Disentangled Variational Autoencoder based Multi-Label Classification with Covariance-Aware Multivariate Probit Model

Junwen Bai*, Shufeng Kong and Carla Gomes
Department of Computer Science, Cornell University
{jb2467, sk2299}@cornell.edu, gomes@cs.cornell.edu

Abstract
Multi-label classification is the challenging task of predicting the presence and absence of multiple targets, involving representation learning and label correlation modeling. We propose a novel framework for multi-label classification, Multivariate Probit Variational AutoEncoder (MPVAE), that effectively learns latent embedding spaces as well as label correlations. MPVAE learns and aligns two probabilistic embedding spaces for labels and features respectively. The decoder of MPVAE takes in the samples from the embedding spaces and models the joint distribution of output targets under a Multivariate Probit model by learning a shared covariance matrix. We show that MPVAE outperforms the existing state-of-the-art methods on important computational sustainability applications as well as on other application domains, using public real-world datasets. MPVAE is further shown to remain robust under noisy settings. Lastly, we demonstrate the interpretability of the learned covariance by a case study on a bird observation dataset.

1 Introduction

Multi-label classification (MLC) concerns the simultaneous prediction of the presence and absence of multiple labels for each sample of a given sample set. Unlike in the conventional classification task, more than one label or target could be associated with each sample in MLC [Zhang and Zhou, 2013; Zhang et al., 2018]. This setting is important for the study of key biodiversity applications in computational sustainability [Gomes et al., 2019], e.g., concerning joint species distributions and wildlife mapping [Chen et al., 2017], as well as in a variety of other scenarios, such as protein site localization [Alazazidah et al., 2015] and drug side effects [Kuhn et al., 2015]. Furthermore, understanding the label correlations is also important. For instance, in biodiversity applications, species correlation modeling is critical to address core ecological concerns like interactions of species with each other, which could affect species monitoring, protection and policymaking [Evans et al., 2017].

Some early work simply decomposes the multi-label classification into multiple single-label classification problems [Boutell et al., 2004]. Though these methods can be adapted from single-label predictors, they ignore the correlation among labels. To improve this, classifier chains [Read et al., 2009] stack the binary classifiers into a chain and reuse the outputs of previous classifiers as extra information to improve the prediction of the current label. Followup works extend the classifier chains to recurrent neural networks [Wang et al., 2016] to increase capacity and better model the label correlation. Label ordering is critical to these methods since long-term dependencies are typically weaker than short-term dependencies. The model structure also restricts parallel computation. Another straightforward method is to find nearest neighbors in the feature space and assign labels to test samples by Bayesian inference [Zhang and Zhou, 2007; Chiang et al., 2012]. However, either the predefined metric space or the prior may heavily affect the model performance.

Latent embedding learning is a recent technique to match features and labels in the latent space. Pioneer studies [Yu et al., 2014; Chen and Lin, 2012; Bhatia et al., 2015] make low-rank assumptions of labels and features, and transform labels to label embeddings, by dimensionality reduction techniques such as canonical correlation analysis (CCA). Benefiting from the capacity of deep neural networks, more recent latent embedding methods for MLC employ neural networks to build and align the latent spaces for both labels and features [Yeh et al., 2017; Chen et al., 2019a]. The constraints in the conventional dimension reduction models are relaxed and embedded into the deep latent space. For example, C2AE relaxes the orthogonality constraint to the minimization of an $\ell_2$ distance. These embedding methods are believed to implicitly encode the label correlations in the embedding space.

Some other state-of-the-art models initiate the research on using graph neural networks (GNN) to explicitly encode the label correlations [Chen et al., 2019b; Lanchantin et al., 2019]. A graph neural network for labels can build dependencies among labels through learned or given edges between them. Though GNN brings a new way to embed the correlations, the number of stacked GNNs or the iterations of message passing may require extra effort to fine-tune.

We propose the Multivariate Probit Variational Autoen-
coder (MPV AE), which improves both the embedding space learning and label correlation encoding. In particular, (1) MPV AE learns probabilistic latent spaces for both labels and features, unlike most autoencoder (AE) based multi-label models. The probabilistic latent space learned by the VAE can provide three major advantages. First, it gives more control to the latent space [Chung et al., 2015]. In many AE models, one can often observe the label-encoder-decoder branch gives much better performance than feature-encoder-decoder branch. Imposing the VAE structure in the latent space helps balance the difficulty of the learning and aligning of the two subspaces. Second, smoothness in the latent space is often desired [Wu et al., 2018]. Probabilistic models like the VAE naturally bring smoothness on a local scale since the decoder decodes a sample rather than a specific embedding. Third, the VAE model and its variations learn representations with disentangled factors [van Steenkiste et al., 2019]. If both latent spaces for features and labels learn disentangled factors, not only is it helpful to aligning two spaces, but it is also beneficial for the decoding process. (2) MPV AE explicitly learns a shared covariance matrix to build dependencies among labels by adopting the Multivariate Probit (MP) probabilistic model, which is inspired by some recent work in joint distribution modeling [Chen et al., 2018]. The MP assumes an underlying latent multivariate Gaussian distribution. We show that the MP model is a simple and straightforward component of the overall probabilistic generative framework compared to other more complex models such as GNNs. More importantly, the MP model improves the prediction performance and provides the interpretability of the learned covariance matrix. By using the Cholesky decomposition and t-SNE, we demonstrate visually the value of the learned covariance, in applications like species correlation modeling in the computational sustainability domain. (3) MPV AE is optimized with respect to a three-component loss function, which includes a Kullback–Leibler (KL) divergence component for jointly learning and aligning the label and feature embeddings and a cross-entropy and ranking loss for the multi-label prediction in the MP model. (4) We thoroughly test MPV AE with multiple public datasets on a variety of metrics. MPV AE outperforms (or is comparable to) other state-of-the-art multi-label prediction models. We further illustrate MPV AE is still robust even if the training labels are noisy.

2 Other Related Work

Covariance matrices are commonly seen in non-deep multi-label classification models [Bi and Kwok, 2014; Zhang and Yeung, 2013]. Non-deep models often assume a matrix-variate normal distribution on features, weights or labels. Covariances thus play a key role in these models. But the scalability issue limits their applications in large-scale problems. However, these assumptions are not necessary in deep neural networks given their powerful expressiveness.

A recent success of combining deep learning and covariance-based paradigms is the deep Multivariate Probit model (DMVP) [Chen et al., 2018]. It is a deep generalization of the classic Multivariate Probit model. An efficient sampling process was proposed in the paper to avoid the heavy-duty Markov chain Monte Carlo (MCMC) sampling process. Though DMVP performs well on the joint likelihood measure, it lacks enough predictive power for presence-absence (0/1) classification. MPV AE makes one step further and introduces the cross-entropy loss as well as the ranking-loss under the Multivariate Probit paradigm.

Some prior work also applied deep generative models in multi-label classification. [Chu et al., 2018] proposes a deep sequential generative model. Though the model is effective in the setting of missing labels, the concern of unstable training of stacked generative models should not be overlooked.

3 Methods

3.1 Preliminaries

Let $D$ denote the dataset $\{(x_i, y_i)^N_{i=1}\}$, where $x_i \in \mathbb{R}^S$ and $y_i \in \{0, 1\}^L$. $x_i$ is the feature vector and $y_i$ represents the presence (1) or absence (0) of targets.

Variational Autoencoder (VAE)

A VAE assumes a generative process for the observed datapoints $X$: $P(X) = \int p_\theta(X|z)P(z)dz$, by introducing latent variables $z$. Since most z’s contribute little to $P(X)$, Monte Carlo sampling would be inefficient. We instead learn a function $q_\phi(z|X)$ to approximate the intractable $P(z|X)$ for efficient sampling. The KL divergence ($D$) between $q_\phi(z|X)$ and $P(z)$ is given by $D[q_\phi(z|X)||P(z|X)] = \mathbb{E}_{z\sim q_\phi}[\log q_\phi(z|X) - \log P(z|X)]$. Applying Bayes rule and transforming the equation yield: $\log P(X) - D[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim q_\phi}[\log p_\theta(X|z)] - D[q_\phi(z|X)||P(z)]$. The right hand side of the equation is the tractable evidence lower bound (ELBO) to maximize. $P(z)$ is the prior, a standard multivariate normal distribution. The first term in the ELBO encourages the reconstruction of $X$ and the second term penalizes the KL divergence between the approximate distribution and the prior to impose structure on the latent space. Both $p_\theta$ and $q_\phi$ can be parameterized by neural networks. With the help of the reparameterization trick, the whole model can be trained with back-propagation.

VAE and its variations can learn disentangled factors by controlling the capacity of the information bottleneck. For example, $\beta$-VAE [Higgins et al., 2017] is a known disentangled VAE, which is able to learn abstract concepts like size and shape, only with a slight modification to the objective, $\mathbb{E}_{z\sim q_\phi}[\log p_\theta(X|z)] - \beta D[q_\phi(z|X)||P(z)]$. MPV AE follows the structure of $\beta$-VAE with $\beta = 1.1$.

Multivariate Probit (MP) Model

Consider a single sample $(x, y)$ where $x$ is the input feature vector and $y \in \{0, 1\}^L$ is the label. The Multivariate Probit model introduces auxiliary latent variables $y^* \in \mathbb{R}^L$, which follow a multivariate normal distribution $N(x^*, \Sigma)$, where $y^*$ is the weight parameter and $\Sigma$ is the covariance matrix. $y$ is viewed as the indicator for whether $y^*$ is positive or not: $y_i = 1 (y_i^* > 0), i = 1, \ldots, L$. The probability of observing $y$ is given by $P(y|x, \Sigma) = \int A \cdots A_2 \ p(y^*|x, \Sigma)dy^*_L \cdots dy^*_1$, where $A_j = (-\infty, 0]$ if $y_j = 0$ or $A_j = (0, \infty)$ otherwise. $P$ is the probability density function of the normal distribution. The framework can be generalized to a deep model.
simply by replacing the mean \( \mathbf{x} \) with a neural network \( f(\mathbf{x}) \). In MPVAE, the mean is given by the decoder of VAE.

### 3.2 MPVAE

We propose Multivariate Probit Variational Autoencoder, a novel disentangled Variational Autoencoder based framework with covariance-aware Multivariate Probit model for MLC. The illustration of the framework is shown in Fig. 1. The whole model can be viewed as a two-stage generative process. The first stage maps the features and labels to Gaussian subspaces where the means and variances are learned by multi-layer perceptrons. The key task in this stage is to match the two subspaces. The second stage decodes the sample from each subspace and feeds the outputs into a Multivariate Probit module as the means. A global covariance matrix is learned separately. The final output of the second stage gives the predicted labels.

**Learning and Aligning Probabilistic Subspaces**

Given an input pair of feature vector and label \((\mathbf{x}, \mathbf{y})\) where \(\mathbf{x} \in \mathbb{R}^L\) and \(\mathbf{y} \in \{0, 1\}^L\), the feature encoder maps \(\mathbf{x}\) to a Gaussian subspace \(\mathcal{N}(\mu^\psi(\mathbf{x}), \Sigma^\psi(\mathbf{x}))\) and the label encoder maps \(\mathbf{y}\) to another Gaussian subspace \(\mathcal{N}(\mu^\phi(\mathbf{y}), \Sigma^\phi(\mathbf{y}))\). \(\phi, \psi\) are trainable parameters in the encoders. \(\mu^\psi(\mathbf{x}), \Sigma^\psi(\mathbf{x}) \in \mathbb{R}^d\) and \(\mu^\phi(\mathbf{y}), \Sigma^\phi(\mathbf{y}) \in \mathbb{R}^d \times d\), where \(d\) is the dimensionality of the latent space. \(\mathbf{z}_x, \mathbf{z}_y\) denote samples from each of these two distributions respectively.

If we only consider the label encoder-decoder branch (orange branch in Fig. 1), it models a standard (\(\beta\)-)VAE. The ELBO to optimize can be written as,

\[
E_{z \sim q_\phi} [\log p_\theta(\mathbf{y}|\mathbf{z})] - \beta D[q_\phi(\mathbf{z}|\mathbf{y})\|p(\mathbf{z})] \tag{1}
\]

The issue with this label autoencoder is the lack of connections between \(\mathbf{x}\) and \(\mathbf{z}\). Even if a good generative model is learned, prediction given \(\mathbf{x}\) is impossible since the prior \(p(\mathbf{z})\) is unrelated to \(\mathbf{x}\). Our simple fix is to replace \(P(\mathbf{z})\) with a prior distribution dependent on \(\mathbf{x}\), \(q_\phi(\mathbf{z}|\mathbf{x})\). That’s where the feature encoder comes from. The feature encoder is also a neural network parameterized by learnable \(\psi\). With the feature encoder, given the input \(\mathbf{x}\), we can sample from \(q_\psi(\mathbf{z}|\mathbf{x})\),

which is approximately equal to \(q_0(\mathbf{z}|\mathbf{y})\). The challenges thereafter are twofold: a) how to learn \(\psi\) and b) how to align \(\mathcal{N}(\mu^\psi(\mathbf{x}), \Sigma^\psi(\mathbf{x}))\) and \(\mathcal{N}(\mu^\phi(\mathbf{y}), \Sigma^\phi(\mathbf{y}))\). We propose to extend the objective function to:

\[
\frac{1}{2} (E_{\mathbf{x} \sim q_\phi} [\log p_\theta(\mathbf{y}|\mathbf{z})] + E_{\mathbf{z} \sim q_\phi} [\log p_\theta(\mathbf{y}|\mathbf{z})]) - \beta D[q_\phi(\mathbf{z}|\mathbf{y})\|q_\psi(\mathbf{z}|\mathbf{x})]
\]

The first two terms will be handled by the Multivariate Probit model in the next subsection. The last term is simply the KL divergence between two multivariate normal distributions. Since both distributions have diagonal covariance matrices, we can derive the KL loss term for \(D[q_\phi(\mathbf{z}|\mathbf{y})\|q_\psi(\mathbf{z}|\mathbf{x})]\):

\[
L_{KL}(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \beta \sum_{i=1}^{d} \log \frac{\Sigma_{\psi,i}(\mathbf{x})}{\Sigma_{\phi,i}(\mathbf{y})} - d + \sum_{i=1}^{d} \frac{\Sigma_{\psi,i}(\mathbf{x})}{\Sigma_{\phi,i}(\mathbf{y})} + \sum_{i=1}^{d} \left( \mu_{\psi,i}(\mathbf{x}) - \mu_{\phi,i}(\mathbf{y}) \right)^2 \tag{2}
\]

As in the conditional VAE, our implementation further adds an improvement to concatenate each of \(\mathbf{y}, \mathbf{z}_x, \mathbf{z}_y\) with \(\mathbf{x}\), which basically does not affect the derivations above, but uses extra information for inference.

**Reconstruction and Prediction**

The Multivariate Probit (MP) is a classic latent variable model for data with presence-absence relationships. Unlike the typical softmax transformation in most deep methods, the model maps a sample from a multivariate normal distribution to its own cumulative distribution (CDF), to bound the output range within \([0, 1]\). The major drawback of the MP model is that the integration step is intractable for large-scale data and is usually approximated with MCMC. [Chen et al., 2018] provides an alternative parallelizable sampling process for estimating the CDF. GPUs can thus expedite the estimation. We adopt this sampling method in our model, and find it works well in MPVAE.

Label information is available for both the training and testing phases in [Chen et al., 2018]. However, as a predictive model, MPVAE does not have access to labels until the final predicted targets are given, when the loss can be computed.
between the ground-truth and predicted labels for training, or the predicted targets are directly given as the output during testing. In this case, we let MPVAE calculate the integral only w.r.t. the \((0, \infty)\) region for each target(dimension). If the corresponding target is present, the CDF should be close to 1. Otherwise, it should be near 0.

**Sampling Process.** The shared decoder reads the samples \(x_s, z_b\), and outputs \(m_s, m_b \in \mathbb{R}^d\) respectively as the means in the MP model. The covariance matrix \(\Sigma_g \in \mathbb{R}^{d \times d}\) in the MP only models the dependencies among targets and is unrelated to the features. Thus \(\Sigma_g\) can be learned as a shared parameter. This stabilizes the training and helps interpret the label correlation (shown in experiments). Instead of directly sampling \(y^*\) from \(N(\mu_g, \Sigma_g)\), we sample twice by decomposing \(\Sigma_g\) to \(V + \Sigma_s\) where \(V\) is a diagonal positive definite matrix and \(\Sigma_s\) is the residual. For 2 random variables \(w \sim N(0, V)\), \(s \sim N(\mu_s, \Sigma_s)\), the difference \((w - s) \sim N(-\mu_s, \Sigma_s)\). Note that since \(V\) is diagonal, the different dimensions of \(w\) are independent. To estimate the probability of presence \(P(y^*_i \geq 0|\mu_s, \Sigma_s)\), \(i \in [1,L]\),

\[
P(y^*_i \geq 0|\mu_s, \Sigma_s) = P(y^*_i \leq 0 - \mu_s, \Sigma_s) = P(w_i - s_i \leq 0) \sim N(0, V), s \sim N(\mu_s, \Sigma_s) = \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}[P(w_i \leq s_i|s_i)] \sim N(0, V) \tag{3}
\]

where \(\Phi\) is the CDF of a univariate standard normal distribution. Since \(V\) is simply a scaling factor, by definition, \(\mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}[P(w_i \leq s_i|s_i)] = \Phi(\sqrt{s_i}/\sqrt{v_i})\).

**Binary Cross Entropy (BCE) Loss in MP.** With the efficient sampling scheme, the reconstruction losses \(\mathbb{E}_{x \sim q_0}[\log p_0(y|x)]\) and \(\mathbb{E}_{x \sim q_0}[\log p_0(y|x)]\) can be concretized. For a binary prediction task for each target, a Bernoulli likelihood assumption is valid, which leads to the binary cross entropy loss between the labels and predictions:

\[
\mathcal{L}_{CE}(y, z) = -\log p_0(y|x, \Sigma_g) = -\log p_0(y|x, \Sigma_s)
\]

\[
= -\log \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}[\prod_{i=1}^{L} \Phi(s_i)^{y_i} (1 - \Phi(s_i))^{1-y_i}]
\]

\[
= -\log \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}[\exp[\sum_{i=1}^{L} y_i \log (\Phi(s_i)) + (1 - y_i) \log (1 - \Phi(s_i))]]
\]

\[
\approx -\log \frac{1}{M} \sum_{k=1}^{M} \exp[\sum_{i=1}^{L} y_i \log (\Phi(s_{ik})) + (1 - y_i) \log (1 - \Phi(s_{ik}))]
\]

The last step is a Monte Carlo approximation. \(M\) is the preset number of samples. The log-sum-exp trick is applied for computation to avoid overflow issues. Since \(\mathbb{E}_{x \sim q_0}[\log p_0(y|x)]\) is typically approximated by a single sample from \(q_0\) and minimization is preferred for training, the objective function can be written as

\[
-\mathbb{E}_{x \sim q_0}[\log p_0(y|x)] \approx -\mathbb{E}_{x \sim q_0}[\log p_0(y|x)]
\]

**Algorithm 1 Training MPVAE**

**Input:** \(\{(x_i, y_i)\}_{i=1}^N\), batch size \(b\)

1. **for # of iterations do**
   2. Sample a minibatch \(\{(x^{(k)}, y^{(k)})\}_{k=1}^b\)
   3. Compute \(\{(\mu^k(x^{(k)}), \Sigma^k(x^{(k)})), (\mu_0^{(k)}(y^{(k)}), \Sigma_0^{(k)}(y^{(k)}))\}\)
   4. Derive \(\mathcal{L}_{KL}\) by Eq (2)
   5. Sample latent variables \(z_i^{(k)}\) \(\tilde{z}_i^{(k)}\)
   6. Decode \(\tilde{z}_i^{(k)}\) and obtain \(\{m_i^{(k)}, \{m_b^{(k)}\}\}\)
   7. Calculate the mean loss by Eq (5) for the batch
   8. Compute gradients and update parameters with Adam
   9. **return** \(\psi, \theta, \Sigma_r\)

**Ranking Loss in MP.** Ranking loss [Zhang and Zhou, 2013] is also widely used in many multi-label tasks. It is a loss to measure the correlations between positive labels and negative labels. The idea behind the ranking loss is simple: the gap between the logits for positive and negative labels should be as large as possible. Suppose \(y\) is the ground-truth label set. Let \(y_0\) denote the set of indices of negative labels and \(y_1\) the positive labels. \(s\) denotes the sample from \(N(\mu_s, \Sigma_s)\), which depends on \(z\). Thus the ranking loss on \(s\) can also be viewed as a loss on \(z\). We define the ranking loss in MP as

\[
\mathcal{L}_{RL}(y, z) = \mathcal{L}_{RL}(y, s) = \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}\left[\frac{1}{|y_0||y_1|} \sum_{(s_i, s_j) \in y_0 \times y_1} \exp[-(\Phi(s_i) - \Phi(s_j))]\right]
\]

\[
= \frac{1}{M} \sum_{k=1}^{M} \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}\left[\frac{1}{|y_0||y_1|} \sum_{(s_i, s_j) \in y_0 \times y_1} \exp[-(\Phi(s_{ik}) - \Phi(s_{jk})]]\right]
\]

The ranking loss can be defined for both \(z_s\) and \(z_b\).

**Entropy Loss in MP.** Some multi-label datasets have very sparse positive labels even if there exist multiple labels for one feature. For example, in the mirflickr dataset, the positive label rate is as low as 12%. Therefore, for sparse datasets, we add an extra entropy loss (define \(p_i = \frac{\exp(\Phi(s_i))}{\sum_{j=1}^{L} \exp(\Phi(s_j))}\)),

\[
\mathcal{L}_{Ent}(z) = \mathcal{L}_{Ent}(s) = \mathbb{E}_{s \sim N(\mu_s, \Sigma_s)}[-\log p_i] \tag{4}
\]

Entropy loss acts as a self-regularizer and only depends on the predicted values.

**Overall Loss Function**

Let \(\mathcal{L}_{prior} = \lambda_1 \mathcal{L}_{CE}(y, z_b) + \lambda_2 \mathcal{L}_{RL}(y, z_b) + \lambda_3 \mathcal{L}_{Ent}(z_b)\) and \(\mathcal{L}_{recon} = \lambda_1 \mathcal{L}_{CE}(y, z_s) + \lambda_2 \mathcal{L}_{RL}(y, z_s) + \lambda_3 \mathcal{L}_{Ent}(z_s)\). Together with the KL loss \(\mathcal{L}_{KL}\), defined in the previous subsection, the overall loss function can be described as

\[
\mathcal{L} = \mathcal{L}_{recon} + \mathcal{L}_{prior} + \beta \mathcal{L}_{KL} \tag{5}
\]

\(\mathcal{L}_{prior}\) encompasses the losses in the feature-encoder-decoder branch (blue branch) for the prior, while \(\mathcal{L}_{recon}\) encompasses the losses in the label-encoder-decoder branch (orange
Table 1: Top: example-F1 scores, Middle: micro-F1 scores, and Bottom: macro-F1 scores for different methods on all the datasets. The best scores are in bold. Each score is the average after 3 runs.

Table 2: Hamming accuracies of different methods across all the datasets. Every accuracy is the average after 3 runs.

### 4 Experiments

#### 4.1 Datasets

MPVAE is validated on 9 real-world datasets from a variety of fields including ecology, biology, images, texts, etc. The datasets are eBird [Munson et al., 2011], North American fish [Morley et al., 2018], mirflickr [Huiskes and Lew, 2008], NUS-WIDE2 [Chua et al., 2009], yeast [Nakai and Kanehisa, 1992], scene [Boutell et al., 2004], sider [Kuhn et al., 2016], bibtex [Katakis et al., 2008], and delicious [Tsoumakas et al., 2008]. eBird is a crowd-sourced bird presence-abscence dataset collected from birders' observations. North American fish (fish) is a fish distribution dataset collected from the trawlers in the North Atlantic. yeast is a biology database of the protein localization sites and sider is another database of drug side-effects. mirflickr, NUS-WIDE (nuswide), and scene datasets are from the image domain. Finally, the bibtex dataset contains a large number of BibTex files online and the delicious dataset contains web bookmarks.

The datasets represent a large variety of different scales. The dimensionality varies from 15 (eBird) to 1836 (bibtex). The number of labels ranges from 6 (scene) to 983 (delicious). The size of the datasets could be as high as 200,000 (nuswide), or as low as 1427 (sider). Most datasets are available on a public website3. The rest can be found in the related papers. If a dataset has been split a priori, we follow those divisions. Otherwise, we separate the dataset into training (80%), validation (10%) and testing (10%). The datasets are also preprocessed to fit the requirements of the input formats for the different methods. For example, we use the input features with word embeddings for LaMP.

#### 4.2 Implementation and Model Comparison

The encoders and decoder of MPVAE are parameterized by 3-layer fully connected neural networks with latent dimensionalities 512 and 256. The compared models share the same neural network structure for fair comparison, in cases where neural networks are used. The activation function in the neural networks is set to ReLU. 

3These default hyperparameter values are inherited from the existing well-trained DMVP [Chen et al., 2018] and β-VAE [Higgins et al., 2017] models. We achieve the best performance for our own model in the neighborhood of this default set of parameters via grid search. We also use grid search to find the best learning rate, learning rate decay ratio and dropout ratio hyperparameters. Note that since not all datasets have sparse labels, λ3 could be set to 0 or close to 0 in such scenarios. In practice, we found larger λ3 gives better performance because the average ranking loss of mul-

---

3We only use the 128-d eVALD features as the input.

http://mulan.sourceforge.net/datasets-mlc.html

---

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLKNN</th>
<th>MLARAM</th>
<th>SLEEC</th>
<th>C2AE</th>
<th>DMVP</th>
<th>LaMP</th>
<th>MPVVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBird</td>
<td>0.5103</td>
<td>0.5101</td>
<td>0.2578</td>
<td>0.7646</td>
<td>0.7444</td>
<td>0.5291</td>
<td>0.4768</td>
</tr>
<tr>
<td>fish</td>
<td>0.7641</td>
<td>0.5072</td>
<td>0.7790</td>
<td>0.7664</td>
<td>0.7664</td>
<td>0.5291</td>
<td>0.4768</td>
</tr>
<tr>
<td>nuswide</td>
<td>0.3420</td>
<td>0.3964</td>
<td>0.4312</td>
<td>0.4354</td>
<td>0.4537</td>
<td>0.3670</td>
<td>0.4684</td>
</tr>
<tr>
<td>yeast</td>
<td>0.6176</td>
<td>0.6292</td>
<td>0.6426</td>
<td>0.6142</td>
<td>0.6335</td>
<td>0.6242</td>
<td>0.6479</td>
</tr>
<tr>
<td>scene</td>
<td>0.6913</td>
<td>0.7166</td>
<td>0.7148</td>
<td>0.6978</td>
<td>0.6886</td>
<td>0.7279</td>
<td>0.7505</td>
</tr>
<tr>
<td>sider</td>
<td>0.7382</td>
<td>0.7222</td>
<td>0.5807</td>
<td>0.7682</td>
<td>0.7686</td>
<td>0.7662</td>
<td>0.7687</td>
</tr>
<tr>
<td>bibtex</td>
<td>0.1826</td>
<td>0.3530</td>
<td>0.4490</td>
<td>0.3346</td>
<td>0.4456</td>
<td>0.4449</td>
<td>0.4534</td>
</tr>
<tr>
<td>delicious</td>
<td>0.2590</td>
<td>0.2670</td>
<td>0.3808</td>
<td>0.3257</td>
<td>0.3639</td>
<td>0.3720</td>
<td>0.3720</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLKNN</th>
<th>MLARAM</th>
<th>SLEEC</th>
<th>C2AE</th>
<th>DMVP</th>
<th>LaMP</th>
<th>MPVVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBird</td>
<td>0.5573</td>
<td>0.5732</td>
<td>0.4124</td>
<td>0.5459</td>
<td>0.5609</td>
<td>0.5170</td>
<td>0.5933</td>
</tr>
<tr>
<td>fish</td>
<td>0.7349</td>
<td>0.5177</td>
<td>0.7563</td>
<td>0.7387</td>
<td>0.7426</td>
<td>0.7598</td>
<td>0.7648</td>
</tr>
<tr>
<td>nuswide</td>
<td>0.3679</td>
<td>0.4151</td>
<td>0.4277</td>
<td>0.4724</td>
<td>0.4912</td>
<td>0.4720</td>
<td>0.4923</td>
</tr>
<tr>
<td>yeast</td>
<td>0.6252</td>
<td>0.6350</td>
<td>0.6531</td>
<td>0.6258</td>
<td>0.6326</td>
<td>0.6407</td>
<td>0.6554</td>
</tr>
<tr>
<td>scene</td>
<td>0.6667</td>
<td>0.6927</td>
<td>0.6993</td>
<td>0.7131</td>
<td>0.6935</td>
<td>0.7156</td>
<td>0.7422</td>
</tr>
<tr>
<td>sider</td>
<td>0.7718</td>
<td>0.7535</td>
<td>0.6965</td>
<td>0.7978</td>
<td>0.7961</td>
<td>0.7977</td>
<td>0.8002</td>
</tr>
<tr>
<td>bibtex</td>
<td>0.1782</td>
<td>0.3645</td>
<td>0.4074</td>
<td>0.3884</td>
<td>0.4801</td>
<td>0.4733</td>
<td>0.4800</td>
</tr>
<tr>
<td>delicious</td>
<td>0.2639</td>
<td>0.2734</td>
<td>0.3333</td>
<td>0.3479</td>
<td>0.3791</td>
<td>0.3868</td>
<td>0.3934</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLKNN</th>
<th>MLARAM</th>
<th>SLEEC</th>
<th>C2AE</th>
<th>DMVP</th>
<th>LaMP</th>
<th>MPVVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBird</td>
<td>0.3379</td>
<td>0.4735</td>
<td>0.3625</td>
<td>0.4260</td>
<td>0.4391</td>
<td>0.3806</td>
<td>0.4936</td>
</tr>
<tr>
<td>fish</td>
<td>0.6377</td>
<td>0.4272</td>
<td>0.6570</td>
<td>0.6466</td>
<td>0.6379</td>
<td>0.6865</td>
<td>0.6925</td>
</tr>
<tr>
<td>nuswide</td>
<td>0.8683</td>
<td>0.1565</td>
<td>0.1354</td>
<td>0.1742</td>
<td>0.1633</td>
<td>0.2031</td>
<td>0.2105</td>
</tr>
<tr>
<td>yeast</td>
<td>0.4716</td>
<td>0.4484</td>
<td>0.4251</td>
<td>0.4272</td>
<td>0.4747</td>
<td>0.4802</td>
<td>0.4817</td>
</tr>
<tr>
<td>scene</td>
<td>0.6932</td>
<td>0.7131</td>
<td>0.6990</td>
<td>0.7284</td>
<td>0.7160</td>
<td>0.7449</td>
<td>0.7504</td>
</tr>
<tr>
<td>sider</td>
<td>0.6674</td>
<td>0.6491</td>
<td>0.5917</td>
<td>0.6674</td>
<td>0.6033</td>
<td>0.6684</td>
<td>0.6904</td>
</tr>
<tr>
<td>bibtex</td>
<td>0.0727</td>
<td>0.2267</td>
<td>0.2937</td>
<td>0.2680</td>
<td>0.3732</td>
<td>0.3763</td>
<td>0.3863</td>
</tr>
<tr>
<td>delicious</td>
<td>0.0526</td>
<td>0.0739</td>
<td>0.1418</td>
<td>0.1019</td>
<td>0.1806</td>
<td>0.1951</td>
<td>0.1814</td>
</tr>
</tbody>
</table>
tiple samples could provide good guidance for training the model. $\beta$ is the tradeoff between the capacity of the information bottleneck and the learnability of the decoder. In our experiments, the best values for $\beta$ are in the vicinity of 1.1.

MPVAE is compared with 6 other state-of-the-art methods for multi-label prediction. MLKNN [Zhang and Zhou, 2007] is a nearest neighbor based algorithm. Bayesian inference is applied for testing. MLARAM [Benites and Sapozhnikova, 2015] is a scalable extension to the adaptive resonance associative map neural network designed for large-scale multi-label classification. SLEEC [Bhatia et al., 2015] learns a small ensemble of embeddings preserving local distances. It makes low-rank assumptions and can be improved with neural networks (implemented for comparison). C2AE [Yeh et al., 2017] is a recently proposed approach that learns a deep latent space through an autoencoder structure. Features and labels are encoded through deep neural networks into a latent space, where the latent embeddings for features and labels are associated by deep canonical correlation analysis (DCCA). DMVP [Chen et al., 2018] is proposed for joint likelihood modeling but can be used for prediction if the trained model follows the sampling process in section 3.3 in the test phase. LaMP [Lanchantin et al., 2019] is the state-of-the-art GNN-based model for multi-label prediction. It encodes the correlations among labels to a GNN, and predicts unseen instances with the trained GNN.

The major evaluation metrics for multi-label predictions are example-based F1 (example-F1), micro-averaged F1 (micro-F1) and macro-averaged F1 (macro-F1) scores.

Example-F1 measures the proportion of true positive predictions among the aggregation of the positive ground-truth labels and positive predicted labels:

$$\frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{j=1}^{L} \frac{2y_i^j y_i^\hat{j}}{y_i^j + y_i^\hat{j} + y_i^j(1-y_i^\hat{j}) + y_i^\hat{j}(1-y_i^j)}$$

where $N_t$ is the number of test samples, $y_i^j$ is the $j$-th actual label of test sample $i$ and $\hat{y}_i^j$ is the $j$-th predicted label of test sample $i$. The top table in Table 1 shows the performances of different methods on all the datasets. Each F1 score is the average of 3 runs (same for the numbers in other tables and figures). MPVAE outperforms other methods on this metric. On average, MPVAE yields a 6% improvement compared to LaMP and a 9% improvement compared to C2AE.

Micro-F1 computes the average F1 scores over all samples:

$$\frac{1}{L} \sum_{i=1}^{L} \sum_{j=1}^{L} \frac{2y_i^j y_i^\hat{j}}{y_i^j + y_i^\hat{j} + y_i^j(1-y_i^\hat{j}) + y_i^\hat{j}(1-y_i^j)}$$

The results are given in the middle table in Table 1. MPVAE is only slightly worse than DMVP on bibtex, but compared to DMVP, MPVAE performs better by 2.5% on average. The third F1-related metric is the macro-F1 score, which is the averaged F1 score over all labels:

$$\frac{1}{L} \sum_{i=1}^{L} \sum_{j=1}^{L} \frac{2y_i^j y_i^\hat{j}}{y_i^j + y_i^\hat{j} + y_i^j(1-y_i^\hat{j}) + y_i^\hat{j}(1-y_i^j)}$$

Results on the bottom table in Table 1 illustrate that MPVAE outperforms other methods except on delicious (second best).

Besides the 3 most commonly used metrics, we test MPVAE on 2 other metrics: Hamming accuracy, and Precision@K. Hamming accuracy measures how many labels are predicted correctly:

$$\frac{1}{N_t} \frac{1}{L} \sum_{i=1}^{N_t} \sum_{j=1}^{L} \mathbb{1}[y_i^j = \hat{y}_i^j]$$

regardless of pos/neg. Accuracies are collected in Table 2. The other metric Precision@K is defined as the percent-

![Figure 2](image-url) Figure 2: Top: Precision@K evaluations on eBird. Bottom: Precision@K evaluations on sider.

![Figure 3](image-url) Figure 3: Micro-F1 scores of the methods under different noise levels. Top: fish dataset. Bottom: mirflickr dataset.
Multivariate Probit model (MPVAE) for multi-label prediction. MPVAE comprises a feature encoder, a label encoder, a shared decoder and a Multivariate Probit model. Encoders are learned for the features and labels respectively to map them to a probabilistic subspace. The samples from the subspaces are decoded under the Multivariate Probit model to give the prediction. The disentangled $\beta$-VAE module improves the label embedding learning as well as feature embedding learning. The Multivariate Probit module provides a simple and convenient way to capture the label correlations. More importantly, we claim that the learned covariance matrix in the MP model is interpretable as shown in a real-world dataset. MPVAE performs favorably against other state-of-the-art methods on 9 public datasets and remains effective under noisy settings, which verifies the usefulness and robustness of our proposed model.

Acknowledgments

This work is supported by National Science Foundation awards OIA-1936950 and CCF-1522054. We also want to thank the Cornell Lab of Ornithology and Gulf of Maine Research Institute for providing data, resources and advice.

References


[van Steenkiste et al., 2019] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In Advances in Neural


