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).

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 $\mathbf{Y}^* \in \mathbb{N}^{V \times N_{\text{test}}}$, which is split into the observed word matrix $\mathbf{Y}^{*\text{o}} \in \mathbb{N}^{V \times N_{\text{test}}}$ and the unobserved word matrix $\mathbf{Y}^{*\text{u}} \in \mathbb{N}^{V \times N_{\text{test}}}$, where $\mathbf{Y}^* = \mathbf{Y}^{*\text{o}} + \mathbf{Y}^{*\text{u}}$. The predictive rates of the testing documents are estimated with $\mathbf{Y}^{*\text{o}}$ and used to compute the perplexity of $\mathbf{Y}^{*\text{u}}$, detailed as follows[2]:

$$\text{Perplexity} = \exp - \left(\frac{1}{y_{\cdot\cdot}^{*\text{u}}} \sum_{j}^{N_{\text{test}}} \sum_{v}^{V} y_{vj}^{*\text{u}} \log \frac{l_{vj}}{l_{\cdot j}} \right), \quad (1)$$

where $y_{\cdot\cdot}^{*\text{u}} = \sum_j^{N_{\text{test}}} \sum_v^V y_{vj}^{*\text{u}}$. Note that l_{vj} is the predictive rate, whose derivation is model specific shown in Table

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.

Dataset	N_{train}	N_{test}	V	#Nonzeros	Density
20NG	11,315	7,531	2,000	774,984	0.0343
RCV	794,414	10,000	10,000	58,637,816	0.0074
Wiki	10,000,000	1,000	7,702	82,311,745	0.0107

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.

[1] https://github.com/zhegan27/dpfa_icml2015

[2] Our 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.

- We use the same network architecture for the two parametric functions in the decoder, $f_{\theta^r}(\cdot)$ and $f_{\theta^p}(\cdot)$.
- The architecture of $f_\phi(\cdot)$ is symmetric to those of $f_{\theta^r}(\cdot)$ and $f_{\theta^p}(\cdot)$. 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 optimisation of the VAE models was done by Adam (Kingma and Ba, 2014) with 0.003 as the learning rate.
- We use the same KL annealing procedure mentioned in the MultiVAE paper (Liang et al., 2018).

For the baselines, we use the original model settings provided in the code published by the authors. 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

ML-10M and ML-20M are downloaded from `https://grouplens.org/datasets/movielens/`; Netflix is downloaded from `http://www.netflixprize.com/`; MSD (Bertin-Mahieux et al., 2011) is downloaded from `https://labrosa.ee.columbia.edu/millionsong/`. All the datasets are preprocessed and binarised by the Python code provided by Liang et al. (2018), using the same settings described in the paper. The statistics of the datasets are shown in Table 1. Note that following Liang et al. (2018), we also generate a validation set with the same size of the testing set.

2.2 Evaluation Metrics

Two ranking-based metrics are used, which are Recall@R and the truncated normalized discounted cumulative gain (NDCG@R). To compute those metrics, following Liang et al. (2018), we first estimate the predictive rate $\boldsymbol{l}'_j$ of user j given the observed items $\boldsymbol{y}^{*o}_j$, and then rank the unobserved items $\boldsymbol{y}^{*u}_j$ by sorting $\boldsymbol{l}'_j$. The metrics are computed as follows:

$$\text{Recall@}R = \frac{\sum_{r=1}^{R} \mathbf{1}\left(y^{*u}_{\omega(r)j} = 1\right)}{\min(R, y^{*u}_{\cdot j})}, \tag{2}$$

$$\text{DCG@}R = \sum_{r=1}^{R} \frac{2^{\mathbf{1}\left(y^{*u}_{\omega(r)j}=1\right)} - 1}{\log(r+1)}, \tag{3}$$

where $\omega(r) \in \{1, \cdots, V\}$ is the item at rank r, obtained by sorting the predictive rate of the user; $\mathbf{1}\left(y^{*u}_{\omega(r)j} = 1\right)$ indicates whether the item is actually clicked on by user j; NDCG@R is computed by linearly normalising DCG@R into $[0, 1]$. Intuitively, 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 NDCG@R assigns larger discounts to lower ranked items. In the experiments, we use the code provided in (Liang et al., 2018) to compute the above two metrics. Moreover, we report the testing performance of the models with the best NDCG@50 on the validation set.

Table 2: Statistics of the datasets in collaborative filtering. N_{train}: number of training instances, N_{test}: number of test instances. The number of nonzeros and density are computed of each whole dataset.

Dataset	N_{train}	N_{test}	V	#Nonzeros	Density
ML-10M	49,167	10,000	10,066	4,131,372	0.0059
ML-20M	116,677	10,000	20,108	9,128,733	0.0033
Netflix	383,435	40,000	17,769	50,980,816	0.0062
MSD	459,330	50,000	36,716	29,138,887	0.0014

2.3 Experimental Settings

For NBVAE and NBVAE$_{\text{b}}$, 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 β, which is set to 0.2, detailed in Liang et al. (2018).

Note that all the above settings are consistent with those in (Liang et al., 2018) 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_{train}: number of training instances, N_{test}: number of test instances, D: number of features, V: number of labels.

Dataset	N_{train}	N_{test}	D	V
Delicious	12920	3185	500	983
Mediamill	30993	12914	120	101
EURLex	15539	3809	5000	3993

3 EXPERIMENTS ON MULTI-LABEL LEARNING

3.1 Datasets

All the datasets are downloaded from `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$_{\text{c}}$, given the feature vector of a testing sample j^*, we can feed $\boldsymbol{x}_{j^*}$ into the feature encoder to sample the latent representation, $\boldsymbol{z}_{j^*}$, then feed it into the decoder to get the predictive rate $\boldsymbol{l'}_{j^*}$. 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$_{\text{c}}$ in multi-label learning, we used the same settings as NBVAE$_{\text{b}}$ in the text analysis experiments, specifically:

- In the Delicious and Mediamill datasets, we use [200-600] for two hidden layers in the encode 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.

Model	20NG	RCV	Wiki
MultiVAE	0.12	48.21	42.18
NBVAE	0.17	49.49	48.57

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

Model	ML-10M	ML-20M	Netflix	MSD
MultiVAE	2.91	14.79	46.90	105.33
NBVAE	3.47	17.54	52.12	124.43

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_{θ^r} and f_{θ^p} (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 temp $= 1 - (1 - p_j)^{r_j}$ 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$_{\text{b}}$) 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$_{\text{b}}$) and MultiVAE in the training phase.

References

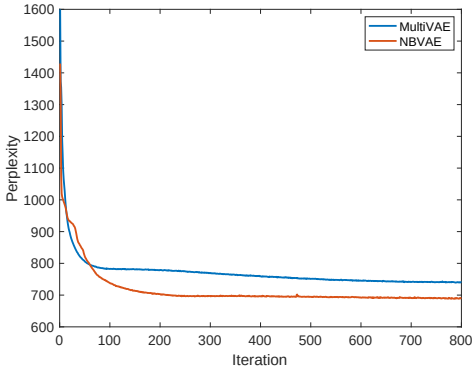
T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *International Conference on Music Information Retrieval*, 2011.

Y. Cong, B. Chen, H. Liu, and M. Zhou. Deep latent Dirichlet allocation with topic-layer-adaptive stochastic gradient Riemannian MCMC. In *ICML*, pages 864–873, 2017.

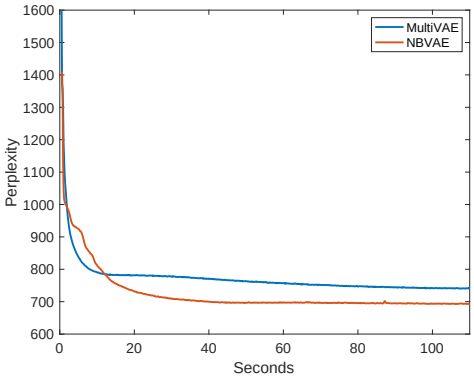
Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. Scalable deep Poisson factor analysis for topic modeling. In *ICML*, pages 1823–1832, 2015.

R. Henao, Z. Gan, J. Lu, and L. Carin. Deep Poisson factor modeling. In *NIPS*, pages 2800–2808, 2015.

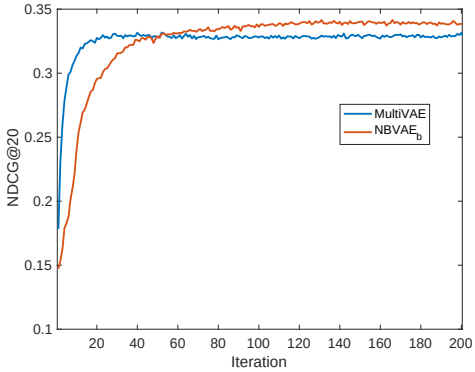
(a) Perplexity on 20NG over iterations.



(b) Perplexity on 20NG over seconds.



(c) NDCG@20 on ML-10M over iterations.



(d) NDCG@20 on ML-10M over seconds.

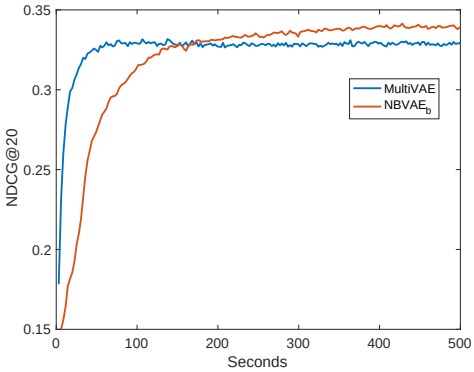


Figure 1: Performance of NBVAE and MultiVAE on the validation set during training.

M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent Dirichlet allocation. In *NIPS*, pages 856–864, 2010.

V. Jain, N. Modhe, and P. Rai. Scalable generative models for multi-label learning with missing labels. In *ICML*, pages 1636–1644, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R. Krishnan, D. Liang, and M. Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *AISTATS*, pages 143–151, 2018.

D. Liang, R. G. Krishncan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *WWW*, pages 689–698, 2018.

Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *ICML*, pages 1727–1736, 2016.

Y. Miao, E. Grefenstette, and P. Blunsom. Discovering discrete latent topics with neural variational inference. In *ICML*, pages 2410–2419, 2017.

H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *ICML*, pages 1105–1112, 2009.