Variational Autoencoders for Collaborative Filtering

Dawen Liang
Netflix
Los Gatos, CA
dliang@netflix.com

Rahul G. Krishnan
MIT
Cambridge, MA
rahulgk@mit.edu

Matthew D. Hoffman
Google AI
San Francisco, CA
mhoffman@google.com

Tony Jebara
Netflix
Los Gatos, CA
tjebara@netflix.com

ABSTRACT

We extend variational autoencoders (VAEs) to collaborative filtering for implicit feedback. This non-linear probabilistic model enables us to go beyond the limited modeling capacity of linear factor models which still largely dominate collaborative filtering research. We introduce a generative model with multinomial likelihood and use Bayesian inference for parameter estimation. Despite widespread use in language modeling and economics, the multinomial likelihood receives less attention in the recommender systems literature. We introduce a different regularization parameter for the learning objective, which proves to be crucial for achieving competitive performance. Remarkably, there is an efficient way to tune the parameter using annealing. The resulting model and learning algorithm has information-theoretic connections to maximum entropy discrimination and the information bottleneck principle. Empirically, we show that the proposed approach significantly outperforms several state-of-the-art baselines, including two recently-proposed neural network approaches, on several real-world datasets. We also provide extended experiments comparing the multinomial likelihood with other commonly used likelihood functions in the latent factor collaborative filtering literature and show favorable results. Finally, we identify the pros and cons of employing a principled Bayesian inference approach and characterize settings where it provides the most significant improvements.

KEYWORDS

Recommender systems, collaborative filtering, implicit feedback, variational autoencoder, Bayesian models

ACM Reference Format:
Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of The 2018 Web Conference (WWW 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186150

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186150

1 INTRODUCTION

Recommender systems are an integral component of the web. In a typical recommendation system, we observe how a set of users interacts with a set of items. Using this data, we seek to show users a set of previously unseen items they will like. As the web grows in size, good recommendation systems will play an important part in helping users interact more effectively with larger amounts of content.

Collaborative filtering is among the most widely applied approaches in recommender systems. Collaborative filtering predicts what items a user will prefer by discovering and exploiting the similarity patterns across users and items. Latent factor models [13, 19, 38] still largely dominate the collaborative filtering research literature due to their simplicity and effectiveness. However, these models are inherently linear, which limits their modeling capacity. Previous work [27] has demonstrated that adding carefully crafted non-linear features into the linear latent factor models can significantly boost recommendation performance. Recently, a growing body of work involves applying neural networks to the collaborative filtering setting with promising results [14, 41, 51, 54].

Here, we extend variational autoencoders (VAEs) [24, 37] to collaborative filtering for implicit feedback. VAEs generalize linear latent-factor models and enable us to explore non-linear probabilistic latent-variable models, powered by neural networks, on large-scale recommendation datasets. We propose a neural generative model with multinomial conditional likelihood. Despite being widely used in language modeling and economics [5, 30], multinomial likelihoods appear less studied in the collaborative filtering literature, particularly within the context of latent-factor models. Recommender systems are often evaluated using ranking-based measures, such as mean average precision and normalized discounted cumulative gain [21]. Top-N ranking loss is difficult to optimize directly and previous work on direct ranking loss minimization resorts to relaxations and approximations [49, 50]. Here, we show that the multinomial likelihoods are well-suited for modeling implicit feedback data, and are a closer proxy to the ranking loss relative to more popular likelihood functions such as Gaussian and logistic.

Though recommendation is often considered a big-data problem (due to the huge numbers of users and items typically present in a recommender system), we argue that, in contrast, it represents a uniquely challenging "small-data" problem: most users only interact with a tiny proportion of the items and our goal is to collectively

make informed inference about each user's preference. To make use of the sparse signals from users and avoid overfitting, we build a probabilistic latent-variable model that shares statistical strength among users and items. Empirically, we show that employing a principled Bayesian approach is more robust regardless of the scarcity of the data.

Although VAEs have been extensively studied for image modeling and generation, there is surprisingly little work applying VAEs to recommender systems. We find that two adjustments are essential to getting state-of-the-art results with VAEs on this task:

- First, we use a multinomial likelihood for the data distribution. We show that this simple choice realizes models that outperform the more commonly used Gaussian and logistic likelihoods.
- Second, we reinterpret and adjust the standard VAE objective, which we argue is over-regularized. We draw connections between the learning algorithm resulting from our proposed regularization and the information-bottleneck principle and maximum-entropy discrimination.

The result is a recipe that makes VAEs practical solutions to this important problem. Empirically, our methods significantly outperform state-of-the-art baselines on several real-world datasets, including two recently proposed neural-network approaches.

2 METHOD

We use $u \in \{1, \dots, U\}$ to index users and $i \in \{1, \dots, I\}$ to index items. In this work, we consider learning with implicit feedback [19, 34]. The user-by-item interaction matrix is the click[1] matrix $\mathbf{X} \in \mathbb{N}^{U \times I}$. The lower case $\mathbf{x}_u = [x_{u1}, \dots, x_{uI}]^\top \in \mathbb{N}^I$ is a bag-of-words vector with the number of clicks for each item from user u. For simplicity, we binarize the click matrix. It is straightforward to extend it to general count data.

2.1 Model

The generative process we consider in this paper is similar to the deep latent Gaussian model [37]. For each user u, the model starts by sampling a K-dimensional latent representation $\mathbf{z}_u$ from a standard Gaussian prior. The latent representation $\mathbf{z}_u$ is transformed via a non-linear function $f_\theta(\cdot) \in \mathbb{R}^I$ to produce a probability distribution over I items $\pi(\mathbf{z}_u)$ from which the click history $\mathbf{x}_u$ is assumed to have been drawn:

$$\begin{aligned} \mathbf{z}_u &\sim \mathcal{N}(0, \mathbf{I}_K), \quad \pi(\mathbf{z}_u) \propto \exp\{f_\theta(\mathbf{z}_u)\}, \\ \mathbf{x}_u &\sim \mathrm{Mult}(N_u, \pi(\mathbf{z}_u)). \end{aligned} \tag{1}$$

The non-linear function $f_\theta(\cdot)$ is a multilayer perceptron with parameters θ. The output of this transformation is normalized via a softmax function to produce a probability vector $\pi(\mathbf{z}_u) \in \mathbb{S}^{I-1}$ (an $(I-1)$-simplex) over the entire item set. Given the total number of clicks $N_u = \sum_i x_{ui}$ from user u, the observed bag-of-words vector $\mathbf{x}_u$ is assumed to be sampled from a multinomial distribution with probability $\pi(\mathbf{z}_u)$. This generative model generalizes the latent-factor model — we can recover classical matrix factorization [38] by setting $f_\theta(\cdot)$ to be linear and using a Gaussian likelihood.

[1] We use the verb "click" for concreteness; this can be any type of interaction, including "watch", "purchase", or "listen".

The log-likelihood for user u (conditioned on the latent representation) is:

$$\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u) \stackrel{c}{=} \sum_i x_{ui} \log \pi_i(\mathbf{z}_u). \tag{2}$$

This multinomial likelihood is commonly used in language models, e.g., latent Dirichlet allocation [5], and economics, e.g., multinomial logit choice model [30]. It is also used in the cross-entropy loss[2] for multi-class classification. For example, it has been used in recurrent neural networks for session-based sequential recommendation [8, 15, 16, 42, 45] and in feedward neural networks applied to Youtube recommendation [9]. The multinomial likelihood is less well studied in the context of latent-factor models such as matrix factorization and autoencoders. A notable exception is the collaborative competitive filtering (CCF) model [53] and its successors, which take advantage of more fine-grained information about what options were presented to which users. (If such information is available, it can also be incorporated into our VAE-based approach.)

We believe the multinomial distribution is well suited to modeling click data. The likelihood of the click matrix (Eq. 2) rewards the model for putting probability mass on the non-zero entries in $\mathbf{x}_u$. But the model has a limited budget of probability mass, since $\pi(\mathbf{z}_u)$ must sum to 1; the items must compete for this limited budget [53]. The model should therefore assign more probability mass to items that are more likely to be clicked. To the extent that it can, it will perform well under the top-N ranking loss that recommender systems are commonly evaluated on.

By way of comparison, we present two popular choices of likelihood functions used in latent-factor collaborative filtering: Gaussian and logistic likelihoods. Define $f_\theta(\mathbf{z}_u) \equiv [f_{u1}, \dots, f_{uI}]^\top$ as the output of the generative function $f_\theta(\cdot)$. The Gaussian log-likelihood for user u is

$$\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u) \stackrel{c}{=} -\sum_i \frac{c_{ui}}{2}(x_{ui} - f_{ui})^2. \tag{3}$$

We adopt the convention in Hu et al. [19] and introduce a "confidence" weight $c_{x_{ui}} \equiv c_{ui}$ where $c_1 > c_0$ to balance the unobserved 0's which far outnumber the observed 1's in most click data. This is also equivalent to training the model with unweighted Gaussian likelihood and negative sampling. The logistic log-likelihood[3] for user u is

$$\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u) = \sum_i x_{ui} \log \sigma(f_{ui}) + (1 - x_{ui}) \log(1 - \sigma(f_{ui})), \tag{4}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic function. We compare multinomial likelihood with Gaussian and logistic in Section 4.

2.2 Variational inference

To learn the generative model in Eq. 1, we are interested in estimating θ (the parameters of $f_\theta(\cdot)$). To do so, for each data point we need to approximate the intractable posterior distribution $p(\mathbf{z}_u \mid \mathbf{x}_u)$. We resort to variational inference [22]. Variational inference approximates the true intractable posterior with a simpler variational

[2] The cross-entropy loss for multi-class classification is a multinomial likelihood under a single draw from the distribution.

[3] Logistic likelihood is also cross-entropy loss for binary classification.

distribution $q(\mathbf{z}_u)$. We set $q(\mathbf{z}_u)$ to be a fully factorized (diagonal) Gaussian distribution:

$$q(\mathbf{z}_u) = \mathcal{N}(\boldsymbol{\mu}_u, \text{diag}\{\boldsymbol{\sigma}_u^2\}).$$

The objective of variational inference is to optimize the free variational parameters $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$ so that the Kullback-Leiber divergence $\text{KL}(q(\mathbf{z}_u)\|p(\mathbf{z}_u|\mathbf{x}_u))$ is minimized.

2.2.1 Amortized inference and the variational autoencoder: With variational inference the number of parameters to optimize $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$ grows with the number of users and items in the dataset. This can become a bottleneck for commercial recommender systems with millions of users and items. The variational autoencoder (VAE) [24, 37] replaces individual variational parameters with a data-dependent function (commonly called an *inference model*):

$$g_\phi(\mathbf{x}_u) \equiv [\mu_\phi(\mathbf{x}_u), \sigma_\phi(\mathbf{x}_u)] \in \mathbb{R}^{2K}$$

parametrized by ϕ with both $\mu_\phi(\mathbf{x}_u)$ and $\sigma_\phi(\mathbf{x}_u)$ being K-vectors and sets the variational distribution as follows:

$$q_\phi(\mathbf{z}_u \mid \mathbf{x}_u) = \mathcal{N}(\mu_\phi(\mathbf{x}_u), \text{diag}\{\sigma_\phi^2(\mathbf{x}_u)\}).$$

That is, using the observed data $\mathbf{x}_u$ as input, the inference model outputs the corresponding variational parameters of variational distribution $q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)$, which, when optimized, approximates the intractable posterior $p(\mathbf{z}_u \mid \mathbf{x}_u)$.[4] Putting $q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)$ and the generative model $p_\theta(\mathbf{x}_u \mid \mathbf{z}_u)$ together in Figure 2c, we end up with a neural architecture that resembles an autoencoder — hence the name variational autoencoder.

VAEs make use of amortized inference [12]: they flexibly reuse inferences to answer related new problems. This is well aligned with the ethos of collaborative filtering: analyze user preferences by exploiting the similarity patterns inferred from past experiences. In Section 2.4, we discuss how this enables us to perform prediction efficiently.

Learning VAEs: As is standard when learning latent-variable models with variational inference [4], we can lower-bound the log marginal likelihood of the data. This forms the objective we seek to maximize for user u (the objective function of the dataset is obtained by averaging the objective function over all the users):

$$\begin{aligned}\log p(\mathbf{x}_u; \theta) &\geq \mathbb{E}_{q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)}[\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u)] - \text{KL}(q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)\|p(\mathbf{z}_u)) \\ &\equiv \mathcal{L}(\mathbf{x}_u; \theta, \phi)\end{aligned} \tag{5}$$

This is commonly known as the evidence lower bound (ELBO). Note that the ELBO is a function of both θ and ϕ. We can obtain an unbiased estimate of ELBO by sampling $\mathbf{z}_u \sim q_\phi$ and perform stochastic gradient ascent to optimize it. However, the challenge is that we cannot trivially take gradients with respect to ϕ through this sampling process. The *reparametrization trick* [24, 37] sidesteps this issue: we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_K)$ and reparametrize $\mathbf{z}_u = \mu_\phi(\mathbf{x}_u) + \boldsymbol{\epsilon} \odot \sigma_\phi(\mathbf{x}_u)$. By doing so, the stochasticity in the sampling process is isolated and the gradient with respect to ϕ can be back-propagated through the sampled $\mathbf{z}_u$. The VAE training procedure is summarized in Algorithm 1.

[4] In the implementation, the inference model will output the log of the variance of the variational distribution. We continue to use $\sigma_\phi(\mathbf{x}_u)$ for notational brevity.

Algorithm 1: VAE-SGD Training collaborative filtering VAE with stochastic gradient descent.

Input: Click matrix $\mathbf{X} \in \mathbb{R}^{U \times I}$
Randomly initialize θ, ϕ
while *not converged* **do**
  Sample a batch of users $\mathcal{U}$
  forall $u \in \mathcal{U}$ **do**
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_K)$ and compute $\mathbf{z}_u$ via reparametrization trick
    Compute noisy gradient $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$ with $\mathbf{z}_u$
  end
  Average noisy gradients from batch
  Update θ and ϕ by taking stochastic gradient steps
end
return θ, ϕ

2.2.2 Alternative interpretation of ELBO. We can view ELBO defined in Eq. 5 from a different perspective: the first term can be interpreted as (negative) reconstruction error, while the second KL term can be viewed as regularization. It is this perspective we work with because it allows us to make a trade-off that forms the crux of our method. From this perspective, it is natural to extend the ELBO by introducing a parameter β to control the strength of regularization:

$$\begin{aligned}\mathcal{L}_\beta(\mathbf{x}_u; \theta, \phi) \equiv \mathbb{E}_{q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)}[\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u)] \\ -\beta \cdot \text{KL}(q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)\|p(\mathbf{z}_u)).\end{aligned} \tag{6}$$

While the original VAE (trained with ELBO in Eq. 5) is a powerful generative model; we might ask whether we need *all* the statistical properties of a generative model for tackling problems in recommender systems. In particular, if we are willing to sacrifice the ability to perform ancestral sampling, can we improve our performance? The regularization view of the ELBO (Eq. 6) introduces a trade-off between how well we can fit the data and how close the approximate posterior stays to the prior during learning.

We propose using $\beta \neq 1$. This means we are no longer optimizing a lower bound on the log marginal likelihood. If $\beta < 1$, then we are also weakening the influence of the prior constraint $\frac{1}{U}\sum_u q(\mathbf{z} \mid \mathbf{x}_u) \approx p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I}_K)$ [18]; this means that the model is less able to generate novel user histories by ancestral sampling.

But ultimately our goal is to make good recommendations, not to maximize likelihood or generate imagined user histories. Treating β as a free regularization parameter therefore costs us nothing, and, as we will see, yields significant improvements in performance.

Selecting β: We propose a simple heuristic for setting β: we start training with $\beta = 0$, and gradually increase β to 1. We linearly anneal the KL term slowly over a large number of gradient updates to θ, ϕ and record the best β when its performance reaches the peak. We found this method to work well and it does not require the need for training multiple models with different values of β, which can be time-consuming. Our procedure is inspired by KL annealing [7], a common heuristic used for training VAEs when there is concern that the model is being underutilized.

Figure 1 illustrates the basic idea (we observe the same trend consistently across datasets). Here we plot the validation ranking metric without KL annealing (blue solid) and with KL annealing all the way to $\beta = 1$ (green dashed, β reaches 1 at around 80 epochs). As we can see, the performance is poor without any KL annealing. With annealing, the validation performance first increases as the training proceeds and then drops as β gets close to 1 to a value that is only slightly better than doing no annealing at all.

Having identified the best β based on the peak validation metric, we can retrain the model with the same annealing schedule, but stop increasing β after reaching that value (shown as red dot-dashed in Figure 1).[5] This might be sub-optimal compared to a thorough grid search. However, it is much more efficient, and gives us competitive empirical performance. If the computational budget is scarce, then within a single run, we can stop increasing β when we notice the validation metric dropping. Such a procedure incurs no additional runtime to learning a standard VAE. We denote this partially regularized VAE with multinomial likelihood as Mult-VAE$^{\text{PR}}$.

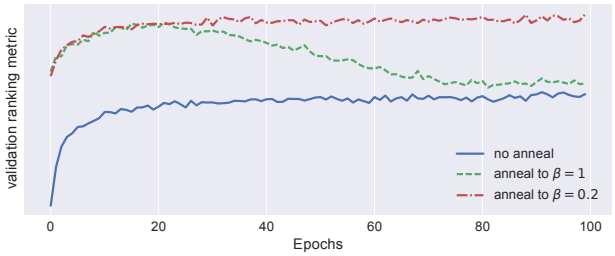


Figure 1: Validation ranking metrics with different annealing configurations. For the green dashed curve, β reaches 1 at around 80 epochs.

2.2.3 Computational Burden. Previous collaborative filtering models with neural networks [14, 51] are trained with stochastic gradient descent where in each step a single (user, item) entry from the click matrix is randomly sampled to perform a gradient update. In Algorithm 1 we subsample users and take their entire click history (complete rows of the click matrix) to update model parameters. This eliminates the necessity of negative sampling (and consequently the hyperparameter tuning for picking the number of negative examples), commonly used in the (user, item) entry subsampling scheme.

A computational challenge that comes with our approach, however, is that when the number of items is huge, computing the multinomial probability $\pi(\mathbf{z}_u)$ could be computationally expensive, since it requires computing the predictions for all the items for normalization. This is a common challenge for language modeling where the size of the vocabulary is in the order of millions or more [32]. In our experiments on some medium-to-large datasets with less than 50K items (Section 4.1), this has not yet come up as a computational bottleneck. If this becomes a bottleneck when working with larger item sets, one can easily apply the simple and effective method proposed by Botev et al. [6] to approximate the normalization factor for $\pi(\mathbf{z}_u)$.

[5] We found this to give slightly better results than keeping β at the best value throughout the training.

2.3 A taxonomy of autoencoders

In Section 2.2, we introduced maximum marginal likelihood estimation of VAEs using approximate Bayesian inference under a non-linear generative model (Eq. 1). We now describe our work from the perspective of learning autoencoders. Maximum-likelihood estimation in a regular autoencoder takes the following form:

$$\begin{aligned}\theta^{\text{AE}}, \phi^{\text{AE}} &= \arg\max_{\theta,\phi} \sum_u \mathbb{E}_{\delta(\mathbf{z}_u - g_\phi(\mathbf{x}_u))} \left[\log p_\theta(\mathbf{x}_u \mid \mathbf{z}_u)\right] \\ &= \arg\max_{\theta,\phi} \sum_u \log p_\theta(\mathbf{x}_u \mid g_\phi(\mathbf{x}_u))\end{aligned} \tag{7}$$

There are two key distinctions of note: (1) The autoencoder (and denoising autoencoder) effectively optimizes the first term in the VAE objective (Eq. 5 and Eq. 6) using a delta variational distribution $q_\phi(\mathbf{z}_u \mid \mathbf{x}_u) = \delta(\mathbf{z}_u - g_\phi(\mathbf{x}_u))$ — it does not regularize $q_\phi(\mathbf{z}_u \mid \mathbf{x}_u)$ towards any prior distribution as the VAE does. (2) the $\delta(\mathbf{z}_u - g_\phi(\mathbf{x}_u))$ is a δ distribution with mass only at the output of $g_\phi(\mathbf{x}_u)$. Contrast this to the VAE, where the learning is done using a variational *distribution*, i.e., $g_\phi(\mathbf{x}_u)$ outputs the parameters (mean and variance) of a Gaussian distribution. This means that VAE has the ability to capture per-data-point variances in the latent state $\mathbf{z}_u$.

In practice, we find that learning autoencoders is extremely prone to overfitting as the network learns to put all the probability mass to the non-zero entries in $\mathbf{x}_u$. By introducing dropout [43] at the input layer, the denoising autoencoder (DAE) is less prone to overfitting and we find that it also gives competitive empirical results. In addition to the Mult-VAE$^{\text{PR}}$, we also study a denoising autoencoder with a multinomial likelihood. We denote this model Mult-DAE. In Section 4 we characterize the tradeoffs in what is gained and lost by explicitly parameterizing the per-user variance with Mult-VAE$^{\text{PR}}$ versus using a point-estimation in Mult-DAE.

To provide a unified view of different variants of autoencoders and clarify where our work stands, we depict variants of autoencoders commonly found in the literature in Figure 2. For each one, we specify the model (dotted arrows denote a sampling operation) and describe the training objective used in parameter estimation.

In Figure 2a we have autoencoder. It is trained to reconstruct input with the same objective as in Eq. 7. Adding noise to the input (or the intermediate hidden representation) of an autoencoder yields the denoising autoencoder in Figure 2b. The training objective is the same as that of an autoencoder. Mult-DAE belongs to this model class. Collaborative denoising autoencoder [51] is a variant of this model class. The VAE is depicted in Figure 2c. Rather than using a delta variational distribution, it uses an inference model parametrized by ϕ to produce the mean and variance of the approximating variational distribution. The training objective of the VAE is given in Eq. 6. Setting β to 1 recovers the original VAE formulation [24, 37]. Higgins et al. [17] study the case where $\beta > 1$. Our model, Mult-VAE$^{\text{PR}}$ corresponds to learning VAEs with $\beta \in [0, 1]$.

2.4 Prediction

We now describe how we make predictions given a trained generative model of the form Eq. 1. For both, Mult-VAE$^{\text{PR}}$ (Section 2.2)

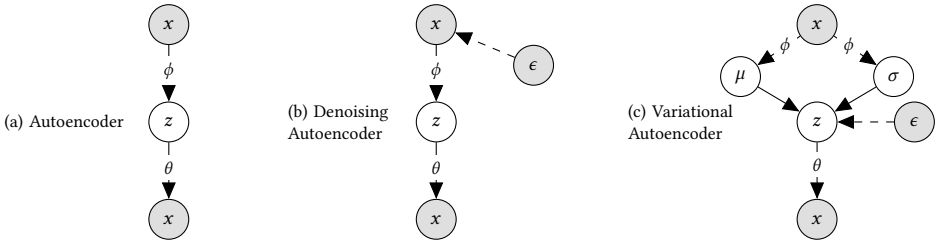


Figure 2: A taxonomy of autoencoders. The dotted arrows denote a sampling operation.

or Mult-DAE (Section 2.3), we make predictions in the same way. Given a user's click history $\mathbf{x}$, we rank all the items based on the un-normalized predicted multinomial probability $f_\theta(\mathbf{z})$. The latent representation $\mathbf{z}$ for $\mathbf{x}$ is constructed as follows: For Mult-VAE$^{\text{PR}}$, we simply take the mean of the variational distribution $\mathbf{z} = \mu_\phi(\mathbf{x})$; for Mult-DAE, we take the output $\mathbf{z} = g_\phi(x)$.

It is easy to see the advantage of using autoencoders. We can effectively make predictions for users by evaluating two functions – the inference model (encoder) $g_\phi(\cdot)$ and the generative model (decoder) $f_\theta(\cdot)$. For most of the latent factor collaborative filtering model, e.g., matrix factorization [13, 19], when given the click history of a user that is not present in the training data, normally we need to perform some form of optimization to obtain the latent factor for this user. This makes the use of autoencoders particularly attractive in industrial applications, where it is important that predictions be made cheaply and with low latency.

3 RELATED WORK

VAEs on sparse data. Variational autoencoders (VAEs) [24, 37] have seen much application to images since their inception. Doersch [10] presents a review on different applications of VAE to image data. Miao et al. [31] study VAEs on text data. More recent results from Krishnan et al. [25] find that VAEs (trained with Eq. 5) suffer from underfitting when modeling large, sparse, high-dimensional data. We notice similar issues when fitting VAE without annealing (Figure 1) or annealing to $\beta = 1$. By giving up the ability to perform ancestral sampling in the model, and setting $\beta \leq 1$, the resulting model is no longer a proper *generative* model though for collaborative filtering tasks we always make predictions *conditional on* users' click history.

Information-theoretic connection with VAE. The regularization view of the ELBO in Eq. 6 resembles maximum-entropy discrimination [20]. Maximum-entropy discrimination attempts to combine discriminative estimation with Bayesian inference and generative modeling. In our case, in Eq. 6, β acts as a knob to balance discriminative and generative aspects of the model.

The procedure in Eq. 6 has information-theoretic connections described in Alemi et al. [1]. The authors propose the deep variational information bottleneck, which is a variational approximation to the information bottleneck principle [46]. They show that as a special case they can recover the learning objective used by VAEs. They report more robust supervised classification performance with $\beta < 1$. This is consistent with our findings as well. Higgins et al. [17] proposed β-VAE, which leads to the same objective as Eq. 6. They motivate β-VAE for the goal of learning disentangled representations from images (basic visual concepts, such as shape, scale, and color). Their work, however, sets $\beta \gg 1$, effectively imposing a stronger independent prior assumption on the latent code $\mathbf{z}$. While their motivations are quite different from ours, it is interesting to note orthogonal lines of research emerging from exploring the full spectrum of values for β.

Neural networks for collaborative filtering. Early work on neural-network-based collaborative filtering models focus on explicit feedback data and evaluates on the task of rating predictions [11, 39, 41, 54]. The importance of implicit feedback has been gradually recognized, and consequently most recent research, such as this work, has focused on it. The two papers that are most closely related to our approaches are collaborative denoising autoencoder [51] and neural collaborative filtering [14].

Collaborative denoising autoencoder (CDAE) [51] augments the standard denoising autoencoder, described in Section 2.3, by adding a per-user latent factor to the input. The number of parameters of the CDAE model grows linearly with both the number of users as well as the number of items, making it more prone to overfitting. In contrast, the number of parameters in the VAE grows linearly with the number of items. The CDAE also requires additional optimization to obtain the latent factor for unseen users to make predicion. In the paper, the authors investigate the Gaussian and logistic likelihood loss functions — as we show, the multinomial likelihood is significantly more robust for use in recommender systems. Neural collaborative filtering (NCF) [14] explore a model with non-linear interactions between the user and item latent factors rather than the commonly used dot product. The authors demonstrate improvements of NCF over standard baselines on two small datasets. Similar to CDAE, the number of parameters of NCF also grows linearly with both the number of users as well as items. We find that this becomes problematic for much larger datasets. We compare with both CDAE and NCF in Section 4.

Asymmetric matrix factorization [35] may also be interpreted as an autoencoder, as elaborated in Steck [44]. We can recover this work by setting both $f_\theta(\cdot)$ and $g_\phi(\cdot)$ to be linear.

Besides being applied in session-based sequential recommendation (see Section 2.1), various approaches [2, 28, 47, 48] have applied neural networks to incorporate *side information* into collaborative filtering models to better handle the cold-start problem. These approaches are complementary to ours.

Table 1: Attributes of datasets after preprocessing. Interactions are non-zero entries. % of interactions refers to the density of the user-item click matrix X. # of the held-out users is the number of validation/test users out of the total number of users in the first row.

	ML-20M	Netflix	MSD
# of users	136,677	463,435	571,355
# of items	20,108	17,769	41,140
# of interactions	10.0M	56.9M	33.6M
% of interactions	0.36%	0.69%	0.14%
# of held-out users	10,000	40,000	50,000

4 EMPIRICAL STUDY

We evaluate the performance of Mult-VAE^{PR} and Mult-DAE. We provide insights into their performance by exploring the resulting fits. We highlight the following results:

- Mult-VAE^{PR} achieves state-of-the-art results on three real-world datasets when compared with various baselines, including recently proposed neural-network-based collaborative filtering models.
- For the denoising and variational autoencoder, the multinomial likelihood compares favorably over the more common Gaussian and logistic likelihoods.
- Both Mult-VAE^{PR} and Mult-DAE produce competitive empirical results. We identify when parameterizing the uncertainty explicitly as in Mult-VAE^{PR} does better/worse than the point estimate used by Mult-DAE and list pros and cons for both approaches.

The source code to reproduce the experimental results is available on GitHub[6].

4.1 Datasets

We study three medium- to large-scale user-item consumption datasets from various domains:

MovieLens-20M (ML-20M): These are user-movie ratings collected from a movie recommendation service. We binarize the explicit data by keeping ratings of four or higher and interpret them as implicit feedback. We only keep users who have watched at least five movies.

Netflix Prize (Netflix): This is the user-movie ratings data from the Netflix Prize[7]. Similar to ML-20M, we binarize explicit data by keeping ratings of four or higher. We only keep users who have watched at least five movies.

Million Song Dataset (MSD): This data contains the user-song play counts released as part of the Million Song Dataset [3]. We binarize play counts and interpret them as implicit preference data. We only keep users with at least 20 songs in their listening history and songs that are listened to by at least 200 users.

Table 1 summarizes the dimensions of all the datasets after preprocessing.

[6]https://github.com/dawenl/vae_cf
[7]http://www.netflixprize.com/

4.2 Metrics

We use two ranking-based metrics: Recall@R and the truncated normalized discounted cumulative gain (NDCG@R). For each user, both metrics compare the predicted rank of the held-out items with their true rank. For both Mult-VAE^{PR} and Mult-DAE, we get the predicted rank by sorting the un-normalized multinomial probability $f_\theta(\mathbf{z})$. While Recall@R considers all items ranked within the first R to be equally important, NDCG@R uses a monotonically increasing discount to emphasize the importance of higher ranks versus lower ones. Formally, define $\omega(r)$ as the item at rank r, $\mathbb{I}[\cdot]$ is the indicator function, and I_u is the set of held-out items that user u clicked on.

Recall@R for user u is

$$\text{Recall@}R(u, \omega) := \frac{\sum_{r=1}^{R} \mathbb{I}[\omega(r) \in I_u]}{\min(M, |I_u|)}.$$

The expression in the denominator is the minimum of R and the number of items clicked on by user u. This normalizes Recall@R to have a maximum of 1, which corresponds to ranking all relevant items in the top R positions.

Truncated discounted cumulative gain (DCG@R) is

$$\text{DCG@}R(u, \omega) := \sum_{r=1}^{R} \frac{2^{\mathbb{I}[\omega(r) \in I_u]} - 1}{\log(r+1)}.$$

NDCG@R is the DCG@R linearly normalized to [0, 1] after dividing by the best possible DCG@R, where all the held-out items are ranked at the top.

4.3 Experimental setup

We study the performance of various models under strong generalization [29]: We split all users into training/validation/test sets. We train models using the entire click history of the training users. To evaluate, we take part of the click history from held-out (validation and test) users to learn the necessary user-level representations for the model and then compute metrics by looking at how well the model ranks the rest of the unseen click history from the held-out users.

This is relatively more difficult than weak generalization where the user's click history can appear during both training and evaluation. We consider it more realistic and robust as well. In the last row of Table 1, we list the number of held-out users (we use the same number of users for validation and test). For each held-out user, we randomly choose 80% of the click history as the "fold-in" set to learn the necessary user-level representation and report metrics on the remaining 20% of the click history.

We select model hyperparameters and architectures by evaluating NDCG@100 on the validation users. For both Mult-VAE^{PR} and Mult-DAE, we keep the architecture for the generative model $f_\theta(\cdot)$ and the inference model $g_\phi(\cdot)$ symmetrical and explore multilayer perceptron (MLP) with 0, 1, and 2 hidden layers. We set the dimension of the latent representation K to 200 and any hidden layer to 600. As a concrete example, recall I is the total number of items, the overall architecture for a Mult-VAE^{PR}/Mult-DAE with 1-hidden-layer MLP generative model would be $[I \to 600 \to 200 \to 600 \to I]$. We find that going deeper does not improve performance. The best performing architectures are MLPs with either 0 or 1 hidden layers. We use a tanh non-linearity as the activation function between layers.

Note that for Mult-VAE^{PR}, since the output of $g_\phi(\cdot)$ is used as the mean and variance of a Gaussian random variable, we do not apply an activation function to it. Thus, the Mult-VAE^{PR} with 0-hidden-layer MLP is effectively a log-linear model. We tune the regularization parameter β for Mult-VAE^{PR} following the procedure described in Section 2.2.2. We anneal the Kullback-Leibler term linearly for 200,000 gradient updates. For both Mult-VAE^{PR} and Mult-DAE, we apply dropout at the input layer with probability 0.5. We apply a weight decay of 0.01 for Mult-DAE. We do not apply weight decay for any VAE models. We train both Mult-VAE^{PR} and Mult-DAE using Adam [23] with batch size of 500 users. For ML-20M, we train for 200 epochs. We train for 100 epochs on the other two datasets. We keep the model with the best validation NDCG@100 and report test set metrics with it.

4.4 Baselines

We compare results with the following standard state-of-the-art collaborative filtering models, both linear and non-linear:

Weighted matrix factorization (WMF) [19]: a linear low-rank factorization model. We train WMF with alternating least squares; this generally leads to better performance than with SGD. We set the weights on all the 0's to 1 and tune the weights on all the 1's in the click matrix among $\{2, 5, 10, 30, 50, 100\}$, as well as the latent representation dimension $K \in \{100, 200\}$ by evaluating NDCG@100 on validation users.

SLIM [33]: a linear model which learns a sparse item-to-item similarity matrix by solving a constrained ℓ_1-regularized optimization problem. We grid-search both of the regularization parameters over $\{0.1, 0.5, 1, 5\}$ and report the setting with the best NDCG@100 on validation users. We did not evaluate SLIM on MSD because the dataset is too large for it to finish in a reasonable amount of time (for the Netflix dataset, the parallelized grid search took about two weeks). We also found that the faster approximation of SLIM [26] did not yield competitive performance.

Collaborative denoising autoencoder (CDAE) [51]: augments the standard denoising autoencoder by adding a per-user latent factor to the input. We change the (user, item) entry subsampling strategy in SGD training in the original paper to the user-level subsampling as we did with Mult-VAE^{PR} and Mult-DAE. We generally find that this leads to more stable convergence and better performance. We set the dimension of the bottleneck layer to 200, and use a weighted square loss, equivalent to what the square loss with negative sampling used in the original paper. We apply tanh activation at both the bottleneck layer as well as the output layer.[8] We use Adam with a batch size of 500 users. As mentioned in Section 3, the number of parameters for CDAE grows linearly with the number of users and items. Thus, it is crucial to control overfitting by applying weight decay. We select the weight decay parameter over $\{0.01, 0.1, \cdots, 100\}$ by examining the validation NDCG@100.

Neural collaborative filtering (NCF) [14]: explores non-linear interactions (via a neural network) between the user and item latent factors. Similar to CDAE, the number of parameters for NCF grows linearly with the number of users and items. We use the publicly available source code provided by the authors, yet cannot obtain competitive performance on the datasets used in this paper — the validation metrics drop within the first few epochs over a wide range of regularization parameters. The authors kindly provided the two datasets (ML-1M and Pinterest) used in the original paper, as well as the training/test split, therefore we separately compare with NCF on these two relatively smaller datasets in the empirical study. In particular, we compare with the hybrid NeuCF model which gives the best performance in He et al. [14], both with and without pre-training.

We also experiment with Bayesian personalized ranking (BPR) [36]. However, the performance is not on par with the other baselines above. This is consistent with some other studies with similar baselines [40]. Therefore, we do not include BPR in the following results and analysis.

4.5 Experimental results and analysis

In this section, we quantitatively compare our proposed methods with various baselines. In addition, we aim to answer the following two questions:

1. How does multinomial likelihood compare with other commonly used likelihood models for collaborative filtering?
2. When does Mult-VAE^{PR} perform better/worse than Mult-DAE?

Quantitative results. Table 2 summarizes the results between our proposed methods and various baselines. Each metric is averaged across all test users. Both Mult-VAE^{PR} and Mult-DAE significantly outperform the baselines across datasets and metrics. Mult-VAE^{PR} significantly outperforms Mult-DAE on ML-20M and Netflix data-sets. In most of the cases, non-linear models (Mult-VAE^{PR}, Mult-DAE, and CDAE) prove to be more powerful collaborative filtering models than state-of-the-art linear models. The inferior results of CDAE on MSD are possibly due to overfitting with the huge number of users and items, as validation metrics drop within the first few epochs even though the training objective continues improving.

We compare with NCF on the two relatively smaller datasets used in Hu et al. [19]: ML-1M (6,040 users, 3,704 items, 4.47% density) and Pinterest (55,187 users, 9,916 items, 0.27% density). Because of the size of these two datasets, we use Mult-DAE with a 0-hidden-layer MLP generative model — the overall architecture is $[I \to 200 \to I]$. (Recall Mult-VAE^{PR} with a 0-hidden-layer MLP generative model is effectively a log-linear model with limited modeling capacity.) Table 3 summarizes the results between Mult-DAE and NCF. Mult-DAE significantly outperforms NCF without pre-training on both datasets. On the larger Pinterest dataset, Mult-DAE even improves over the pre-trained NCF model by a big margin.

How well does multinomial likelihood perform? Despite being commonly used in language models, multinomial likelihoods have typically received less attention in the collaborative filtering literature, especially with latent-factor models. Most previous work builds on Gaussian likelihoods (square loss, Eq. 3) [19, 33, 51] or logistic likelihood (log loss, Eq. 4) [14, 51] instead. We argue in Section 2.1 that multinomial likelihood is in fact a good proxy for the top-N ranking loss and is well-suited for implicit feedback data. To demonstrate the effectiveness of multinomial likelihood, we take the best-performing Mult-VAE^{PR} and Mult-DAE model on each dataset and swap the likelihood distribution model for the data while keeping everything else exactly the same.

[8] Wu et al. [51] used sigmoid activation function but mentioned tanh gave similar results. We use tanh to be consistent with our models.

Table 2: Comparison between various baselines and our proposed methods. Standard errors are around 0.002 for ML-20M and 0.001 for Netflix and MSD. Both Mult-VAE^{PR} and Mult-DAE significantly outperform the baselines across datasets and metrics. We could not finish SLIM within a reasonable amount of time on MSD.

(a) ML-20M

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.395**	**0.537**	**0.426**
Mult-DAE	0.387	0.524	0.419
WMF	0.360	0.498	0.386
SLIM	0.370	0.495	0.401
CDAE	0.391	0.523	0.418

(b) Netflix

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.351**	**0.444**	**0.386**
Mult-DAE	0.344	0.438	0.380
WMF	0.316	0.404	0.351
SLIM	0.347	0.428	0.379
CDAE	0.343	0.428	0.376

(c) MSD

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.266**	**0.364**	**0.316**
Mult-DAE	**0.266**	**0.363**	0.313
WMF	0.211	0.312	0.257
SLIM	—	—	—
CDAE	0.188	0.283	0.237

Table 3: Comparison between NCF and Mult-DAE with $[I \to 200 \to I]$ architecture. We take the results of NCF from He et al. [14]. Mult-DAE model significantly outperforms NCF without pre-training on both datasets and further improves on Pinterest even comparing with pre-trained NCF.

(a) ML-1M

	NCF	NCF (pre-train)	Mult-DAE
Recall@10	0.705	**0.730**	0.722
NDCG@10	0.426	**0.447**	0.446

(b) Pinterest

	NCF	NCF (pre-train)	Mult-DAE
Recall@10	0.872	0.880	**0.886**
NDCG@10	0.551	0.558	**0.580**

Table 4: Comparison of Mult-VAE^{PR} and Mult-DAE with different likelihood functions at the output layer on ML-20M. The standard error is around 0.002 (the results on the other two datasets are similar.) The multinomial likelihood performs better than the other two commonly-used likelihoods from the collaborative filtering literature.

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.395**	**0.537**	**0.426**
Gaussian-VAE^{PR}	0.383	0.523	0.415
Logistic-VAE^{PR}	0.388	0.523	0.419
Mult-DAE	**0.387**	**0.524**	**0.419**
Gaussian-DAE	0.376	0.515	0.409
Logistic-DAE	0.381	0.516	0.414

Table 4 summarizes the results of different likelihoods on ML-20M (the results on the other two datasets are similar.) We tune the hyperparameters for each likelihood separately.[9] The multinomial likelihood performs better than the other likelihoods. The gap between logistic and multinomial likelihood is closer — this is understandable since multinomial likelihood can be approximated by individual binary logistic likelihood, a strategy commonly adopted in language modeling [32, 52].

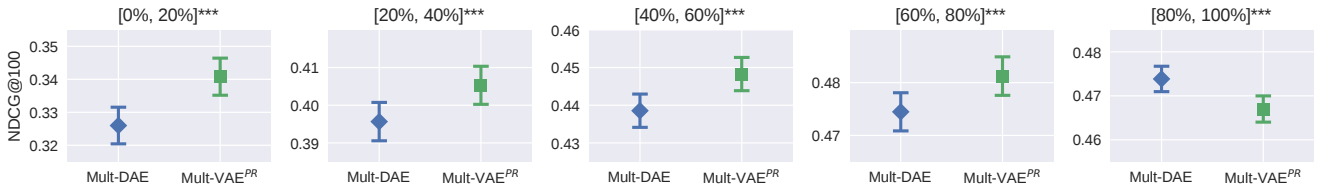
We wish to emphasize that the choice of likelihood remains data-dependent. For the task of collaborative filtering, the multinomial likelihood achieves excellent empirical results. The methodology behind the partial regularization in Mult-VAE^{PR}, however, is a technique we hypothesize will generalize to other domains.

When does Mult-VAE^{PR} perform better/worse than Mult-DAE? In Table 2 we can see that both Mult-VAE^{PR} and Mult-DAE produce competitive empirical results with Mult-VAE^{PR} being comparably better. It is natural to wonder when a variational Bayesian inference approach (Mult-VAE^{PR}) will win over using a point estimate (Mult-DAE) and vice versa.

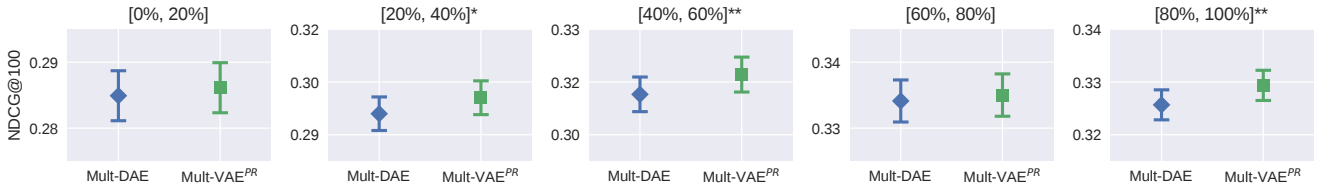
Intuitively, Mult-VAE^{PR} imposes stronger modeling assumptions and therefore could be more robust when user-item interaction data is scarce. To study this, we considered two datasets: ML-20M where Mult-VAE^{PR} has the largest margin over Mult-DAE and MSD where Mult-VAE^{PR} and Mult-DAE have roughly similar performance. The results on the Netflix dataset are similar to ML-20M. We break down test users into quintiles based on their activity level in the fold-in set which is provided as input to the inference model $g_\phi(\cdot)$ to make prediction. The activity level is simply the number of items each user has clicked on. We compute NDCG@100 for each group of users using both Mult-VAE^{PR} and Mult-DAE and plot results in Figure 3. This summarizes how performance differs across users with various levels of activity.

In Figure 3, we show performance across increasing user activity. Error bars represents one standard error. For each subplot, a paired t-test is performed and statistical significance is marked. Although there are slight variations across datasets, Mult-VAE^{PR} consistently improves recommendation performance for users who have only clicked on a small number of items. This is particularly prominent for ML-20M (Figure 3a). Interestingly, Mult-DAE actually

[9] Surprisingly, partial regularization seems less effective for Gaussian and logistic.



(a) ML-20M



(b) MSD

Figure 3: NDCG@100 breakdown for users with increasing levels of activity (starting from 0%), measured by how many items a user clicked on in the fold-in set. The error bars represents one standard error. For each subplot, a paired t-test is performed and * indicates statistical significance at $\alpha = 0.05$ level, ** at $\alpha = 0.01$ level, and * at $\alpha = 0.001$ level. Although details vary across datasets, Mult-VAE^{PR} consistently improves recommendation performance for users who have only clicked on a small number of items.**

outperforms Mult-VAE^{PR} on the most active users. This indicates the stronger prior assumption could potentially hurt the performance when a lot of data is available for a user. For MSD (Figure 3b), the least-active users have similar performance under both Mult-VAE^{PR} and Mult-DAE. However, as we described in Section 4.1, MSD is pre-processed so that a user has at least listened to 20 songs. Meanwhile for ML-20M, each user has to watch at least 5 movies. This means that the first bin of ML-20M has much lower user activity than the first bin of MSD.

Overall, we find that Mult-VAE^{PR}, which may be viewed under the lens of a principled Bayesian inference approach, is more robust than the point estimate approach of Mult-DAE, regardless of the scarcity of the data. More importantly, the Mult-VAE^{PR} is less sensitive to the choice of hyperparameters – weight decay is important for Mult-DAE to achieve competitive performance, yet it is not required for Mult-VAE^{PR}. On the other hand, Mult-DAE also has advantages: it requires fewer parameters in the bottleneck layer — Mult-VAE^{PR} requires two sets of parameters to obtain the latent representation $\mathbf{z}$: one set for the variational mean $\mu_\phi(\cdot)$ and another for the variational variance $\sigma_\phi(\cdot)$ — and Mult-DAE is conceptually simpler for practitioners.

5 CONCLUSION

In this paper, we develop a variant of VAE for collaborative filtering on implicit feedback data. This enables us to go beyond linear factor models with limited modeling capacity.

We introduce a generative model with a multinomial likelihood function parameterized by neural network. We show that multinomial likelihood is particularly well suited to modeling user-item implicit feedback data.

Based on an alternative interpretation of the VAE objective, we introduce an additional regularization parameter to partially regularize a VAE (Mult-VAE^{PR}). We also provide a practical and efficient way to tune the additional parameter introduced using KL annealing. We compare the results obtained against a denoising autoencoder (Mult-DAE).

Empirically, we show that the both Mult-VAE^{PR} and Mult-DAE provide competitive performance with Mult-VAE^{PR} significantly outperforming the state-of-the-art baselines on several real-world datasets, including two recently proposed neural-network-based approaches. Finally, we identify the pros and cons of both Mult-VAE^{PR} and Mult-DAE and show that employing a principled Bayesian approach is more robust.

In future work, we would like to futher investigate the trade-off introduced by the additional regularization parameter β and gain more theoretical insight into why it works so well. Extending Mult-VAE^{PR} by *condition* on side information might also be a way to improve performance.

REFERENCES

[1] Alexander Alemi, Ian Fischer, Joshua Dillon, and Kevin Murphy. 2017. Deep Variational Information Bottleneck. In *5th International Conference on Learning Representations*.
[2] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 147–154.
[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset.. In *ISMIR*, Vol. 2. 10.
[4] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational Inference: A Review for Statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.
[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.
[6] Aleksandar Botev, Bowen Zheng, and David Barber. 2017. Complementary Sum Sampling for Likelihood Approximation in Large Scale Classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and*

Statistics. 1030–1038.
[7] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).
[8] Sotirios Chatzis, Panayiotis Christodoulou, and Andreas S. Andreou. 2017. Recurrent Latent Variable Networks for Session-Based Recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*.
[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
[10] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).
[11] Kostadin Georgiev and Preslav Nakov. 2013. A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines. In *Proceedings of the 30th International Conference on Machine Learning*. 1148–1156.
[12] Samuel Gershman and Noah Goodman. 2014. Amortized inference in probabilistic reasoning. In *Proceedings of the Cognitive Science Society*, Vol. 36.
[13] Prem Gopalan, Jake M. Hofman, and David M. Blei. 2015. Scalable Recommendation with Hierarchical Poisson Factorization. In *Uncertainty in Artificial Intelligence*.
[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.
[15] Balázs Hidasi and Alexandros Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *arXiv preprint arXiv:1706.03847* (2017).
[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations*.
[18] Matthew D. Hoffman and Matthew J. Johnson. 2016. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*.
[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. 263–272.
[20] Tommi Jaakkola, Marina Meila, and Tony Jebara. 2000. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems*. 470–476.
[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
[22] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine learning* 37, 2 (1999), 183–233.
[23] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[25] Rahul G. Krishnan, Dawen Liang, and Matthew D. Hoffman. 2017. On the challenges of learning with inference networks on sparse, high-dimensional data. *arXiv preprint arXiv:1710.06085* (2017).
[26] Mark Levy and Kris Jack. 2013. Efficient top-n recommendation by linear regression. In *RecSys Large Scale Recommender Systems Workshop*.
[27] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*. 59–66.
[28] Dawen Liang, Minshu Zhan, and Daniel P.W. Ellis. 2015. Content-Aware Collaborative Music Recommendation Using Pre-trained Neural Networks.. In *ISMIR*. 295–301.
[29] Benjamin Marlin. 2004. *Collaborative filtering: A machine learning perspective*. University of Toronto.
[30] Daniel McFadden et al. 1973. Conditional logit analysis of qualitative choice behavior. (1973), 105–142 pages.
[31] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*. 1727–1736.
[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[33] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. 497–506.
[34] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. 502–511.
[35] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, Vol. 2007. 5–8.
[36] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. 452–461.
[37] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning*. 1278–1286.
[38] Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. *Advances in neural information processing systems* (2008), 1257–1264.
[39] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*. 791–798.
[40] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Darius Braziunas. 2016. On the Effectiveness of Linear Models for One-Class Collaborative Filtering.. In *AAAI*.
[41] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. 111–112.
[42] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*.
[43] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.
[44] Harald Steck. 2015. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 115–122.
[45] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 17–22.
[46] Naftali Tishby, Fernando Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057* (2000).
[47] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*. 2643–2651.
[48] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.
[49] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. 2008. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*. 1593–1600.
[50] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, Vol. 11. 2764–2770.
[51] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 153–162.
[52] Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. 2011. Efficient subsampling for training complex language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1128–1136.
[53] Shuang-Hong Yang, Bo Long, Alexander J. Smola, Hongyuan Zha, and Zhaohui Zheng. 2011. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 295–304.
[54] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *Proceedings of The 33rd International Conference on Machine Learning*. 764–773.