A Collective Variational Autoencoder for Top-N Recommendation with Side Information

Yifan Chen
University of Amsterdam
y.chen4@uva.nl

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

ABSTRACT

Recommender systems have been studied extensively due to their practical use in real-world scenarios. Despite this, generating effective recommendations with sparse user ratings remains a challenge. Side information has been widely utilized to address rating sparsity. Existing recommendation models that use side information are linear and, hence, have restricted expressiveness. Deep learning has been used to capture non-linearities by learning deep item representations from side information but as side information is high-dimensional, existing deep models tend to have large input dimensionality, which dominates their overall size. This makes them difficult to train, especially with insufficient inputs.

Rather than learning item representations, in this paper, we propose to learn feature representations through deep learning from side information. Learning feature representations ensures a sufficient number of inputs to train a deep network. To achieve this, we propose to simultaneously recover user ratings and side information, by using a Variational Autoencoder (VAE). Specifically, user ratings and side information are encoded and decoded collectively through the same inference network and generation network. This is possible as both user ratings and side information are associated with items. To account for the heterogeneity of user ratings and side information, the final layer of the generation network follows different distributions.The proposed model is easy to implement and efficient to optimize and is shown to outperform state-of-the-art top-N recommendation methods that use side information.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

Top-N recommendation, Side information, Collective Variational autoencoder

ACM Reference Format:
Yifan Chen and Maarten de Rijke. 2018. A Collective Variational Autoencoder for Top-N, Recommendation with Side Information. In *3rd Workshop on Deep Learning for Recommender Systems (DLRS 2018), October 6, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3270323.3270326

DLRS 2018, October 6, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-6617-5/18/10…$15.00
https://doi.org/10.1145/3270323.3270326

1 INTRODUCTION

Recommender systems have become increasingly indispensable. Applications include top-N recommendations, which are widely adopted to recommend users ranked lists of items. For e-commerce, typically only a few recommendations are shown to the user each time and recommender systems are often evaluated based on the performance of the top-N recommendations.

Collaborative Filtering (CF) based methods are a fundamental building block in many recommender systems. CF based recommender systems predict what items a user will prefer by discovering and exploiting similarity patterns across users and items. The performance of CF-based methods often drops when ratings are very sparse. With the increased availability of *side information*, that is, additional information associated with items such as product reviews, movie plots, etc., there is great interest in taking advantage of such information so as to compensate for the sparsity of ratings.

Existing methods utilizing side information are linear models [13], which have a restricted model capacity. Recent work generalizes linear model by deep learning to explore non-linearities for large-scale recommendations [3, 15, 19, 23]. State-of-the-art performance is achieved by applying Variational Autoencoders (VAEs) [5] for CF [7, 9, 10]. These deep models learn item representations from side information. Thus, the dimension of side information determines the input dimension of the network, which dominates the overall size of the model. This is problematic since side information is generally high-dimensional [1]. As we will see, existing deep models fail to beat linear models due to the high-dimensionality of side information and an insufficient number of samples.

To avoid the impact from high-dimensionality while exploiting the effectiveness of VAEs, we learn feature representations from side information. The dimensions of the side information correspond to the number of samples rather than the input dimension of the deep network. To instantiate this idea, we propose collective Variational Autoencoder (cVAE), which learns to recover user ratings and side information simultaneously through VAE. While user ratings and side information are different sources of information, both are information associated with items. Thus, we take ratings from each user and each dimension of side information over all items as the input for VAE, so that samples from both sources of information have the same dimensionality (number of items). We can then feed ratings and side information into the same inference network and generation network. cVAE complements the sparse ratings with side information, as feeding side information into the same VAE increases the number of samples for training. The high-dimensionality of side information is not a problem for cVAE, as it increases the sample size rather than the network scale. To account for the heterogeneity of user rating and side information, the final

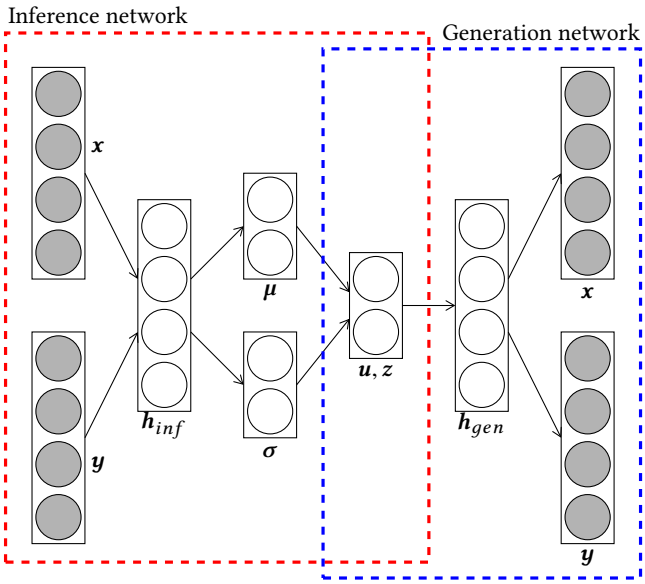


Figure 1: Collective Variational Autoencoder.

layer of the generation network follows different distributions depending on the type of information. Training a VAE by feeding it side information as input acts like a pre-training step, which is a crucial step for developing a robust deep network. Our experiments show that the proposed model,cVAE, achieves state-of-the-art performance for top-N recommendation with side information.

The remainder of the paper is organized as follows. We present preliminaries in Section 2. We introduce the cVAE model and optimization in Section 3. Section 4 describes the experimental setup and results. We review related work in Section 5 and conclude in Section 6.

2 PRELIMINARIES

2.1 Notation

We introduce relevant notation in this section. We use m, n and d to denote the number of users, items and the dimension of side information, respectively. We study the problem of top-N recommendation with high-dimensional side information, where $d \gg n$. We write $X \in \mathbb{R}^{d \times n}$ for the matrix for side information and $Y \in \mathbb{R}^{m \times n}$ for user ratings. We summarize our notation in Table 1.

2.2 Linear models for top-N recommendation

The Sparse LInear Method (SLIM) [12] achieves state-of-the-art performance for top-N recommendation. SLIM learns to reproduce the user rating marix Y through:

$$Y \sim YW.$$

Here, $W \in \mathbb{R}^{n \times n}$ is the coefficient matrix, which is analogous to the item similarity matrix. The performance of SLIM is heavily affected by the rating sparsity [4]. Side information has been utilized to overcome this issue [1, 13, 22]. As a typical example of a method that uses side information, collective SLIM (cSLIM) learns W from both user rating and side information. Specifically, X, Y are both

Table 1: Notation used in the paper.

Notation	Description
m	number of users
n	number of items
d	dimension of side information
k	dimension of latent item representation
N	number of recommended items
$X \in \mathbb{R}^{d \times n}$	matrix of side information
$Y \in \mathbb{R}^{m \times n}$	matrix of user rating
$U \in \mathbb{R}^{m \times k}$	matrix of latent user representation
$V \in \mathbb{R}^{n \times k}$	matrix of latent item representation
$Z \in \mathbb{R}^{d \times k}$	matrix of latent feature representation
$\boldsymbol{h}_{inf}$	hidden layer of inference network
$\boldsymbol{h}_{gen}$	hidden layer of generation network
$\boldsymbol{\mu} \in \mathbb{R}^k$	the mean of latent input representation
$\boldsymbol{\sigma} \in \mathbb{R}^k$	the variance of latent user or feature representation
$f_\phi(\cdot)$	non-linear transformation of inference network
$f_\theta(\cdot)$	non-linear transformation of generation network
$\mu(\cdot)$	the activation function to get $\boldsymbol{\mu}$
$\sigma(\cdot)$	the activation function to get $\boldsymbol{\sigma}$
$\varsigma(\cdot)$	the sigmoid function

reproduced through:

$$Y \sim YW, \quad X \sim XW.$$

cSLIM learns the coefficient matrix W collectively from both side information X and user rating Y, a strategy that can help to overcome rating sparsity by side information. However, cSLIM is restricted by the fact that it is a linear model, which has limited model capacity.

2.3 Autoencoders for collaborative filtering

Recently, autoencoders have been used to address CF problems [15, 17, 19, 24]. Autoencoders are neural networks popularized by Kramer [6]. They are unsupervised networks where the output of the network aims to be a reconstruction of the input.

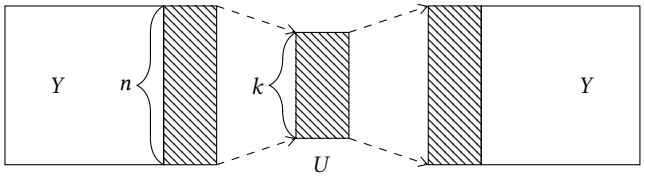
In the context of CF, the autoencoder is fed with incomplete rows (resp. columns) of the user rating matrix Y. It then outputs a vector that predicts the missing entries. These approaches perform a non-linear low-rank approximation of Y in two different ways, using a User-side Autoencoder (UAE) (Figure 2(a)) or Item-side Autoencoder (IAE) (Figure 2(b)), which recover Y respectively through:

$$U \sim f(Y), \quad Y \sim g(U),$$

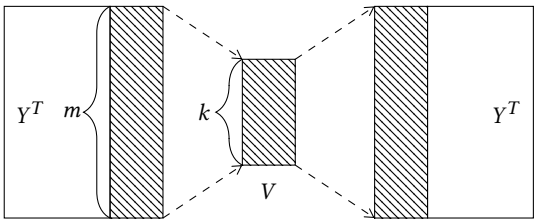
and

$$V \sim f(Y^T), \quad Y^T \sim g(V),$$

where $U \in \mathbb{R}^{m \times k}$ is the user representation and $V \in \mathbb{R}^{n \times k}$ is the item representation. Moreover, $f(\cdot)$ and $g(\cdot)$ are the encode network and decode network, respectively. UAEs encode Y to learn a user latent representation U and then recover Y from U. In contrast, IAEs encode the transpose of Y to learn item latent representation V and then recover the transpose of Y from V. Note that UAEs work in a similar way as SLIM, as both can be viewed as reproducing Y through $Y \sim g(f(Y))$, which also captures item similarities.



(a) User-side Autoencoder



(b) Item-side Autoencoder

Figure 2: Autoencoders for collaborative filtering.

When side information associated with items is available, the Feature-side Autoencoder (FAE) is utilized to learn item representations:

$$V' \sim f(X^T), \quad X^T \sim g(V'),$$

where $V' \in \mathbb{R}^{n \times k}$ is the item representation. Existing hybrid methods incorporate FAE with IAE as both learn item representations. However, this way of incorporating side information needs to estimate two separate VAEs, which is not an effective way to address rating sparsity. They are also vulnerable to the high dimensionality of side information.

3 METHOD

In this section, we propose a new way to incorporate side information with user ratings by combining the effectiveness of both cSLIM and autoencoders. We propose to reproduce X by a FAE and Y by a UAE. In this way, the input for autoencoders of both X and Y are of the same dimension, i.e., the number of items n. Thus, we can feed X and Y into the same autoencoder rather than two different autoencoders, which helps to overcome rating sparsity.

3.1 Collective variational autoencoder

We propose a collective Variational Autoencoder (cVAE) to generalize the linear models for top-N recommendation with side information to non-linear models, by taking advantage of Variational Autoencoders (VAEs). Specifically, we propose to recover X, Y through

$$U \sim f_\phi(Y), \quad Y \sim f_\theta(U),$$
$$Z \sim f_\phi(X), \quad X \sim f_\theta(Z),$$

where $f_\phi(\cdot)$ and $f_\theta(\cdot)$ correspond to the inference network and generation network parameterized by ϕ and θ, respectively. An overview of cVAE is depicted in Figure 1. Unlike previous work utilizing VAEs, the proposed model encodes and decodes user rating and side information through the same inference and generation networks. Our model can be viewed as a non-linear generalization of cSLIM, so as to learn item similarities collectively from user ratings and side information. While user ratings and side information are two different types of information, cSLIM fails to distinguish them. In contrast, cVAE assumes the output of the generation network to follow different distributions according to the type of input it has been fed.

Next, we describe the cVAE model in detail. Following common practice for VAEs, we first assume the latent variables $\boldsymbol{u}$ and $\boldsymbol{z}$ to follow a Gaussian distribution:

$$\boldsymbol{u} \sim \mathcal{N}(0, I), \quad \boldsymbol{z} \sim \mathcal{N}(0, I),$$

where $I \in \mathbb{R}^{k \times k}$ is an identity matrix. While X and Y are fed into the same network, we would like to distinguish them via different distributions. In this paper, we assume that Y is binarized to capture implicit feedback, which is a common setting for top-N recommendation [12]. Thus we follow Lee et al. [7] and assume that the rating of user j over all items follows a Bernoulli distribution:

$$\boldsymbol{y}_j \mid \boldsymbol{u}_j \sim Bernoulli(\varsigma(f_\theta(\boldsymbol{u}_j))),$$

where $\varsigma(\cdot)$ is the sigmoid function. This defines the loss function when feeding user rating as input, i.e., the logistic log-likelihood for user j:

$$\log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j) = \sum_{i=1}^{n} y_{ji} \log \varsigma(f_{ji}) + (1 - y_{ji}) \log \left(1 - \varsigma(f_{ji})\right), \quad (1)$$

where f_{ji} is the i-th element of the vector $f_\theta(\boldsymbol{u}_j)$ and $f_\theta(\boldsymbol{u}_j)$ is normalized through a sigmoid function so that f_{ji} is within $(0, 1)$.

For side information, we study numerical features so that we assume the j-th dimension of side information from all items follows a Gaussian distribution:

$$\boldsymbol{x}_j \mid \boldsymbol{z}_j \sim \mathcal{N}(f_\theta(\boldsymbol{z}_j), I).$$

This defines the loss function when feeding side information as input, i.e., the Gaussian log-likelihood for dimension j:

$$\log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j) = \sum_{i=1}^{n} -\frac{1}{2}(x_{ji} - f_{ji})^2, \quad (2)$$

where f_{ji} is the i-th element of vector $f_\theta(\boldsymbol{z}_j)$. Note that although we assume $\boldsymbol{x}$ and $\boldsymbol{y}$ to be generated from $\boldsymbol{z}$ and $\boldsymbol{u}$ respectively, the generation has shared parameters θ.

The generation procedure is summarized as follows:

(1) for each user $j = 1, \ldots, m$:
  (a) draw $\boldsymbol{u}_j \sim \mathcal{N}(0, I)$;
  (b) draw $\boldsymbol{y}_j \sim Bernoulli\left(\varsigma(f_\theta(\boldsymbol{u}_j))\right)$
(2) for each dimension of side information $j = 1, \ldots, d$:
  (a) draw $\boldsymbol{z}_j \sim \mathcal{N}(0, I)$;
  (b) draw $\boldsymbol{x}_j \sim \mathcal{N}(f_\theta(\boldsymbol{z}_u), I)$.

Once the cVAE is trained, we can generate recommendations for each user j with items ranked in descending order of $f_\theta(\boldsymbol{u}_j)$. Here, $\boldsymbol{u}_j$ is calculated as $\boldsymbol{u}_j = \mu(f_\phi(\boldsymbol{y}_j))$, that is, we take the mean of $\boldsymbol{u}_j$ for prediction.

Next, we discuss how to perform inference for cVAE.

3.2 Variational inference

The log-likelihood of cVAEs is intractable due to the non-linear transformations of the generation network. Thus, we resort to variational inference to approximate the distribution. Variational inference approximates the true intractable posterior with a simpler variational distribution $q(U, Z)$. We follow the mean-field assumption [20] by setting $q(U, Z)$ to be a fully factorized Gaussian distribution:

$$q(U, Z) = \prod_{j=1}^{m} q(\boldsymbol{u}_j) \prod_{j=1}^{d} q(\boldsymbol{z}_j),$$
$$q(\boldsymbol{u}_j) = \mathcal{N}(\boldsymbol{\mu}_j, \mathrm{diag}(\boldsymbol{\sigma}_j^2)),$$
$$q(\boldsymbol{z}_j) = \mathcal{N}(\boldsymbol{\mu}_{m+j}, \mathrm{diag}(\boldsymbol{\sigma}_{m+j}^2)),$$

While we can optimize $\{\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j\}$ by minimizing the Kullback-Leiber divergence $\mathbb{KL}(q_\phi \parallel p_\theta)$, the number of parameters to learn grows with the number of users and dimensions of side information. This can become a bottleneck for real-world recommender systems with millions of users and high-dimensional side information. The VAE replaces individual variational parameters with a data-dependent function through an inference network parameterized by ϕ, i.e., f_ϕ, where $\boldsymbol{\mu}_j$ and $\boldsymbol{\sigma}_j$ are generated as:

$$\boldsymbol{\mu}_j = \mu(f_\phi(\boldsymbol{y}_j)), \quad \boldsymbol{\sigma}_j = \sigma(f_\phi(\boldsymbol{y}_j)), \quad \forall j = 1, \ldots, m$$
$$\boldsymbol{\mu}_{m+j} = \mu(f_\phi(\boldsymbol{x}_j)), \quad \boldsymbol{\sigma}_{m+j} = \sigma(f_\phi(\boldsymbol{x}_j)), \quad \forall j = 1, \ldots, d.$$

Putting together $p_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and $p_\phi(\boldsymbol{u} \mid \boldsymbol{y})$ with $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ and $p_\theta(\boldsymbol{y} \mid \boldsymbol{u})$ forms the proposed cVAE (Figure 1).

Next we derive the Evidence Lower Bound (ELBO):

$$\mathcal{L}(q) = \mathbb{E}_{q_\phi}\left[\log p_\theta(X, Y \mid U, Z)\right] - \mathbb{KL}\left(q_\phi \parallel p(U, Z)\right). \quad (3)$$

We use a Monte Carlo gradient estimator [14] to infer the expectation in Equation (3). We draw L samples of $\boldsymbol{u}_j$ and $\boldsymbol{z}_j$ from q_ϕ and perform stochastic gradient ascent to optimize the ELBO. In order to take gradients with respect to ϕ through sampling, we follow the reparameterization trick [5] to sample $\boldsymbol{u}_j$ and $\boldsymbol{z}_j$ as:

$$\boldsymbol{u}_j^{(l)} = \mu(f_\phi(\boldsymbol{y}_j)) + \boldsymbol{\epsilon}_1^{(l)} \odot \sigma(f_\phi(\boldsymbol{y}_j)),$$
$$\boldsymbol{z}_j^{(l)} = \mu(f_\phi(\boldsymbol{x}_j)) + \boldsymbol{\epsilon}_2^{(l)} \odot \sigma(f_\phi(\boldsymbol{x}_j)),$$
$$\boldsymbol{\epsilon}_1^{(l)} \sim \mathcal{N}(0, I), \quad \boldsymbol{\epsilon}_2^{(l)} \sim \mathcal{N}(0, I).$$

As the $\mathbb{KL}$-divergence can be analytically derived [5], we can then rewrite $\mathcal{L}(q)$ as:

$$\mathcal{L}(q) = \frac{1}{L} \sum_{l=1}^{L} \left(\sum_{j=1}^{m} \log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j^{(l)}) + \sum_{j=1}^{d} \log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j^{(l)}) \right) + \sum_{j=1}^{d+m} \left(1 + 2\log(\boldsymbol{\sigma}_j) - \boldsymbol{\mu}_j^2 - \boldsymbol{\sigma}_j^2\right). \quad (4)$$

We then maximize ELBO given in Equation (4) to learn θ and ϕ.

3.3 Implementation details

We discuss the implementation of cVAE in detail. As we feed the user rating matrix Y and the item side information X through the same input layer with n neurons, we need to ensure that the input from both types of information are of the same format. In this paper, we assume that user ratings are binarized to capture implicit feedback and that side information is represented as a bag-of-words. We propose to train cVAEs through a two-phase algorithm. We first feed it side information to train, which works as pre-training. We then refine the VAE by feeding user ratings. We follow the typical setting by taking f_θ as a Multi-Layer Perceptron (MLP); f_ϕ is also taken to be a MLP of the identical network structure with f_θ. We also introduce two parameters, i.e., α and β, to extend the model and make it more suitable for the recommendation task.

3.3.1 Parameter α. Generally, item feature X and user rating Y can be very sparse, that is, the positive entry (x_{ji} or $y_{ji} = 1$) is much less than the negative entry (x_{ji} or $y_{ji} = 0$). We introduce a parameter α to balance between positive samples and negative samples. Specifically, the loss functions in Equation (1) and (2) become

$$\log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j) \simeq \sum_{i=1}^{n} \alpha \cdot y_{ji} \log \varsigma(f_{ji}) + (1 - y_{ji}) \log\left(1 - \varsigma(f_{ji})\right),$$

and

$$\log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j) = \sum_{i=1}^{n} \frac{c_{ji}}{2} (x_{ji} - f_{ji})^2,$$

where

$$c_{ji} = \begin{cases} \alpha, & \text{if } x_{ji} > 0, \\ 1, & \text{otherwise.} \end{cases}$$

3.3.2 Parameter β. We can adopt different perspectives about the ELBO derived in Equation (3) as: the first term can be interpreted as the reconstruction error, while the second $\mathbb{KL}$ term can be viewed as regularization. The ELBO is often over-regularized for recommendation tasks [10]. Therefore, a parameter β is introduced to control the strength of regularization, so that the ELBO becomes:

$$\mathcal{L}(q) = \mathbb{E}_{q_\phi}\left[\log p_\theta(X, Y \mid U, Z)\right] - \beta \mathbb{KL}\left(q_\phi \parallel p(U, Z)\right).$$

We propose to train the cVAE in two phases. We first pre-train the cVAE by feeding it side information only. We then refine the model by feeding it user ratings. While Liang et al. [10] suggests to set β small to avoid over-regularization, we opt for a larger value for β during refinement, for two reasons: (1) the model is effectively pre-trained with side information; it would be reasonable to require the posterior to comply more with this prior; and (2) refinement with user ratings can easily overfit due to the sparsity of ratings; it would be reasonable to regularize heavier so as to avoid overfitting.

4 EXPERIMENTS

4.1 Experimental setup

4.1.1 Dataset. We conduct experiments on two datasets, Games and Sports, constructed from different categories of Amazon products [11]. For each category, the original dataset contains transactions between users and items, indicating implicit user feedback. The statistics of the datasets are presented in Table 2. We use the product reviews as item features. We extract unigram features from the review articles and remove stopwords. We represent each product item as a bag-of-words feature vector.

Table 2: Statistics of the datasets used.

Dataset	#User	#Item	#Rating	#Dimension	#Feature
Games	5,195	7,163	96,316	20,609	5,151,174
Sports	5,653	11,944	86,149	31,282	3,631,243

4.1.2 Methods for comparison. We contrast the performance of cVAE with that of existing existing VAE-based methods for CF: cfVAE [9] and rVAE [10]. Note that the performance of cfVAE will be affected greatly by the high-dimensionality of side information. Besides, as cfVAE was designed originally for the rating prediction task, the recommendations provided by cfVAE are likely be less effective. While rVAE is effective for top-N recommendation, it suffers from rating sparsity as side information is not utilized.

We also compare with the state-of-the-art linear model for top-N recommendation with side information, i.e., cSLIM [13]. By comparing with cSLIM, we can evaluate the capacity of cVAE as it can be regarded as a deep extension of cSLIM. We also compare with fVAE, which is the pre-trained model of cVAE with side information only. cVAE extends fVAE with user ratings.

For all VAE-based methods, we follow [5] to set the batch size as 100 so that we can set $L = 1$. We choose a two-layer network architecture for the inference network and generation network. For cfVAE and rVAE, the scale is 200-100 for inference network and 100-200 for generation network. For fVAE and cVAE, the scale is 1000-100 and 100-1000, respectively. The reason that the network scales for cfVAE and rVAE are smaller is that (1) the input for cfVAE is high-dimensional with relatively fewer samples; and (2) the input for rVAE is sparse, which easily overfits with larger networks. In comparison, we can select more hidden neurons for fVAE as it takes each dimension of the features over all items as input, so that the input for the network has relatively fewer dimensions and the number of samples is sufficient. This is similar with cVAE, which uses side information to overcome rating sparsity.

4.1.3 Evaluation method. To evaluate the performance on the top-N recommendation task, we split the user rating matrix R into R_{train}, R_{valid} and R_{test}, respectively, for training the model, selecting parameters, and testing the recommendation accuracy. Specifically, for each user, we randomly hold 10% of the ratings in the validation set and 10% in the test set and put the other ratings in the training set. For each user, the unrated items are sorted by predicted score in decreasing order and the first N items are returned as the top-N recommendations for that user.

Given the list of top-N recommended items for user u, Precision at N (Pre@N) and Recall at N (Rec@N) are defined as

$$Pre@N = \frac{|\text{relevant items} \cap \text{recommended items}|}{N},$$
$$Rec@N = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|}.$$

Average precision at N (AP@N) is a ranked precision metric that gives larger credit to correctly recommended items in the top-N ranks. AP@N is defined as the average of precision scores computed at all positions with an adopted item, namely

$$AP@N = \frac{\sum_{k=1}^{N} Pre@k \times rel(k)}{\min\{N, |\text{relevant items}|\}},$$

Table 3: Parameter selection.

Method	Parameters	
	Games	Sports
cSLIM	$\alpha = 10^{-2}, \beta = 10^{-3}, \lambda = 1$	$\alpha = 0, \beta = 10, \lambda = 10$
cfVAE	$\lambda_u = 10, \lambda_v = 0.8$	$\lambda_u = 10, \lambda_v = 1$
rVAE	$\alpha = 4, \beta = 0.1$	$\alpha = 8, \beta = 0.1$
fVAE	$\alpha = 6, \beta = 0.1$	$\alpha = 5, \beta = 0.9$
cVAE	$\alpha = 2, \beta = 2$	$\alpha = 1, \beta = 2$

where Pre@k is the precision at cut-off k in the top-N recommended list. Here, $rel(k)$ is an indicator function

$$rel(k) = \begin{cases} 1 & \text{if the item ranked at } k \text{ is relevant,} \\ 0 & \text{otherwise.} \end{cases}$$

Mean average precision at N (MAP@N) is defined as the mean of the AP scores for all users. As in [19], the list of recommended items is evaluated with R_{test} using Rec@N and MAP@N.

4.2 Experimental results

4.2.1 Parameter selection. To compare the performance of alternative top-N recommendation methods, we first select parameters for all the methods through validation. Specifically, for cSLIM, we select α, β and λ from 0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10. For cfVAE, we select λ_u from 0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10 and λ_v from $0, 0.1, 0.2, \ldots, 1$. For rVAE and fVAE, we select α from $1, 2, \ldots, 10$ and β from $0, 0.1, 0.2, \ldots, 1$. For cVAE, we select α from $1, 2, \ldots, 10$ and β from $0, 0.5, 1, \ldots, 3$. Note that we tune β with larger values to possibly regularize heavier during the refinement.

The result of parameter selection is shown in Table 3.

4.2.2 Performance comparison. We present the results in terms of Rec@N and MAP@N in Table 4, where N is respectively set as $N = 5, 10, 15, 20$. We show the best score in boldface. We attach asterisks to the best score if the improvement over the second best score is statistically significant; we conducted two-sided tests for the null hypothesis that cVAE and the second best have identical average values ($^{*}\ p < 0.1$, $^{**}\ p < 0.05$).

As shown in Table 4, cVAE outperforms other methods according to all metrics and on both datasets. The improvement is also significant in many cases. The significance of the improvements becomes more evident when N gets larger. The other three methods utilizing VAE are less effective with high-dimensional side information. They even fail to beat linear models. In contrast, cVAE improves over cSLIM by using VAEs for non-linear low-rank approximation. This demonstrates the effectiveness of our proposed cVAE model.

On the Games dataset, cVAE shows significant improvements over the state-of-the-art. Apart from cVAE, cfVAE provides the best recommendation among all VAE-based CF methods, though it fails to beat cSLIM. This is followed by fVAE, which utilizes side information only. rVAE performs the worst, due to the rating sparsity. On the Sports dataset, significant improvements can only be observed for Rec@15 and Rec@20. The results yield an interesting insight. If we look at the parameter selection for cSLIM, we can see that α is set to 0, which means cSLIM generates the best recommendations when no side information is utilized. This does not necessarily mean

Table 4: Comparison of top-N recommendation performance. * indicates a statistically significant difference between cVAE and the best performing baseline, where * indicates $p < 0.1$ and ** indicates $p < 0.05$

Method	Rec@5	Rec@10	Rec@15	Rec@20	MAP@5	MAP@10	MAP@15	MAP@20
	Games							
cSLIM	0.0761	0.1162	0.1474	0.1734	0.0590	0.0337	0.0240	0.0188
cfVAE	0.0685	0.1065	0.1359	0.1608	0.0519	0.0298	0.0212	0.0165
rVAE	0.0137	0.0206	0.0270	0.0375	0.0106	0.0060	0.0043	0.0034
fVAE	0.0495	0.0796	0.1072	0.1276	0.0390	0.0230	0.0167	0.0131
cVAE	**0.0858***	**0.1376****	**0.1731****	**0.2081****	**0.0668***	**0.0394****	**0.0279****	**0.0218****
	Sports							
cSLIM	0.0419	0.0622	0.0776	0.0911	0.0263	0.0148	0.0104	0.0080
cfVAE	0.0315	0.0512	0.0639	0.0768	0.0206	0.0119	0.0084	0.0065
rVAE	0.0171	0.0249	0.0328	0.0390	0.0109	0.0062	0.0044	0.0034
fVAE	0.0284	0.0437	0.0602	0.0732	0.0190	0.0109	0.0078	0.0061
cVAE	**0.0441**	**0.0655**	**0.0857***	**0.1035***	**0.0268**	**0.0152**	**0.0107**	**0.0084**

that the side information of Sports is useless for the recommendation task. Actually, fVAE provides acceptable recommendations by utilizing side information only. Therefore, the way of incorporating side information by cSLIM is not effective. In comparison, cVAE improves over cSLIM by utilizing side information.

4.2.3 Effect of the number of recommended items. Table 4 reveals a possible trend that the recommendation improvement becomes more evident when more items are recommended. We use Figure 3 to illustrate this, where N is increased from 5 to 1000. As depicted in Figure 3(a), the gaps between cVAE and other methods is getting larger with as N increases. It is interesting to note that fVAE surpasses cfVAE when $N = 500$ and $N = 1000$. This further demonstrates the effectiveness of a pre-training phase with side information proposed in this model. In Figure 3(c), both fVAE and cfVAE outperform cSLIM when $N \geq 200$, and fVAE outperforms cfVAE when $N \geq 100$. This shows that deep models are superior to linear models when more items are recommended. In comparison, the improvement achieved by cVAE is more evident when $N \geq 100$, and the gap between cVAE and the second best method is always substantial. In contrast, the performance w.r.t. MAP@N does not show big differences when N grows. Note that on the Games dataset (Figure 3(b)), cVAE performs much better than cSLIM when N is small. The improvement becomes less evident when N grows.

5 RELATED WORK

Related work concerns linear models for top-N recommendation with side information and deep models for collaborative filtering.

5.1 Top-N recommendation with side information

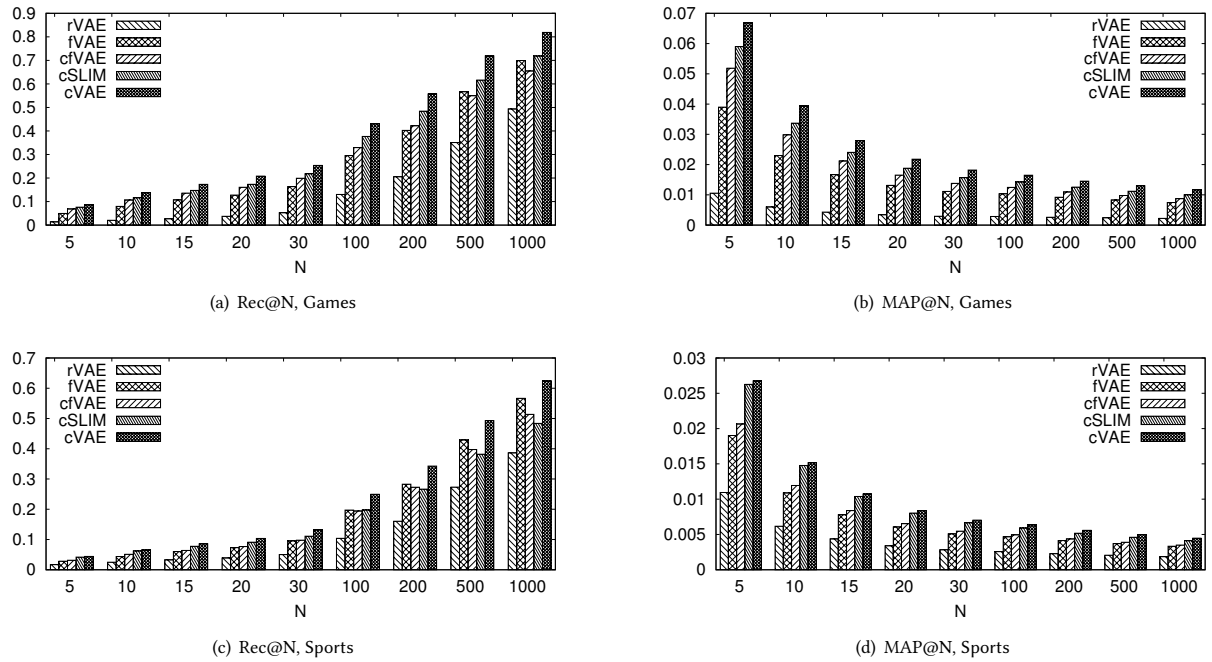
Various methods have been developed to incorporate side information in recommender systems. Most of these methods have been developed in the context of the rating prediction problem, whereas the top-N recommendation problem has received less attention. In the rest of this section we only review methods addressing top-N recommendation problems. Ning and Karypis [13] propose several methods to incorporate side information with SLIM [12]. Among these methods, cSLIM generally achieves the best performance as it can well compensate for sparse ratings with side information. Zhao et al. [22] propose a joint model to combine self-recovery for user rating and predication from side information. Side information is also utilized to address cold-start top-N recommendation. Elbadrawy and Karypis [2] learn feature weights for calculating item similarities. Sharma et al. [16] further improve over [2] by studying feature interactions. While these methods deliver state-of-the-art performance for top-N recommendation, they are all linear models, which have a restricted model capacity.

5.2 Deep learning for hybrid recommendation

Several authors have attempted to combine deep learning with collaborative filtering. Wu et al. [19] utilize a denoising autoencoder to encode ratings and recover the score prediction. Zhuang et al. [24] propose a dual-autoencoder to learn representations for both users and items. He et al. [3] generalize matrix factorization for collaborative filtering by a neural network. These methods utilize user ratings only, that is, side information is not utilized. Wang et al. [18] propose stacked denoising autoencoders to learn item representations from side information and form a collaborative deep learning method. Later, Li et al. [8] reduce the computational cost of training by replacing stacked denoising autoencoders by a marginalized denoising autoencoder. Rather than manually corrupting the input, variational autoencoders were later utilized for representation learning [9]. These models achieve state-of-the-art performance among hybrid recommender systems, but they are less effective when side information is high-dimensional. For more discussions on deep learning based recommender systems, see [21].

6 CONCLUSION

In this paper, we proposed a new way to feed side information to a neural network so as to overcome the high-dimensionality. We propose collective Variational Autoencoder (cVAE), which can be regarded as a non-linear generalization of cSLIM. cVAE overcomes rating sparsity by feeding both ratings and side information into the same inference network and generation network. To cater for the heterogeneity of information, i.e., rating and side information,



(a) Rec@N, Games

(b) MAP@N, Games

(c) Rec@N, Sports

(d) MAP@N, Sports

Figure 3: Performance on the top-N recommendation task.

we assume them to follow different distributions, which is reflected in the use of different loss functions. As for the implementation, we introduce a parameter α to balance the positive samples and negative samples. We also introduce β as a parameter for regularization, which controls how much the latent variable should comply with the prior distribution. We conduct experiments using two Amazon datasets. The results show the superiority of cVAE over other methods in scenarios with high-dimensional side information.

Deep models are effective if the number of inputs is sufficient. Thus, using side information to pre-train cVAE helps to overcome the high-dimensionality. As a rule-of-thumb, one should regularize cVAE lightly during pre-training and heavily when fine tuning.

SOURCE CODE

Source code to reproduce the experiments in this paper is available at https://github.com/shikamaruChen/cVAE.

ACKNOWLEDGMENTS

This research was partially supported by NSFC Grant No. 61872446.

REFERENCES

[1] Yifan Chen, Xiang Zhao, and Maarten de Rijke. 2017. Top-N Recommendation with High-Dimensional Side Information via Locality Preserving Projection. In *SIGIR*. ACM, 985–988.
[2] Asmaa Elbadrawy and George Karypis. 2015. User-Specific Feature-Based Similarity Models for Top-*n* Recommendation of New Items. *TIST* 6, 3 (2015), 33:1–33:20.
[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[4] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *SIGKDD*. ACM, 659–667.
[5] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2013).
[6] Mark A Kramer. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal* 37, 2 (1991), 233–243.
[7] Wonsung Lee, Kyungwoo Song, and Il-Chul Moon. 2017. Augmented Variational Autoencoders for Collaborative Filtering with Auxiliary Information. In *CIKM*. ACM, 1139–1148.
[8] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *CIKM*. ACM, 811–820.
[9] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *SIGKDD*. ACM, 305–314.
[10] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. ACM, 689–698.
[11] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. ACM, 165–172.
[12] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *ICDM*. IEEE, 497–506.
[13] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *RecSys*. ACM, 155–162.
[14] John William Paisley, David M. Blei, and Michael I. Jordan. 2012. Variational Bayesian Inference with Stochastic Search. In *ICML*. JMLR.
[15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *WWW*. ACM, 111–112.
[16] Mohit Sharma, Jiayu Zhou, Junling Hu, and George Karypis. 2015. Feature-based factorized Bilinear Similarity Model for Cold-Start Top-*n* Item Recommendation. In *SDM*. SIAM, 190–198.
[17] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *DLRS*. ACM, 11–16.
[18] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *SIGKDD*. ACM, 1235–1244.
[19] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. ACM, 153–162.
[20] Eric P Xing, Michael I Jordan, and Stuart Russell. 2002. A generalized mean field algorithm for variational inference in exponential families. In *UAI*. Morgan Kaufmann Publishers Inc., 583–591.
[21] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.07435 (2017).
[22] Feipeng Zhao, Min Xiao, and Yuhong Guo. 2016. Predictive Collaborative Filtering with Side Information. In *IJCAI*. 2385–2391.
[23] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *ICML*, Vol. 48. JMLR, 764–773.
[24] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. 2017. Representation learning via Dual-Autoencoder for recommendation. *Neural Networks* 90 (2017), 83–89.