Scalable Generative Models for Multi-label Learning with Missing Labels

Vikas Jain [1 *] **Nirbhay Modhe** [1 *] **Piyush Rai** [1]

Abstract

We present a scalable, generative framework for multi-label learning with missing labels. Our framework consists of a latent factor model for the binary label matrix, which is coupled with an *exposure model* to account for label missingness (i.e., whether a zero in the label matrix is indeed a zero or denotes a missing observation). The underlying latent factor model also assumes that the low-dimensional embeddings of each label vector are directly conditioned on the respective feature vector of that example. Our generative framework admits a simple inference procedure, such that the parameter estimation reduces to a sequence of simple weighted least-square regression problems, each of which can be solved easily, efficiently, and in parallel. Moreover, inference can also be performed in an online fashion using mini-batches of training examples, which makes our framework scalable for large data sets, even when using moderate computational resources. We report both quantitative and qualitative results for our framework on several benchmark data sets, comparing it with a number of state-of-the-art methods.

1. Introduction

Multi-label learning (Gibaja & Ventura, 2015; 2014) is the problem of assigning to an object a subset of labels from a potentially very large label vocabulary (Prabhu & Varma, 2014; Jain et al., 2016; Babbar & Schölkopf, 2017). In contrast to binary or multi-class classification, in multi-label learning, each example is associated with a binary label vector (potentially very large), denoting the presence/absence (relevance/irrelevance) of each label. Multi-label learning has applications in several domains such as computer vision (Wang et al., 2016), computational advertising and recommender systems (Prabhu & Varma, 2014; Jain et al., 2016), etc.

Several state-of-the-art methods for multi-label learning are based on certain *structural assumptions* on the binary label matrix. Some of the key structural assumptions that have been used in prior work include low-rank assumption (Yu et al., 2014), locally low-rank assumption (Bhatia et al., 2015), and low-rank plus sparse assumption (Xu et al., 2016), and clusters/topics of labels assumption (Cissé et al., 2016; Rai et al., 2015). Models based on these assumptions are broadly dubbed as embedding based methods for multi-label learning and offer two key advantages: (1) The relatedness/correlation among labels can be easily modeled/captured, and (2) the label vector for each example can be represented as a low-dimensional embedding, which faciliates developing computationally scalable models for multi-label learning. A more detailed discussion of prior work is provided in the Related Work section.

Despite the considerable recent interest and progress on the problem of multi-label learning (Yu et al., 2014; Bhatia et al., 2015; Wang et al., 2016; Cissé et al., 2016), a number of important issues still remain. One of such issues, especially for the embdding based methods, is the ambiguity regarding the zeros vs unobserved (missing) entries in the binary label vector of each example. Since, in practice, the true value (0/1) for only a small subset of all the labels can be obtained, the zeros in the label vector do not necessarily represent negative labels. A typical heuristic employed by multi-label learning algorithms is to simply treat all such the zeros in the label vector as are true negatives (Yu et al., 2014). Another heuristic is to assign different weights to the zeros and ones in the binary label matrix (Yu et al., 2017), which is inspired by matrix factorization based collaborative filtering models that learn from implicit (binary) feedback data (Hu et al., 2008). However, a more principled strategy to address this issue is highly desirable.

Another important desideratum is scalability, especially in the case of *extreme* multi-label learning problems (Prabhu & Varma, 2014; Jain et al., 2016; Babbar & Schölkopf, 2017), which are characterized by a massive number of labels, features, and examples. Although a number of recent multi-label learning models have been proposed that can scale to large-scale problems, these models usually require

[*]Equal contribution [1]Department of Computer Science and Enginerring, IIT Kanpur, Kanpur 208016, UP, India. Correspondence to: Vikas Jain <vikasj@iitk.ac.in>, Nirbhay Modhe <nirbhaym@iitk.ac.in>, Piyush Rai <piyush@cse.iitk.ac.in>

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

large computational resources to truly scale to massive data sets (Babbar & Schölkopf, 2017; Bhatia et al., 2015; Jain et al., 2016). Moreover, most of the scalable multi-label learning algorithms only operate in batch setting and are usually not designed to work (Prabhu & Varma, 2014; Bhatia et al., 2015; Jain et al., 2016) in online settings with continuous stream of training examples.

In this paper, we present a scalable, generative framework for multi-label learning, that not only bring to bear the modeling flexibility of probabilistic, generative models for the multi-label learning problem (Kapoor et al., 2012; Rai et al., 2015), but is also designed to handle the above-mentioned challenges in a principled way. Our framework is based on a latent factor model for the binary label matrix, and has the following distinguishing aspects: (1) It naturally handles the issue of missing vs negative labels via a principled generative model with a *exposure model* (Liang et al., 2016) for the label matrix; (2) It is accompanied by a simple and scalable inference procedure (both via Gibbs sampling and via fast point estimation); and (3) Inference can also be easily performed in an online fashion, enabling us to apply it on large-scale problems, even when using moderate computational resources.

2. The Model

In the multi-label learning problem, we assume that we are given N training examples $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $\boldsymbol{y}_n \in \{0,1\}^L$, $n = 1, \ldots, N$. We will denote $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \in \mathbb{R}^{N \times D}$ to be the feature matrix and $\mathbf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\} \in \{0,1\}^{N \times L}$ to be the label matrix. Given training data $\{\mathbf{X}, \mathbf{Y}\}$, the goal in multi-label learning is to learn a model that can predict the label vector $\boldsymbol{y}_* \in \{0,1\}^L$ for a new test input $\boldsymbol{x}_* \in \mathbb{R}^D$.

Note that an entry $y_{n\ell} = 0$ in the label matrix $\mathbf{Y}$ may not necessarily mean a negative label but could simply mean that this label is missing (and its *true* value could be 0 or 1). As we shall show, our generative model can infer the missingness of a label $y_{n\ell} = 0$ by associating another binary latent variable $\xi_{n\ell}$ (called *exposure* variable). These exposure variables will be incorporated in a latent factor model (Sec. 2.1) for the label matrix $\mathbf{Y}$ and are jointly learned along with the rest of the model parameters.

2.1. An Exposure-based Latent Factor Model for the Binary Label Matrix

We model the binary label matrix $\mathbf{Y}$ using a latent factor model. Specifically, we assume that each training example $n = 1, \ldots, N$ is associated with a latent factor $\boldsymbol{u}_n \in \mathbb{R}^K$ and each label $\ell = 1, \ldots, L$ is associated with a latent factor $\boldsymbol{v}_\ell \in \mathbb{R}^K$. We further condition $\boldsymbol{u}_n$ on the feature vector $\boldsymbol{x}_n \in \mathbb{R}^D$ of example n by assuming that the prior distribution of $\boldsymbol{u}_n$ is conditioned on $\boldsymbol{x}_n$, as $p(\boldsymbol{u}_n|\boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{u}_n|\mathbf{W}\boldsymbol{x}_n, \lambda_u^{-1}\mathbf{I}_K)$. Here, $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]^\top \in \mathbb{R}^{K \times D}$ which denotes the matrix of regression weights that map the feature vector $\boldsymbol{x}_n$ to the mean of the Gaussian prior on $\boldsymbol{u}_n$. We further assume a zero-mean Gaussian prior $p(\boldsymbol{v}_\ell) = \mathcal{N}(\boldsymbol{v}_\ell|\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ on label latent factors $\boldsymbol{v}_\ell$, $\ell = 1, \ldots, L$. Note that, although we do not consider it here, our model can also be easily extended to incorporate *label features* (if available) by conditioning Gaussian prior on $\boldsymbol{v}_\ell$ on those label features, in the same manner we condition the prior on $\boldsymbol{u}_n$ on input features.

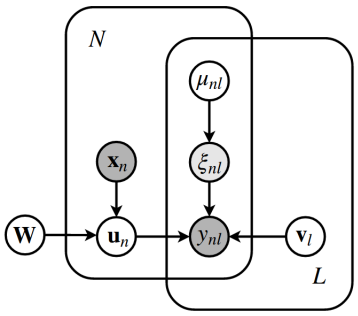

Figure 1. Our generative model in plate notation. Note: Hyperparameters not shown for brevity.

The complete generative story for each label $y_{n\ell}$ of the binary label matrix $\mathbf{Y}$ is given by

$$\boldsymbol{u}_n|\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{u}_n|\mathbf{W}\boldsymbol{x}_n, \lambda_u^{-1}\mathbf{I}_K) \quad (1)$$

$$\boldsymbol{v}_\ell \sim \mathcal{N}(\boldsymbol{v}_\ell|\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K) \quad (2)$$

$$\xi_{n\ell} \sim \text{Bernoulli}(\mu_{n\ell}) \quad (3)$$

$$y_{n\ell} \sim \begin{cases} \text{Bernoulli}\left(y_{n\ell}|\sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)\right), & \text{if } \xi_{n\ell} = 1 \\ \delta_0, & \text{if } \xi_{n\ell} = 0 \end{cases} \quad (4)$$

where $\sigma(z) = 1/(1 + \exp(-z))$ denotes the logistic function. Note that we have associated a binary exposure latent variable $\xi_{n\ell}$ with each label $y_{n\ell}$ such that $\xi_{n\ell} = 0$ implies that $y_{n\ell}$ is 0 because it is *missing* (not exposed), and $\xi_{n\ell} = 1$ implies that $y_{n\ell}$ is exposed (and could be 0 or 1 depending on the outcome of the Bernoulli draw). In Eq. 4, δ_0 denotes a point-mass at zero, which means that, if $\xi_{n\ell} = 0$, then $y_{n\ell}$ is zero with probability 1. Otherwise, we draw the observed label $y_{n\ell}$ from a Bernoulli distribution as $y_{n\ell} \sim \text{Bernoulli}(y_{n\ell}|\sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell))$. Note that, effectively, each $y_{n\ell}$ is being modeled using a mixture of two distributions - a Bernoulli with probability given by the sigmoid $\sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell))$ and a point-mass at 0.

$$y_{n\ell} \sim \xi_{n\ell}\text{Bern}\left(y_{n\ell}|\sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)\right) + (1 - \xi_{n\ell})\mathbb{I}[y_{n\ell} = 0] \quad (5)$$

Note that the latent variable $\xi_{n\ell}$ decides which of the two distributions from this mixture generates $y_{n\ell}$. Figure 1 shows our model in the plate notation. Also note that if $y_{n\ell} = 1$ then $\xi_{n\ell} = 1$ with probability 1 and therefore $\xi_{n\ell}$ only needs to be inferred for entries for which $y_{n\ell} = 0$.

The generative model specified in Eq (1)-(4) has two additional parameters: $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]^\top \in \mathbb{R}^{K \times D}$ which denotes the matrix of regression weights that map each input feature vector $\boldsymbol{x}_n$ to the corresponding latent factor $\boldsymbol{u}_n \in \mathbb{R}^K$, and a probability parameter $\mu_{n\ell} \in (0, 1)$ which denotes the probability of the label $y_{n\ell}$ being exposed (but note that $y_{n\ell}$ can be 0 or 1, depending on the outcome of Bernoulli$(y_{n\ell}|\sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_\ell))$). We refer to $\mu_{n\ell}$ as the *exposure probability* of label ℓ for example n.

We assume each regression weight vector $\boldsymbol{w}_k$ to have a Gaussian prior, i.e., $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{w}_k|\mathbf{0}, \lambda_w^{-1}\mathbf{I}_D)$. Note that the spherical covariance of this prior can also be replaced by a more flexible diagonal covariance, which will give the model ability to perform feature selection.

For the exposure probability $\mu_{n\ell}$, we consider two types of priors. In the first case, we simply assume $\mu_{n\ell} = \mu_\ell, \forall n$, which means that the probability that a label ℓ is observed is the same for all the examples (i.e., the label exposure for the label ℓ is *global*, not example specific). In this case, we assume a Beta prior on μ_ℓ, i.e., $\mu_\ell \sim \text{Beta}(\alpha_1, \alpha_2)$. In the second case, we assume access to some contexual information (often available in applications such as recommender systems) that we may have for each example-label pair (n, ℓ), in form of some given covariates $\boldsymbol{\phi}_{n\ell} \in \mathbb{R}^M$. Given these covariates, we model the label exposure probability as $\mu_{n\ell} = \sigma(\boldsymbol{\beta}^\top \boldsymbol{\phi}_{n\ell})$, where $\boldsymbol{\beta} \in \mathbb{R}^M$ is a vector of regression coefficients. We assume a Gaussian prior on $\boldsymbol{\beta}$, i.e., $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda_\beta^{-1}\mathbf{I}_M)$

3. Inference

Although the generative model specified in Eq. 1-4 is not readily conjugate because the logistic-Bernoulli likelihood is not conjugate to the Gaussian prior on the latent factors, we can leverage data-augmentation techniques (Polson et al., 2013) to make the model locally conjugate. This enables us to develop a simple Gibbs sampling algorithm for doing inference in our model. The conjugacy also allows us to design an *online* expectation maximization (EM) algorithm (Cappé & Moulines, 2009), which enables us to apply our model on large-scale problems.

We handle the non-conjugate logistic-Bernoulli likelihood using the Pólya-gamma augmentation technique (Polson et al., 2013), which 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 $p(\omega) = \text{PG}(b, 0)$ 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., Bernoulli, binomial, negative-binomial) as a Gaussian distribution, when conditioned on a PG random variable $\omega|\psi \sim \text{PG}(b, \psi)$. Specifically, using PG augmentation, we can write the logistic-Bernoulli likelihood from Eq. 4 as a Gaussian when conditioned on $\omega_{n\ell} \sim \text{PG}(1, \boldsymbol{u}_n^\top \boldsymbol{v}_\ell)$. In particular, $\psi_{n\ell} = \boldsymbol{u}_n^\top \boldsymbol{v}_\ell$, conditioned on $\omega_{n\ell}$, becomes a Gaussian

$$p(\psi_{n\ell}|\omega_{n\ell}) \propto \exp\left(\kappa_{n\ell}\psi_{n\ell} - \frac{1}{2}\omega_{n\ell}\psi_{n\ell}^2\right) \quad (6)$$

where $\kappa_{n\ell} = y_{n\ell} - 0.5$. This likelihood with the Gaussian priors on the latent factors $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$ results in Gaussian posteriors on $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$. When doing EM, this also leads to subproblems that are like least square regression problems.

3.1. Gibbs Sampling

Using the PG augmentation, we can derive the posterior distributions of all the latent variables in our model, and perform Gibbs sampling for doing inference in our model. Due to conjugacy, the inference updates are straightforward to derive as are summarized below.

Sampling $\xi_{n\ell}$: Note that if $y_{n\ell} = 1$ then $\xi_{n\ell} = 1$ with probability one, and therefore need not be inferred. For $y_{n\ell} = 0$, we sample $\xi_{n\ell}$ from the posterior

$$p(\xi_{n\ell} = 1|.) \propto \mu_{n\ell}\sigma(-\boldsymbol{u}_n^\top \boldsymbol{v}_\ell) \quad (7)$$

$$p(\xi_{n\ell} = 0|.) \propto (1 - \mu_{n\ell}) \times 1 \quad (8)$$

Sampling $\mu_{n\ell}$: For the case when $\mu_{n\ell} = \mu_\ell, \forall n$, with Beta$(\alpha_1, \alpha_2)$ prior on each μ_ℓ, the posterior will be

$$p(\mu_{n\ell}|.) = \text{Beta}(\alpha_1 + \sum_{n=1}^N \xi_{n\ell}, \alpha_2 + N - \sum_{n=1}^N \xi_{n\ell}) \quad (9)$$

Note that, if we parameterize each $\mu_{n\ell}$ as $\mu_{n\ell} = \sigma(\boldsymbol{\beta}^\top \boldsymbol{\phi}_{n\ell})$ where $\boldsymbol{\phi}_{n\ell}$ is the interaction feature vector for the example-label pair, and the regresssion weight $\boldsymbol{\beta}$ is assumed to have a Gaussian prior, the model is not conjugate. However, using the PG augmentation allows us to easily derive a closed-form Gaussian posterior for $\boldsymbol{\beta}$.

Sampling $\boldsymbol{u}_n$: Given the PG variables $\Omega_{n,:} = \{\omega_{n\ell}\}_{\ell=1}^L$ and the other latent variables, the posterior of $\boldsymbol{u}_n$ will be $\boldsymbol{u}_n \sim \mathcal{N}(\boldsymbol{u}_n|\boldsymbol{\mu}_{\boldsymbol{u}_n}, \boldsymbol{\Sigma}_{\boldsymbol{u}_n})$ where the covariance is given by $\boldsymbol{\Sigma}_{\boldsymbol{u}_n} = (\sum_{\ell=1}^L \xi_{n\ell}\omega_{n\ell}\boldsymbol{v}_\ell\boldsymbol{v}_\ell^\top + \lambda_u\mathbf{I}_K)^{-1}$ and the mean is given by $\boldsymbol{\mu}_{\boldsymbol{u}_n} = \boldsymbol{\Sigma}_{\boldsymbol{u}_n}(\sum_{\ell=1}^L \xi_{n\ell}\kappa_{n\ell}\boldsymbol{v}_\ell + \lambda_u\mathbf{W}\boldsymbol{x}_n)$. Note that if a label ℓ is inferred as *not* exposed for example n, i.e., $\xi_{n\ell} = 0$, it does not contribute to the update of $\boldsymbol{u}_n$.

Sampling $\boldsymbol{v}_\ell$: Given $\Omega_{:,\ell} = \{\omega_{n\ell}\}_{n=1}^N$ and the other latent variables, the posterior $\boldsymbol{v}_\ell$ will be $\boldsymbol{v}_\ell \sim \mathcal{N}(\boldsymbol{v}_\ell|\boldsymbol{\mu}_{\boldsymbol{v}_\ell}, \boldsymbol{\Sigma}_{\boldsymbol{v}_\ell})$ where covariance $\boldsymbol{\Sigma}_{\boldsymbol{v}_\ell} = (\sum_{n=1}^N \xi_{n\ell}\omega_{n\ell}\boldsymbol{u}_n\boldsymbol{u}_n^\top + \lambda_v\mathbf{I}_K)^{-1}$ and the mean $\boldsymbol{\mu}_{\boldsymbol{v}_\ell} = \boldsymbol{\Sigma}_{\boldsymbol{v}_\ell}(\sum_{n=1}^N \xi_{n\ell}\kappa_{n\ell}\boldsymbol{u}_n)$. Note that if an example n is inferred as *not* exposed to label ℓ, i.e., $\xi_{n\ell} = 0$, it does not contribute to the update of $\boldsymbol{v}_\ell$.

Sampling $\mathbf{W}$: Each row $\{\boldsymbol{w}_k\}_{k=1}^K$ of the regression weights matrix $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]^\top \in \mathbb{R}^{K \times D}$ will have a Gaussian posterior given by $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{w}_k|\boldsymbol{\mu}_{\boldsymbol{w}_k}, \boldsymbol{\Sigma}_{\boldsymbol{w}_k})$ where covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}_k} = (\mathbf{X}^\top\mathbf{X} + \lambda_w\mathbf{I}_D)^{-1}$, the mean $\boldsymbol{\mu}_{\boldsymbol{w}_k} = \boldsymbol{\Sigma}_{\boldsymbol{w}_k}(\mathbf{X}^\top\mathbf{U})$, and $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N] \in \mathbb{R}^{K \times N}$.

3.2. Scalable Inference via EM and Online EM

Although the Gibbs sampler (Sec. 3.1) is easy to derive and implement in practice, sampling tends to be slow in practice and convergence may be slow. We therefore present an *online* expectation maximization algorithm (Cappé & Moulines, 2009) for doing efficient inference in our model. We first show the *batch* EM updates for our model parameters and then describe the online EM algorithm which can process the training data in small mini-batches of examples, and results in faster convergence in practice.

3.2.1. The EM Algorithm

The EM algorithm for our model alternates between computing the *expectations* of the *local* latent variables, namely the Pólya-gamma variables $\{\omega_{n\ell}\}$ and the binary exposure latent variables $\{\xi_{n\ell}\}$ in the E step, and then using these expectations to estimate the other model parameters $\boldsymbol{u}_n$, $\boldsymbol{v}_\ell$, $\mathbf{W}$, and exposure probabilities $\{\mu_{n\ell}\}$ in the M step.

The E Step: The E step involves computing the expectations of the latent variables $\{\omega_{n\ell}\}$ and $\{\xi_{n\ell}\}$, given the current values of the other model parameters $\boldsymbol{u}_n$, $\boldsymbol{v}_\ell$, $\mathbf{W}$, and $\mu_{n\ell}$ 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 known to be available in closed form (Scott & Sun, 2013), and are given by

$$\eta_{n\ell} = \mathbb{E}[\omega_{n\ell}|\psi_{n\ell}] = \frac{1}{2\psi_{n\ell}} \tanh\left(\frac{\psi_{n\ell}}{2}\right) \tag{10}$$

  where $\psi_{n\ell} = \boldsymbol{u}_n^\top \boldsymbol{v}_\ell$ is computed using the estimates of $\boldsymbol{u}_n$ and $\boldsymbol{v}_m$ from the previous M step.
- Expectations of each of the binary exposure variables $\xi_{n\ell}$, $\forall n, \ell$, are given by

$$p_{n\ell} = \mathbb{E}[\xi_{n\ell}|\psi_{n\ell}] = \frac{\mu_{n\ell}\sigma(-\psi_{n\ell})}{\mu_{n\ell}\sigma(-\psi_{n\ell}) + (1-\mu_{n\ell})} \tag{11}$$

The M Step: Given the expectations of the latent variables computed in the E step, the M step maximizes the following expected complete data log-likelihood plus log-prior terms, which we denote as $\mathcal{Q}(\mathbf{U}, \mathbf{V}, \mathbf{W}, \boldsymbol{\mu})$, where $\mathbf{U} = \{\boldsymbol{u}_n\}_{n=1}^N$, $\mathbf{V} = \{\boldsymbol{v}_\ell\}_{\ell=1}^L$, $\mathbf{W}$, and $\boldsymbol{\mu} = \{\mu_{n\ell}\}$, $\forall n, \ell$

$$\begin{aligned}\mathcal{Q}(\mathbf{U}, \mathbf{V}, \mathbf{W}, \boldsymbol{\mu}) = &-\frac{1}{2}\sum_{n,\ell} p_{n\ell}\frac{(\kappa_{n\ell} - \eta_{n\ell}\boldsymbol{u}_n^\top \boldsymbol{v}_\ell)^2}{\eta_{n\ell}} \\ &+ \sum_{n,\ell} \log \text{Bernoulli}(p_{n\ell}|\mu_{n\ell}) - \lambda_u \sum_{n=1}^N ||\boldsymbol{u}_n - \mathbf{W}\boldsymbol{x}_n||^2 \\ &- \lambda_v \sum_{\ell=1}^L ||\boldsymbol{v}_\ell||^2 - \lambda_w ||\mathbf{W}||^2 + \sum_{n,\ell} \log \text{Beta}(\mu_{n\ell}|\alpha_1, \alpha_2)\end{aligned} \tag{12}$$

Note that the first term in the objective function given in Eq. 12 is due to the logistic likelihood transformed into a Gaussian (using PG augmentation). This term is akin to a weighted least squares objective where each label being associated with a weight $p_{n\ell} = \mathbb{E}[\xi_{n\ell}|\psi_{n\ell}]$. Intuitively, in the first term, the contribution of each label $y_{n\ell}$ to the log-likelihood gets modulated based on its expected exposure.

Maximizing $\mathcal{Q}(\mathbf{U}, \mathbf{V}, \mathbf{W}, \boldsymbol{\mu})$ w.r.t. each of the model parameters $\mathbf{U}, \mathbf{V}, \mathbf{W}, \boldsymbol{\mu}$, fixing the rest, yields closed-form updates for each of these. The updates are as follows:

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

$$\boldsymbol{u}_n = \boldsymbol{\Sigma}_{\boldsymbol{u}_n}\left(\sum_{\ell=1}^L p_{n\ell}\kappa_{n\ell}\boldsymbol{v}_\ell + \lambda_u \mathbf{W}\boldsymbol{x}_n\right) \tag{13}$$

  where $\boldsymbol{\Sigma}_{\boldsymbol{u}_n} = (\sum_{\ell=1}^L p_{n\ell}\eta_{n\ell}\boldsymbol{v}_\ell\boldsymbol{v}_\ell^\top + \lambda_u \mathbf{I}_K)^{-1}$. Note that the updates for $\{\boldsymbol{u}_n\}_{n=1}^N$ are all independent of each other and are easily parallelizable.
- Estimating each of the label latent factors $\{\boldsymbol{v}_\ell\}_{\ell=1}^L$ is a weighted ridge-regression problem with solution

$$\boldsymbol{v}_\ell = \boldsymbol{\Sigma}_{\boldsymbol{v}_\ell}\left(\sum_{n=1}^N p_{n\ell}\kappa_{n\ell}\boldsymbol{u}_n\right) \tag{14}$$

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

$$\mathbf{W}^\top = (\mathbf{X}^\top\mathbf{X} + \lambda_w \mathbf{I}_D)^{-1}(\mathbf{X}^\top\mathbf{U}) \tag{15}$$

  Note that solving Eq. (15) 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 $\mathbf{W}$ efficient using gradient based methods, such as 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.
- Given $p_{n\ell}$ from the E step, the updates for $\mu_{n\ell}$ for the case when $\mu_{n\ell} = \mu_\ell$, $\forall n$, is simply the MAP solution

$$\mu_\ell = \frac{\alpha_1 + \sum_{n=1}^N p_{n\ell} - 1}{\alpha_1 + \alpha_2 + N - 2} \tag{16}$$

  For the other case when each $\mu_{n\ell}$ in modeled as $\mu_{n\ell} = \sigma(\boldsymbol{\beta}^\top \boldsymbol{\phi}_{n\ell})$ with a Gaussian prior on $\boldsymbol{\beta}$, estimating $\boldsymbol{\beta}$ reduces to solving a regression problem with the training data being $\{\boldsymbol{\phi}_{n\ell}, p_{n\ell}\}$, $\forall n, \ell$, where $\boldsymbol{\phi}_{n\ell}$ is the given feature vector for the input-label pair n, ℓ and $p_{n\ell}$ is estimated in the E step. Ignoring the prior term (equivalent to ℓ_2 regularizer on $\boldsymbol{\beta}$), we can estimate $\boldsymbol{\beta}$ iteratively using gradient-descent updates

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \frac{\tau}{NL}\sum_{n,\ell}(\sigma(\boldsymbol{\beta}^\top \boldsymbol{\phi}_{n\ell}) - p_{n\ell})\boldsymbol{\phi}_{n\ell} \tag{17}$$

  where τ denotes the learning rate.

3.2.2. ONLINE EM

The EM algorithm described in Section 3.2.1 is more efficient than the Gibbs sampler described in Section 3.1. It is also highly parallelizable since the updates for $\{\boldsymbol{u}\}_{n=1}^N$ and $\{\boldsymbol{v}_\ell\}_{\ell=1}^L$ can be easily parallelized, and solve for $\mathbf{W}$ efficiently using CG updates. However, it is a batch procedure and requires going over the entire training data in every iteration. For large-scale multi-label learning problems, which are characterized by large N, D, and L, the batch setting may not be feasible in practice, especially when having access to moderate computational resources and storage.

We therefore present an efficient online version of the EM algorithm for our model which allows it to scale up to massive-sized data sets even on machines with moderate hardware. As we show in our experiments, this enables us to apply our model to be run efficiently on massive data sets (e.g., one of the data sets we experiment with has more than $600k$ examples with about $50k$ features per example) even on a standard laptop with very moderate hardware.

The online EM algorithm works by maintaining sufficient statistics of all the model parameters and updates these sufficient statistics with every mini-batch of data. For each mini-batch of training examples, the E step computes the relevant expectations associated with these observations and then uses the expectations to update the sufficient statistics of the parameters to be estimated in the M step.

For example, noting that the sufficient statistics for updating the label latent factors $\boldsymbol{v}_\ell = \mathbf{A}^{-1}\boldsymbol{b}$ are given by $\mathbf{A} = \sum_{n=1}^N p_{n\ell}\eta_{n\ell}\boldsymbol{u}_n\boldsymbol{u}_n^\top + \lambda_v\mathbf{I}_K$ and $\boldsymbol{b} = \sum_{n=1}^N p_{n\ell}\kappa_{n\ell}\boldsymbol{u}_n$, we can update $\mathbf{A}$ and $\boldsymbol{b}$ using a small mini-batch containing N_b examples as $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N_b}$ as follows

$$\mathbf{A}^{(t+1)} = (1-\gamma_t)\mathbf{A}^{(t)} + \gamma_t\mathbf{A}^{(new)} \tag{18}$$

$$\boldsymbol{b}^{(t+1)} = (1-\gamma_t)\boldsymbol{b}^{(t)} + \gamma_t\boldsymbol{b}^{(new)} \tag{19}$$

where $\mathbf{A}^{(new)} = (\sum_{n=1}^{N_b} p_{n\ell}\eta_{n\ell}\boldsymbol{u}_n\boldsymbol{u}_n^\top + \lambda_v\mathbf{I}_K)$, and $\boldsymbol{b}^{(new)} = \sum_{n=1}^{N_b} p_{n\ell}\kappa_{n\ell}\boldsymbol{u}_n$ are computing using only the current mini-batch. The sufficient statistics of the other model parameters can also be updated in the same manner. Here γ_t is a decaying learning rate (or a forgetting factor), which also acts as a trade-off between the contribution from the old sufficient statistics computed thus far and the sufficient statistics contribution from the new mini-batch of data. We set $\gamma_t = (a_0 + t)^{-\epsilon}$ with $a_0 = 1$ and ϵ to be close to 0.5 (Cappé & Moulines, 2009).

3.2.3. PREDICTION

Given a new test input $\boldsymbol{x}_*$, we first predict its latent factor $\boldsymbol{u}_* \in \mathbb{R}^K$ as $\mathbf{W}\boldsymbol{x}_*$ and then predict each entry of its label vector $\boldsymbol{y}_*$ as $\mathbb{E}[y_{*\ell}|\boldsymbol{u}_*, \boldsymbol{v}_\ell] = \sigma(\boldsymbol{u}_*^\top\boldsymbol{v}_\ell)$. If we are only interested in the top few labels, fast search methods such as maximum inner product search (Fraccaro et al., 2016) can be used to reduce the computational cost at test time.

4. Related Work

A prominent line of work on multi-label learning has been based on models that learn a low-dimensional embedding of the label vectors (Chen & Lin, 2012; Yu et al., 2014; Rai et al., 2015; Bhatia et al., 2015). Note that this amounts to assuming that the label matrix is low-rank.

Since many real-world data sets have a large number of rare labels, sometimes the low-rank assumption may not be appropriate. To address this issue, (Bhatia et al., 2015) proposed a method which assumes the label matrix to be *locally* low-rank. One way to impose this assumption is to learn embeddings that only try to preserve distances in a small neighborhood of each example. Another approach to handle the rare labels is to assume that the label matrix is a sum of a low-rank and a sparse matrix (Xu et al., 2016).

Note that our latent factor model is equivalent to imposing a low-rank assumption on the label matrix, and is therefore similar in spirit to the label-embedding approaches. However, unlike the existing label-embedding based approaches, our generative framework has a principled mechanism to handle/infer the unobserved labels. Moreover, none of the existing label-embedding methods can work in online fashion, and scaling up these methods to large-scale problems requires large computational resources. In addition, our model readily allows incorporating the label features (if available) by a simple modification to the prior on the label latent factors.

Apart from the label-embedding based multi-label learning methods, tree-based methods for multi-label learning (Agrawal et al., 2013; Prabhu & Varma, 2014; Jain et al., 2016) are also popular due to being fast at test time, especially when the number of labels is large. However, these models usually have high training costs and cannot be trained easily in an online fashion, unlike our model. On the other hand, for faster predictions at test time, our framework model can easily be adapting by replacing the Gaussian prior on the label latent factors $\boldsymbol{v}_\ell$ by a von Mises-Fisher prior (Fraccaro et al., 2016), which naturally facilitates using maximum inner-product search techniques, without the requirement of any post-processing.

Among other models to address the missing labels problem in multi-label learning, recently, (Kanehira & Harada, 2016) proposed a ranking based framework for learning from positive and unlabeled data in the context of multi-label learning. Although this is similar in spirit to our model in terms of not treating the unobserved labels as zeros, the approach in (Kanehira & Harada, 2016) is fundamentally different than ours. Moreover, their setting is not amenable to online learning, nor does it leverage the low-rank structure of label matrices with a huge number of labels. Other approaches that try to handle missing labels in-

clude (Bucak et al., 2011) which uses group LASSO adaptation of a multi-label ranking objective, and (Kong et al., 2014), which learns a model using a positive and unlabeled (PU) stochastic gradient descent procedure. However, it works in batch setting, uses stacking to leverage label correlations, and does not scale to large number of labels.

One-class matrix factorization (OCMF) is also an approach (Yu et al., 2017) to solve the missing labels problem by assigning different (but fixed) weights to the ones and zeros. In contrast to this method, our generative framework can *learn* the weight for each label by modeling these weights as latent variables. In another recent work, (Liang et al., 2016) proposed an exposure model for recommender system problems posed as matrix factorization of implicit feedback data. Their approach of modeling the exposure similar in spirit to our framework.

Some of the early works on generative models for multi-label learning problems include models specifically designed for image annotation problems (Barnard et al., 2003; Feng et al., 2004). Other recent attempts on doing multi-label learning in more general problem settings include models such as Bayesian compressive sensing (Kapoor et al., 2012) and multi-label learning using Bayesian non-negative matrix factorization (Rai et al., 2015). However, these models do not have a mechanism to distinguish between unobserved and negative labels, have complicated inference, and do not scale to large-scale problems.

Our generative framework is also amenable for various interesting extensions. 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. Note that 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 framework is that is 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. Moreover, our framework is flexible and inference in our model can be performed in a fully Bayesian manner (e.g., MCMC or variational inference) as well as fast point estimation methods such as (online) EM, that we used in this work.

To summarize, our generative framework offers a flexible way to model the label generation mechanism for real-world multi-label data sets, which most of the existing models currently lack. We can model label missingness/observability rigorously under our framework and infer the model parameters easily using a simple inference procedure. Moreover, the simplicity of the inference procedure makes it easy to design scalable inference algorithms, such as online EM for our model, which enables updating the model whenever fresh training data is available. This is in contrast to some of the other state-of-the-art multi-label learning methods, which although scalable (Bhatia et al., 2015; Prabhu & Varma, 2014; Jain et al., 2016), are not suitable to be applied in such online settings.

5. Experiments

We evaluate our framework on a number of benchmark data sets and compare it with several state-of-the-art methods. Our baselines include both label-embedding methods as well as tree-based methods. The statistics of data sets we use in our experiments are summarized in Table 1.

Dataset	N	N_{test}	D	L
Bibtex	4880	2515	1836	159
Mediamill	30993	12914	120	101
Eurlex-4K	15539	3809	5000	3993
Movielens	4000	2040	29	3952
RCV	623847	155962	47236	2456
Wikipedia	14146	6616	101938	30938

Table 1. Dataset used for the experiments with their properties. D: number of features, L: number of labels, N: number of training examples, N_{test}: number of test examples

We report both quantitative results (in terms of label prediction accuracies) as well as some qualitative results, namely looking at the relationship of empirical label frequencies and label exposure. Note that the label frequency for a given label denotes how many examples had this label as 1, while label exposure $\boldsymbol{\mu}_\ell \in (0, 1)$ in general refers to how popular the label ℓ is.

In our experiments, we compare with the following state-of-the-art baselines.

- **LEML:** This is a low-rank embedding based multi-label learning model (Yu et al., 2014). LEML assumes the label matrix $\mathbf{Y}$ to be modeled as $\mathbf{Y} \approx \mathbf{UV}$ where $\mathbf{U} = \mathbf{XW}$. LEML considers various types of loss functions such as squared loss, logistic loss, hinge loss, etc. Interestingly, note that LEML with logistic loss can be seen as a special non-probabilistic case of our model when also considering $\lambda_u \to \infty$, and the label exposure model turned off.
- **BCS:** Bayesian Compressive Sensing (BCS) is a generative model (Kapoor et al., 2012) for the label vector. It assumes a compressive sensing model for the label vectors and is essentially a low-rank model.
- **FastXML:** This is a fast tree-based multi-label learning model which uses an ensemble of trees (Prabhu & Varma, 2014).
- **PfasterXML:** This is an extension of FastXML and uses *propensity*-weighted scores to improve performance on rare labels (Jain et al., 2016).

- **PD-Sparse:** This model takes a different approach as compared to label-embedding methods and uses a margin-maximizing loss for the multi-label learning problem (Yen et al., 2016).

For the baselines, the reported results are either obtained using publicly available implementations (with the recommended hyperparameter settings), or the publicly known best results. We refer to our model as **GenEML** (for Generative Exposure-based model for Multi-label Learning)

Hyperparameter Settings: For our model, we set the hyperparameters λ_u and λ_v to 0.001, which works well on all the data sets we experimented with. We select the other two hyperparameters λ_w and K (number of latent factors) using cross-validation. On small-/medium-scale data, both EM and online EM perform comparably and we only report the results using online EM. On large data sets, we only use online EM. On the small and medium-scale data, we however also show a separate experiment comparing EM and online EM for our model in terms of convergence speed versus accuracy. For the conjugate gradient (CG) method used by the M step of our inference algorithm, we run 5 iterations, which was found to be sufficient. For online EM, for each data set, we use mini-batch sizes of 1024 and 4096 and report the one which gives better results.

5.1. Quantitative Results

5.1.1. Benefit of Exposure Model

In our first experiment, we assess the benefit of using the exposure model. For this, we apply our model with and without exposure on a synthetic data set. For this experiment, we generate a synthetic data set with N=500, D=100, and L=20 and use varying degrees of exposure probabilities $\mu_\ell \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.9\}$ for the different labels $\ell = 1, \ldots, 20$. We also create a test set with 500 test examples.

The results are shown in Table 2. As the results show, our model with exposure turned on outperforms the model when the exposure is turned off. This clearly demonstrate the benefit of the exposure model when a significant fraction of labels are missing (i.e., not exposed). Our model also outperforms LEML which does not have a mechanism to model label exposure.

	GenEML	GenEML w/o Exposure	LEML
P@1	**87.8**	79.2	78.8
P@2	**75.2**	68.9	69.1
P@3	**65.2**	59.0	58.9

Table 2. Precision@k values on synthetic data obtained by our model with and without using the *label exposure*

5.1.2. Prediction Accuracies

In our next set of experiments, in Table 3 we compare our model (with exposure on) with the other baselines, in terms of Precision@1, Precision@3, and Precision@5 scores. As Table 3 shows, our model outperforms the other baselines in most of the cases, except for the RCV and Wikipedia data, on which our model is outperformed by LEML and/or PfasterXML. Note, however, that these state-of-the-art baselines use batch inference methods whereas we only ran our model in the online setting on a moderate 4 core processor with 8GB RAM. Moreover, our results may further improve with a more careful hyperparameter tuning (including selection of minibatch size). The point of the large-scale data experiment was to mainly show that the our model can be feasibly run on such large-scale data sets, on standard machines with moderate computational resources. Most of the other existing models for multi-label learning are infeasible to run under such restrictive settings.

5.1.3. Batch EM vs Online EM

The online version of our EM algorithm is scalable and faster than its batch counterpart. Fig 2 shows that online EM converges faster and to a precision score which is very similar to the batch EM on Bibtex and Mediamill datasets.

Furthermore, online inference is also more effecient, storage-wise, due the need of maintaining just the sufficient statistics as in Eq 19 for the updates of each latent factor $\boldsymbol{u}_n$ and $\boldsymbol{v}_\ell$. For very large datasets, the size of the the sufficient statistics (a $D \times D$ covariance matrix) for updating the regression weight matrix $\mathbf{W}$ might not be feasible to store and update. Therefore, we use cheap, first-order gradient based updates for finding an approximate solution to the update equation of $\mathbf{W}$ in each iteration of the EM algorithm (note that we need not solve for $\mathbf{W}$ exactly; the EM algorithm just requires a few steps of updates for $\mathbf{W}$ in the M step). This further reduces the memory requirement of our model, while also speeding up inference due to faster computation of gradients as compared to CG updates.

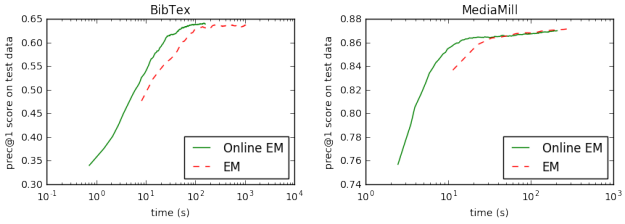

Figure 2. Convergence time comparison of the batch and online EM algorithm for inference in our model

5.2. Qualitative Results

Finally, we do some qualitative analyses of our model's behavior. We investigate whether the global frequency of a label necessarily correlates to its exposure probability.

Dataset		BCS	WSABIE	FastXML	PfasterXML	PD-Sparse	LEML	GenEML
Bibtex	P@1	60.24	54.78	63.42	63.46	61.29	62.54	**64.09**
	P@3	34.87	32.29	39.23	39.22	35.82	38.41	**40.14**
	P@5	24.48	23.98	28.86	29.14	25.74	28.21	**29.47**
Mediamill	P@1	-	81.29	84.22	83.98	81.86	84.01	**87.15**
	P@3	-	64.74	67.33	67.37	62.52	67.20	**69.98**
	P@5	-	49.83	53.04	53.02	45.11	52.80	**55.21**
Eurlex-4K	P@1	-	68.55	71.36	75.45	76.43	63.40	**77.75**
	P@3	-	55.11	59.90	62.70	60.37	50.35	**63.98**
	P@5	-	45.12	50.39	52.51	49.72	41.28	**53.24**
Movielens	P@1	-	-	-	-	-	54.22	**55.78**
	P@3	-	-	-	-	-	50.44	**50.62**
	P@5	-	-	-	-	-	48.63	**48.86**
RCV-2K	P@1	-	-	-	-	-	**89.09**	87.21
	P@3	-	-	-	-	-	**70.96**	69.22
	P@5	-	-	-	-	-	**50.73**	49.53
Wiki10-31K	P@1	-	-	83.03	**83.57**	-	73.47	76.38
	P@3	-	-	67.47	**68.61**	-	62.43	44.49
	P@5	-	-	57.76	**59.10**	-	54.35	32.06

Table 3. Performance Comparison using *Prec@k* of the model with other baselines. The - denotes that either these results were not available or the method was infeasible to run on that data set. On the large-scale data sets (RCV and Wiki), our model was run using online EM based inference.

While it may be the case for some data sets where high label frequency implies a high inferred label exposure probability (e.g., see Fig. 3 for Bibtex and Mediamill data), it need not be the case with other data sets. For example, for Movielens data, each user-movie (example-label) pair has an some context information (user and movie features) available for it. As we show in Fig 4, the inferred exposure probability (which depends on the context features) of the same movie (label) indeed turns out to be different for different users (examples).

Fig. 4 shows the plot of inferred exposure probabilities μ_{nl} for two users (one female, one male) plotted against the label frequencies (movie popularities).

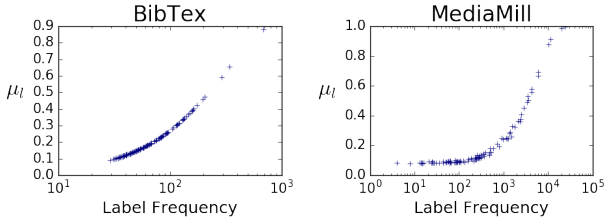

Figure 3. Inferred label exposure probabilities for Bibtex and Mediamill data sets

As Fig. 4 shows, our model infers that, a popular movie (shown in *red* dot in Fig 4) has a high exposure probability for the *left* user (Female, 25, Healthcare/Doctor) while it has a low exposure probability for the *right* user (Male, 35, artist). This example illustrates that a high label frequency does not necessarily imply a high exposure probability, which can be context (user in this case) dependent.

6. Conclusion

We presented a flexible and scalable generative framework for multi-label learning. Our framework is based on a latent factor model for the label matrix and does not assume that the zeros in the label matrix are necessarily negative labels. We use a set of label exposure latent variables to model this, and infer these exposure probabilities from data. Incorporating these latent variables leads to improve multi-label classification accuracies, and also enables doing interesting qualitative analyses. Our model admits a simple inference procedure which can be implemeted using Gibbs sampling or EM. We further develop a highly scalable online EM algorithm for performing inference in our model, which allows our model to be applied on large-scale data sets, even on standard machines with moderate hardware. The generative framework makes it easy to extend our model in many interesting ways. 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.

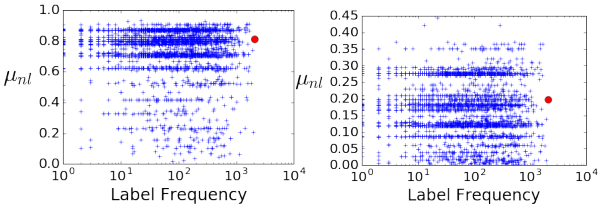

Figure 4. Inferred user-specific label exposure probabilties for two users on Movielens data set: Female, 25, Healthcare/Doctor (*left*) and Male, 35, artist (*right*), with label frequency for each label. The *red circle* shows the most frequent movie (*Hairspray(1988) Comedy, Drama*) and its exposure probability for both the users.

Acknowledgements: PR acknowledges support from Extreme Classification research grant from Microsoft Research India, DST-SERB Early Career Research Award, Dr. Deep Singh and Daljeet Kaur Fellowship, and Research-I Foundation, IIT Kanpur.

References

Agrawal, Rahul, Gupta, Archit, Prabhu, Yashoteja, and Varma, Manik. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, 2013.

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

Barnard, Kobus, Duygulu, Pinar, Forsyth, David, Freitas, Nando de, Blei, David M, and Jordan, Michael I. Matching words and pictures. *JMLR*, 2003.

Bertsekas, Dimitri P. *Nonlinear programming*. Athena scientific Belmont, 1999.

Bhatia, Kush, Jain, Himanshu, Kar, Purushottam, Varma, Manik, and Jain, Prateek. Sparse local embeddings for extreme multi-label classification. In *NIPS*, 2015.

Bucak, Serhat Selcuk, Jin, Rong, and Jain, Anil K. Multi-label learning with incomplete class assignments. In *CVPR*, 2011.

Cappé, Olivier and Moulines, Eric. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2009.

Chen, Yao-Nan and Lin, Hsuan-Tien. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, 2012.

Cissé, Moustapha, Al-Shedivat, COM Maruan, and Bengio, Samy. Adios: Architectures deep in output space. In *ICML*, 2016.

Feng, SL, Manmatha, Raghavan, and Lavrenko, Victor. Multiple bernoulli relevance models for image and video annotation. In *CVPR*, 2004.

Fraccaro, Marco, Paquet, Ulrich, and Winther, Ole. Indexable probabilistic matrix factorization for maximum inner product search. In *AAAI*, 2016.

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

Gibaja, Eva and Ventura, Sebastián. A tutorial on multilabel learning. *ACM Comput. Surv.*, 2015.

Hu, Yifan, Koren, Yehuda, and Volinsky, Chris. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

Jain, Himanshu, Prabhu, Yashoteja, and Varma, Manik. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*, 2016.

Kanehira, Atsushi and Harada, Tatsuya. Multi-label ranking from positive and unlabeled data. In *CVPR*, 2016.

Kapoor, Ashish, Viswanathan, Raajay, and Jain, Prateek. Multilabel classification using bayesian compressed sensing. In *NIPS*, 2012.

Kong, Xiangnan, Wu, Zhaoming, Li, Li-Jia, Zhang, Ruofei, Yu, Philip S, Wu, Hang, and Fan, Wei. Large-scale multi-label learning with incomplete label assignments. In *SDM*, 2014.

Liang, Dawen, Charlin, Laurent, McInerney, James, and Blei, David M. Modeling user exposure in recommendation. In *WWW*, 2016.

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

Prabhu, Yashoteja and Varma, Manik. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, 2014.

Rai, Piyush, Hu, Changwei, Henao, Ricardo, and Carin, Lawrence. Large-scale bayesian multi-label learning via topic-based label embeddings. In *NIPS*, 2015.

Scott, James G and Sun, Liang. Expectation-maximization for logistic regression. *arXiv preprint arXiv:1306.0040*, 2013.

Vasisht, Deepak, Damianou, Andreas, Varma, Manik, and Kapoor, Ashish. Active learning for sparse bayesian multilabel classification. In *KDD*, 2014.

Wang, Jiang, Yang, Yi, Mao, Junhua, Huang, Zhiheng, Huang, Chang, and Xu, Wei. CNN-RNN: A unified framework for multi-label image classification. In *CVPR*, 2016.

Xu, Chang, Tao, Dacheng, and Xu, Chao. Robust extreme multi-label learning. In *KDD*, 2016.

Yen, Ian EH, Huang, Xiangru, Zhong, Kai, Ravikumar, Pradeep, and Dhillon, Inderjit S. PD-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, 2016.

Yu, Hsiang-Fu, Jain, Prateek, Kar, Purushottam, and Dhillon, Inderjit S. Large-scale multi-label learning with missing labels. In *ICML*, 2014.

Yu, Hsiang-Fu, Huang, Hsin-Yuan, Dhillon, Inderjit S, and Lin, Chih-Jen. A unified algorithm for one-class structured matrix factorization with side information. In *AAAI*, 2017.