Variational Autoencoders for Sparse and Overdispersed Discrete Data

He Zhao↵ Piyush Rai† Lan Du↵

Wray Buntine↵ Dinh Phung↵ Mingyuan Zhou†

Faculty of Information Technology, Monash University, Australia
†Department of Computer Science and Engineering, IIT Kanpur, India
‡McCombs School of Business, The University of Texas at Austin, USA

Abstract

Many applications, such as text modelling, high-throughput sequencing, and recommender systems, require analysing sparse, high-dimensional, and overdispersed discrete (count or binary) data. Recent deep probabilistic models based on variational autoencoders (VAE) have shown promising results on discrete data but may have inferior modelling performance due to the insufficient capability in modelling overdispersion and model misspecification. To address these issues, we develop a VAE-based framework using the negative binomial distribution as the data distribution. We also provide an analysis of its properties vis-à-vis other models. We conduct extensive experiments on three problems from discrete data analysis: text analysis/topic modelling, collaborative filtering, and multi-label learning. Our models outperform state-of-the-art approaches on these problems, while also capturing the phenomenon of overdispersion more effectively.1

1 INTRODUCTION

Discrete data are ubiquitous in many applications. For example, in text analysis, a collection of documents can be represented as a word-document count matrix; in recommender systems, users’ shopping history can be represented as a binary (or count) item-user matrix, with each entry indicating whether or not a user has bought an item (or its purchase count); in extreme multi-label learning problems, data samples can be tagged with a large set of labels, presented as a binary label matrix. Such kinds of data are often characterised by high-dimensionality and extreme sparsity. With the ability to handle high-dimensional and sparse matrices, Probabilistic Matrix Factorisation (PMF) (Mnih and Salakhutdinov, 2008) has been a key method of choice for such problems. PMF generates data from a suitable probability distribution, parameterised by some low-dimensional latent factors. For modelling discrete data, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Poisson Factor Analysis (PFA) (Canny, 2004; Zhou et al., 2012) are two representative models that generate data samples using the multinomial and Poisson distributions, respectively. Originally, LDA and PFA can be seen as single-layer models, whose modelling expressiveness may be limited. Several prior works have focused on extending them with hierarchical/deep Bayesian priors (Blei et al., 2010; Paisley et al., 2015; Gan et al., 2015b; Zhou et al., 2016; Lim et al., 2016; Zhao et al., 2018a). However, increasing model complexity with hierarchical priors can also complicate inference, which hinders their usefulness in analysing large-scale data.

The recent success of deep generative models such as Variational Autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) on modelling real-valued data such as images has motivated machine learning practitioners to adapt VAEs to discrete data, as done in recent works (Miao et al., 2016, 2017; Krishnan et al., 2018; Liang et al., 2018). Instead of using the Gaussian distribution as the data distribution for real-valued data, the multinomial distribution has been used for discrete data (Miao et al., 2016; Krishnan et al., 2018; Liang et al., 2018). Following Liang
et al. (2018), we refer to these VAE-based models as “MultiVAE” (Multi for multinomial). MultiVAE can be viewed as a deep nonlinear PMF model, with non-linearity modelled by deep neural networks. Compared with conventional hierarchical Bayesian models, MultiVAE enjoys better modelling capacity without sacrificing the scalability, with the use of amortized variational inference (AVI) (Rezende et al., 2014). In this work, we address some key shortcomings of these existing models; in particular, 1) insufficient capability of modelling overdispersion in count-valued data, and 2) model misspecification for binary data.

Specifically, overdispersion (i.e., variance larger than mean) describes the phenomenon that the data variability is large, a key aspect in large-scale count-valued data. For example, overdispersion in text data can behave as word burstiness (Church and Gale, 1995; Madisen et al., 2005; Doyle and Elkan, 2009; Buntine and Mishra, 2014). When a word is seen in a document, it may excite both itself and related ones. Burstiness can cause overdispersion in text data because in a document, it is common that a few bursty words occur multiple times while other words only show up once or never, resulting in high variance in the word counts of the document. As shown in Zhou (2018), the deep-seated cause of the insufficient capability of modelling overdispersion in existing PMF models is their limited ability to handle self- and cross-excitation. Specifically, in the text data example, self-excitation captures the effect that if a word occurs in a document, it is likely to occur more times in the same document; while cross-excitation models the effect that if a word such as “puppy” occurs, it will likely excite the occurrences of the related words such as “dog.” It can be shown that existing PMF models (including VAEs) with Poisson/multinomial likelihood cannot explicitly capture self- and cross-excitations.

Besides count-valued observations, binary-valued data are also prevalent in many applications, such as in collaborative filtering and graph analysis. It may not be proper to directly use multinomial or Poisson to model binary data, which is a common misspecification in many existing models. This is because a multinomial or Poisson may assign more than one count to one position, ignoring the binary nature of data. The misspecification could result in inferior modelling performance (Zhou, 2015; Zhao et al., 2017a).

To address the aforementioned challenges, we propose a Bayesian approach using the negative binomial (NB) distribution as the data distribution in a VAE-based deep generative model, so as to exploit its power in learning nonlinearity from high-dimensional and complex data spaces. We show that using NB as the likelihood can explicitly capture self-excitation, which existing VAE-based methods for discrete data are unable to deal with properly. The use of deep structures further boosts our model capacity for capturing cross-excitation. By sufficiently capturing both kinds of excitations, our proposed method is able to better handle overdispersion. Moreover, the use of NB (instead of multinomial as in Liang et al. (2018); Krishnan et al. (2018)) facilitates developing a link function between the Bernoulli and NB distributions, which enjoys better modelling performance and efficiency for binary data. Putting this together, our resulting Negative-Binomial Variational Autoencoder (NBVAE for short) is a VAE-based framework generating data with a NB distribution. This can be efficiently trained and achieves superior performance on various tasks on discrete data, including text analysis, collaborative filtering, and multi-label learning.

2 PROPOSED METHOD

We start with the introduction of our proposed NBVAE model for count-valued data in Section 2.1, and then give a detailed analysis on how self- and cross-excitations are captured in different models and why NBVAE is capable of better handling them in Section 2.2. Finally, we describe the variants of NBVAE for modelling binary data and for multi-label learning in Sections 2.3 and 2.4, respectively.

2.1 Negative-Binomial Variational Autoencoder (NBVAE)

Like the standard VAE model, NBVAE consists of two components: a decoder for the generative process and an encoder for the inference process. Here we focus on the generative process and discuss the inference procedure in Section 3. Without loss of generality, we present our model in text analysis on word counts, but the model can work with any kind of count-valued matrices. Suppose the word counts of a text corpus are stored in a $V \times N$ count matrix $Y \in \mathbb{N}^{V \times N} = \{y_1, \cdots, y_N\}$, where $N = \{0, 1, 2, \cdots\}$, $N$ and $V$ are the number of documents and size of the vocabulary, respectively. To generate the occurrences of the words for the $j^{th}$ document, $y_j \in \mathbb{N}^V$, we draw a $K$ dimensional latent representation $z_j \in \mathbb{R}^K$ from a standard multivariate normal prior. Given $z_j$, $y_j$ is drawn from a (multivariate) negative-binomial distribution with $r_j \in \mathbb{R}_+^V$ ($\mathbb{R}_+ = \{x : x \geq 0\}$) and $p_j \in (0, 1)^V$ as the parameters. Moreover, $r_j$ and $p_j$ are obtained by transforming $z_j$ from two nonlinear functions, $f_{r\theta}()$ and $f_{p\theta}()$. 

\[ y_j \sim \text{NB}(r_j, p_j) \]
parametrised by $\theta^\alpha$ and $\theta^\beta$, respectively. To generate valid NB parameters, the output of $f_{\theta^\alpha}(\cdot)$ and $f_{\theta^\beta}(\cdot)$ is fed into $\exp(\cdot)$ and $\text{sigmoid}(\cdot)$, respectively. The above generative process can be formulated as follows:

$$z_j \sim \mathcal{N}(0, I_K),$$

$$r_j = \exp\left(f_{\theta^\alpha}(z_j)\right),$$

$$p_j = \text{sigmoid}\left(f_{\theta^\beta}(z_j)\right),$$

$$y_j \sim \text{NB}(r_j, p_j).$$

Due to the equivalence of the draws between a multivariate negative-binomial distribution and a Dirichlet-multinomial distribution (Zhou, 2018, Theorem 1), the proposed NBVAE can be viewed as a deep non-linear generalisation of the models built upon the Dirichlet-Multinomial distribution, which effectively capture word burstiness in topic modelling (Doyle and Elkan, 2009; Buntine and Mishra, 2014).

### 2.2 How NBVAE Captures Self- and Cross-Excitations

We now compare our NBVAE model and other PMF models in terms of their ability of capturing self- and cross-excitations in count-valued data. For easy comparison, we present the related models with a unified framework and show our analysis in the case of generating word counts. In general, $y_j$ can be explicitly generated from the data distribution by $y_j \sim p(y_j \mid l_j)$, where $l_j \in \mathbb{R}_+^V$ is the model parameter. Alternatively, one can first generate the token of each word in document $j$, $\{w_j^1, \ldots, w_j^V\}$, and then we count the occurrences of different tokens by $y_{ij} = \sum_{i=1}^V 1\{w_j^i = v\}$, where $1(\cdot)$ is the indicator function. It can be seen that the two ways are equivalent. As shown later, in both ways, $l_j$ takes a factorised form. Now if we look at the predictive distribution of a word, $w_j^i$, conditioned on the other words’ occurrences in the corpus, $Y^{-ij}$, it can be presented as follows:

$$p(w_j^i = v \mid Y^{-ij}) \propto$$

$$\int p(w_j^i = v \mid l'_j) p(l'_j \mid Y^{-ij}) dl'_j =$$

$$\sum_{l''_j} p(w_j^i = v \mid l''_j) \text{E}_{l'_j}(Y^{-ij}) =$$

where $l'_j$ is the predictive rate computed with the parameters obtained from the posterior.

With the above notation, we can relate (as special cases) various existing models for discrete data to our proposed NBVAE model, such as PFA (Canny, 2004; Zhou et al., 2012), LDA (Blei et al., 2003), MultiVAE (Miao et al., 2016; Krishnan et al., 2018; Liang et al., 2018), and Negative-Binomial Factor Analysis (NBFA) (Zhou, 2018), as follows:

**PFA:** It is easy to see that PFA directly fits into this framework, where $p(y_j \mid l_j)$ is the Poisson distribution and $l_j = \Phi \theta_j$. Here $\Phi \in \mathbb{R}^{V \times K}$ is the factor loading matrix and $\Theta \in \mathbb{R}^{K \times N}$ is the factor score matrix. Their linear combinations determine the probability of the occurrence of $v$ in document $j$.

**LDA:** Originally, LDA explicitly assigns a topic $z_j^i \in \{1, \ldots, K\}$ to $w_j^i$, with the following process: $z_j^i \sim \text{Cat}(\theta_j/\theta_j)$ and $w_j^i \sim \text{Cat}(\phi_{z_j^i})$, where $\theta_j = \sum_{k=1}^K \theta_{kj}$ and “Cat” is the categorical distribution. By collapsing all the topics, we can derive an equivalent representation of LDA, in line with the general framework: $y_j \sim \text{Multi}(y_j, l_j)$, where $l_j = \Phi \theta_j/\theta_j$.

**MultiVAE:** MultiVAE generates data from a multinomial distribution, whose parameters are constructed by the decoder: $y_j \sim \text{Multi}(y_j, l_j)$, where $l_j = \text{softmax}(f_{\theta}(z_j))$. As shown in Krishnan et al. (2018), MultiVAE can be viewed as a nonlinear PMF model.

**NBFA:** NBFA uses a NB distribution as the data distribution, the generative process of which can be represented as: $y_j \sim \text{NB}(l_j, p_j)$, where $l_j = \Phi \theta_j$.

The above comparisons on the data distributions and predictive distributions of those models are shown in Table 1. In particular, $y_{ij}^v$ denotes the number of occurrences of word $v$ in document $j$, excluding the $i$th word. Now we can show a model’s capacity of capturing self- and cross-excitations by analysing its predictive distribution. 1) **Self-excitation:** If we compare PFA, LDA, and MultiVAE against NBFA and NBVAE, it can be seen that the latter two models with NB as their data distributions explicitly capture self-excitation via the term $y_{ij}^{-v}$ in the predictive distribution. Specifically, if $v$ appears more in document $j$, $y_{ij}^{-v}$ will be larger, leading to larger probability that $v$ shows up again. Therefore, the latter three models directly capture word burstiness with $y_{ij}^{-v}$. However, PFA, LDA, and MultiVAE cannot do this because they predict a word purely based on the interactions of the latent representations and pay less attention to the existing word frequencies. Therefore, even with deep structures, their potential for modelling self-excitation is still limited. 2) **Cross-excitation:** For the models with NB, i.e., NBVAE and NBFA, as self-excitation is explicitly captured by $y_{ij}^v$, the interactions of the latent factors are only responsible to model cross-excitation. Specifically, NBFA applies a single-layer linear combination of the latent representations, i.e., $\sum_{k=1}^K \phi_{sk} \theta_{kj}$, while NBVAE can be viewed as a deep extension of NBFA, using deep neural networks to conduct multi-layer nonlinear combinations of the latent representations, i.e., $r_{ij} = \exp(f_{\theta_j}(z_j)^{\top})$ and $p_{ij} = \text{sigmoid}(f_{\theta_j}(z_j))$. Therefore, NBVAE enjoys richer modelling capacity than NBFA on capturing cross-excitation. 3) **Summary:** We summarise our
VAEs for Sparse and Overdispersed Discrete Data

Table 1: Comparison of the data distributions, model parameters, predictive rates, and posteriors. $q(\cdot)$ denotes the encoder in VAE models, which will be introduced in Section 3.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data distribution</th>
<th>Model parameter</th>
<th>Predictive rate</th>
<th>Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFA</td>
<td>$y_j \sim \text{Poisson}(l_i)$</td>
<td>$l_i = \Phi \theta$</td>
<td>$l_i \sim \sum_{k=0}^{K} \alpha_k \theta_k$</td>
<td>$\Phi, \theta \sim p(\Phi, \theta</td>
</tr>
<tr>
<td>LDA</td>
<td>$y_j \sim \text{Multinomial}(y_{-j})$</td>
<td>$l_i = \Phi \theta_j$</td>
<td>$l_i \sim \sum_{k=0}^{K} \alpha_k \theta_k \theta_j$</td>
<td>$\Phi, \theta \sim p(\Phi, \theta</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>$y_j \sim \text{Multinomial}(y_{-j}, l_i)$</td>
<td>$l_i = \text{softmax}(f_d(z_j))$</td>
<td>$l_i \sim \text{softmax}(f_d(z_j))$</td>
<td>$y_j \sim q(z_j</td>
</tr>
<tr>
<td>NBVAE</td>
<td>$y_j \sim \text{NB}(n_i, l_i)$</td>
<td>$l_i = \Phi \theta_j$</td>
<td>$l_i \sim (\alpha_j + \sum_{k=0}^{K} \alpha_k \theta_k) \theta_j$</td>
<td>$\Phi, \theta \sim p(\Phi, \theta, p_j</td>
</tr>
<tr>
<td>NBVAE</td>
<td>$y_j \sim \text{NB}(r_j, p_j)$</td>
<td>$r_j = \exp(f_{d^1}(z_j))$</td>
<td>$r_j \sim (\alpha_j + \exp(f_{d^1}(z_j))) \cdot \text{sigmoid}(f_{d^2}(z_j))$</td>
<td>$y_j \sim q(z_j</td>
</tr>
</tbody>
</table>

Table 2: How self- and cross-excitation are captured.

<table>
<thead>
<tr>
<th>Model</th>
<th>Self-excitation</th>
<th>Cross-excitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFA</td>
<td>Single-layer structure</td>
<td>-</td>
</tr>
<tr>
<td>LDA</td>
<td>Single-layer structure</td>
<td>-</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>Multi-layer neural networks</td>
<td>-</td>
</tr>
<tr>
<td>NBVAE</td>
<td>$y_{-j}$</td>
<td>Single-layer structure</td>
</tr>
<tr>
<td>NBVAE</td>
<td>$y_{-j}$</td>
<td>Multi-layer neural networks</td>
</tr>
</tbody>
</table>

2.3 NBVAE for Binary Data

In many problems, discrete data are in binary form. For example, suppose a binary matrix $Y \in \{0, 1\}^{V \times N}$ stores the buying history of $N$ users on $V$ items, where $y_{ij} = 1$ indicates that user $j$ has brought item $i$, and vice versa. Previous models like MultiVAE (Krishnan et al., 2018; Liang et al., 2018) treat such binary data as counts, which is a model misspecification and is likely to result in inferior performance. To better model binary data, we develop a simple yet effective method that links the generative process of NBVAE with the Bernoulli distribution. Specifically, inspired by the link function used in Zhou (2015), we first generate a latent discrete intensity vector, $m_j \in \mathbb{N}^V$, from the generative process of NBVAE. Next, we generate the binary vector, $y_j$, by thresholding $m_j$ at one as:

$$z_j \sim \mathcal{N}(0, \mathbf{1}_N),$$

$$r_j = \exp(f_{d^1}(z_j)), \quad p_j = \text{sigmoid}(f_{d^2}(z_j)),$$

$$m_j \sim \text{NB}(r_j, p_j), \quad y_j = 1(m_j \geq 1).$$

Intuitively, $m_{ij}$ can be viewed as the (latent) interest of user $j$ on item $i$, and the user will buy this item if and only if $m_{ij} > 0$. As $m_j$ is drawn from NB, we do not have to explicitly generate it. We can marginalise it out and get the following data likelihood:

$$y_j \sim \text{Bernoulli}(1 - (1 - p_j)^{r_j})$$

where $r_j$ and $p_j$ have the same construction of the original NBVAE. Here we refer to this extension of NBVAE as NBVAE$_0$ (b for binary). Note that the NB distribution is a gamma mixed Poisson distribution, so the elements of $m_j$ (from a multivariate NB) can be viewed as being individually generated from a Poisson distribution. This is an important property that makes the link function applicable to NBVAE. In contrast, it is inapplicable to MultiVAE (Liang et al., 2018; Krishnan et al., 2018) as multinomials generates elements dependently.

2.4 NBVAE for Multi-label Learning

Our model admits easy extensions for other problems that require modeling discrete data. In particular, we show how to extend NBVAE for solving the multi-label learning problem, which can be formulated into a supervised task of modelling binary matrices. In a multi-label learning task, there is a large set of labels in a dataset while an individual sample is associated with a small subset of the labels. Suppose there are $N$ samples, each of which is associated with a $D$ dimensional feature vector $x_i \in \mathbb{R}^D$, and a binary label vector $y_i \in \{0, 1\}^V$. $V$ is the number of labels that can be very large, and $y_{ij} = 1$ indicates sample $j$ is labelled with $v$. The goal is to predict the labels of a sample given its features. Here the label matrix $Y \in \{0, 1\}^{V \times N}$ is a large-scale, sparse, binary matrix. Inspired by the idea of Conditional VAE (Sohn et al., 2015) and corresponding linear methods (Zhao et al., 2017a,b), we develop a conditional version of NBVAE, named NBVAE$\_c$ (c for conditional), for extreme multi-label learning$^3$. The general idea is that instead of drawing the latent representation of a sample from an uninformative prior (i.e., standard normal) as in NBVAE$_0$, we use a prior constructed with the sample’s feature in NBVAE$_0$. Specifically, we introduce a parametric function $f_{\phi}(\cdot)$ to transform the features of sample $j$ to the mean and variance of the normal prior, formulated as follows:

$$\mu_j, \sigma_j = f_{\phi}(x_j), \quad z_j \sim \mathcal{N}(\mu_j, \text{diag}(\sigma_j^2)),$$

$$r_j = \exp(f_{d^1}(z_j)), \quad p_j = \text{sigmoid}(f_{d^2}(z_j)),$$

$$m_j \sim \text{NB}(r_j, p_j), \quad y_j = 1(m_j \geq 1).$$

Note that $f_{\phi}(\cdot)$ defines $p(z_j | x_j)$, which encodes the

$^3$Other approaches that incorporate supervised/conditional information in VAEs can also be used in our model. The main purpose here is to demonstrate our model’s appealing potential for modelling binary data as well its model flexibility.
features of a sample into the prior of its latent representation. Intuitively, we name it the feature encoder. With the above construction, given the feature vector of a testing sample \( j^* \), we can feed \( x_j \) into the feature encoder to sample the latent representation, \( z_j \), then feed it into the decoder to predict its labels.

3 VARIATIONAL INFERENCE

The inference of NBVAE and NBVAE\(_c\) follows the standard amortized variational inference procedure of VAEs, where a data-dependent variational distribution \( q(z_j \mid y_j) \) (i.e., encoder) is used to approximate the true posterior, \( p(z_j \mid y_j) \), as follows:

\[
\tilde{\mu}_j, \tilde{\sigma}_j = f_o(y_j), \quad z_j \sim \mathcal{N}(\tilde{\mu}_j, \text{diag}(\tilde{\sigma}_j^2)).
\]

Given \( q(z_j \mid y_j) \), the learning objective is to maximise the Evidence Lower Bound (ELBO) of the marginal likelihood of the data: \( \mathbb{E}_{q(z_j \mid y_j)} [\log p(y_j \mid z_j)] - \text{KL} [q(z_j \mid y_j) \parallel p(z_j)] \), in terms of the decoder parameters \( \theta^e \), \( \theta^p \) and the encoder parameter \( \phi \). Here the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014) is used to sample from \( q(z_j \mid y_j) \).

For NBVAE\(_c\), there are two differences of inference from the above two models: 1) Due to the use of the informative prior conditioned on features, the Kullback-Leibler (KL) divergence on the RHS of the ELBO is calculated between two non-standard multivariate normal distributions: \( \text{KL} [q(z_j \mid y_j) \parallel p(z_j \mid x_j)] \). Thus, the KL divergence of the ELBO does not only serve as a regularizer for \( q(z_j \mid y_j) \) but also helps learn the feature encoder, \( f_o(\cdot) \). 2) In NBVAE\(_c\), the feature encoder plays an important role, as it is the one that generates the latent representations for test samples. To get more opportunities to train it, when sampling \( z_j \) in the training phase, we propose to alternatively draw it between \( z_j \sim q(z_j \mid y_j) \) and \( z_j \sim p(z_j \mid x_j) \). This enables the feature encoder to directly contribute to the generation of labels, which improves its performance in the testing phase.

4 RELATED WORK

Probabilistic matrix factorisation models for discrete data. Several well-known models fall into this category, including LDA (Blei et al., 2003) and PFA (Zhou et al., 2012), as well as their hierarchical extensions such as those in Blei et al. (2010); Teh et al. (2012); Paisley et al. (2015); Gan et al. (2015a); Ranganath et al. (2015); Henao et al. (2015); Lim et al. (2016); Zhou et al. (2016); Zhao et al. (2017b, 2018b). Among various models, the closest ones to ours are NBFA (Zhou, 2018) and non-parametric LDA (Buntine and Mishra, 2014), which generate data with the negative-binomial distribution and Dirichlet-multinomial distribution, respectively. Our models can be viewed as a deep generative extensions to them, providing better model expressiveness, flexibility, as well as inference scalability.

VAEs for discrete data. Miao et al. (2016) proposed Neural Variational Document Model (NVDM), which extended the standard VAE with multinomial likelihood for document modelling. Miao et al. (2017) further built a VAE to generate the document-topic distributions in the LDA framework. Srivastava and Sutton (2017) developed an AVI algorithm for the inference of LDA, which can be viewed as a VAE model. Card et al. (2018) introduced a general VAE framework for topic modelling with meta-data. Groueix et al. (2019) recently proposed a Gaussian mixture VAE with negative-binomial for gene-expression data, which has a different construction to ours and does not consider binary data, multi-label learning, or in-depth analysis. Burkehart and Kramer (2019) proposed a VAE framework for topic modelling, the latent representations of which are drawn from Dirichlet instead of Gaussian. Krishnan et al. (2018) found that using the standard training algorithm of VAEs in large sparse discrete data may suffer from model underfitting and proposed a stochastic variational inference (SVI) (Hoffman et al., 2013) algorithm initialised by AVI to mitigate this issue. In the collaborative filtering domain, Liang et al. (2018) noticed a similar issue and alleviated it by proposing MultiVAE with a training scheme based on KL annealing (Bowman et al., 2016). Note that NVDM, NFA, and MultiVAE are the closest ones to ours, their generative processes are very similar but their inference procedures are different. NFA is reported to outperform NVDM on text analysis (Krishnan et al., 2018) while MultiVAE is reported to have better performance than NFA on collaborative filtering tasks (Liang et al., 2018). Compared with them, we improve the modelling performance in a different way, i.e., by better capturing self- and cross-excitations so as to better handle overdispersion. Moreover, NFA and MultiVAE use the multinomial likelihood, which may not work well for binary data.

VAEs for multi-label learning. In various applications, such as image/document tagging, recommender system, ad-placement, multi-label learning/extreme classification problems have drawn a significant attention (Yu et al., 2014; Mencia and Fürnkranz, 2008; Yen et al., 2016; Bhatia et al., 2015; Rai et al., 2015; Jain et al., 2016; Prabhu et al., 2018a; You et al., 2018; Prabhu et al., 2018b). Most of the existing state-of-the-art methods adopt either multiple steps of processing of the labels and features or complex optimisation algorithms. In contrast, NBVAE\(_c\) achieves compara-
ble performance but with much simpler model structures. To our knowledge, the adaptation of VAEs to the multi-label learning area is rare, because most existing VAE models are unable to deal with large-scale sparse binary label matrices effectively.

5 EXPERIMENTS

In this section, we evaluate the proposed models on three important applications of large-scale discrete data: text analysis, collaborative filtering, and multi-label learning. We run our models multiple times and report the average results. The details of the datasets, evaluation metrics, experimental settings, and running time comparison are provided in the appendix.

5.1 Experiments on Text Analysis

Our first set of experiments is on text analysis. We consider three widely-used corpora (Srivastava et al., 2013; Gan et al., 2015a; Henao et al., 2015; Cong et al., 2017): 20 News Group (20NG), Reuters Corpus Volume (RCV), and Wikipedia (Wiki). Following Wallach et al. (2009), we report per-heldout-word perplexity of all the models, which is a widely-used metric for text analysis. Note that we use the same perplexity calculation for all the compared models, detailed in the appendix. We compare our proposed NBVAE with the following three categories of state-of-the-art models for text analysis: 1) Bayesian deep extensions of PFA and LDA: Deep latent Dirichlet allocation (DLDA) (Cong et al., 2017), Deep Poisson Factor Modelling (DPFM) (Henao et al., 2015), Deep Poisson Factor Analysis (DPFA) (Gan et al., 2015b) with different kinds of inference algorithms such as Gibbs sampling, stochastic variational inference, and stochastic gradient MCMC (SGMCMC) (Chen et al., 2014); 2) NBFA (Zhou, 2018), a recently-proposed single-layer factor analysis model with negative-binomial likelihood, where the truncated version is used and the inference is done by Gibbs sampling. 3) MultiVAE (Li et al., 2018; Krishnan et al., 2018), a recent VAE model for discrete data with the multinomial distribution as the data distribution. We use the implementation in Liang et al. (2018) for MultiVAE.

The perplexity results are shown in Table 3. Following Gan et al. (2015a); Henao et al. (2015); Cong et al. (2017), we report the performance of DLDA, DPFM, and DPFA with two and/or three hidden layers, which are the best results reported in their papers. For the VAE-based models, we vary the network architecture with one and two hidden layers and vary the depths and widths of the layers, as shown in Table 3. We observe the following from the results: 1) If we compare NBFA with DLDA, DPFM, and DPFA, it can be seen that modelling self-excitation with the NB distribution in NBFA has a large contribution to the modelling performance. 2) It can be observed that the single-layer VAE models (i.e., MultiVAEs with one layer) achieve no better results than NBFA. However, when multi-layer structures were used, VAE models largely improve their performance. This shows the increased model capacity with deeper neural networks is critical to getting better modelling performance via cross-excitation. 3) Most importantly, our proposed NBVAE significantly outperforms all the other models, which demonstrates the necessity of modelling self-excitation explicitly and modelling cross-excitation with deep structures.

In Table 3, we use 128 as the maximum layer width, following Gan et al. (2015a); Henao et al. (2015); Cong et al. (2017). Given the sizes of RCV and Wiki, we increase the maximum to 256 and 512 to study how the width of layer matters, the results of which are shown in Table 4. By looking at Tables 3 and 4, we find: 1) With the increased layer widths, all the models gained significant improvements; 2) Among all the models,

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference</th>
<th>Layers</th>
<th>RCV</th>
<th>Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLDA</td>
<td>TLASGR</td>
<td>128-64-32</td>
<td>751</td>
<td>815</td>
</tr>
<tr>
<td>DLDA</td>
<td>Gibbs</td>
<td>128-64-32</td>
<td>752</td>
<td>802</td>
</tr>
<tr>
<td>DPFM</td>
<td>SVI</td>
<td>128-64</td>
<td>818</td>
<td>961</td>
</tr>
<tr>
<td>DPFM</td>
<td>MCMC</td>
<td>128-64</td>
<td>780</td>
<td>908</td>
</tr>
<tr>
<td>DPFM-SBN</td>
<td>Gibbs</td>
<td>128-64-32</td>
<td>827</td>
<td>-</td>
</tr>
<tr>
<td>DPFM-SBN</td>
<td>SGNHT</td>
<td>128-64-32</td>
<td>846</td>
<td>1143</td>
</tr>
<tr>
<td>DPFA-RBM</td>
<td>SGNHT</td>
<td>128-64-32-32</td>
<td>896</td>
<td>920</td>
</tr>
<tr>
<td>NBFA</td>
<td>Gibbs</td>
<td>128</td>
<td>699</td>
<td>702</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>VAE</td>
<td>128-64</td>
<td>746</td>
<td>632</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>VAE</td>
<td>128</td>
<td>772</td>
<td>786</td>
</tr>
<tr>
<td>NBVAE</td>
<td>VAE</td>
<td>128-64</td>
<td>688</td>
<td>579</td>
</tr>
<tr>
<td>NBVAE</td>
<td>VAE</td>
<td>128</td>
<td>714</td>
<td>694</td>
</tr>
</tbody>
</table>

Table 3: Perplexity comparisons. “Layers” indicate the architecture of the hidden layers (for VAE models, it is the hidden layer architecture of the encoder.). Best results are in boldface. TLASGR and SGNHT are the algorithms of SGMCMC, detailed in the papers of DLDA (Cong et al., 2017) and DPFA (Gan et al., 2015a). Some results of the models with Gibbs sampling on RCV and Wiki are not reported because of the scalability issue. All the experimental settings here are consistent with those in Gan et al. (2015a); Henao et al. (2015); Cong et al. (2017).

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference</th>
<th>Layers</th>
<th>RCV</th>
<th>Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLDA</td>
<td>TLASGR</td>
<td>256-128-64</td>
<td>710</td>
<td>682</td>
</tr>
<tr>
<td>NBFA</td>
<td>Gibbs</td>
<td>256</td>
<td>649</td>
<td>-</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>VAE</td>
<td>256-128</td>
<td>587</td>
<td>589</td>
</tr>
<tr>
<td>NBVAE</td>
<td>VAE</td>
<td>256-128</td>
<td>535</td>
<td>451</td>
</tr>
<tr>
<td>DLDA</td>
<td>TLASGR</td>
<td>512-256-128</td>
<td>656</td>
<td>602</td>
</tr>
<tr>
<td>MultiVAE</td>
<td>VAE</td>
<td>512-256</td>
<td>552</td>
<td>558</td>
</tr>
<tr>
<td>NBVAE</td>
<td>VAE</td>
<td>512-256</td>
<td>512</td>
<td>445</td>
</tr>
</tbody>
</table>

Table 4: Perplexity comparisons with larger layer width.
small entropy. This is because self-excitation is better. Our model is closer to the origin of coordinates, which shows that, in our proposed NBVAE, the centre of the histogram is closer to the origin of coordinates, which means that a smaller document would relatively spread over all the words. Essentially, it means that a smaller document is expected to focus on. Therefore, if a model takes self-excitation into account, the entropy of a document is expected to be small, as the model’s predictive distribution will put large mass on the words that do not take into account self-excitation. In contrast, if a model does not take into account self-excitation, its predictive distribution of a document can be interpreted as the effective number of unique words that the document is expected to focus on. Therefore, if a model takes self-excitation into account, the entropy of a document is expected to be small, as the model’s predictive distribution will put large mass on the words that already occur in the document. In contrast, if a model does not take into account self-excitation, its predictive distribution of a document would relatively spread over all the words. Essentially, it means that a smaller entropy indicates better capture of self-excitation. After computing the entropy of all the test documents, we plot the histograms in Figure 1. It can be observed that, in our proposed NBVAE, the centre of the histogram is closer to the origin of coordinates, which shows that our model has more documents that with small entropy. This is because self-excitation is better captured by our model.

5.2 Experiments on Collaborative Filtering

Our second set of experiments is on collaborative filtering, where the task is to recommend items to users using their clicking history. We evaluate our models’ performance on four user-item consumption datasets: MovieLens-10M (ML-10M), MovieLens-20M (ML-20M), Netflix Prize (Netflixt), and Million Song Dataset (MID) (Bertin-Mahieux et al., 2011). Following Liang et al. (2018), we report two evaluation metrics: Recall@R and the truncated normalized discounted cumulative gain (NDCG@R). Figure 2 shows the NDCG@R and Recall@R of the models on the four datasets, where we used R ∈ {1, 5, 10, 20, 50}. In general, our proposed NBVAEb outperforms the baselines (i.e., MultiVAE and MultiDAE) on almost all of the datasets. In particular, the margin is notably large while the R value is small, such as 1 or 5. It indicates that the top-ranked items in NBVAEb are always more accurate than those in MultiVAE and MultiDAE. This fact is also supported by the large gap of NDCG@R between NBVAEb and the two baselines, as NDCG@R penalises the true items that are ranked low.

To further demonstrate the benefit of using NBVAEb, we compare it with NBVAE on the ML-10M dataset, where the latter treats binary data as count-valued data. The results of NDCG@R and Recall@R on ML-10M are shown in Table 5, where NBVAEb’s results are significantly better than NBVAE, showing the necessity of dealing with binary data separately from count-valued data.

5.3 Experiments on Multi-Label Learning

Finally, we compare NBVAEb with several recent advances in multi-label learning, including LEML (Yu et al., 2014), PfastXML (Jain et al., 2016), PD-Sparse (Yen et al., 2016), and GenEMIL (Jain et al., 2017). We use three multi-label learning benchmark datasets: Delicious (Tsoumakas et al., 2008), Mediumill (Snoek et al., 2006), and EURLex (Mencia and Fürnkranz, 2008). We report Precision@R (R ∈ {1, 3, 5}), which is a widely-used evaluation metric for
multi-label learning, following Jain et al. (2017). For the baselines, the reported results are the publicly known best ones.

Table 6 shows the performance comparisons on the multi-label learning datasets. It can be observed that the proposed NBVAE outperforms the others on the prediction precision, showing its promising potential on multi-label learning problems. Note that the baselines are specialised to the multi-label learning problem, many of which either take multiple steps of processing of the labels and features or use complex optimisation algorithms. Compared with those models, the model simplicity of NBVAE is an appealing advantage, which gives VAE models great potential in multi-label learning and related problems.

6 CONCLUSION

In this paper, we have focused on analysing and addressing two shortcomings of probabilistic modelling on large-scale, sparse, discrete data: insufficient capability of modelling overdispersion in count-valued data and model misspecification in binary data. To address this we use the NB distribution, which conquers overdispersion in count-valued data by better capturing self- and cross-excitations as well as deals with binary data in a proper and effective way.

The advantages of our methods on count-valued and binary data have been demonstrated by their superior performance on text analysis and collaborative filtering. Remarkably, due to the great model capacity and flexibility of modelling binary data, our model can be extended to multi-label learning, obtaining state-of-the-art performance in the comparison with many advanced custom-designed algorithms. Extensive experiments have shown the great potential of our models on modelling various kinds of discrete data.

Finally, we believe that our methods are excellent examples of hybridising the valuable knowledge in Bayesian probabilistic modelling with newly-developed deep learning techniques, which would inspire considerable future work in various areas.
Acknowledgements: MZ was supported in part by NSF IIS-1812699. PR acknowledges support from Visvesvaraya Young Faculty Fellowship. DP gratefully acknowledges funding support from the Australian Research Council under the ARC DP160109394 Project.

References


VAEs for Sparse and Overdispersed Discrete Data


Appendix for “Variational Autoencoders for Sparse and Overdispersed Discrete Data”

He Zhao⇤ Piyush Rai† Lan Du⇤

Wray Buntine⇤ Dinh Phung⇤ Mingyuan Zhou‡

⇤Faculty of Information Technology, Monash University, Australia
†Department of Computer Science and Engineering, IIT Kanpur, India
‡McCombs School of Business, The University of Texas at Austin, USA

1 EXPERIMENTS ON TEXT ANALYSIS

1.1 Datasets

The statistics of the datasets used in the text analysis experiments are shown in Table 1. The 20NG and RCV datasets were downloaded from the code repository of Gan et al. (2015)1. The Wiki dataset was downloaded from Wikipedia using the scripts provided in Hoffman et al. (2010).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ntrain</th>
<th>Ntest</th>
<th>V</th>
<th>#Nonzeros</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>11,315</td>
<td>7,531</td>
<td>2,000</td>
<td>774,984</td>
<td>0.0343</td>
</tr>
<tr>
<td>RCV</td>
<td>794,414</td>
<td>10,000</td>
<td>10,000</td>
<td>58,637,816</td>
<td>0.0074</td>
</tr>
<tr>
<td>Wiki</td>
<td>10,000,000</td>
<td>1,000</td>
<td>7,702</td>
<td>82,311,745</td>
<td>0.0107</td>
</tr>
</tbody>
</table>

1Our perplexity calculation is the same with the ones in Gan et al. (2015); Henao et al. (2015); Cong et al. (2017), but different from the ones in Miao et al. (2017) 2016; Krishnan et al. (2018), which use ELBO obtained from all the words of a testing document without splitting it. The results of Miao et al. (2017) 2016; Krishnan et al. (2018) can only be compared with models with variational inference.

1.2 Evaluation Metric

We report per-heldout-word perplexity of all the models, which is a widely-used metric for text analysis. Following the approach in Wallach et al. (2009), after training a model with the training documents, we randomly select some words as the observed words and use the remaining words as the unobserved words in each testing document, then use the observed words to estimate the predictive probability, and finally compute the perplexity of the unobserved words. Specifically, suppose that the matrix of the testing documents is \( Y^* \in \mathbb{N}^{V \times N_{test}} \), which is split into the observed word matrix \( Y^{*o} \in \mathbb{N}^{V \times N_{test}} \) and the unobserved word matrix \( Y^{*u} \in \mathbb{N}^{V \times N_{test}} \), where \( Y^* = Y^{*o} + Y^{*u} \). The predictive rates of the testing documents are estimated with \( Y^{*o} \) and used to compute the perplexity of \( Y^{*u} \), detailed as follow:

\[
\text{Perplexity} = \exp \left( \frac{1}{y^{*u}_{ij}} \sum_{v} y^{*u}_{ij} \log \frac{l_{ij}}{l_{v}} \right),
\]

where \( y^{*u}_{ij} = \sum_{v} y^{*u}_{ij} \). Note that \( l_{ij} \) is the predictive rate, whose derivation is model specific shown in Table

1.3 Experimental Settings

In the experiments of text analysis, in terms of model settings of our proposed models, following Liang et al. (2018), we basically use the settings as for MultiVAE. Specifically, for both MultiVAE and NBVAE,

- We apply the fully connected multi-layer perceptrons (MLP) with tanh as the nonlinear activation function between the layers of the encoder and the decoder.

Table 1: Statistics of the datasets in text analysis. \( N_{train} \): number of training instances, \( N_{test} \): number of test instances. The number of nonzeros and density are computed of each whole dataset.

https://github.com/zhegan27/dpfa_icml2015
We use the same network architecture for the two parametric functions in the decoder, \( f_o(\cdot) \) and \( f_u(\cdot) \).

The architecture of \( f_o(\cdot) \) is symmetric to those of \( f_v(\cdot) \) and \( f_w(\cdot) \). For example, if we use \([32, 64, 128]\) as the architecture of the hidden layers for the decoder, then \( K = 32 \) is the dimension of the latent representations and the architecture of the hidden layers for the encoder would be \([128, 64, 32]\).

The output layers of the encoder and decoder have no activation function.

We set the batch size to 500 and 2000 for 20NG and the other two larger datasets, respectively.

The number of training epochs was set to 800 and the perplexity averaged over multiple samples in the last iteration of the training phrase, whereas for the VAE-based models, we report the perplexity computed with the parameters (the encoder and decoder) in the last iteration of the training phrase, whereas for models with MCMC sampling (e.g., NBFA), we report the perplexity averaged over multiple samples in the collection iterations.

## 2 EXPERIMENTS ON COLLABORATIVE FILTERING

### 2.1 Datasets


### 2.2 Evaluation Metrics

Two ranking-based metrics are used, which are Recall@R and the truncated normalized discounted cumulative gain \( \text{NDCG}_R @ R \). To compute those metrics, following [Liang et al.](https://labrosa.ee.columbia.edu/millionsong/), we first estimate the predictive rate \( \hat{y}_{uj} \) of user \( j \) given the observed items \( y_{uj}^o \), and then rank the unobserved items \( y_{uj}^u \) by sorting \( \hat{y}_{uj} \). The metrics are computed as follows:

\[
\text{Recall}_R = \frac{1}{R} \sum_{r=1}^{R} \mathbb{1} \left( \omega(r) = 1 \right) \frac{\mathbb{1} \left( \omega(r) = 1 \right)}{\min(R, y_{uj}^R)}.
\]

\[
\text{DCG}_R = \frac{1}{\log(r + 1)} \sum_{r=1}^{R} 2^r \mathbb{1} \left( \omega(r) = 1 \right) - 1.
\]

where \( \omega(r) \in \{1, \ldots, V\} \) is the item at rank \( r \), obtained by sorting the predictive rate of the user; \( 1 \left( \omega(r) = 1 \right) \) indicates whether the item is actually clicked on by user \( j \); \( \text{NDCG}_R \) is computed by linearly normalising \( \text{DCG}_R \) into \([0, 1]\). Intuitively, \( \text{Recall}_R \) measures the number of the \( R \) predicted items that are within the set of the ground-truth items but does not consider the item rank in \( R \), while \( \text{NDCG}_R \) assigns larger discounts to lower ranked items. In the experiments, we use the code provided in [Liang et al.](https://labrosa.ee.columbia.edu/millionsong/) to compute the above two metrics. Moreover, we report the testing performance of the models with the best \( \text{NDCG}@50 \) on the validation set.

### 2.3 Experimental Settings

For NBVAE and NBVAE\(_o\), we used the same settings as in the text analysis experiments, except that:

- The batch size is set to 500 for all the datasets.
- We use two hidden layers in the encoder with \([200-600]\) (The architectures of the two parametric functions in the decoder are symmetric to that in the encoder).
- Following MultiVAE, we use the annealing cap \( \beta \), which is set to 0.2, detailed in [Liang et al.](https://labrosa.ee.columbia.edu/millionsong/).

Note that all the above settings are consistent with those in [Liang et al.](https://labrosa.ee.columbia.edu/millionsong/) The original code of MultiVAE and MultiDAE and their best settings provided by the authors are used in the comparison.
Table 3: The statistics of the datasets used in the experiments. $N_{\text{train}}$: number of training instances, $N_{\text{test}}$: number of test instances, $D$: number of features, $V$: number of labels.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$N_{\text{train}}$</th>
<th>$N_{\text{test}}$</th>
<th>$D$</th>
<th>$V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delicious</td>
<td>12920</td>
<td>3185</td>
<td>500</td>
<td>983</td>
</tr>
<tr>
<td>Mediamill</td>
<td>30993</td>
<td>12914</td>
<td>120</td>
<td>101</td>
</tr>
<tr>
<td>EURLex</td>
<td>15539</td>
<td>3809</td>
<td>5000</td>
<td>3993</td>
</tr>
</tbody>
</table>

3 EXPERIMENTS ON MULTI-LABEL LEARNING

3.1 Datasets

All the datasets are downloaded from [http://manikvarma.org/downloads/XC/XMLRepository.html](http://manikvarma.org/downloads/XC/XMLRepository.html) and the statistics of the datasets are shown in Table 3.

3.2 Evaluation Metrics

We report Precision@$R$ ($R \in \{1, 3, 5\}$), which is a widely-used ranking-based evaluation metric for multi-label learning, following Jain et al. (2017). To compute this metric, after training NBVAE, given the feature vector of a testing sample $j^*$, we can feed $x_j^*$ into the feature encoder to sample the latent representation, $z_j^*$, then feed it into the decoder to get the predictive rate $l_j^*$. With the predictive rate, we can rank the labels and compute Precision@$R$, which is similar to the computation of Recall and NDCG used in collaborative filtering.

3.3 Experimental Settings

For NBVAE in multi-label learning, we used the same settings as NBVAE in the text analysis experiments, specifically:

- In the Delicious and Mediamill datasets, we use [200-600] for two hidden layers in the encoder and for EURLex, we use one hidden layer in the encoder with 600 units.
- We use relu as the activation function for Delicious and EURLex and tanh as the activation function for Mediamill.

4 RUNNING SPEED COMPARISON

Here we compare the running speeds of NBVAE and MultiVAE.

Table 4: Running time (seconds) per iteration on the text datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>20NG</th>
<th>RCV</th>
<th>Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiVAE</td>
<td>0.12</td>
<td>48.21</td>
<td>42.18</td>
</tr>
<tr>
<td>NBVAE</td>
<td>0.17</td>
<td>49.49</td>
<td>48.57</td>
</tr>
</tbody>
</table>

Table 5: Running time (seconds) per iteration on the collaborative filtering datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>ML-10M</th>
<th>ML-20M</th>
<th>Netflix</th>
<th>MSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiVAE</td>
<td>2.91</td>
<td>14.79</td>
<td>46.90</td>
<td>105.33</td>
</tr>
<tr>
<td>NBVAE</td>
<td>3.47</td>
<td>17.54</td>
<td>52.12</td>
<td>124.43</td>
</tr>
</tbody>
</table>

Analytically, NBVAE has additional computational cost over MultiVAE in two aspects: (1) there are two parameters $p$ and $r$ in NBVAE so we use two decoders $f_p$ and $f_r$ (one more than MultiVAE); (2) an additional term in the log likelihood for zeros, as pointed out by the reviewer. In terms of (2), according to our description under Eq. (3) of the main paper, we just need to compute the Bernoulli parameter $\text{temp} = 1 - (1 - p_j)^r$ and then compute $y_j \log \text{temp} + (1 - y_j) \log (1 - \text{temp})$, where the RHS is for zeros.

Empirically, as our models are implemented in TensorFlow running on GPUs, the overhead as compared to MultiVAE is not large. Here we report the running time (seconds) per iteration of NBVAE (NBVAE) and MultiVAE on the text and collaborative filtering datasets in Table 4 and Table 5 respectively. All the experiments are implemented in TensorFlow and with the same settings in the paper, and run on the same machine with Nvidia Tesla P100 GPU.

In addition, in Figure 1, we plot the validation set performance of NBVAE (NBVAE) and MultiVAE in the training phase.

References


Figure 1: Performance of NBVAE and MultiVAE on the validation set during training.


