A Probabilistic Framework for Zero-Shot Multi-Label Learning

Abhilash Gaure, Aishwarya Gupta, Vinay Kumar Verma, Piyush Rai
{agaure,aish,vkverma,piyush}@cse.iitk.ac.in
Department of Computer Science and Engineering
IIT Kanpur, India

Abstract

We present a probabilistic framework for multi-label learning for the setting when the test data may require predicting labels that were not available at training time (i.e., the zero-shot learning setting). We develop a probabilistic model that leverages the co-occurrence statistics of the labels via a joint generative model for the label matrix (which denotes the label presence/absence for each training example) and for the *label co-occurrence matrix* (which denotes how many times a pair of labels co-occurs with each other). In addition to handling the unseen labels at test time, leveraging the co-occurrence information may also help in the standard multi-label learning setting, especially if the number of training examples is very small and/or the label matrix of training examples has a large fraction of missing entries. Our experimental results demonstrate the efficacy of our model in handling unseen labels.

1 Introduction

Multi-label learning (Gibaja and Ventura, 2015, 2014) refers to the problem of annotating an object with a subset of labels from a large label vocabulary (Prabhu and Varma, 2014; Jain et al., 2016; Babbar and Schölkopf, 2017). Unlike standard classification paradigms such as binary or multi-class classification, which associate each object with a *single* label (binary/discrete), in multi-label learning, each object is associated with a binary *label vector* (potentially very large), denoting the presence/absence of each label. Multi-label learning has a wide range of applications in diverse domains such as computer vision (Wang et al., 2016), computational advertising and recommender systems (Prabhu and Varma, 2014; Jain et al., 2016), etc.

To handle the large number of labels and to leverage the label correlations, existing multi-label learning algorithms usually employ certain structural assumptions on the binary label matrix. One such commonly used assumption is the low-rank assumption on the label matrix Yu et al. (2014); Rai et al. (2015). In a probabilistic setting, this is equivalent to modeling the label matrix using a generative latent factor model (Jain et al., 2017) which assumes that each example n and each label ℓ are associated with latent factors $\boldsymbol{u}_n \in \mathbb{R}^K$ and $\boldsymbol{v}_\ell \in \mathbb{R}^K$, respectively, and the *relevance* of the label ℓ for the example n is based on how *similar* $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$ are.

One important challenge for multi-label learning that has not been adequately addressed so far is the setting where not all the labels are available at the time of training the model (Zhang et al., 2016; Mensink et al., 2014). In many real-world multi-label learning problems, e.g., in recommender systems, the set of possible labels (e.g., items/products) keeps on increasing and it may not be feasible to re-train the model every time new labels are added to the set of possible labels. This requires collecting fresh training data and annotating it with both existing as well as new labels. This process may be expensive and time-consuming.

To address this issue, we present a generative framework that is based on leveraging the co-occurrence statistics of the *seen* labels (i.e., those present in the training data) with all the labels (both seen and unseen labels). These co-occurrence statistics can usually be obtained from an external source (e.g., a text corpus, such as Wikipedia) and are in form of *counts* of how many times a pair of labels is found to co-occur with each other. We use the train data label matrix and the label co-occurrence matrix to learn the embeddings (in form of low-dimensional latent factors) of both seen and unseen labels using a generative model. Our generative framework employs a negative binomial latent factor model for modeling the (count-valued) co-occurrence of each pair of labels.

We further combine this negative binomial latent factor model with another logistic-Bernoulli latent factor model that is learned exclusively on the binary label matrix of the training data. In order to share information between both generative models, we maintain another set of latent factors (defined only for the seen labels), that are shared between both latent factor models.

We leverage the first set of latent factors (defined for both seen and unseen classes), learned exclusively using the label co-occurrences, and the second set of *shared* latent factors (defined for only seen classes), learned using both label co-occurrences and the binary label matrix, to then learn the label predictors for each of the unseen labels. These label predictors can be used at test time to predict the presence/absence of each of the unseen labels.

In addition to its ability to leverage the label co-occurrences to handle the unseen labels at test time, one of the appealing aspects of our framework is its simplicity. Our generative framework admits a very simple inference procedure, for which both fully Bayesian inference or point estimation based inference can be used. Moreover, using data-augmentation techniques (Polson et al., 2013), our framework enjoys full local-conjugacy, which enables us to develop simple Gibbs sampling, variational inference, or Expectation Maximization (EM) algorithms, for doing inference in our model. In particular, we propose an efficient EM algorithm for our framework, which is extremely simple to implement, with each parameter update only requiring solving a simple weighted ridge-regression style problem.

2 The Model

2.1 Background and Notations

In multi-label learning, the goal is to learn a model to predict, for any example $\boldsymbol{x} \in \mathbb{R}^D$, a binary label vector $\boldsymbol{y} \in \{0,1\}^L$, which denotes the presence/absence of L binary-valued labels for that example. We consider a particularly challenging setting where the training data may only have presence/absence information available for $L_s < L$ labels, with $L = L_s + L_u$, where the subscripts s and u refer to the *seen* and *unseen* labels, respectively. On the other hand, the set of possible labels for test examples can include both seen as well as unseen labels. Our goal is to learn a model that is able to predict the presence/absence for both seen and unseen labels for the test examples.

To learn the model, we assume that we are given N training examples as $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $\boldsymbol{y}_n \in \{0,1\}^{L_s}$, $n = 1, \ldots, N$. We define $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ to denote the $N \times D$ feature matrix and $\mathbf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ to denote the $N \times L_s$ binary label matrix. Note again that the training data only has the label information for the seen labels.

In addition to the $N \times L_s$ label matrix $\mathbf{Y}$ for the training examples, we also assume that we have access to the label co-occurrence information of each seen label with *all* the labels (both seen and unseen). This information can usually be obtained *unsupervisedly* from an external corpus (e.g., Wikipedia). We assume this information is available in form of an $L_s \times L$ *count-valued* matrix $\mathbf{M}$ where $m_{\ell\ell'}$ denotes the number of times a seen class label $\ell \in \{1, \ldots, L_s\}$ has been found to co-occur with a seen/unseen class label $\ell' \in \{1, \ldots, L\}$ in the external source of data used to compute the label co-occurrences. Our goal will be to leverage the label co-occurrence information to enable our model to also predict the presence/absence of the L_u unseen labels for which have no information in the label matrix $\mathbf{Y}$ of the training data.

Also note that even when $L_s = L$, i.e., when all the labels are available at training time, using the $L \times L$ label co-occurrence matrix $\mathbf{M}$ may still be useful if a large fraction of entries in the label matrix $\mathbf{Y}$ are missing, as is often the case in many multi-label learning problems with partial label information (Yu et al., 2014).

2.2 Modeling the Label Matrix

Given the training data in form of the $N \times D$ feature matrix $\mathbf{X}$ and the $N \times L_s$ label matrix $\mathbf{Y}$ (possibly only partially observed), we assume that each *observed* entry $y_{n\ell} \in \{0,1\}$ of $\mathbf{Y}$ is generated from a latent factor model. Specifically, we assume that each input $\boldsymbol{x}_n$, $n = 1, \ldots, N$ is associated with an *input latent factor* $\boldsymbol{u}_n \in \mathbb{R}^K$ and each label $\ell = 1, \ldots, L_s$ is associated with a *label latent factor* $\boldsymbol{v}_\ell \in \mathbb{R}^K$. These latent factors can be thought of as low-dimensional embeddings. Conditioned on these latent factors, we assume the following model for each observed binary label $y_{n\ell}$ in $\mathbf{Y}$

$$p(y_{n\ell} = 1 | \boldsymbol{u}_n, \boldsymbol{v}_\ell) = \frac{1}{1 + \exp(-\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)} \qquad (1)$$

We also assume the following prior distributions on the latent factors $\boldsymbol{u}_n, n = 1, \ldots, N$ and $\boldsymbol{v}_\ell, \ell = 1, \ldots, L_s$.

$$\begin{aligned} p(\boldsymbol{u}_n | \boldsymbol{x}_n, \mathbf{W}) &= \mathcal{N}(\boldsymbol{u}_n | \mathbf{W}\boldsymbol{x}_n, \lambda_u^{-1}\mathbf{I}) && (2) \\ p(\boldsymbol{v}_\ell) &= \mathcal{N}(\boldsymbol{v}_\ell | \mathbf{0}, \lambda_v^{-1}\mathbf{I}) && (3) \end{aligned}$$

Note that we condition the latent factors $\boldsymbol{u}_n \in \mathbb{R}^K$ on the corresponding inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ by assuming that mean of the Gaussian prior on $\boldsymbol{u}_n$ depends on $\boldsymbol{x}_n$ via a regression model. Specifically, in Eq. 2, $\mathbf{W}$ is a $K \times D$ matrix of regression weights which maps the feature vector $\boldsymbol{x}_n$ to the latent factors $\boldsymbol{u}_n$. Conditioning the latent factors

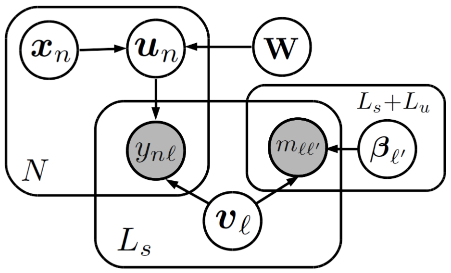

Figure 1: Our generative model in the plate notation. Note: Hyperparameters not shown for brevity.

this way also allows us to predict the latent factors of any test example $\boldsymbol{x}_*$ as $\mathbb{E}[\boldsymbol{u}_*|\boldsymbol{x}_*\mathbf{W}] \approx \mathbf{W}\boldsymbol{x}_*$, which is required to predict the label vector for this test example.

2.3 Modeling Label Co-occurrences

To incorporate the label co-occurrence information in our model, we assume another generative model for the $L_s \times L$ label co-occurrence matrix $\mathbf{M}$ and combine this generative model with the generative model for the label matrix $\mathbf{Y}$, that we described in Sec. 2.2.

To model the label co-occurrence matrix $\mathbf{M}$, we assume another set of label latent factors for each seen/unseen label, $\{\boldsymbol{\beta}_{\ell'}\}_{\ell'=1}^{L}$, with $\boldsymbol{\beta}_{\ell'} \in \mathbb{R}^K$ having a Gaussian prior

$$p(\beta_{\ell'}) = \mathcal{N}(\beta_{\ell'}|\mathbf{0}, \lambda_\beta^{-1}\mathbf{I}) \tag{4}$$

We can then model each count-valued entry $m_{\ell\ell'}$, $\ell = 1, \ldots, L_s$, $\ell' = 1, \ldots, L$, of $\mathbf{M}$ using a negative binomial distribution, as follows

$$p(m_{\ell\ell'}|r, p_{\ell\ell'}) = \text{NB}(m_{\ell\ell'}|r, p_{\ell\ell'}) \tag{5}$$

where
$$p_{\ell\ell'} = \frac{1}{1+\exp(-\boldsymbol{v}_\ell^\top \beta_{\ell'})} \tag{6}$$

and r is the overdispersion parameter of the negative binomial distribution,where the negative binomial is

$$\text{NB}(m = k|r, p) \propto (1-p)^r p^k$$

Fig. 1 shows the overall generative model. Note that the label latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$, learned exclusively for the seen labels, are shared by both $\mathbf{Y}$ and $\mathbf{M}$, whereas the label latent factors $\{\beta_{\ell'}\}_{\ell'=1}^{L}$ which are defined for both seen and unseen are only used to model $\mathbf{M}$.

One particular benefit of modeling the label co-occurrences via a negative binomial is the fact that it better models the overdispersion in the distribution of count-valued data, as compared to other distributions such as Poisson. Another benefit is that the form of the negative binomial enables us to develop an efficient algorithm based on Pólya-gamma augmentation (Polson et al., 2013) (Sec. 4), which results in very simple ridge-regression like update equations for the latent factors.

3 Handling Unseen Labels at Test Time

The model described in Sec. 2 is able to leverage the label co-occurrences, with the latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$, shared by $\mathbf{Y}$ and $\mathbf{M}$, acting as a "conduit". This leads to improved estimates of the model parameters, especially when the label matrix $\mathbf{Y}$ might have a large fraction of its entries as missing. However, this model cannot be used to predict the presence/absence of unseen labels for any test example. To see this, note that although our model learns the label latent factors $\{\beta_{\ell'}\}_{\ell'=1}^{L}$ for seen as well as unseen labels, the $\{\beta_{\ell'}\}_{\ell'=1}^{L}$ only appear in the generative model for the label co-occurrence matrix $\mathbf{M}$ (cf., Fig. 1 and Eq. 5). Therefore these cannot be used to predict the unseen labels $\ell = L_s + 1, \ldots, L$ in the label vector $\boldsymbol{y}_*$ of a test example $\boldsymbol{x}_*$.

In particular, note that, as per Eq. 1, predicting each binary label $y_{*\ell}$ in the label vector $\boldsymbol{y}_* \in \{0,1\}^L$ associated with the test input $\boldsymbol{x}_* \in \mathbb{R}^D$ would require computing

$$p(y_{*\ell} = 1|\boldsymbol{u}_*, \boldsymbol{v}_\ell) = \frac{1}{1+\exp(-\boldsymbol{u}_*^\top \boldsymbol{v}_\ell)} \tag{7}$$

where the test input's latent factors $\boldsymbol{u}_*$ can be approximated using the test input's feature vector $\boldsymbol{x}_* \in \mathbb{R}^D$ and the $K \times D$ regression weight matrix $\mathbf{W}$ as $\boldsymbol{u}_* \approx \mathbf{W}\boldsymbol{x}_*$.

However, the label latent factors $\boldsymbol{v}_\ell$ required in the evaluation of $p(y_{*\ell} = 1|\boldsymbol{u}_*, \boldsymbol{v}_\ell)$ in Eq. 7 are only available for the *seen* class labels since these are learned using only the $N \times L_s$ label matrix of the training data. In order to compute $p(y_{*\ell} = 1|\boldsymbol{u}_*, \boldsymbol{v}_\ell)$ to predict the presence/absence of labels not seen at training time, we first need an estimate of the latent factors $\boldsymbol{v}_\ell$ for all the unseen labels $\ell = L_s + 1, \ldots, L$. To handle this, we augment our framework with another regression model that will allow us to estimate $\{\boldsymbol{v}_\ell\}_{\ell=L_s+1}^{L}$ for the unseen labels.

To elaborate our regression-based approach for doing this, note that our model, as described in Sec. 2, learns *two* types of latent factors for each label. We will refer to these as *type-1* latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$ that are learned for only the seen labels, and *type-2* latent factors $\{\beta_{\ell'}\}_{\ell'=1}^{L}$ are learned for both seen as well as unseen labels.

To predict type-1 latent factors $\{\boldsymbol{v}_\ell\}_{\ell=L_s+1}^{L}$ for unseen labels, we learn a regression model that learns to map the type-2 latent factors to type-1 latent factors using the obtained estimates of type-1 and type-2 latent factors of the *seen* labels. Specifically, given the estimates $\{\hat{\boldsymbol{v}}_\ell\}_{\ell=1}^{L_s}$ and $\{\hat{\boldsymbol{\beta}}_\ell\}_{\ell=1}^{L_s}$ of the type-1 and type-2 latent factors of seen labels learned by our generative model, we learn a regression function f, s.t. $\hat{\boldsymbol{v}}_\ell \approx f(\hat{\boldsymbol{\beta}}_\ell), \ell = 1, \ldots, L_s$, and then use the learned regression function f to predict the type-1 latent factors $\{\boldsymbol{v}_\ell\}_{\ell=L_s+1}^{L}$ for the unseen labels using their corresponding type-2 latent factors.

The function $f : \mathbb{R}^K \to \mathbb{R}^K$ can be modeled as a linear regression model or a nonlinear model (e.g., kernel regression or a deep neural net). In this paper, we consider a linear model defined by a $K \times K$ regression matrix $\mathbf{\Psi}$

$$\arg\min_{\mathbf{\Psi}} \sum_{\ell=1}^{L_s} ||\hat{\boldsymbol{v}}_\ell - \mathbf{\Psi}\hat{\boldsymbol{\beta}}_\ell||^2 + \lambda_{\mathbf{\Psi}}||\mathbf{\Psi}||^2 \tag{8}$$

However, the linear model $\mathbf{\Psi}$ can be replaced by any nonlinear regression model as well.

4 Inference

Inference in our model requires estimating the latent factors $\mathbf{U} = \{\boldsymbol{u}_n\}_{n=1}^N$ of the training examples, the label latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$ of the seen labels, the label latent factors $\{\beta_{\ell'}\}_{\ell'=1}^{L}$ of the seen and unseen labels, and the regression matrix $\mathbf{W}$. We learn the regression matrix $\mathbf{\Psi}$ by solving Eq. 8 once we have estimated $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$ and $\{\beta_{\ell'}\}_{\ell'=1}^{L}$. Note however that learning $\mathbf{\Psi}$ can also be integrated with learning of the rest of the model parameters but we found that learning it separately in the end does almost similarly and therefore we follow this strategy.

Both fully Bayesian inference as well as point estimation is possible under our generative framework. Fully Bayesian posterior inference for our probabilistic framework is, in general, intractable but a number of approximate inference methods such as MCMC (Andrieu et al., 2003) or variational inference (Blei et al., 2017) can be used. Note that our model is not *natively* conjugate due to the presence of the logistic-Bernoulli likelihood (for binary labels - Eq. 1) and negative binomial (for label co-occurrences - Eq. 5) and the Gaussian priors for the model parameters. However, we are able to leverage the recently developed Pólya-gamma augmentation (Polson et al., 2013) technique to handle these non-conjugate likelihoods and are able to transform these likelihoods into Gaussian likelihoods, when conditioned on auxiliary variables. This enables us to develop efficient Gibbs sampling, variational inference, or expectation maximization algorithm for doing inference.

The Pólya-gamma augmentation technique (Polson et al., 2013) is based on the following identity

$$\frac{(\exp(\psi)^a}{(1+\exp(\psi))^b} = 2^{-b}\exp(\kappa\psi)\int_0^\infty \exp\left(-\omega\psi^2/2\right)p(\omega)d\omega$$

where $\kappa = a - b/2$ and $\omega|\psi \sim \text{PG}(b, \psi)$, and PG denotes the Pólya-Gamma distribution (Polson et al., 2013). This identity allows us to write any likelihood of the form $\frac{(\exp(\psi)^a}{(1+\exp(\psi))^b}$ (e.g., logistic-Bernoulli, binomial, negative-binomial) as a Gaussian distribution, when conditioned on another random variable $\omega|\psi \sim \text{PG}(b, \psi)$.

We can thus re-express the logistic-Bernoulli likelihood $p(y_{n\ell} = 1|\boldsymbol{u}_n, \boldsymbol{v}_\ell) = \frac{\exp(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)}{1+\exp(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)}$ on the binary labels as a Gaussian, when conditioning on PG auxiliary variables drawn as $\omega_{n\ell} \sim \text{PG}(1, \boldsymbol{u}_n^\top \boldsymbol{v}_\ell)$. In particular, the distribution of $\xi_{n\ell} = \boldsymbol{u}_n^\top \boldsymbol{v}_\ell$, conditioned on $\omega_{n\ell}$, becomes a Gaussian

$$p(\xi_{n\ell}|\omega_{n\ell}) \propto \exp\left(-\kappa_{n\ell}^{(y)}\xi_{n\ell} - \omega_{n\ell}\xi_{n\ell}^2\right) \tag{9}$$

where $\kappa_{n\ell}^{(y)} = y_{n\ell} - 0.5$. This likelihood, combined with the Gaussian priors on the latent factors $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$, results in Gaussian posteriors for the $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$.

Likewise,the negative binomial likelihood model

$$\begin{aligned} p(m_{\ell\ell'}|r, p_{\ell\ell'}) &= \text{NB}(m_{\ell\ell'}|r, p_{\ell\ell'}) && (10) \\ &\propto (1 - p_{\ell\ell'})^r p_{\ell\ell'}^{m_{\ell\ell'}} && (11) \end{aligned}$$

with $p_{\ell\ell'} = \frac{1}{1+\exp(-\boldsymbol{v}_\ell^\top \beta_{\ell'})}$, can be written in the form $\frac{(\exp(\psi)^a}{(1+\exp(\psi))^b}$, with $a = m_{\ell\ell'}$ and $b = r + m_{\ell\ell'}$.

The Pólya-gamma technique can be applied to this likelihood as well (Polson et al., 2013), which makes it a Gaussian, when conditioned on another set of PG auxiliary variables $\tau_{\ell\ell'}$. In particular, the distribution of $\gamma_{\ell\ell'} = \boldsymbol{v}_\ell^\top \boldsymbol{\beta}_{\ell'}$, conditioned on $\tau_{\ell\ell'}$, becomes a Gaussian

$$p(\gamma_{\ell\ell'}|\tau_{n\ell}) \propto \exp\left(-\kappa_{\ell\ell'}^{(m)}\gamma_{\ell\ell'} - \tau_{\ell\ell'}\gamma_{\ell\ell'}^2\right) \tag{12}$$

where $\kappa_{\ell\ell'}^{(m)} = \frac{(m_{\ell\ell'}+r)}{2}$. This likelihood, combined with the Gaussian priors on the latent factors $\boldsymbol{v}_n$ and $\boldsymbol{\beta}_{\ell'}$, results in Gaussian posteriors for the $\boldsymbol{v}_\ell$ and $\boldsymbol{\beta}_{\ell'}$.

The Pólya-gamma technique therefore enables us to derive closed-form posteriors on all the latent variables and we can easily perform Gibbs sampling. However, Gibbs sampling can be slow to converge in practice, especially when dealing with large data sets. Another alternative would be to develop a variational Bayes (VB) inference algorithm, which is computationally more efficient. Here we take the approach related to VB inference and develop a fast expectation maximization (EM) algorithm for this model (note that the EM algorithm can also be easily extended to a full VB algorithm).

The EM algorithm we propose also benefits from the Pólya-gamma augmentation because having Gaussian likelihoods and Gaussian priors implies that, *given* the Pólya-gamma variables (estimated in the E step) the maximum-a-posteriori (MAP) objective function to be optimized in the M step will have a structure like a least squares problem, which can be solved efficiently (Scott and Sun, 2013) in closed form.

4.1 The EM Algorithm

The (conditional) EM algorithm for our model alternates between computing the *expectations* of the Pólya-gamma variables $\omega_{n\ell}$ and $\tau_{\ell\ell'}$, where $n = 1, \ldots, N; \ell = 1, \ldots, L_s; \ell' = 1, \ldots, L$, in the E step, and then using these expectations to estimate the other model parameters $\boldsymbol{u}_n$, $\boldsymbol{v}_\ell$, $\boldsymbol{\beta}_{\ell'}$, and $\mathbf{W}$, in the M step.

The E Step: The E step involves computing the expectations of the latent variables $\omega_{n\ell}$ and $\tau_{\ell\ell'}$, given the current values of the other model parameters $\boldsymbol{u}_n$, $\boldsymbol{v}_\ell$, $\boldsymbol{\beta}_{\ell'}$, and $\mathbf{W}$ estimated in the previous M step. The E step update equations are given below:

- Expectations of Pólya-gamma variables $\{\omega_{n\ell}\}$, $\forall n, \ell$ are available in closed form and are given by the following expressions (Scott and Sun, 2013)

$$\hat{\omega}_{n\ell} = \mathbb{E}[\omega_{n\ell}|\xi_{n\ell}] = \frac{1}{2\xi_{n\ell}} \tanh\left(\frac{\xi_{n\ell}}{2}\right) \tag{13}$$

  where $\xi_{n\ell} = \boldsymbol{u}_n^\top \boldsymbol{v}_\ell$ is computed using the estimates of $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$ from the previous M step.

- Likewise, expectations of Pólya-gamma variables $\tau_{\ell\ell'}$, $\forall \ell, \ell'$ are also available in closed form and are given by the following expressions (Scott and Sun, 2013)

$$\hat{\tau}_{\ell\ell'} = \mathbb{E}[\tau_{\ell\ell'}|\gamma_{\ell\ell'}] = \frac{(m_{\ell\ell'} + r)}{2\gamma_{\ell\ell'}} \tanh\left(\frac{\gamma_{\ell\ell'}}{2}\right) \tag{14}$$

  where $\gamma_{\ell\ell'} = \boldsymbol{v}_\ell^\top \boldsymbol{\beta}_{\ell'}$ is computed using the estimates of $\boldsymbol{v}_\ell$ and $\boldsymbol{\beta}_{\ell'}$ and from the previous M step.

The M Step: Given the expectations computed in the E step, the M step minimizes the following negative expected complete data log-likelihood $\mathcal{Q}(\mathbf{U}, \mathbf{V}, \boldsymbol{\mu})$, w.r.t. the rest of the model parameters, namely $\mathbf{U} = \{\boldsymbol{u}_n\}_{n=1}^N$, $\mathbf{V} = \{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$, $\mathbf{B} = \{\beta_{\ell'}\}_{\ell'=1}^{L}$, and $\mathbf{W}$.

$$\mathcal{Q}(\mathbf{U}, \mathbf{V}, \mathbf{B}, \mathbf{W}) = \sum_{n,\ell} \frac{(\kappa_{n\ell}^{(y)} - \hat{\omega}_{n\ell}\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)^2}{\hat{\omega}_{n\ell}}$$

$$\sum_{\ell,\ell'} \frac{(\kappa_{\ell\ell'}^{(m)} - \hat{\tau}_{\ell\ell'}\boldsymbol{v}_\ell{}^\top \boldsymbol{\beta}_{\boldsymbol{\ell'}})^2}{\hat{\tau}_{\ell\ell'}} + \lambda_u \sum_{n=1}^N ||\boldsymbol{u}_n - \mathbf{W}\boldsymbol{x}_n||^2$$

$$+ \lambda_v \sum_{\ell=1}^{L_s} ||\boldsymbol{v}_\ell||^2 + \lambda_\beta \sum_{\ell'=1}^{L} ||\boldsymbol{\beta}_{\ell'}||^2 + \lambda_w ||\mathbf{W}||^2 \tag{15}$$

Note that the objective function in Eq. 15 can be broken down into a bunch of independent *weighted* least squares problems (where the weights depend on the expectations of the PG variables estimated in the E step), with ℓ_2 regularizers on the model parameters. Therefore minimizing $\mathcal{Q}(\mathbf{U}, \mathbf{V}, \mathbf{B}, \mathbf{W})$ w.r.t. $\mathbf{U}, \mathbf{V}, \mathbf{B}, \mathbf{W}$ reduces to solving these weighted, regularized least squares problems. We take an alternating minimization scheme and solve for one variable at a time, treating all other variables as fixed. This yields closed-form updates for each of these variables. Denoting the expectations $\mathbb{E}[\omega_{n\ell}|\xi_{n\ell}] = \hat{\omega}_{n\ell}$ and $\mathbb{E}[\tau_{\ell\ell'}|\gamma_{\ell\ell'}] = \hat{\tau}_{\ell\ell'}$, these update equations for these variables will be as follows:

- Estimating the latent factors $\{\boldsymbol{u}_n\}_{n=1}^N$ is a weighted ridge-regression problem with the updates

$$\boldsymbol{u}_n = \boldsymbol{\Sigma}_{\boldsymbol{u}_n} \left(\sum_{\ell=1}^{L_s} \kappa_{n\ell}^{(y)} \boldsymbol{v}_\ell + \lambda_u \mathbf{W}\boldsymbol{x}_n \right) \tag{16}$$

  where $\boldsymbol{\Sigma}_{\boldsymbol{u}_n} = (\sum_{\ell=1}^{L_s} \hat{\omega}_{n\ell}\boldsymbol{v}_\ell\boldsymbol{v}_\ell^\top + \lambda_u \mathbf{I}_K)^{-1}$. Note that the updates for $\{\boldsymbol{u}_n\}_{n=1}^N$ are all independent of each other and are easily parallelizable.

- Likewise, estimating the latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^{L_s}$ is a weighted ridge-regression problem with the updates

$$\boldsymbol{v}_\ell = \boldsymbol{\Sigma}_{\boldsymbol{v}_\ell} \left(\sum_{n=1}^{N} \kappa_{n\ell}^{(y)} \boldsymbol{u}_n + \sum_{\ell'=1}^{L} \kappa_{\ell\ell'}^{(m)} \boldsymbol{\beta}_{\ell'} \right) \tag{17}$$

  where $\boldsymbol{\Sigma}_{\boldsymbol{v}_\ell} = (\mathbf{A} + \lambda_v \mathbf{I}_K)^{-1}$, where $\mathbf{A} = \sum_{n=1}^N \hat{\omega}_{n\ell}\boldsymbol{u}_n\boldsymbol{u}_n^\top + \sum_{\ell'=1}^{L} \hat{\tau}_{\ell\ell'}\boldsymbol{\beta}_{\boldsymbol{\ell'}}\boldsymbol{\beta}_{\boldsymbol{\ell'}}^\top$. Again, note that the updates for $\{\boldsymbol{v}_n\}_{\ell=1}^{L}$ are all independent of each other and are easily parallelizable.

- The updates for estimating the latent factors $\{\boldsymbol{\beta}_{\boldsymbol{\ell'}}\}_{\ell'=1}^{L}$ have the form

$$\boldsymbol{\beta}_{\boldsymbol{\ell'}} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}_{\boldsymbol{\ell'}}} \left(\sum_{\ell=1}^{L_s} \kappa_{\ell\ell'}^{(m)} \boldsymbol{v}_\ell \right) \tag{18}$$

  where $\boldsymbol{\Sigma}_{\boldsymbol{\beta}_{\boldsymbol{\ell'}}} = (\sum_{\ell=1}^{L_s} \hat{\tau}_{\ell\ell'}\boldsymbol{v}_\ell\boldsymbol{v}_\ell^\top + \lambda_\beta \mathbf{I}_K)^{-1}$. The updates for $\{\boldsymbol{\beta}_{\boldsymbol{\ell'}}\}_{\ell'=1}^{L}$ are all independent of each other and are easily parallelizable.

Updates for the regression matrices $\mathbf{W}$ and $\boldsymbol{\Psi}$ are also available in simple closed form. For brevity, we skip it here and provide the update equations in the appendix.

Although the inference algorithm described here works in a batch fashion, it is easy to extend the algorithm to operate in an online fashion using an online EM algorithm (Cappé and Moulines, 2009) where, in each iteration, we only process a small minibatch of training examples. To see this, note that the updates of the latent factors $\boldsymbol{v}_\ell$ depend on sufficient statistics that require summing over latent variables that are computed for all the training example. In the online version, we can compute these latent variables only for the examples in the current minibatch and update the sufficient statistics using a weighted sum of the old sufficient statistics and the contribution to the sufficient statistics from the new minibatch of data.

5 Related Work

A number of methods have been proposed for multi-label learning with the main focus being on exploiting the relatedness of the labels. To this end, a major thrust has been on developing methods that capture the label relatedness by learning a low-dimensional label embedding for each label (Chen and Lin, 2012; Yu et al., 2014; Kapoor et al., 2012; Rai et al., 2015; Bhatia et al., 2015). This amounts to the label matrix being a low-rank matrix. However, none of these methods are designed to handle the challenging setting where the test examples may contain labels that were not used at the time of training the model. Therefore, none of these methods can be applied to the setting of predicting previously unseen labels.

One way to handle the issue of previously unseen labels, motivated by applications in recommender systems, could be to use inductive matrix completion methods (Jain and Dhillon, 2013; Natarajan and Dhillon, 2014; Rai, 2017). Here the label matrix could be modeled using an inductive matrix factorization or inductive matrix completion model. These methods assume that we are given features for both examples as well as labels. However, it may be difficult/impossible to get a predefined set of good features for the labels. Our framework circumvents this need by *unsupervisedly* learning a good set of features (in form of the label latent factors) directly from the label co-occurrence statistics. In this sense, co-occurrence based label embedding aspect of our model is akin to word-embedding methods (Mikolov et al., 2013); however, instead of using models like skip-gram, we are directly using a negative binomial latent factor model for the label co-occurrences, and augment this latent factor model with another latent factor model for the label matrix. This results in a model that is customized for the multi-label learning problem.

Our work is somewhat similar in spirit to (Liang et al., 2016a) which developed a model for matrix factorization based recommender system by leveraging item-item co-occurrences. However, our model differs in several key aspects. In particular, (1) The zero-shot multi-label learning setting requires extrapolating to unseen labels; and (2) while (Liang et al., 2016a) assumed a real-valued user-item matrix and real-valued item-item co-occurrence (as point-wise mutual information), we explicitly model the *binary* labels and *count-valued* label co-occurrences.

We would like to point out that, although, much of the prior work on zero-shot learning models is typically for multi-class learning problems (Socher et al., 2013; Norouzi et al., 2013; Changpinyo et al., 2016), our model is designed for multi-label learning, where we also need to learn and leverage the structural properties of the label matrix (which we accomplish by having another latent factor model on the label matrix itself).

Although some prior works (Zhang et al., 2016; Mensink et al., 2014; Sandouk and Chen, 2016; Fu et al., 2015) have directly applied solutions of zero-shot multi-class learning for the multi-label learning problem, these methods do not take into account the structure of the label matrix, which is known to be of extreme importance, as evidenced by success of multi-label learning algorithms that do take into account such structure (Chen and Lin, 2012; Yu et al., 2014; Kapoor et al., 2012; Rai et al., 2015; Bhatia et al., 2015).

The idea of jointly factorizing multiple matrices with some latent factors shared between matrices has also been used in collective matrix factorization models (Bouchard et al., 2013). However, these are several key difference in motivation as well as the methodology. In particular, the goal in collective matrix factorization is typically only limited to matrix completion and not in solving problems like multi-label learning, where we need the model to predict label vectors for test examples. Moreover, in our framework, we do not factorize every matrix; in particular, the feature matrix $\mathbf{X}$ is used to *condition* the latent factors, which is needed to make the prediction for test examples. Finally, our generative model is customized to specifically handle binary label matrix and count-valued co-occurrence matrix via appropriate likelihood models for the observations.

Another distinguishing aspect of our model is that, unlike most existing model for zero-shot multi-class learning, which require label attributes to be given beforehard, our framework *learns* these attributes (in form the label latent factors) directly using the label co-occurrence statistics. Learning of these embeddings is directly influenced by the information in the label matrix (since the two latent factor models are learned jointly) and therefore these embeddings are customized for the task.

Our generative framework is also amenable for various interesting extensions, which we leave for future work. For example, it can be be extended to a *mixture* of latent factor models, which can handle the situation when the label matrix is not low-rank but a mixture of several low-rank matrices. Such an extension would be a fully generative counter-part of the model in (Bhatia et al., 2015) which learns a locally low-rank model but has to rely on an *ad-hoc* clustering step beforehand, which is known to be unstable in practice (Bhatia et al., 2015). Another nice aspect of our probabilistic framework is that it naturally allows active learning (Kapoor et al., 2012; Vasisht et al., 2014) where we can selectively ask for most informative labels for an unannotated example.

6 Experiments

To evaluate the efficacy of our proposed framework, we simulate a setting where the training data only has the label presence/absence information for a subset of all the labels. To do this, we train the model using only $L_s < L$ labels (randomly chosen) from the training data label matrix. However, for the model to be able to predict the unseen labels, we also provide the model the co-occurrence statistics of these L_s labels with all the L labels (seen and unseen) that the model is required to predict at test time. While, in general, this information can be obtained from an external source (e.g., a text corpus, such as Wikipedia), for our experiments, we generate it *from the training data* itself as follows: Since the *original* training data (from which we simulate the zero-shot setting) consists of all the L label, we simply count how many training examples share a pair of labels (seen/unseen). In particular, we count the number of training examples in which each of the L_s seen labels co-occurs with each of the labels (seen/unseen). This gives us the $L_s \times L$ matrix $\mathbf{M}$ (count-valued). Note that the unseen labels are *not* used in the label matrix $\mathbf{Y}$ used to train our model and therefore the label matrix $\mathbf{Y}$ used by our model is only of size $N \times L_s$. Again, note that, although in our simulated experimental setting, we have generated the co-occurrence statistics this way, in real-world applications, these statistics can also be obtained from an external text corpus (e.g., Wikipedia).

In our experiments, we use half of the total labels as the seen labels and the remaining labels are treated as the unseen labels. At test time, the task is to predict the presence/absence of both seen as well as unseen labels.

Evaluation Protocol: To evaluate our model and the other baselines, we use two evaluation protocols: (1) the combined average prediction accuracy on both seen and unseen labels, and (2) prediction accuracy exclusively on the unseen labels. We use precision@k as the measure of the predict accuracy. The precision@k for each test example is the fraction of the top-k predicted labels (based on the predicted scores) that are indeed 1s in the ground truth label vector of that test example. We report the precision@k scores averaged over the entire set of test examples.

We evaluate our model on three benchmark data sets (the details in Table 1). In the experiments, we will refer to our model as **ML-LCS** (abbreviated for **M**ulti-label **L**earning with **L**abel **C**o-occurrence **S**tatistics)

We compare our model with the following baselines:

- The first baseline we use is a state-of-the-art multi-label learning algorithm based on a latent factor model (Yu et al., 2014) for the label matrix $\mathbf{Y}$ but cannot incorporate any information about the unseen class labels. As expected, this baseline would be able to predict the seen labels well but would not be able to reliably predict the unseen class labels (our experimental results corroborate this). We refer to this baseline as LFM-SL (latent factor model with seen labels only). Note that although such a model can be extended to handle unseen label, it would require pre-defined feature vectors for the seen and unseen labels (Jain and Dhillon, 2013; Natarajan and Dhillon, 2014). This is not a realistic assumption in the problem setting being considered here.

- Although most of the existing multi-label learning algorithms cannot be applied for predicting unseen labels, one baseline that can however be used in this setting is the COSTA algorithm (Mensink et al., 2014). The COSTA algorithm uses the normalized label co-occurrences of seen labels with each of the unseen labels and uses it to "synthesize" predictors for the unseen labels. To do so, COSTA uses a weighted combination of the predictors for seen labels to construct the predict of each unseen label. However, COSTA first requires training *independent* classifiers for each of the seen labels and then combines these independent classifiers to synthesize the predictors for unseen classes. Since the original COSTA algorithm assumes that the seen labels are independent of each other, which is not true in practice, we designed and implemented a variant of COSTA which builds on top of a latent factor model (Yu et al., 2014). The variant uses a latent factor model to factorize the label matrix (using only seen class labels), which gives the latent factors ($\boldsymbol{v}_l, l = 1...L_s$) of all the seen labels. It then computes the latent factors of any unseen label $\ell = L_s + 1, \ldots, L$ using a *weighted combination* of the seen class latent factors ($\boldsymbol{v}_l, l = 1...L_s$). The combination weights are given by the normalized co-occurrence information of that unseen label with all the seen labels.

Dataset	N	N_{test}	D	L_s	L_u
Bibtex	4880	2515	1836	80	79
Mediamill	30993	12914	120	50	51
Delicious	12920	3185	500	500	483

Table 1: Dataset used for the experiments. D: number of features Dimensionality, L_s: number of seen labels, L_u: number of unseen labels, N: number of training examples, and N_{test}: number of test examples.

Experimental Settings: In our experiments, we fix the

	Bibtex			MediaMill			Delicious		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
LFM-SL	0.42	0.23	0.17	0.64	0.34	0.24	0.41	0.28	0.23
COSTA	0.43	0.31	0.25	0.63	0.22	0.13	0.44	0.37	0.30
ML-LCS	**0.51**	**0.37**	**0.30**	**0.73**	**0.60**	**0.48**	**0.52**	**0.45**	**0.41**

Table 2: Precision@k scores comparison (considering both seen and unseen labels) of our model with the other baselines on various data sets.

regularization hyperparameters for the various model parameters equal to 1 (via a 5-fold cross-validation over the range 0.1-10). The negative binomial overdispersion parameter was set to 5. The number of latent factors was set to 80% of the total number of labels. More fine-grained tuning via cross-validation can be tried and can potentially improve the results further. We run the EM algorithm for 500 iterations; however, the algorithm exhibits a fast convergence and converges in about 100 iterations in all the cases (Sec. 6.4 shows a convergence plot).

6.1 Zero-Shot Multi-Label Learning

Table 2 shows the comparison of our model with the two baselines when considering how well the models predict both seen and unseen labels. As shown in Table 2, our model ML-LCS outperforms both LFM-SL (which only uses the seen labels) as well as COSTA which uses information from both seen as well as unseen labels. Between the two baselines, COSTA (which uses label co-occurrences) outperforms LFA-SL (which does not use label co-occurrences) on two out of the three data sets.

Table 3 shows the comparison (on bibtex data) of our model with COSTA when considering how well the models exclusively predict the unseen examples. Since we are reporting precision scores, it shows how well the models can *rank* the unseen labels. Expectedly, the average accuracies are lower for both models; however, our model outperforms COSTA, which shows that our model. For Table 3, we do not report the accuracies of LFM-SL as LFM-SL accuracies were expectedly very poor since it does not use any information about the unseen labels.

	P@1	P@3	P@5
COSTA	0.23	0.16	0.12
ML-LCS	**0.34**	**0.20**	**0.14**

Table 3: Precision@k scores (considering only unseen labels) of our model with COSTA on bibtex data.

6.2 Standard Multi-Label Learning

The label co-occurrence statistics is also expected to help even in the standard multi-label learning setting where all the labels are available at the training time ($L_s = L$). To demonstrate this, we conduct an experiment on bibtex data for this setting and compare our model with the standard latent factor model based multi-label learning baseline (LFM-SL) which does not use label co-occurrences. Note that, for this setting, our model does not requie learning the regression matrix $\mathbf{\Psi}$. As Table 4 show, our model outperforms LFM-SL. Interestingly, note that both models use the same amount of information (the label co-occurrence matrix $\mathbf{M}$ used in our model is constructed using the label matrix $\mathbf{Y}$) and this experiment shows that a simple "re-encoding" of the training data (Liang et al., 2016a), leveraged appropriately, can also help in extracting better latent factors, leading to improved performance.

	P@1	P@3	P@5
LFM-SL	0.628	0.374	0.273
ML-LCS	**0.643**	**0.398**	**0.293**

Table 4: Precision@k scores for the standard multi-label learning setting.

6.3 Varying Fractions of Seen vs Unseen Labels

We also perform an experiment when we vary the fraction of seen labels from 20%-80% in the increments of 10% and evaluate our model and COSTA in terms of the precision@1 and precision@3 scores. As shown in Fig. 2, despite both ML-LCS and COSTA having access to the label co-occurrence information, our model consistently outperform COSTA for the entire range of fraction of split between seen and unseen labels, which shows the effectiveness of our generative model that jointly models the label presence/absence and the label co-occurrences.

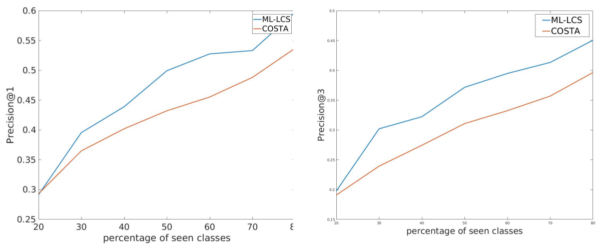

Figure 2: Precision@1 and Prediction@3 comparison of our model (blue curve) with COSTA (red curve) on bibtex data.

6.4 Convergence

The EM algorithm used for our model can only converge to a local optima. However, in practice, we find that our algorithm converges very fast. For example, on the bibtex data, our algorithm converges in about 100 iterations or so, while achieving a significantly better precision@1 score as compared to the COSTA variant we used as a baseline. The convergence plot (also comparing the convergence of COSTA) is shown in Fig. 3. In terms of per-iteration running time, our algorithm is only slightly more expensive than the COSTA baseline as it needs to factorize the label co-occurrence matrix, too.

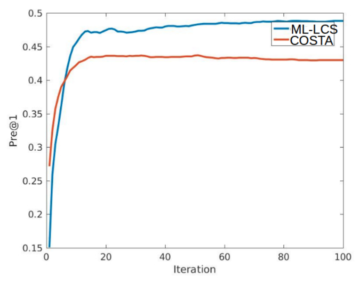

Figure 3: Convergence plot for precision@1 score vs iterations for our model (blue curve) and COSTA (red curve) on bibtex data.

7 Conclusion and Discussion

We have presented a probabilistic framework for multi-label learning that does not require all the labels to be present at the time of training the model. We accomplish this via a generative model that jointly models the label matrix of the training examples and the pairwise label co-occurrence statistics of seen labels (the ones present at training time) with all the labels (seen/unseen). Importantly, the co-occurrence statistics can be computed in an unsupervised fashion using freely available external source of data (e.g., Wikipedia). The joint modeling of the label matrix and the label co-occurrences enables us to learn the predictors for both seen and unseen labels that can be used for predicting the presence/absence of both seen/unseen labels at test time. The generative nature of our framework allows us to also handle missing labels in the label matrix or missing label co-occurrence statistics in the label co-occurrence matrix. Our framework brings to bear the advantage of latent factor models for multi-label learning, while still being capable of handling the labels that were not present at training time. The generative framework also makes it easy to extend our model. For example, it can be extended to a *mixture* of latent factor models, which will allow handling the cases where a single low-rank model does not adequately capture the structure of the label matrix, or replaced by a variational autoencoder style model (Kingma and Welling, 2013; Rezende et al., 2014). Another interesting possible future extension would be to introduce an exposure model (Jain et al., 2017; Liang et al., 2016b) which infers whether a zero in the label matrix is indeed zero or a missing entry (the existing multi-label learning algorithms treat the zeros in the label matrix as true zeros, which may not be the case). Finally, another possible future extension would be to be active learning (Kapoor et al., 2012) in our framework, which can be naturally incorporated since our model produces probabilistic predictions and posterior distributions of various model parameters can be obtained in closed-form.

Appendix

Updating $\mathbf{W}$: Estimating the regression weight matrix $\mathbf{W}$ is equivalent to solving a vector-valued linear regression problem $\boldsymbol{u}_n \approx \mathbf{W}\boldsymbol{x}_n, \forall n$, with the following updates

$$\mathbf{W}^\top = (\mathbf{X}^\top\mathbf{X} + \lambda_w \mathbf{I}_D)^{-1}(\mathbf{X}^\top\mathbf{U}) \tag{19}$$

Note that solving Eq. (19) exactly requires inverting a $D \times D$ matrix which will be expensive for large D. However, the EM algorithm does not require solving for $\mathbf{W}$ *exactly* in each M step. We therefore solve for $\mathbf{W}$ using the conjugate-gradient (CG) method (Bertsekas, 1999), which allows us to also leverage the sparsity in the feature matrix $\mathbf{X}$. Typically, a small number of CG iterations are sufficient in practice.

Updating $\boldsymbol{\Psi}$: Estimating the regression weight matrix $\boldsymbol{\Psi}$ is equivalent to solving a vector-valued linear regression problem $\hat{\boldsymbol{v}}_\ell \approx \boldsymbol{\Psi}\hat{\boldsymbol{\beta}}_\ell, \forall n$, with the following updates

$$\boldsymbol{\Psi}^\top = (\sum_{\ell=1}^{L_s} \hat{\boldsymbol{\beta}}_\ell \hat{\boldsymbol{\beta}}_\ell^\top + \lambda_\Psi \mathbf{I}_K)^{-1}(\hat{\mathbf{B}}_s^\top \hat{\mathbf{V}}_s) \tag{20}$$

where $\hat{\mathbf{B}}_s$ denotes the $L_s \times K$ matrix consisting of the $\hat{\boldsymbol{\beta}}_\ell$'s along its rows and $\hat{\mathbf{V}}_s$ denotes the $L_s \times K$ matrix consisting of the $\hat{\boldsymbol{v}}_\ell$'s along its rows. If L_s is large, the closed-form update can be replaced by a CG style update for better efficiency.

Acknowledgements: This work is supported by DST (India) SERB Early Career Research Award, a grant from Tower Research CSR, Dr. Deep Singh and Daljeet Kaur Fellowship, and Research-I Foundation, IIT Kanpur. Vinay Verma acknowledges support from Visvesvaraya Ph.D. fellowship.

References

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43.

Babbar, R. and Schölkopf, B. (2017). DiSMEC- distributed sparse machines for extreme multi-label classification. In *WSDM*.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Bhatia, K., Jain, H., Kar, P., Varma, M., and Jain, P. (2015). Sparse local embeddings for extreme multi-label classification. In *NIPS*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*.

Bouchard, G., Yin, D., and Guo, S. (2013). Convex collective matrix factorization. In *AISTATS*, volume 13, pages 144–152.

Cappé, O. and Moulines, E. (2009). On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613.

Changpinyo, S., Chao, W.-L., Gong, B., and Sha, F. (2016). Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336.

Chen, Y.-N. and Lin, H.-T. (2012). Feature-aware label space dimension reduction for multi-label classification. In *NIPS*.

Fu, Y., Yang, Y., Hospedales, T., Xiang, T., and Gong, S. (2015). Transductive multi-label zero-shot learning. *arXiv preprint arXiv:1503.07790*.

Gibaja, E. and Ventura, S. (2014). Multilabel learning: A review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

Gibaja, E. and Ventura, S. (2015). A tutorial on multilabel learning. *ACM Comput. Surv.*

Jain, H., Prabhu, Y., and Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*.

Jain, P. and Dhillon, I. S. (2013). Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626*.

Jain, V., Modhe, N. M., and Rai, P. (2017). Scalable generative models for multi-label learning with missing labels. In *ICML*.

Kapoor, A., Viswanathan, R., and Jain, P. (2012). Multilabel classification using bayesian compressed sensing. In *NIPS*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Liang, D., Altosaar, J., Charlin, L., and Blei, D. M. (2016a). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*.

Liang, D., Charlin, L., McInerney, J., and Blei, D. M. (2016b). Modeling user exposure in recommendation. In *WWW*.

Mensink, T., Gavves, E., and Snoek, C. G. (2014). Costa: Co-occurrence statistics for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2441–2448.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Natarajan, N. and Dhillon, I. S. (2014). Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*.

Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S., and Dean, J. (2013). Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

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

Prabhu, Y. and Varma, M. (2014). FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.

Rai, P. (2017). Non-negative inductive matrix completion for discrete dyadic data. In *AAAI*.

Rai, P., Hu, C., Henao, R., and Carin, L. (2015). Large-scale bayesian multi-label learning via topic-based label embeddings. In *NIPS*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Sandouk, U. and Chen, K. (2016). Multi-label zero-shot learning via concept embedding. *arXiv preprint arXiv:1606.00282*.

Scott, J. G. and Sun, L. (2013). Expectation-maximization for logistic regression. *arXiv preprint arXiv:1306.0040*.

Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Vasisht, D., Damianou, A., Varma, M., and Kapoor, A. (2014). Active learning for sparse bayesian multilabel classification. In *KDD*.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., and Xu, W. (2016). CNN-RNN: A unified framework for multi-label image classification. In *CVPR*.

Yu, H.-F., Jain, P., Kar, P., and Dhillon, I. S. (2014). Large-scale multi-label learning with missing labels. In *ICML*.

Zhang, Y., Gong, B., and Shah, M. (2016). Fast zero-shot image tagging. In *CVPR*.