}^{(t+1)}]^{-} = \alpha$; since $0 < \alpha < 0.5$, we have $[\boldsymbol{v}^{(t+1)}]^{+} > [\boldsymbol{v}^{(t+1)}]^{-}$. Therefore, $[\boldsymbol{v}^{(t)}]^{+} > [\boldsymbol{v}^{(t)}]^{-}$ holds for any $t \ge 0$, i.e., $\boldsymbol{v} \simeq \boldsymbol{y}$.

2) Proving that $\boldsymbol{v} \simeq \boldsymbol{y}$ holds for any $\alpha \in [2^{-1}, \alpha_{\boldsymbol{y}}^{\star})$. Say there exists $\alpha_0 \in [2^{-1}, \alpha_{\boldsymbol{y}}^{\star})$ such that $v_i < v_j$, where $i \in \{i \mid y_i = 1\}$ and $j \in \{j \mid y_j = 0\}$, then there must exist $\alpha' \in [2^{-1}, \alpha_0)$ such

that $v_i = v_j$ (since $v_i > v_j$ always holds for $0 < \alpha < 2^{-1}$, and $\boldsymbol{v} = \varphi(\boldsymbol{y}, \alpha)$ is continuous), which contradicts the fact that $\alpha_{\boldsymbol{y}}^{\star}$ is the minimum of the roots (between 2^{-1} and 1) of all equations $v_i = v_j$ (where $(i,j) \in \{(i,j) \mid y_i = 1 \wedge y_j = 0\}$). Therefore, $\boldsymbol{v} \simeq \boldsymbol{y}$ holds for any $\alpha \in (2^{-1}, \alpha_{\boldsymbol{y}}^{\star})$.

Therefore, $\boldsymbol{v} \simeq \boldsymbol{y}$ holds for any $\alpha \in (0, \alpha_{\boldsymbol{y}}^{\star})$. □

Example of calculating $\alpha_{\boldsymbol{y}}^{\star}$ in Theorem 1

Suppose that $\boldsymbol{y} = [1, 0, 1]^{\top}$, Laplacian matrix $\mathbf{P}$ of the label co-occurrence graph can be factorized as:

$$\mathbf{P} = \begin{bmatrix} 2^{-1} & 3^{-1} & 6^{-1} \\ 3^{-1} & 2^{-1} & 6^{-1} \\ 6^{-1} & 6^{-1} & 2 \times 3^{-1} \end{bmatrix} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.17 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \mathbf{U}^{\top},$$

$$\text{where } \mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3] = \begin{bmatrix} -0.58 & -0.71 & -0.41 \\ -0.58 & 0.71 & -0.41 \\ -0.58 & 0 & 0.82 \end{bmatrix}.$$

Then, we have:

$$\begin{aligned} \frac{1}{1-\alpha}\boldsymbol{v} &= \frac{\boldsymbol{u}_1\boldsymbol{u}_1^{\top}\boldsymbol{y}}{1-\alpha} + \frac{\boldsymbol{u}_2\boldsymbol{u}_2^{\top}\boldsymbol{y}}{1-0.17\alpha} + \frac{\boldsymbol{u}_3\boldsymbol{u}_3^{\top}\boldsymbol{y}}{1-0.5\alpha} \\ &= \begin{bmatrix} 0.67(1-\alpha)^{-1} + 0.5(1-0.17\alpha)^{-1} - 0.17(1-0.5\alpha)^{-1} \\ 0.67(1-\alpha)^{-1} - 0.5(1-0.17\alpha)^{-1} - 0.17(1-0.5\alpha)^{-1} \\ 0.67(1-\alpha)^{-1} + 0(1-0.17\alpha)^{-1} + 0.33(1-0.5\alpha)^{-1} \end{bmatrix}. \end{aligned}$$

Finally, solving $v_1 = v_2$ or $v_3 = v_2$ both yield $\alpha \in \varnothing$ (when $0.5 < \alpha < 1$), then we have $\mathbb{A} = \varnothing, \alpha_{\boldsymbol{y}}^{\star} = \min(\mathbb{A} \cup \{1\}) = 1$.

Author contributions YL contributes to conceptualization, methodology, experiments, and writing. WL contributes to reviewing. HL contributes to reviewing. XJ contributes to conceptualization, methodology, and reviewing.

Funding This work was supported by the National Natural Science Foundation of China (62176123).

Availability of data and materials The datasets used in this paper are public. We have provided the corresponding reference of each dataset.

Code availability The code is available at https://github.com/yunan-lu/GLERB.

Declarations

Conflict of interest The authors declare that they have no known competing interests or conflict of interest that are relevant to the content of this paper.

Ethics approval Not applicable.

Consent to participate Not applicable. All experiments in this paper do not involve animals, plants, or human entities.

Consent for publication Not applicable. The paper does not include data or images requiring permissions.

References

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *International conference on machine learning* (pp. 89–96).

Chen, K., Kämäräinen, J., & Zhang, Z. (2016). Facial age estimation using robust label distribution. In *ACM international conference on multimedia* (pp. 77–81).

Crammer, K., & Singer, Y. (2001). Pranking with ranking. In *Advances in neural information processing systems*.

Deng, Z., Zhao, M., Liu, H., Yu, Z., & Feng, F. (2020). Learning neighborhood-reasoning label distribution (NRLD) for facial age estimation. In *IEEE international conference on multimedia and expo* (pp. 1–6).

Gao, B., Zhou, H., Wu, J., & Geng, X. (2018). Age estimation using expectation of label distribution learning. In *International joint conference on artificial intelligence* (pp. 712–718).

Gayar, N., Schwenker, F., & Palm, G. (2006). A study of the robustness of knn classifiers trained using soft labels. In *Artificial neural networks in pattern recognition* (pp. 67–80).

Geng, X., & Xia, Y. (2014). Head pose estimation based on multivariate label distribution. In *IEEE conference on computer vision and pattern recognition* (pp. 1837–1842).

Geng, X. (2016). Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering, 28*(7), 1734–1748.

Geng, X., Smith-Miles, K., & Zhou, Z.-H. (2013). Facial age estimation by learning from label distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*, 2401–2412.

Guiver, J., & Snelson, E. (2009). Bayesian inference for Plackett-Luce ranking models. In *International conference on machine learning* (pp. 377–384).

Guo, B., Han, S., Han, X., Huang, H., & Lu, T. (2021). Label confusion learning to enhance text classification models. In *AAAI conference on artificial intelligence* (pp. 12929–12936).

Hou, P., Geng, X., & Zhang, M.-L. (2016). Multi-label manifold learning. In *AAAI conference on artificial intelligence* (pp. 1680–1686).

Hou, P., Geng, X., Huo, Z., & Lv, J. (2017). Semi-supervised adaptive label distribution learning for facial age estimation. In *AAAI conference on artificial intelligence* (pp. 2015–2021).

Jebara, T. (2012). Machine learning: Discriminative and generative, Vol. 755.

Jia, X., Lu, Y., & Zhang, F. (2023). Label enhancement by maintaining positive and negative label relation. *IEEE Transactions on Knowledge and Data Engineering, 35*(2), 1708–1720.

Jiang, X., Yi, Z., & Lv, J. (2006). Fuzzy SVM with a new fuzzy membership function. *Neural Computing & Applications, 15*(3), 268–276.

Jia, X., Shen, X., Li, W., Lu, Y., & Zhu, J. (2023). Label distribution learning by maintaining label ranking relation. *IEEE Transactions on Knowledge and Data Engineering, 35*(2), 1695–1707.

Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE conference on computer vision and pattern recognition* (pp. 7482–7491).

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *International conference on learning representations*.

Kipf, T., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Kumaraswamy, P. (1980). A generalized probability density function for double-bounded random processes. *Journal of Hydrology, 46*(1), 79–88.

Li, Y., Zhang, M.-L., & Geng, X. (2015). Leveraging implicit relative labeling-importance information for effective multi-label learning. In *IEEE international conference on data mining* (pp. 251–260).

Ling, M.-L., & Geng, X. (2019). Indoor crowd counting by mixture of Gaussians label distribution learning. *IEEE Transactions on Image Processing, 28*, 5691–5701.

Liu, T., Venkatachalam, A., Sanjay Bongale, P., & Homan, C. (2019). Learning to predict population-level label distributions. In *World wide web conference* (pp. 1111–1120).

Liu, P., Wang, X., Wang, S., Ye, W., Xi, X., & Zhang, S. (2021). Improving embedding-based large-scale retrieval via label enhancement. In *Conference on empirical methods in natural language processing* (pp. 133–142).

Liu, X., Zhu, J., Li, Z., Tian, Z., Jia, X., & Chen, L. (2021). Unified framework for learning with label distribution. *Information Fusion, 75*, 116–130.

Liu, X., Zhu, J., Zheng, Q., Li, Z., Liu, R., & Wang, J. (2021). Bidirectional loss function for label enhancement and distribution learning. *Knowledge-Based System, 213*, 106690.

Lv, J.-Q., Xu, N., Zheng, R.-Y., & Geng, X. (2019). Weakly supervised multi-label learning via label enhancement. In *AAAI conference on artificial intelligence* (pp. 3101–3107).

Peng, K., Chen, T., Sadovnik, A., & Gallagher, A. (2015). A mixed bag of emotions: Model, predict, and transfer emotion distributions. In *IEEE conference on computer vision and pattern recognition* (pp. 860–868).
Ren, T., Jia, X., Li, W., Chen, L., & Li, Z. (2019). Label distribution learning with label-specific features. In *International joint conference on artificial intelligence* (pp. 3318–3324).
Shao, R., Geng, X., & Xu, N. (2018). Multi-label learning with label enhancement. In *IEEE international conference on data Mining*, 437–446.
Tan, C., Chen, S., Ji, G., & Geng, X. (2022). A novel probabilistic label enhancement algorithm for multi-label distribution learning. *IEEE Transactions on Knowledge and Data Engineering, 34*(11), 5098–5113.
Tang, H., Zhu, J., Zheng, Q., Wang, J., Pang, S., & Li, Z. (2020). Label enhancement with sample correlations via low-rank representation. In *AAAI conference on artificial intelligence* (pp. 5932–5939).
Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining, 3*(3), 1–13.
Wang, J., & Geng, X. (2019). Classification with label distribution learning. In *International joint conference on artificial intelligence* (pp. 3712–3718).
Wang, J., & Geng, X. (2021). Label distribution learning machine. In *International conference on machine learning*.
Wang, J., & Geng, X. (2021). Learn the highest label and rest label description degrees. In *IJCAI* (pp. 3097–3103).
Wang, J., Geng, X., & Xue, H. (2022). Re-weighting large margin label distribution learning for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(9), 5445–5459.
Wang, K., Xu, N., Ling, M.-L., & Geng, X. (2023). Fast label enhancement for label distribution learning. *IEEE Transactions on Knowledge and Data Engineering, 35*(2), 1502–1514.
Weerasooriya, T., Liu, T., & Homan, C. (2020). Neighborhood-based pooling for population-level label distribution learning. In *European conference on artificial intelligence* (pp. 490–497).
Wen, X., Li, B., Guo, H., Liu, Z., Hu, G., Tang, M., & Wang, J. (2020). Adaptive variance based label distribution learning for facial age estimation. In *European conference on computer vision* (pp. 379–395).
Wen, T., Li, W., Chen, L., & Jia, X. (2022). Semi-supervised label enhancement via structured semantic extraction. *International Journal of Machine Learning and Cybernetics, 13*, 1131–1144.
Xu, N., Lv, J., & Geng, X. (2019). Partial label learning via label enhancement. In *AAAI conference on artificial intelligence* (pp. 5557–5564).
Xu, N., Shu, J., Liu, Y., & Geng, X. (2020). Variational label enhancement. In *International conference on machine learning* (pp. 10597–10606).
Xu, N., Tao, A., & Geng, X. (2018). Label enhancement for label distribution learning. In *International joint conference on artificial intelligence* (pp. 2926–2932).
Xu, N., Liu, Y.-P., & Geng, X. (2021). Label enhancement for label distribution learning. *IEEE Transactions on Knowledge and Data Engineering, 33*, 1632–1643.
Yang, J., Sun, M., & Sun, X. (2017). Learning visual sentiment distributions via augmented conditional probability neural network. In *AAAI conference on artificial intelligence* (pp. 224–230).
Zeng, X., Chen, Q., Chen, S., & Zuo, J. (2021). Emotion label enhancement via emotion wheel and lexicon. *Mathematical Problems in Engineering* (pp. 1–11).
Zhang, Q.-W., Zhong, Y., & Zhang, M.-L. (2018). Feature-induced labeling information enrichment for multi-label learning. In *AAAI conference on artificial intelligence* (pp. 4446–4453).
Zhang, Z., Wang, M.-L., & Geng, X. (2015). Crowd counting in public video surveillance by label distribution learning. *Neurocomputing, 166*, 151–163.
Zhang, M.-L., & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering, 26*, 1819–1837.
Zheng, Q., Zhu, J., Tang, H., Liu, X., Li, Z., & Lu, H. (2023). Generalized label enhancement with sample correlations. *IEEE Transactions on Knowledge and Data Engineering, 35*(1), 482–495.
Zhou, D., Zhang, X., Zhou, Y., Zhao, Q., & Geng, X. (2016). Emotion distribution learning from texts. In *Conference on empirical methods in natural language processing* (pp. 638–647).
Zhu, W., Jia, X., & Li, W. (2020). Privileged label enhancement with multi-label learning. In *International joint conference on artificial intelligence* (pp. 2376–2382).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.