공학석사 학위논문

Enhancing VAEs for Collaborative Filtering: Flexible Priors & Gating Mechanisms

복잡 사전분포 및 Gating 구조를 이용한 VAE 기반 협업필터링 성능 향상

2019년 8월

서울대학교 융합과학기술대학원

융합과학부 디지털정보융합전공

김 대 룡

Abstract

Since Matrix Factorization based linear models have been dominant in the Collaborative Filtering context for a long time in the past, Neural Network based CF Models for recommendation have started to gain attention recently. One branch of research is based on using deep generative models to model user preferences and Variational Autoencoders where shown to give state-of-the-art results.

However, there are some potentially problematic characteristics of the current Variational Autoencoder for CF. The first is the too simplistic prior VAEs incorporate for learning the latent representations of user preference, which may be restricting the model from learning more expressive and richer latent variables that could boost recommendation performance. The other is the model's inability to learn deeper representations with more than one hidden layer.

Our goal is to incorporate appropriate techniques in order to mitigate the aforementioned problems of Variational Autoencoder CF and further improve the recommendation performance of VAE based Collaborative Filtering. We bring the VampPrior, which successfully made improvements for image generation to tackle the restrictive prior problem. We also adopt Gated Linear Units (GLUs) which were used in stacked convolutions for language modeling to control information flow in the *"easily deepening"* autoencoder framework.

We show that such simple priors (in original VAEs) may be too restrictive to fully model user preferences and setting a more flexible prior gives significant gains. We also show that VAMP priors coupled with gating mechanisms outperform SOTA results including the Variational Autoencoder for Collaborative Filtering by meaningful margins on 4 benchmark datasets (MovieLens, Netflix, Pinterest, Melon).

Keywords: Recommender Systems, Deep Learning, Neural Collaborative Filtering, Deep Generative Models, Variational Auto-Encoders, Machine Learning, Information Systems

Student number: 2017-29907

Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 Background and Motivation

Deep Learning is the hot topic in almost every research field in todays' era and recommender systems are not an exception. Since the Netflix Prize in 2006, recommender systems have gained much attention in both academia and industry with Matrix-Factorization based collaborative filtering algorithms [15, 23, 26] being the long-standing king in the field of recommender systems. Matrix-Factorization methods have been popular for collaborative filtering because of its' impressive performance despite its' simple and intuitive idea of learning latent variables through matrix decomposition.

However, Matrix-Factorization methods are restricted to linear models and like other fields of machine learning there has been effort to apply neural networks and deep learning to recommender systems. Neural networks were used to conduct non-linear matrix factorization or learn rich non-linear latent variables of user preference. The results showed superior results compared to traditional matrix factorization techniques and recently Neural Network based collaborative filtering models have started to gain attention.

Deep Learning based Recommender Systems

There have been different kinds of recent studies incorporating deep learning into recommender systems. There are researches using deep learning for collaborative filtering. Extending the traditional matrix factorization framework to non-linear matrix factorization using neural networks [11], session-based recommendation using recurrent neural networks (RNNs) [12, 24, 37], recommendation with autoencoders and generative models [19, 28, 36, 38], and many others including hybrid methods using extraction of high-level content features through deep learning [32, 35].

In this work we focus on the branch of research using autoencoders and generative models which model latent variables of user preference. Recommendation can be done by using the latent variables of a given user to reconstruct the users' history for recommendation. There has been work using vanilla autoencoders [28], denoising autoencoders [38], and most recently variational autoencoders (VAEs) [19] to model user preference for collaborative filtering. To the best of our knowledge, Variational Autoencoders for collaborative filtering currently gives state-of-the-art results in the context of collaborative filtering.

Variational Autoencoders and Collaborative Filtering

Variational Autoencoders have been at the core of attention in Artificial Intelligence (AI) research in the last year. Especially in the domain of computer vision and signal processing, deep generative models such has VAEs were used in the task of image and audio generation. Also, a few shortcomings of the original VAEs were suggested and many interesting new researches were proposed tailoring VAEs for better performance in image generation.

However, Variational Autoencoders for Collaborative Filtering is in its' simplest form and there is definitely room left for further improvement. While many new variations of VAEs are being proposed in the domain of image and audio generation, there has not yet been much research that has yielded further success in the Collaborative Filtering task for recommender systems.

1.2 Research Goal

In this work we aim to overcome the problems of Variational Autoencoders in the task of Collaborative Filtering and appropriately tailor VAEs in order to further improve model performance and make high quality recommendations. While many new progresses have been made on VAEs in different domains, not all of them are suitable for the task of recommendation while some of them are. Our work incorporates ideas from different domains such as computer vision and natural language processing (NLP) to help Variational Autoencoders to better model user preferences for recommendation.

1.3 Enhancing VAEs for Collaborative Filtering

Two main motivations led our research. 1) The current prior distribution used in VAEs may be too restrictive for the Collaborative Filtering task, hindering the models from learning richer latent variables of user preference which is crucial to model performance. 2) Learning from user-item interaction history is different from learning from pixels (images) and may have its' own more effective architectures to model the characteristic of such data.

Flexible Priors

Original VAEs, including the research of VAEs for Collaborative filtering, use a unimodal multivariate standard Gaussian distribution for the prior distribution of the latent variables. The encoder, is trained to encode each data point to a posterior distribution matching the prior distribution of the latent variable. The main idea behind using such a simple prior distribution is that a flexible neural network is used as an encoder, expecting that whatever prior distribution we choose the encoder network will learn a posterior distribution matching the prior. However, this is a very idealistic assumption that the Encoder (and Decoder) will capture all the complex dependencies and factor the latent variables to a very simple distribution. There has been research in domains of image generation that this is not the case in real world applications, which lead to the question: could this also be hurting the expressiveness of latent variables learned by VAEs in Collaborative Filtering?

We implement Hierarchical Variational Autoencoders with VampPrior (Variational Mixture of Posteriors Prior) to learn richer latent representations of user preferences from interaction history. VampPrior is a very recent idea found effective in image generation relaxing the original restrictive prior to a more flexible prior which is an approximation to the optimal prior w.r.t the Evidence Lower Bound (ELBO).

Gating Mechanisms

Another variation we adopted different from the original research of VAEs for CF is that we used Gated Linear Units (GLUs) to successfully increase the depth of our model. Gated Linear Units are similar to Gated Recurrent Units (GRUs) in Recurrent Neural Networks. While GRUs control the information flow during the recurrent process of RNNs, Gated Linear Units control the information flow of the information upstream starting from the data at the bottom of the network to the output.

The Gating Mechanism enables to effectively control information flow of deep networks by learning ***when*** each item or feature contribute to certain units. Coupling the Gating Mechanism with the aforementioned VampPrior significantly boosted the performance of the Variational Autoencoding CF framework and outperformed current state-of-the-art Collaborative Filtering algorithms.

1.4 Experiment

Experiments were carried out on three popular public benchmark datasets and one private dataset of different domains and size: MovieLens 20M, Netflix, Pinterest and Melon. Our proposed method was compared to baseline models including state-of-the-art Matrix Factorization and Autoencoder based methods. Evaluation was done under the *strong generalization* setting where users were split into train/validation/test sets so that all click history information of a held-out user (for evaluation) was totally blocked at the training step.

1.5 Contributions

The key contributions of our work are as follows:

- Our work is the first to address the restrictive prior problem for the VAE-CF framework and shows that relaxing the prior to a more flexible distribution yields better recommendation performance.
- We show introducing gating mechanisms are also very helpful for

autoencoder based CF in learning deeper and more sophisticated representations of interaction history.

- Our proposed model using hierarchical VAEs with VampPrior and Gated Linear Units gives new State-Of-The-Art results on standard benchmark datasets in the task of collaborative filtering.

2 RELATED WORK

Prior studies which were essential to our research are arranged in four main themes: Collaborative Filtering, Deep Generative Models (VAE), Variational Autoencoders for Collaborative Filtering, Recent research in Computer Vision & Deep Learning.

2.1 Collaborative Filtering

Today, the immense size and diversity of Web-based services make it nearly impossible for individual users to effectively search and find online content without the help of recommender systems. Recommender systems is a domain in machine learning which has the goal of understanding and modeling the factors of user preference and predicting future behavior. Due to its direct practical use in e-commerce and close relatedness to human behavior, recommender systems are an important topic receiving much attention from both academia and the industry.

Collaborative Filtering (CF) is a popular technique used in recommender systems and designates a whole class of machine learning algorithms that uses collaborative information of users, items, etc. for recommendation. Collaborative

Filtering algorithms usually consume a very large history of user-item interaction to make personalized recommendations. The key idea of Collaborative Filtering is that people often get the best recommendations from someone with tastes similar to themselves. Collaborative Filtering uses the user-item interaction history to leverage information about the user preferences and make recommendations based on users with similar interest.

The boom of Collaborative Filtering algorithms was steered by the Netflix Prize. The Netflix Prize in 2009 was an open competition to develop to best collaborative filtering algorithm to predict user ratings for films, using the previous user-item consumption history. The competition was held by Netflix with a grand prize of $1,000,000 and triggered an explosive attention to the field of recommender systems. The winning solution of the Netflix Prize was an ensemble of predictors based on neighborhood models, Matrix-Factorization models and Restricted Boltzmann Machines (RBM).

In this section, we first review traditional neighborhood models and Matrix-Factorization based CF methods that has been the most popular in the field of recommender systems for a long time. Then we get to a more recent approach incorporating neural networks which is very closely related to our research: an Autoencoding framework to perform Collaborative Filtering. The Autoencoder also has a connection to the Restricted Boltzmann Machine (RBM) in the sense that they are an unsupervised latent variable model.

2.1.1 Traditional methods & Matrix-Factorization based CF

Traditionally Collaborative Filtering methods could be divided into neighborhood-based methods and model-based methods. Nowadays CF algorithms are mostly model-based, including the Matrix-Factorization based methods and Neural Network based methods.

Neighborhood-based methods

Before the Matrix-Factorization based CF gained popularity, a common Collaborative Filtering method was the neighborhood-based approach and had the form of user-based CF and item-based CF. User-based CF for example is made up of the following two steps:

1. Calculate users who have similar consumption patterns as the given user
2. Use the ratings from the similar users of step 1 to calculate the predicted ratings of items for given user

Item-based collaborative filtering follows similar steps in an item-centric manner. These methods were called Neighborhood-Based approaches as it finds explicit neighbors to calculate predicted ratings from. The subsequent Matrix-Factorization Collaborative Filtering can be categorized as a Model-based approach.

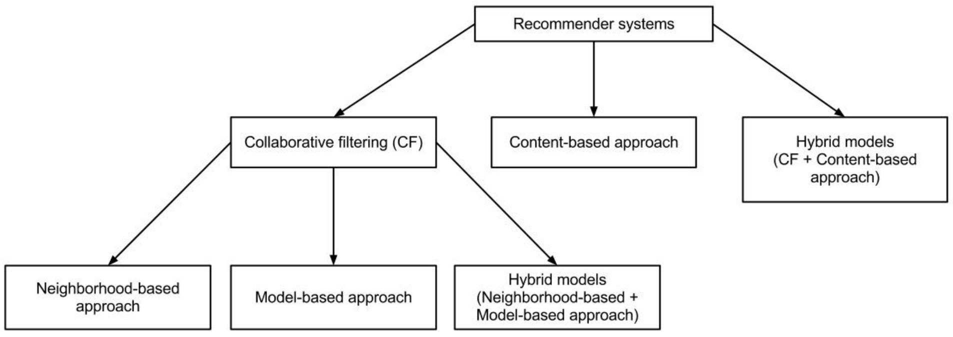
Recommender systems
Collaborative filtering (CF)
Content-based approach
Hybrid models (CF + Content-based approach)
Neighborhood-based approach
Model-based approach
Hybrid models (Neighborhood-based + Model-based approach)

Figure 2-1. Types of Collaborative Filtering in Recommender Systems

Matrix Factorization CF

While neighborhood-based approaches took a user-centric or an item-centric viewpoint to make recommendations, Matrix-Factorization methods learn user latent factors and item latent factors simultaneously. The initial idea was formed by applying low-rank matrix decomposition techniques on the user-item history matrix such as Singular Value Decomposition (SVD) and Principle Component Analysis (PCA).

A simple form of Matrix-Factorization Collaborative Filtering is as follows. For every user item pair u, i the user's rating for the item can be predicted as below.

$$\hat{r}_{ui} = x_u^T y_i$$

$x_u \in R^d$ is the latent vector of user u and $y_i \in R^d$ is the latent vector of item i. The dimension of the latent vectors d is usually much smaller than the size of users or items ($d \ll N, M$). The latent vectors $X \in R^{N\times d}$ and $Y \in R^{M\times d}$ can be learnt through optimizing the following objective function for all u, i pairs that have a given rating (or all pairs with unobserved ratings as 0):

$$\min_{x_*, y_*} \sum_{r_{ui} \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

λ is a regularizing parameter to control for the norm of x_u and y_i. A diagram visualizing the Matrix-Factorization algorithm can be seen in figure 2-2.

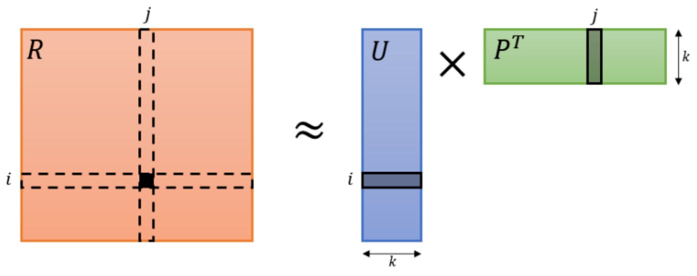

Figure 2-2. Visualization of Matrix-Factorization CF

Recently, researches focus more on implicit feedback of users such as click or purchase rather than explicit ratings. The reason is that many users using Web-based services do not bother to give explicit ratings to the items they consumed and therefore explicit feedback is hardly available. A popular realization of Matrix-Factorization CF on implicit feedback is the Weighted Matrix-Factorization (WMF) algorithm [15].

First, it models binary variables p_{ui} indicating implicit preference of user u to item i. The p_{ui} values can be derived by binarizing the r_{ui} values in the explicit case:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

And the model can be computed by minimizing the corresponding objective function:

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

c_{ui} is a *confidence level* or *weight* chosen by the researcher reflecting the amount of preference or importance of the user-item pair. c_{ui} may set to incorporate the value of r_{ui} or simply the existence of interaction if only implicit data is available.

The Matrix-Factorization model can be optimized in various ways: applying matrix decomposition (SVD, PCA, …) directly on the preference matrix, Stochastic Gradient Descent (SGD), Alternating Least Squares (ALS) and more. Also, the MF model can be further extended by adding additional terms in the model formula. User and item intercepts, temporal dynamics, content and contextual features are popular choices. There are also probabilistic versions of MF so called Probabilistic Matrix Factorization (PMF) by using probabilistic graphical models to solve the Matrix-Factorization problem.

Matrix-Factorization methods for Collaborative Filtering have been thoroughly researched and used in many industrial applications. It has impressive performance despite its simplicity and also scales to large size datasets making it favorable to industrial applications. However, one important drawback is that MF methods are restricted to linear models. This led to the research of using neural networks, a non-linear universal function approximator, in the field of recommender systems.

2.1.2 Autoencoders for Collaborative Filtering

Even though Matrix-Factorization based methods of CF had many advantages, there were needs of more sophisticated non-linear modeling in CF. The heroic breakthroughs Neural Networks (NN) has made in other domains such as object recognition in ImageNet has led attempts to apply Neural Networks to Collaborative Filtering. NN based CF methods conduct non-linear transformation techniques to model user-item preferences. The ability of Neural Networks to conduct non-linear transformations was shown to improve the performance of recommender systems compared to the traditional linear models.

Neural Network based CF

Several branches of research using Neural Networks for Collaborative filtering are being actively studied. The most typical is Non-Linear Matrix Factorization, it uses the Neural Network architecture to learn non-linear latent factors of users and items. The latent factors go through several layers of a Multilayer Perceptron (MLP) to predict the user-item rating. This is different from the original linear Matrix-Factorization in the sense that it can learn any non-linear function of the two latent factors instead of just the dot product. Figure 2-3 shows a diagram of Neural Collaborative Filtering [11] conducting non-linear matrix factorization.

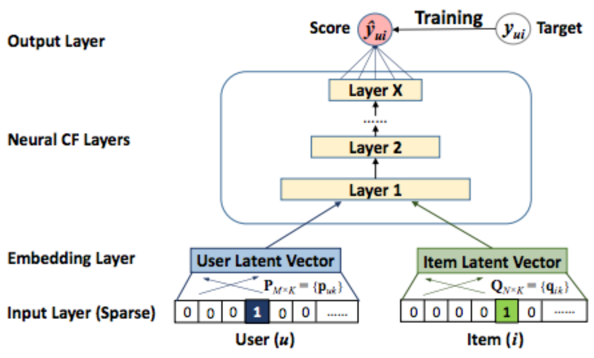

Figure 2-3. Neural Collaborative Filtering Framework [11]

Another line of studies uses Recurrent Neural Networks (RNNs) to make Session-based recommendations. RNNs take in the sequence of item IDs in a given session and predict the next item likely to be consumed. The user ID need not to be tracked and the sequence of item IDs in the session recurrently updates the hidden state of the model. The model predicts the next item based on the current hidden state. A popular work is the Session-Based Recommendations with Recurrent Neural Networks [12] shown in figure 2-4, advances of the research continued after on [24, 37].

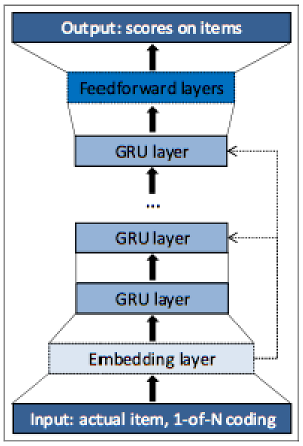

Figure 2-4. Session-Based Recommendations with RNNs [24]

Algorithms using Autoencoders and Generative models also take a position in the lines of research in NN based recommendation. This line of research is directly connected to this current work and will be discussed in more detail as we go on.

There are more various kinds of studies incorporating Neural Networks for recommendations. A popular method is using NNs for extracting deep content features from the items [32, 35]. There are also Autoregressive methods [41], Deep Reinforcement Learning based methods [40], and much more. In this work we focus on the works using autoencoders and generative models for recommendation.

Autoencoder based Recommendation

The Autoencoder based recommendation algorithm was first proposed as AutoRec [28] by Sedhain et al. Autoencoders are similar to RBMs in the fact that they are unsupervised latent variable models and they can become the same under certain objectives [34]. RBMs explicitly model the joint distribution of the *hidden* and *visible* variables. Autoencoders are much more intuitional, consisting of the input, encoder, latent code, decoder, and reconstruction.

Autoencoders take in whatever input and encodes it into a latent code, typically in a lower dimension. The decoder then decodes the latent code to reconstruct the original input. The difference between the original input and reconstructed input, namely the reconstruction error, is used as the objective function to train autoencoders.

See figure 2-5 for an example of using Autoencoders for collaborative filtering [28].

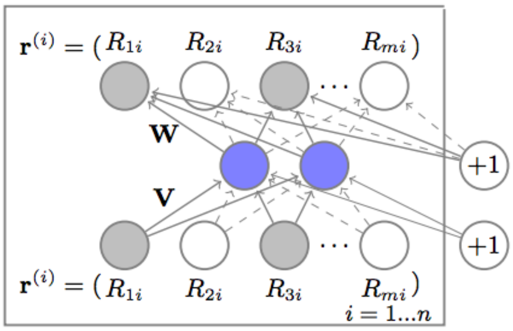
$\mathbf{r}^{(i)} = (R_{1i} \quad R_{2i} \quad R_{3i} \quad R_{mi})$
W
V
+1
+1
$\mathbf{r}^{(i)} = (R_{1i} \quad R_{2i} \quad R_{3i} \quad R_{mi})$
$i = 1...n$

Figure 2-5. Collaborative Filtering with Autoencoders [28]

The input of the autoencoder in the case of collaborative filtering is the rating history $r^{(i)}$ of a specific user i. $r^{(i)}$ is a vector of size N, the of the number of items with the given users' ratings for each item if rated (it can be binary in the case of implicit feedback). The encoder learns a low-dimension mapping from the input to a latent variable which can be interpreted as the users' preference. The decoder then tries to reconstruct the original ratings from the users' preference. Its prediction gives estimates for the missing (unobserved) ratings and they can be used for recommendation.

A one-layer Autoencoders reconstruction of input $\boldsymbol{r} \in R^N$ can be written as the following,

$$h(\boldsymbol{r}; \theta) = f(\boldsymbol{W} \cdot g(\boldsymbol{V}\boldsymbol{r} + \boldsymbol{\mu}) + \boldsymbol{b})$$

with activation functions $f(\cdot)$, $g(\cdot)$, parameters $\theta \in \{\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{\mu}, \boldsymbol{b}\}$ with transformations $\boldsymbol{W} \in R^{N\times k}$, $\boldsymbol{V} \in R^{k\times N}$ and biases $\boldsymbol{\mu} \in R^k$, $\boldsymbol{b} \in R^N$. This is a 1-layer AutroRec model with k-dimensional latent code. The Model can be trained following the optimization of the objective function:

$$\min_{\theta} \sum_{i=1}^{n} \left\| \mathbf{r}^{(i)} - h\left(\mathbf{r}^{(i)}; \theta\right) \right\|_{O}^{2} + \frac{\lambda}{2} \cdot (\|\mathbf{W}\|_F^2 + \|\mathbf{V}\|_F^2)$$

with λ as the regularization parameter.

The AutoRec approach [28], which is the algorithm of using vanilla autoencoders for collaborative filtering, was shown to give superior results compared to linear MF methods and RBM methods.

Further research was made using Denoising Autoencoders (DAEs) [38]. Random noise was injected at the input layer of the autoencoder, and the model was trained to reconstruct input $\boldsymbol{r} \in R^N$ from its (partially) corrupted version $\tilde{\boldsymbol{r}}$. The noise was anticipated to force the hidden layer discover more robust features and prevent it from simply learning the identity function. The noise injection technique had positive regularization effects and further improved the model performance of autoencoder based CF.

2.2 Deep Generative Models (VAEs)

Deep Generative Models are in the spotlight of AI and Machine Learning research today. It has eye-catching outputs bringing much interest to the potential of AI research (see figure 2-6). Deep Generative Models have been successful in domains such as image generation, voice synthesis, style-transfer and more.

Deep Generative Models are *generative models* using the power of Deep Learning. Generative models aim at learning the underlying true data distribution from training data. If we can recover the data probability distribution $P(X)$ we can do almost everything we want with data: conduct arbitrary inference through conditionalization and marginalization (regression and classification are also inference), generate new samples from the data distribution $X_{new} \sim P(X)$.

Two most commonly used and efficient approaches are Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). Variational Autoencoders explicitly model the data distribution by maximizing the lower bound of the data log-likelihood and Generative Adversarial Networks implicitly learn the data distribution through adversarial training.

In this section we will focus on Variational Autoencoders (VAEs) as it is the base framework used in our research for recommendation. We review the idea of general VAEs and how it is trained through Variational Inference.

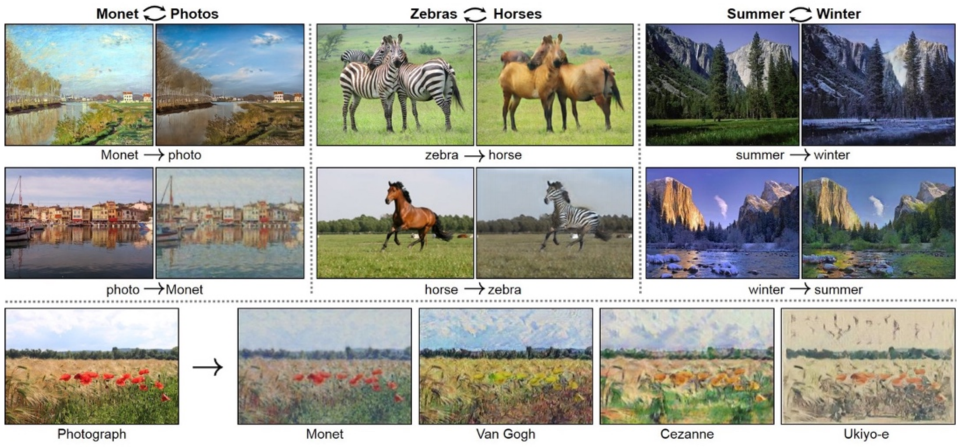

Figure 2-6. Style-Transfer with Deep Generative Models [7]

2.2.1 Variational Bayes

Variational Bayes, or Variational Bayesian methods are a family of techniques for approximating intractable integrals arising in Bayesian inference and machine learning. In the setting where there are unobserved latent variables and observed training data, Variational Bayes can be used to make analytical approximations to the true intractable posterior probability of latent variables which in turn can be used to derive a lower bound for the marginal likelihood of the observed data. This lower bound, so called the Evidence Lower Bound (ELBO) can be used to optimize the Variational Autoencoder (VAE). We will see the detailed implementation of the idea for the VAE framework in the following section 2.2.2.

2.2.2 Variational Autoencoders

A Variational Autoencoder (VAE) [16] is a generative model which attempts to model the data distribution $p(X)$ with the assumption there exists a latent structure z using neural networks.

The scenario consists of two steps: (1) a value z^i from some prior distribution $p_{\theta^*}(z)$ is generated, (2) a data point x^i is generated from the conditional distribution $p_{\theta^*}(x|z)$. The distributions are parameterized by θ with the true parameters θ^* and we would like to find them by maximizing the marginal likelihood $p_\theta(x) = \int p_\theta(x|z)p_\theta(z)\,dz$ w.r.t θ. However, we would like to use the expressive deep neural networks to model the generation process $p_\theta(x|z)$ which then makes the marginal likelihood $p_\theta(x)$ and true posterior $p_\theta(z|x) = \frac{p_\theta(X|Z)p_\theta(z)}{p_\theta(x)}$ intractable.

Variational Autoencoders solve the problem using the Variational Bayes approach. In order to do so, we set a *generative model (decoder)* $p_\theta(x|z)$ which is a neural network parameterized by θ, a *prior distribution* of latent variables $p_\lambda(z)$, and an approximation to the unknown posterior $p_\theta(z|x)$ with a *recognition model (encoder)* $q_\phi(z|x)$ also with neural networks.

The Variational Lower Bound

We will be applying Stochastic Gradient Decent (SGD) methods for batches of data and the marginal log-likelihood is composed of the sum of the marginal log-likelihood $\log p_\theta(x^{(i)})$ for each data point in the batch. Using the generative model, recognition model and prior defined above, each marginal log-likelihood can be rewritten as:

$$\log p_\theta(\mathbf{x}^{(i)}) = D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\|p_\theta(\mathbf{z}|\mathbf{x}^{(i)})\big) + \boldsymbol{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

Since the Kullbak-Leibler divergence can take on only non-negative values, the second term of the RHS becomes a lower bound of the marginal log-likelihood. Which is the following:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}\left(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})\|p_{\theta}(\mathbf{z})\right) + \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z})]$$

This is Variational Lower bound also called the Evidence Lower Bound (ELBO) [16]. The objective is to optimize the lower bound w.r.t both the encoder parameters ϕ and decoder parameters θ in order to indirectly optimize the marginal log-likelihood.

Standard Normal Prior

The Variational Autoencoder chooses the prior distribution of z as the standard normal distribution considering that we used a flexible neural network for approximating the posterior. This leads to an analytic solution to the KL-divergence part of the ELBO resulting in a realization of the following:

$$\log q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}\right)$$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$$
$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

L is the number of samples used to approximate the negative reconstruction error which can be set to 1 for a large enough batch size. The aggregated ELBO will be optimized to train the model.

2.3 Variational Autoencoder for Collaborative Filtering

After the attempt of using vanilla Autoencoders [28] and Denoising Autoencoders [38] for collaborative filtering, the Variational Autoencoder was adapted for collaborative filtering [19]. To the best of our knowledge, the Variational

Autoencoder for Collaborative Filtering currently shows the state-of-the-art (SOTA) performance and is one of the most important researches to our proposed work.

2.3.1 VAE for CF

Variational Autoencoders for CF uses the VAE framework in the task of Collaborative Filtering. It has a few adaptions such as introducing the beta parameter controlling the impact of the prior and using the multinomial likelihood to calculate the reconstruction error instead of the original binary cross-entropy.

The model starts by sampling a K-dimensional latent representation z^u for each user u from a standard normal prior distribution and is transformed by a neural network generative model to produce the probability distribution over the user's item consumption history x^u, a bag-of-words vector indicating whether the user has consumed each item, assuming a multinomial distribution:

$$\mathbf{z}_u \sim N(0, \mathbf{I}_K), \quad \pi(\mathbf{z}_u) \propto \exp\{f_\theta(\mathbf{z}_u)\}$$
$$\mathbf{x}_u \sim Multi\big(N_u, \pi(\mathbf{z}_u)\big)$$

$f_\theta(\cdot)$ is a non-linear function parameterized by θ and $\pi(z_u)$ is the probability vector over the entire item set. Then the same Evidence Lower Bound (ELBO) can be derived as in section 2.2.2:

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

$p_\theta(x|z)$ is the generative model and $q_\phi(z|x)$ is the recognition model (variational posterior).

The model makes slight changes to ELBO by introducing the parameter β to control for the impact of the KL-divergence term and takes it as the new objective function [19]:

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{\beta}}(\mathbf{x}_u; \boldsymbol{\theta}, \boldsymbol{\phi}) \equiv \mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{x}_u)}[\log p_\theta(\mathbf{x}_u|\mathbf{z}_u)] - \beta \cdot \mathrm{KL}\left(q_\phi(\mathbf{z}_u|\mathbf{x}_u)||p(\mathbf{z}_u)\right)$$

If we look at the objective function, the first term of the RHS is the negative reconstruction error (this is the objective function of vanilla autoencoders) while the second term is a regularizer forcing individual posteriors of data points to match the prior of z. The new objective added the β parameter too loosen the effect of KL-divergence for better recommendations. The idea was that posterior and prior need not to match as much in the task of collaborative filtering. We only need to reconstruct the original user history to make predictions for unseen items, we do not want to sample from a random z to generate new samples from $p_\theta(x)$.

Adequately tuning the parameter β led to superior performances compared to prior autoencoder based CF algorithms (obviously it beat linear MF methods as well) and showed state-of-the-art performance for Collaborative Filtering. See figure 2-7 for a taxonomy of autoencoders used for CF.

It can be interpreted that the VAE framework, learning a stochastic latent representation of the user preference, acted as a regularizer and helped the model to learn more useful representations.

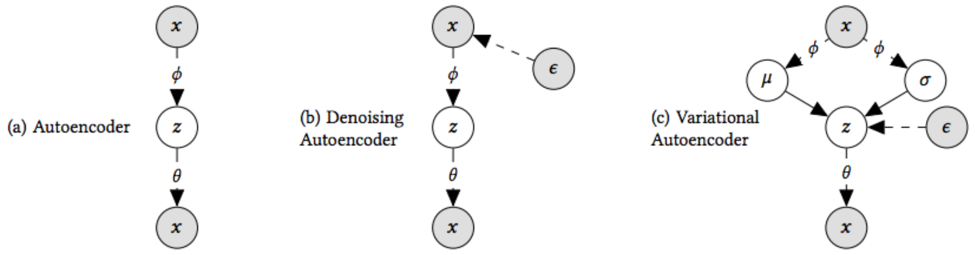

Figure 2-7. Taxonomy of autoencoders used for CF [19]

Is this the best we can do?

However, in this original research of VAEs for CF the variational posterior (distribution of latent variables) of the data is pulled towards a very simple standard normal distribution. To make things worse, the model is shallow with one hidden layer (simply adding more layers does not improve performance). What this means is that we should be suspicious of the ideal assumption that the encoder is currently capturing all the complex dependencies and factoring the latent variables to a very simple distribution. If this is not the case, the simple standard normal prior may be restricting the model from learning richer latent representations which is crucial to making recommendations with CF.

In our work we claim that we need a more effective structure to help the model learn deeper representations and a more flexible prior that will not restrict the model from learning richer latent representations.

2.4 Recent research in Computer Vision & Deep Learning

VAEs for CF currently shows the state-of-the-art results on Collaborative Filtering, but as we pointed out at the end of the last section there are some questions to be answered.

To tackle the proposed problems, we incorporated techniques developed in different domains such as computer vision and natural language processing. The two main ideas we borrowed are from VAE with VampPrior [31] and Gated CNN [4].

2.4.1 VampPrior

There have been a line of research concerning the simple prior of vanilla VAEs [13, 20, 31]. A recent research [31] proposed the "Variational Mixture of Posteriors" prior, or VampPrior for short, and showed successive results in the domain of image generation. It showed that relaxing the original restrictive prior by setting a more flexible prior improved performance.

The motivation of VampPrior starts from the Evidence Lower Bound (ELBO) we saw previously in VAEs. The ELBO, originally interpreted as the negative reconstruction error and KL-divergence term, can be further decomposed into three terms:

$$\begin{aligned}\boldsymbol{\mathcal{L}}(\theta,\phi,\lambda) = {} & \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]\right] \\ & +\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\mathbb{H}[q_\phi(\mathbf{z}|\mathbf{x})]\right] \\ & -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[-\log p_\lambda(\mathbf{z})]\end{aligned}$$

Maximizing $\mathcal{L}(\phi,\theta,\lambda)$ is our objective while the third term is made up of the prior distribution $p_\lambda(z)$ which is chosen in advance as the standard normal distribution. However, one could find a prior that optimizes the ELBO w.r.t the prior by solving the Lagrange function. Solving the problem simply gives the aggregated posterior as the optimal prior:

$$p_{\lambda}^{*}(\mathbf{z}) = \frac{1}{N}\sum_{u=1}^{N} q_{\phi}(\mathbf{z}|\mathbf{x}_u)$$

However, using this choice as the prior will potentially lead to overfitting and computational issues. Therefore, the VampPrior proposes an approximation of the aggregated posterior by a mixture of variational posteriors with *pseudo-inputs*:

$$p_{\lambda}(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_{\phi}(\mathbf{z}|\mathbf{u}_k)$$

with $K \ll N$ as the number of pseudo-inputs and u_k the N-dimensional (size of item set) vector of the pseudo-input. The values of pseudo-inputs are learned through backpropagation during training and can be viewed as hyperparameters of the prior. If we look at the shape of the prior we can see that it is the form of a *mixture of gaussian* distribution, resulting in a multimodal distribution.

One interesting aspect is that the VampPrior is comprised of the variational posterior (encoder). The work claims that the prior and encoder will “cooperate” during training to yield helpful results.

2.4.2 Gated Convolutional Neural Network

The study of gated Convolutional Neural Network (CNN) was proposed as an application of natural language processing [4]. It introduced the Gated Linear Unit (GLU) and used it with stacked convolutions to produce superior results in language modeling compared to previous models.

Gating mechanisms in neural networks can control the path of how information flows through the network and have been proven to be useful for Recurrent Neural Networks (RNNs). Long Short-Term Memory networks (LSTMs) are a popular choice of gated networks that control the information flow of each cell by activating the input and forget gates. Gating mechanisms mitigate the problem of vanishing gradients during the recurrent structure and selects how to aggregate the sequential information passed on the network.

As the structure of Neural Networks get deeper and deeper, non-recurrent neural nets also have the problem of being unable to properly propagate information from the bottom layer to the top. Following this idea, Gated CNNs applied gates to the non-recurrent CNN architecture and introduced Gated Linear Units (GLUs).

Gated Linear Units consider only output gates, which allow the network to control what information should be propagated through the hierarchy of layers. It has a relatively simple form as shown below:

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

With $\mathbf{X}$ the input of the layer and $\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}$ learned parameters, σ is the sigmoid function. As we can see from the formula, how the gates react on the given transformation of the input $(\mathbf{X} * \mathbf{W} + \mathbf{b})$ is also different depending on the current input. This can also be interpreted as potentially increasing the network's modeling capacity.

Gated Linear Units can also be compared to Rectified Linear Units (ReLUs). We can express the ReLU function as the following:

$$\mathrm{ReLU}(\mathbf{X}) = \mathbf{X} \otimes (\mathbf{X} > 0)$$

Then the ReLU can be seen as a simplification of GLUs with a fixed deterministic gate depending on the sign of the input.

The Gated Linear Units made a vast difference and provided useful modeling capacity in the task of language modeling. Since many word-level NLP techniques also work well with item recommendation techniques (such as the Word2Vec algorithm which is also popular for learning item representations for recommenders), there is potential that the Gating Mechanism will be helpful to recommender systems.

3 Method

Many forms of neural network based Collaborative Filtering (CF) systems are being proposed and the Variational Autoencoder (VAE) framework for CF is showing state-of-the-art (SOTA) performance [19]. However, we see some potentially problematic characteristics of the current Variational Autoencoder CF framework. The first is the too simplistic prior VAEs incorporate for learning the latent representations of user preference, which may be restricting the model from learning more expressive and richer latent variables that could boost recommendation performance. The other is the model's inability to learn deeper representations with more than one hidden layer.

Our goal is to incorporate appropriate techniques in order to mitigate the aforementioned problems of Variational Autoencoder CF and further improve the recommendation performance of VAE based Collaborative Filtering. We bring the VampPrior, which successfully made improvements for image generation [31] to tackle the restrictive prior problem. We also adopt Gated Linear Units (GLUs) which were used in stacked convolutions for language modeling to control information flow in the *"easily deepening"* autoencoder framework.

Thus, our proposed model can be summarized as an extention of the original Variational Autoencoding CF with VampPrior and GLUs.

3.1 Flexible Prior

In this section we describe the methods we used including flexible priors in order to learn richer latent representations of user preference.

3.1.1 Motivation

As we have seen in section 2.3.1 the Variational Autoencoder for CF attempts to maximize the following objective function:

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{\beta}}(\mathbf{x}_u; \boldsymbol{\theta}, \boldsymbol{\phi}) \equiv \mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{x}_u)}[\log p_\theta(\mathbf{x}_u|\mathbf{z}_u)] - \beta \cdot \mathrm{KL}\left(q_\phi(\mathbf{z}_u|\mathbf{x}_u)||p(\mathbf{z}_u)\right)$$

Which is a modified version of the Evidence Lower Bound (ELBO), a lower bound on the marginal log-likelihood $p(X)$. If we look at the objective, the first term can be interpreted as the negative reconstruction error while the second KL term acts as regularizer pulling the variational posterior towards the prior $p(z)$. This results in shaping the aggregated posterior close to the prior.

Giving restrictions to the aggregated posterior (rather than letting it be scattered everywhere) acts as a regularizer and helps the model learn more meaningful latent representations. Which, in turn helps the recommendation performance.

However, the prior $p(z)$ is chosen in advance as the multivariate standard normal distribution. This distribution is unimodal with no covariance structure. Which means the encoder is forced to learn a mapping of user preferences were the latent representation matches a very simple unimodal distribution. A question arises, *"is this reasonable?"*

The original idea of using standard normal priors comes from an **idealistic assumption**: *the neural network will capture all the complex dependencies and factor the latent variables to a very simple distribution*. Neural Networks are indeed flexible and theoretically universal function approximators, but does the assumption really hold for the current context of Collaborative Filtering?

Modeling human preference is a very complicated task, looking at the metrics such as recall@k of current CF algorithms gives the feeling that the models are not yet near perfection. Furthermore, the current autoencoder based CF algorithms (AE, DAE, VAE, ...) fail to learn deep representations of more than 1 hidden layer. Overall, it becomes plausible that the *ideal assumption* is currently not the case and the too simplistic prior may be restricting the model from learning richer representations. This calls for a need to replace the prior to a more flexible one.

3.1.2 VampPrior

The *Variational Mixture of Posteriors* prior (VampPrior) [31] is a recently proposed type of prior which is derived by analyzing the variational Evidence Lower Bound (ELBO). VampPrior consists of a *mixture of gaussian* distribution were components are given by the encoder (variational posterior) conditioned on learnable pseudo-inputs. The VampPrior looks like the following:

$$p_\lambda(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z}|\mathbf{u}_k)$$

u_k are trainable pseudo-inputs and $K(\ll N)$ is the number of pseudo-inputs which controls the level of flexibility. Since individual $q_\phi(z|u_k)$ are gaussians, the resulting distribution becomes a *mixture of gaussians* which is multimodal.

Our model is modified to use the VampPrior as the prior distribution. The prior now indirectly learns from data the appropriate distribution by referring to the encoder distribution and is also multimodal, making the prior much more flexible compared to the original standard normal prior overall. The level of flexibility is controlled by K and calculation of the KL divergence (in the objective function) can be done through Monte-Carlo estimation.

3.1.3 Hierarchical Stochastic Units

In order to learn even richer latent representations, we also followed the approach in VampPrior [31] to change the original latent variable z to a stacked hierarchical structure of z_1 and z_2. There are now two layers of stochastic latent variables instead of one. A visual diagram of the new hierarchical structure is shown is figure 3-1.

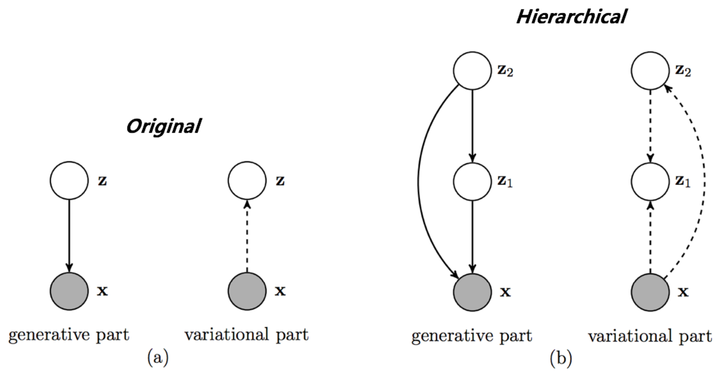

Figure 3-1. Diagram comparing the original and hierarchical stochastic layers [31]

The variational part of the Hierarchical VAE is now:

$$q_{\phi}(\mathbf{z}_1|\mathbf{x},\mathbf{z}_2)\, q_{\psi}(\mathbf{z}_2|\mathbf{x}),$$

and the generative part as follows:

$$p_{\theta}(\mathbf{x}|\mathbf{z}_1,\mathbf{z}_2)\, p_{\lambda}(\mathbf{z}_1|\mathbf{z}_2)\, p(\mathbf{z}_2)$$

with $p(z_2)$ given by a VampPrior, $\{\phi,\psi\}$ the variational parameters and $\{\theta,\lambda\}$ the generative parameters.

3.2 Gating Mechanism

In this section we describe the gating mechanism adopted in our proposed model to help learn deeper representations and increase model capacity.

3.2.1 Motivation

The autoencoding framework consists of the following: input, encoder, latent variables, decoder, reconstructed input. Due to the presence of the encoder and the decoder, as we increase the number of hidden layers the depth of the total network increases much faster. Since in our case we use two layers of latent variables, a VAE with 1 hidden layer actually ends up with a network of depth $2*(1)+2=4$ and a VAE with 2 hidden layers a depth of $2*(2)+2=6$. Eventually, an autoencoder with only a few hidden layers end up as a relatively deep structure (see figure 3-2 for a visualization of a 2-layer autoencoder).

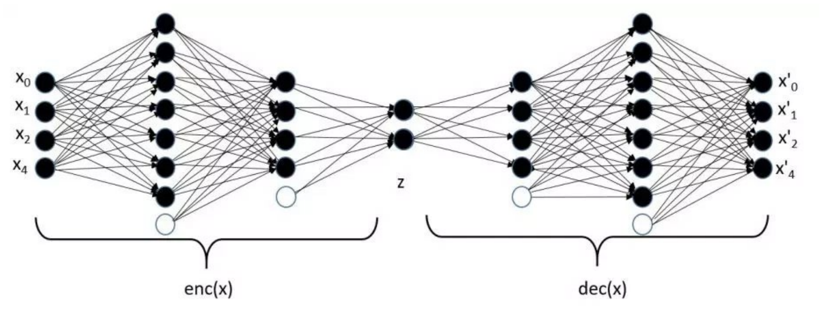

Figure 3-2. Visualization of a 2-layer Autoencoder (the depth of an autoencoder end up relatively deep)

The current problem is, the autoencoder based CF algorithms are having trouble learning representations of more than 1 hidden layer. Preceding researches using vanilla Autoencoders, Denoising Autoencoders, Variational Autoencoders did not achieve significant performance gain by adding additional hidden layers. We anticipate two reasons for this. (1) The nature of the data, extracting preference from consumption history is a complex problem and the current NN structure may not be effectively enforcing it. (2) The relatively *easily deepening* autoencoder structure, deep neural networks are hard to train because information may not properly propagate through the whole network.

Our model adopts Gated Linear Units to control for the information flow of the deep network in order to help train deeper networks. The gating mechanism can also be interpreted as increasing the capability of individual units to capture more complex dependencies.

3.2.2 Gated Linear Units (GLUs)

Gating mechanisms are commonly used in Recurrent Neural Networks (RNNs) to control the path of how information flows through the long recurrent process. Gated Linear Units (GLUs) are a Gating Mechanism that can be used in Non-Recurrent Networks to control information flow in deep networks.

As seen in section 2.4.2, the Gated Linear Unit has a simple formulation as the following:

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

The gate retains the non-linear capability of the unit so no additional activation function is needed. X is the input of the layer, W, b are linear transformations applied to X and V, c are learned parameters for controlling the gates. σ is the sigmoid function.

We adopt the Gated Linear Units as the default type of unit for all units in the network.

4 EXPERIMENTS

Experiments were conducted to evaluate the effect of flexible priors, hierarchical stochastic units and gating mechanisms in the context of collaborative filtering. Our proposed models are compared to other state-of-the-art collaborative filtering models. The experiments were made on three popular benchmark datasets (MovieLens, Netflix, Pinterest) and one private dataset (Melon).

4.1 Setup

The problem setup is for the Collaborative Filtering algorithms to make recommendations using binary implicit feedback. The models can use only the pure user-item interaction history with no information about the context or item content.

4.1.1 Baseline Models

We use the most popular Matrix Factorization models and state-of-the-art Auto-encoder models as baseline models to compare with our model.

- **Weighted Matrix Factorization (WMF)** [15]: A linear low-rank matrix factorization model trained with alternating least squares. The model is explained in detail in section 2.1.1. The weights on all the 0's were set to 1 and the weights on all the 1's were tuned among {2, 5, 10, 30, 50, 100}. Also the dimension of the latent representation was set between {100, 200}.

- **SLIM** [23]: A linear model which learns a sparse item-to-item similarity matrix through solving a L1-regularized constrained optimization problem. The regularization parameters were searched over {0.1, 0.5, 1, 5}.
- **Collaborative Denoising Autoencoder (CDAE)** [38]: An autoencoder collaborative filtering model with additional noise injection and per-user latent factor in the input. The noise injection is used for learning more robust representations and is explained in section 2.1.2. The latent dimension was set to 200 with tanh activations for the network. Since the number of parameters for CDAE grows linearly with the number of users and items, overfitting was controlled by applying weight decay with the parameter examined over {0.01, 0.1, … , 100}.
- **Multi-VAE** [19]: Variational autoencoder with multinomial likelihood. The model is thoroughly explained in section 2.3 and was shown to achieve state-of-the-art results in the collaborative filtering context. Modeling the per-user variances in the latent state z_u led to superior results compared to the original autoencoder. Tanh activations were used and parameters such as beta, dimension of the hidden layers and latent state are tuned in accordance with all the other proposed models following 4.3.

4.1.2 Proposed Models

Models to evaluate the individual effects of flexible priors, HVAE, and gating are the following.

- **Multi-VAE (Gated)**: The Multi-VAE model with gating mechanisms. Gated Linear Units were used for all hidden units in the network. This model was studied for the individual effect of gating on the original VAE for CF and comparison.
- **Vamp**: Variational autoencoder with a VampPrior as the prior distribution instead of the original standard gaussian prior. We can compare with Multi-VAE to evaluate the effect of using flexible priors.
- **H + Vamp**: Hierarchical VAE with the VampPrior, the difference to the Vamp model is that it has hierarchical stochastic units to model the latent representation.

Our final proposed model:

- **H + Vamp (Gated)**: Our final model, additional gating mechanisms are applied to the H + Vamp above. Gated Linear Units are used for all hidden units in the network.

4.1.3 Strong Generalization

The performance of various models was evaluated under the *strong generalization* setting [19, 21]. All users are split into training/validation/test sets. Models are trained using the entire click history of training users. For evaluation, we take 80% of the click history from the validation (or test) dataset to calculate the necessary user-level representations and predict the remaining 20% of the dataset.

4.1.4 Evaluation Metrics

We use the metrics NDCG@K and Recall@K to evaluate the performance of the models. Recall@K can be interpreted as a metric that calculates how much the top-k prediction of the model is actually in the held-out test set. Truncated Normalized Discounted Cumulative Gain (NDCG@K) is a metric that also considers the rankings of the top-k prediction of the model.

Recall@K:

$$Recall@K(u,\omega) := \frac{\sum_{k=1}^{K} \mathbb{I}[\omega(k) \in I_u]}{\min(K,\ |I_u|\}}$$

NDCG@K:

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

NDCG@K is the DCG@K normalized by dividing by the best possible DCG@K, where all held-out items are ranked at the top.

4.2 Datasets

The baseline models and proposed model were evaluated on four datasets. Three are popular benchmark datasets: MovieLens, Netfilx, Pinterest. The other is data from the largest music streaming platform in Korea: Melon.

- MovieLens: the public MovieLens 20M dataset. Ratings are binarized by keeping only ratings of four or higher, interpreting them as implicit feedback. We only keep users who have watched at least 5 movies.
- Netflix: benchmark dataset used at the Netflix Price. Similar to ML20M,

explicit feedback is binarized by keeping ratings of four or higher. We only keep users who have watched at least 5 movies.

- Pinterest: open source public dataset of the social network and image platform Pinterest. We only keep users who have at least 20 interactions (pins) as in [11].
- Melon: streaming history of 8 days from the largest music streaming platform in Korea. The play count is binarized like all the other datasets. We kept users who have listened to at least 100 songs and songs that have been played by at least 3000 users.

	Pinterest	ML-20M	Netflix	Melon
# of users	55,169	136,677	463,435	1,111,652
# of items	9,916	20,108	17,769	17,809
# of interactions	1.5M	10.0M	56.9M	194.4M
% of interactions	0.27%	0.36%	0.69%	0.98%
# of held-out users	5,000	10,000	40,000	100,000

Table 4-1. Summary of datasets after preprocessing

4.3 Configurations

Hyperparameters of the model were tuned through grid search of candidate values. Model selection was done by evaluating the NDCG@100 on the validation set.

In the case of VAE based models: *Multi-VAE*, *Multi-VAE (Gated)*, *Vamp*, *H+Vamp* and *H+Vamp (Gated)*, the models were trained and tuned following the exact same protocol. Note that the Multi-VAE is our strongest baseline and has also been compared rigorously with our proposed models. The Beta parameter, controlling the effectiveness of the prior, was selected between {0.1, 0.2, …, 1.0}.

The width of hidden layers {300, 600}, and size of the bottleneck Z_1 and Z_2 (only Z_1 in the case of original VAE) {100, 200} were chosen for the best performance in each separate model. In the case of models using VampPrior, the number of components K was set to 1000. Warm-up epochs were applied on the parameter beta and early stopping was done if the model's NDCG@100 did not improve on the validation set for over 50 epochs after warm-up. The ADAM optimizer was used for stochastic gradient descent with 200 batch size.

4.4 Results

In this section we report the experimental results comparing baseline models to our proposed model along with the intermediate models studying the effect of flexible priors, hierarchical stochastic units and gating mechanisms. In the case of MovieLens and Netflix dataset, the results of *WMF*, *SLIM* and *CDAE* are taken from [19]. Note that our experimental settings and data preprocessing are consistent with [19] for fair comparison. We also present results of additional analysis further studying the effect of using gates.

4.4.1 Model Performance

Here we present a summary of experimental results of the model performance for the four different real-world datasets: MovieLens 20M, Netflix, Pinterest and Melon. Our proposed model along with intermediate models are compared to state-of-the-art baselines for collaborative filtering. Performance is measured for truncated normalized discounted cumulative gain and recall on different K's and the result for the best performing model on each metric in marked in bold.

MovieLens 20M

Quantitative results on the MovieLens 20M dataset are presented in Table 4-2. The standard errors of the statistics are around 0.002. Multi-VAE was the strongest baseline as expected, it showed equivalent performance to the results reported in the original paper [19]. Vamp shows significant improvement compared to Multi-VAE indicating the benefit of changing the restrictive standard normal prior to a flexible VampPrior. Our final model H+Vamp (Gated) shows the best performance and significantly outperforms the strongest baseline Multi-VAE on all metrics. The final model shows up to 6.52% relative increase in NCDG@20 producing new state-of-the-art results.

MovieLens 20M					
Models	NDCG@100	NDCG@20	Recall@50	Recall@20	Recall@10
WMF [15] †	0.386	-	0.498	0.360	-
SLIM [23] †	0.401	-	0.495	0.370	-
CDAE [38] †	0.418	-	0.523	0.391	-
Mult-VAE [19]	0.42700	0.33804	0.53524	0.39569	0.33285
Vamp	0.43433	0.34892	0.53933	0.40310	0.34413
H+Vamp	0.43684	0.35284	0.53974	0.40524	0.34911
Mult-VAE (Gated)	0.43515	0.34741	0.54498	0.40558	0.34457
H+Vamp (Gated)	**0.44522**	**0.36008**	**0.55109**	**0.41308**	**0.35442**

Table 4-2. Results for MovieLens 20M dataset. Standard errors are around 0.002. †Results are taken from [19], note that our datasets, metrics and experimental settings are consistent with [19].

Netflix

Quantitative results for the Netflix dataset are presented in Table 4-3. Standard errors are around 0.001. Similar to the MovieLens dataset, Multi-VAE is the strongest baseline while Vamp, H+Vamp, H+Vamp(Gated) shows sequentially improving performance. Our final model shows the best performance and shows up to 9.58% relative increase in Recall@10 compared to the strongest baseline.

		Netflix			
Models	NDCG@100	NDCG@20	Recall@50	Recall@20	Recall@10
WMF [15] †	0.351	-	0.404	0.316	-
SLIM [23] †	0.379	-	0.428	0.347	-
CDAE [38] †	0.376	-	0.428	0.343	-
Mult-VAE [19]	0.38711	0.32256	0.44429	0.35248	0.32650
Vamp	0.39589	0.33843	0.44907	0.36327	0.34275
H+Vamp	0.40242	0.34630	0.45605	0.37090	0.35129
Mult-VAE (Gated)	0.39241	0.32927	0.44958	0.35953	0.33377
H+Vamp (Gated)	**0.40861**	**0.35251**	**0.46252**	**0.37678**	**0.35779**

Table 4-3. Results for the Netflix dataset. Standard errors are around 0.001. †Results are taken from [19], note that our datasets, metrics and experimental settings are consistent with [19].

Pinterest

Quantitative results on the Pinterest dataset are presented in Table 4-4. Standard errors are around 0.002. Since WMF, SLIM and CDAE have not been evaluated on the Pinterest dataset in [19], we only compare our models with the strongest baseline Multi-VAE. In case of the Pinterest dataset, our final model H+Vamp (Gated) does show increased performance but the results are not as significant as the other datasets. Especially, the gating mechanism does not show significant improvements if we compare Multi-VAE vs Multi-VAE (Gated) or H+Vamp vs H+Vamp (Gated). Our final model shows a maximum of 3.03% relative increase compared to the baseline. However, the difference for many of the metrics are within two standard errors and therefore it is uncertain to say that there is significant improvement.

Pinterest					
Models	NDCG@100	NDCG@20	Recall@50	Recall@20	Recall@10
Mult-VAE [19]	0.18888	0.11179	0.28485	0.15956	0.10043
Vamp	0.18983	0.11328	0.28648	0.16352	0.09942
H+Vamp	0.19026	0.11284	0.28937	0.16287	0.10082
Mult-VAE (Gated)	0.18810	0.11116	0.28683	0.16064	0.09988
H+Vamp (Gated)	**0.19189**	**0.11416**	**0.28995**	**0.16440**	**0.10134**

Table 4-4. Results for the Pinterest dataset. Standard errors are around 0.002.

Melon

Quantitative results for the Melon streaming dataset are shown in Table 4-5. The Standard errors are around 0.001. Like the Pinterest dataset, we only compare between the Variational Autoencoder based models. The results show significant improvements similar to the MovieLens and Netflix dataset. Vamp, H+Vamp , H+Vamp (Gated) shows sequentially increasing performance beating the baseline Multi-VAE. Results show very significant increases in performance, with the final model showing up to 13.08% relative increase in NDCG@20 compared to the baseline.

Melon					
Models	NDCG@100	NDCG@20	Recall@50	Recall@20	Recall@10
Mult-VAE [19]	0.44325	0.40473	0.38324	0.37033	0.43536
Vamp	0.46478	0.43845	0.39845	0.39566	0.47213
H+Vamp	0.47483	0.44853	0.40731	0.40510	0.48364
Mult-VAE (Gated)	0.45378	0.41813	0.39135	0.38115	0.44966
H+Vamp (Gated)	**0.48486**	**0.45770**	**0.41690**	**0.41389**	**0.49224**

Table 4-5. Results for the Melon dataset. Standard errors are around 0.001.

4.4.2 Further Analysis on the Effect of Gating

We also conducted experiments to further study the effect of using gates. We present the results in ndcg@100 for the Netflix dataset in Table 4-6. In this experiment the number of hidden units in each layer is fixed to 600[1]. A two layer model means that there are two hidden layers in each of the encoder and decoder.

We can see in Table 4-6 that for models with no gates, increasing the depth does not bring performance gain while for gated models it does. This can be interpreted that gating does help the network to propagate information through deeper models. However, we can also see large performance gains in simply adding the gates without additional layers. This tells us that the higher-level interactions the self-attentive gates allow are also very helpful themselves for modeling user preferences. One may point out that the gated model has more parameters, but note that ungated models cannot achieve similar performance by merely adding more units.

Netflix (NDCG@100)	No-Gate	Gated
Mult-VAE (1 Layer)	**0.38711**	0.39229
Mult-VAE (2 Layer)	0.38359	**0.39241**
Vamp (1 Layer)	**0.39589**	0.40169
Vamp (2 Layer)	0.39346	**0.40277**
H + Vamp (1 Layer)	**0.40242**	0.40728
H + Vamp (2 Layer)	0.37970	**0.40861**

Table 4-6. Comparison of performance between Gated and Un-Gated for models of different depth[2]. The model with better performance (1 Layer vs 2 Layers) is marked in bold.

[1] All other hyperparameters except the number of layers were fixed as well.

[2] There was no additional performance gain for adding more hidden layers than two.

5 CONCLUSION

In this work, we extend the VAE for collaborative filtering to adopt flexible priors and gating mechanisms. We show empirically that standard gaussian priors may limit the model capacity and introducing a more flexible prior can learn better representations of the user preference. For three datasets: MovieLens 20M, Netflix and Melon, qualitative results show that accompanying flexible priors, hierarchical stochastic units and gating mechanisms bring sequentially improving performance. Our proposed methods show significant performance gains on large real-world collaborative filtering datasets.

Our final model incorporating Hierarchical VampPrior VAEs with GLUs produces new state-of-the-art results in the collaborative filtering literature. The *H+Vamp (Gated)* model beats the original state-of-the-art baseline on all datasets with up to 13.08% relative increase on the Melon dataset. While the model showed the least amount of performance gain on the Pinterst dataset, the model is still at least on par and marginally better than the original VAE.

We also show that gating mechanisms are suitable for the sparse user-item interaction data. Gates provide valuable modeling capacity as well as helping information propagate through deeper networks. The Gated linear units allow for higher level interactions; for example, it can extract different values of features for the same item depending on which other items the user has consumed it with. This may be an important feature in learning from certain user-item preference datasets as there may be many different intentions to a consumption of the same item.

Overall, this work is the first to address the restrictive prior problem in the VAE-CF framework as well as introducing the potential of gating mechanisms in non-recurrent recommender systems. The results encourage the need for exploring more efficient architectures for variational autoencoders and neural networks in recommender systems.

Bibliography

[1] Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep Variational Information Bottleneck, 1–19. Retrieved from http://arxiv.org/abs/1612.00410

[2] Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., & Murphy, K. (2017). Fixing a Broken ELBO. https://doi.org/10.1080/02568543.2016.1244582

[3] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., & Lerchner, A. (2018). Understanding disentangling in β-VAE, (Nips). Retrieved from http://arxiv.org/abs/1804.03599

[4] Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2016). Language Modeling with Gated Convolutional Networks. https://doi.org/10.1016/j.physa.2015.03.066

[5] Dieng, A. B., Kim, Y., Rush, A. M., & Blei, D. M. (2018). Avoiding Latent Variable Collapse With Generative Skip Models, *89*(2). Retrieved from http://arxiv.org/abs/1807.04863

[6] Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders, (2016), 1–12. Retrieved from http://arxiv.org/abs/1611.02648

[7] Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2414–2423. https://doi.org/10.1109/cvpr.2016.265

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[9] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Networks, 1–9. https://doi.org/10.1017/CBO9781139058452

[10] Goyal, P., Hu, Z., Liang, X., Wang, C., & Xing, E. P. (2017). Nonparametric Variational Auto-Encoders for Hierarchical Representation Learning. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 5104–5112. https://doi.org/10.1109/ICCV.2017.545

[11] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. https://doi.org/10.1145/3038912.3052569

[12] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based Recommendations with Recurrent Neural Networks, 1–10. https://doi.org/10.1103/PhysRevLett.116.151105

[13] Hoffman, M. D., & Johnson, M. J. (2016). ELBO surgery: yet another way to carve up the variational evidence lower bound. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 111, pp. 1177–1183). Retrieved from http://approximateinference.org/accepted/HoffmanJohnson2016.pdf

[14] Hsu, W.-N., Zhang, Y., & Glass, J. (2017). Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data, (Nips). Retrieved from http://arxiv.org/abs/1709.07902

[15] Hu, Y., Park, F., Koren, Y., Volinsky, C., & Park, F. (n.d.). Collaborative Filtering for Implicit Feedback Datasets.

[16] Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes, (Ml), 1–14. Retrieved from http://arxiv.org/abs/1312.6114

[17] Koenigstein, N. (2017). Rethinking Collaborative Filtering. *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, 336–337. https://doi.org/10.1145/3109859.3109919

[18] Li, X., & She, J. (2017). Collaborative Variational Autoencoder for Recommender Systems, 305–314. https://doi.org/10.1145/3097983.3098077

[19] Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational Autoencoders for Collaborative Filtering, 689–698. https://doi.org/10.1145/3178876.3186150

[20] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial Autoencoders. https://doi.org/10.1016/j.msec.2012.07.027

[21] Marlin, B. (2004). *Collaborative filtering: A machine learning perspective* (pp. 2239-2239). Toronto: University of Toronto.

[22] Nalisnick, E., & Smyth, P. (2016). Stick-Breaking Variational Autoencoders, 1–12. Retrieved from http://arxiv.org/abs/1605.06197

[23] Ning, X., & Karypis, G. (2011). SLIM: Sparse LInear Methods for top-N recommender systems. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 497–506. https://doi.org/10.1109/ICDM.2011.134

[24] Quadrana, M., Milano, P., Karatzoglou, A., Cremonesi, P., Milano, P., & Gravity, R. (2017). Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks.

[25] Rakesh, V., Wang, S., & Shu, K. (2019). Linked Variational AutoEncoders for Inferring Substitutable and Supplementary Items, 438–446. https://doi.org/10.1145/3289600.3290963

[26] Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian Personalized Ranking from Implicit Feedback, 452–461. https://doi.org/10.1145/1772690.1772773

[27] Sachdeva, N., Manco, G., Ritacco, E., & Pudi, V. (2018). Sequential Variational Autoencoders for Collaborative Filtering, 600–608. https://doi.org/10.1145/3178876.3186150

[28] Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). AutoRec : Autoencoders Meet Collaborative Filtering. *WWW 2015 Companion: Proceedings of the 24th International Conference on World Wide Web*, 111–112. https://doi.org/10.1145/2740908.2742726

[29] Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016). Ladder Variational Autoencoders, (Nips). https://doi.org/10.1353/cjl.2008.0013

[30] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000

[31] Tomczak, J. M., & Welling, M. (2017). VAE with a VampPrior. https://doi.org/10.1128/AAC.47.9.2831

[32] Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems* (pp. 2643-2651).

[33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention Is All You Need, (Nips). https://doi.org/10.1017/S0140525X16001837

[34] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, *23*(7), 1661-1674.

[35] Wang, H., Wang, N., & Yeung, D.-Y. (2014). Collaborative Deep Learning for Recommender Systems, 1235–1244. https://doi.org/10.1145/2783258.2783273

[36] Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., … Zhang, D. (2017). IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. https://doi.org/10.1145/3077136.3080786

[37] Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., & Jing, H. (2017). Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17* (pp. 495–503). https://doi.org/10.1145/3018661.3018689

[38] Wu, Y., DuBois, C., Zheng, A. X., & Ester, M. (2016). Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM '16*, 153–162. https://doi.org/10.1145/2835776.2835837

[39] Zhang, S., Yao, L., & Sun, A. (2017). Deep Learning based Recommender System : A Survey and New Perspectives, *1*(1), 1–35.

[40] Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., & Li, Z. (2018). DRN: A Deep Reinforcement Learning Framework for News Recommendation. *Www*, *2*, 167–176. https://doi.org/10.1145/3178876.3185994

[41] Zheng, Y., Tang, B., Ding, W., & Zhou, H. (2016). A Neural Autoregressive Approach to Collaborative Filtering, *48*. https://doi.org/10.1109/TPAMI.2015.2476802.

논 문 초 록

최근 뉴럴넷 기반 협업필터링 추천알고리즘이 주목을 받고 있다. 그 중 한 갈래의 연구는 깊은 생성모형 (Deep Generative Model)을 이용해 사용자들의 선호를 모델링하는 방법이다. 이중 Variational Autoencoder 를 (VAE) 이용한 방법이 최근 state-of-the-art (SOTA) 성능을 보여주었다. 그러나 VAE 를 이용한 협업필터링 알고리즘은 현재 몇 가지의 잠재적인 문제점들을 지니고 있다. 첫 번째는 사용자 선호를 압축하는 잠재변수를 학습하는 과정에서 매우 단순한 사전분포를 사용한다는 것이다. 또 다른 문제점은 모델이 현재 여러 단을 이용한 깊은 인코더와 디코더를 사용하지 못하고 있다는 것이다. 본 연구는 최신기술들을 활용하여 앞선 문제점들을 해결하고 VAE 를 이용한 협업필터링 알고리즘의 추천성능을 더욱 높이는 것이 목표이다. 본 연구는 협업필터링 문제에 더 복잡한 사전분포 (Flexible Prior)를 적용한 첫 연구로서, 기존의 단순한 사전분포가 모델의 표현력을 제한할 수 있으며 더 복잡한 사전분포를 정의함으로써 모델의 성능을 더욱 높일 수 있음을 보였다. 이를 위해 이미지 생성 문제에서 좋은 결과를 보인 Vamp-Prior 를 이용해 실험을 진행하였다. 또한 VampPrior 를 Gating Mechanisim 과 함께 사용하였을 때 기존 SOTA 를 넘어서는 성능을 보임을 추천알고리즘에서 사용되는 대표적인 데이터셋들을 통해 보여준다.

주요어: 추천알고리즘, 딥러닝, 오토인코더, 생성모델, 베이지안 기계학습

학 번: 2017-29907