JoVA-Hinge: Joint Variational Autoencoders for Personalized Recommendation with Implicit Feedback

by

Bahare Askari Firoozjayi

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada

December 2020

Thesis Examination Information

Submitted by: **Bahare Askari Firoozjayi**

Master of Science in **Computer Science**

Thesis title: JoVA-Hinge: Joint Variational Autoencoders for Personalized Recommendation with Implicit Feedback

An oral defense of this thesis took place on December, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Robert Bailey
Research Supervisor	Dr. Jarek Szlichta
Research Co-supervisor	Dr. Amirali Salehi-Abari
Examining Committee Member	Dr. Ken Pu
Thesis Examiner	Dr. Faisal Qureshi

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Recently, Variational Autoencoders (VAEs) have shown remarkable performance in collaborative filtering (CF) with implicit feedback. These existing recommendation models learn user representations to reconstruct or predict user preferences. However, existing VAE-based recommendation models learn user and item representations separately. This thesis introduces joint variational autoencoders (JoVA). JoVA, as an ensemble of two VAEs, simultaneously and jointly learns both user-user and item-item correlations and collectively reconstructs and predicts user preferences. Moreover, a variant of JoVA, referred to as JoVA-Hinge, is introduced to improve recommendation quality. JoVA-Hinge incorporates pairwise ranking loss to VAE's losses. Extensive experiments on multiple real-world datasets show that our model can outperform state-of-the-art under a variety of commonly-used metrics. Our empirical experiments also confirm that JoVA-Hinge offers better results than existing methods for cold-start users with limited training data.

Keywords: recommender systems; deep learning; Variational Autoencoder; hinge-based loss function;

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the Ontario Tech University to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Tech University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Bahare Askari Firoozjayi

Statement of Contributions

The development of the framework was entirely done by myself. The work described is currently under submission and review at the conference. The conference remains unknown until the paper gets published. As the first author of the paper co-authored by my supervisors, I performed all the experiments, writing the code and the majority of the model's design.

Acknowledgements

First of all, I would like to express my sincere gratitude to my parents, my sister, Setare, and my boyfriend, Shervin, for their unconditional love and support during my study. I would like to thank my supervisors, Dr. Szlichta and Dr. Salehi-Abari, for their knowledge, invaluable support, guidance, and encouragement. It was truly an honor to work with you. And it has been a pleasure to do this research under your supervision. Finally, my special gratitude goes out to the Ontario Tech University faculty and my friends.

Contents

List of Tables

List of Figures

List of Symbols

n,m ..... (Lowercase) Variables

U ..... Set of users

I ..... Set of items

$\mathbf{R}$ ..... Implicit feedback matrix

$\mathbf{R}_u$ ..... The user u's interaction vector

$\mathbf{R}_i^T$ ..... The item i's interaction vector

I_u^+ ..... set of items that user u has interacted with

I_u^- ..... set of items that user u has not yet interacted with.

f ..... Scoring function

r_{ui} ..... Interaction score

$\hat{r}_{ui}$ ..... Predicted interaction score

$\mathbf{R}_u$ ..... A row of matrix R

$\mathbf{R}_i$ ..... A column of matrix R

$\hat{\mathbf{R}}^{user}$ ..... implicit matrices predicted (or completed) by user VAE

$\hat{\mathbf{R}}^{item}$ ..... implicit matrices predicted (or completed) by item VAE

ω_u ..... ranked list predicted for user u with the ground-truth items

I_u^* ..... u's true preferred items in held-out data

List of Abbreviations

CF ..... Collaborative Filtering

DNNs ..... Deep Neural Networks

AE ..... Autoencoder

CDAE ..... Collaborative Denoising Autoencoder

VAE ..... Variational Autoencoder

CR ..... Collaborative Ranking

TF-IDF ..... Term Frequency-Inverse Document Frequency

TF ..... Term Frequency

IDF ..... Inverse Document Frequency

MLP ..... Multilayer perceptron

CNNs ..... Convolutional Neural Networks

RNN ..... Recurrent Neural Network

LSTM ..... Long Short Term Memory

GRU ..... Gated Recurrent Unit

MF ..... Matrix Factorization

BPR ..... Bayesian Personalized Ranking

KL ..... Kullback-Leibler

JoVA ..... Joint Variational Autoencoder

RMS ..... Root Mean Square

ML1M MovieLens-1M

NDCG ..... Normalized Discounted Cumulative Gain

P@K ..... Precision@K

F1@K ..... F1-score@K

OCCF ..... One-Class Collaborative Filtering

GBPR ..... Group Bayesian Personalized Ranking

ALS ..... Alternating Least Squares

NCF ..... Neural Collaborative Filtering

GMF ..... Generalize Matrix Factorization

VBPR ..... Visual Bayesian Personalized Ranking

CDR ..... Collaborative Deep Ranking

NCR ..... Neural Network based Collaborative Ranking

DSSM ..... Deep Structured Semantic Models

CDAE ..... Collaborative Denoising Autoencoder

JCA ..... Joint Collaborative Autoencoder

FAWMF ..... Fast Adaptively Weighted Matrix Factorization

SGD ..... Stochastic Gradient Descent

AUC ..... Area Under the ROC Curves

MMMF ..... Maximum Margin Matrix Factorization

NeuRec ..... Neural network based Recommendation

POSNs ..... Preference-oriented social networks

CCA ..... Canonical Correlation Analysis

CoNet ..... Collaborative cross networks

ADC ..... Adaptive deep learning strategy

SPF ..... Social Poisson factorization

PF ..... Poisson factorization

Sorec ..... Social Recommendation

Chapter 1

Introduction

1.1 Motivation

Over the past few decades, the overload and diversity of information are among the problems we face. These problems result in several challenges, such as finding the most relevant information. Recommender systems are promising solutions to these problems. Recommender systems can solve the problem of information overload and are extensively used by many online services. They filter irrelevant items and present only the most desirable information and items to users based on their preferences. Recommender systems have been successfully used for movies (Netflix), books (Amazon), music (Spotify).

Recommender systems are usually categorized into *collaborative filtering*, *content-based recommender system* and *hybrid recommender system* based on how their recommendations are being made [1] [2]. Collaborative filtering (CF) is known as the most popular recommendation technique. The key idea behind CF [3] is that users with similar revealed preferences might also rate items similarly in the future. CF

can use either *explicit* (i.e., ratings and reviews) or *implicit* feedback.

Explicit feedback is more informative than its implicit alternative; however, it imposes a more cognitive burden on users through their elicitation. Explicit feedback is also subject to noisy self-reporting [4], and suffers from interpersonal comparison or *calibration* issues [5][6]. In contrast, implicit feedback naturally originates from user behavior based on the assumption that a user's interaction with an item is a signal of his/her interest in the item. Compared to explicit feedback, implicit feedback is more easily collected and abundant as long as user-item interactions are observable. In the research on implicit feedback, only positive implicit feedback, such as purchase history, browsing history, search patterns can be observed [7]. And it indirectly reflects users' preference [8].

In most practical recommendation scenarios, user feedback is implicit, not explicit. This abundance of implicit feedback has made collaborative filtering more intriguing at the cost of some practical challenges. The implicit feedback lacks negative examples as the absence of a user-item interaction does not necessarily indicate user disinterest (e.g., the user is unaware of the item). Also, the user-item interaction data for implicit feedback is large, yet severely *sparse*. It is even sparser than explicit feedback data as the unobserved user-item interactions are a mixture of both missing values and real negative feedback.

Since Deep Neural Networks (DNNs) are very powerful in learning representations, they have been widely explored and applied for recommendation task[9][10]. DNNs can learn non-linear user-item interactions. Recent work [11][12] has used deep learning models for recommendation with implicit feedback. Multilayer perceptron (or feedforward) networks were (arguably) the first class of neural networks successfully

applied for collaborative filtering [10][8]. Neural collaborative ranking [13] is a general collaborative ranking framework based on neural networks. To model a user's pairwise preference between items and estimate ranking, they combined pairwise ranking classification strategy with a neural network.

There has been emerging interest in deploying the variants of Autoencoder (AE) based models. They have shown promising performance in the recommendation task. Collaborative Denoising Auto-Encoder (CDAE) [14] presents a framework for a recommendation with implicit feedback. Users and items representations are learned by a Denoising Auto-Encoder. Joint Collaborative Autoencoder (JCA) [15] presents a joint learning model with AE that captures the correlation between users and items. The loss function of JCA incorporates pairwise ranking loss. But, these models are deterministic. Consequently, they are not able to obtain the uncertainty of the latent representations. On the other hand, Variational autoencoder (VAE) has the power to capture uncertainty and perform efficient inference. Recent work shows that VAE-based models outperform previous neural network-based methods for recommendation task. Mult-VAE [16] utilizes the VAE with a multinomial log-likelihood loss. The CVAE [17] is another VAE-based model that considers both content and rating information.

Despite the effectiveness of VAE-based models, they consider users and items separately. To improve user and item interaction modeling, we use two VAEs to simultaneously model users and items. One VAE captures user representation, and the other one focuses on item representation simultaneously.

1.2 Contribution of This Work

We present *joint variational autoencoder (JoVA)*, an ensemble of two variational autoencoders (VAEs). The two VAEs jointly learn both user and item representations while modeling their uncertainty, and then collectively reconstruct and predict user preferences. This design allows JoVA to capture user-user and item-item correlations simultaneously. We also introduce *JoVA-Hinge*, a variant of JoVA, which extends the JoVA's objective function with a pairwise ranking loss further to specialize it for top-k [1] recommendation with implicit feedback.

The main contributions of this thesis are as follows:

- We propose a VAE-based collaborative filtering (CF) model for top-k recommendation with implicit feedback. Instead of learning user and item representations separately, we use two VAEs to simultaneously capture user-user and item-item correlations.

- We extend the VAE loss with hinge-based loss in combination with reconstruction loss of VAE.

- Through extensive experiments over multiple real-world datasets, we show the accuracy improvements of our proposed solutions over a variety of state-of-the-art methods, under different metrics. Our JoVA-Hinge significantly outperforms other methods in the sparse datasets. Our experiments also demonstrate that JoVA-Hinge can achieve the best performance across all users with varying numbers of training data. Our findings confirm that the ensemble of VAEs

[1] In top-k recommendation, for each user, k most preferred items are recommended from all his un-interacted items (see section 4.1)

equipped with pairwise loss improves recommendations with implicit feedback.

1.3 Thesis Outline

The remainder of this thesis is organized as follows.

In **chapter 2**, we present the background concepts on recommender systems and deep learning.

In **chapter 3**, we review a survey of related works on implicit feedback recommendation and deep learning models employed for recommender systems.

In **chapter 4**, we present proposed Joint Variational Autoencoders for Recommendation with Implicit Feedback (JoVA) and its variant (JoVA-Hinge).

In **chapter 5**, the experimental design, analysis of the experimental results, and the discussion of research questions are presented.

In **chapter 6**, we present a discussion of the conclusion and direction for future research.

Chapter 2

Background

Our goal is to provide personalized item recommendations with the presence of implicit feedback. In this section, we present the essential background. We review the personalized recommendation and its three categories: Collaborative Filtering, Content-based and Hybrid Methods. We also introduce deep neural networks and two primary neural networks that are related to our work.

2.1 Personalized Recommendation

Recommender systems aim to help users find relevant information such as products to buy, movies to watch, or restaurants to eat. We can categorize the recommendation task into two groups based on the form of outputs: *rating prediction* and *top-k recommendation*. The recommender system with rating prediction considers explicit feedback. The top-k recommendation provides users a ranked list of k items. This research thesis focuses on the top-k recommendation.

Recommender systems usually fall into three categories: collaborative filtering,

content-based recommender system, and hybrid recommender system based on how recommendations are made [1][18].

2.1.1 Collaborative Filtering Methods

Collaborative filtering (CF)—a well-recognized approach in recommender systems—is based on the idea that users with similar revealed preferences might have similar preferences in the future [19]. CF can use explicit (i.e., ratings and reviews) or implicit feedback (i.e., purchase history, browsing history, search patterns, and clicks).

CF methods can be grouped into *point-wise* and *pairwise* approaches based on the form of the loss function and training data. CF methods that consider the preference prediction problem in a point-wise way are such as the recommender systems that consider ratings. Although point-wise approaches can work well, they have some drawbacks. The same rating given by different users can represent different preferences. This drawback is known as calibration [5][6]. For example, a rating of 1 for some user A might be comparable to a rating of 2 for another user B. Also, the user's judgment may change during the rating process. Some users also may not be comfortable giving numerical ratings to items. But, Pairwise ranking based models of CF, which are referred to as *Collaborative Ranking (CR)* [5] can solve these problems. CR methods evaluate the rankings of items for each user, instead of predicting score or rating for each item. And also, in most cases, users do not change their pairwise comparisons after seeing new information. For example, a user who prefers item i over item j will still do so after he has seen other items. The ratings of other users are not used in content-based methods.

Collaborative filtering methods are also classified into two classes: *neighborhood-*

based and *model-based* methods [20]. Neighborhood-based recommender systems first compute the similarities between users. Then, they compute an estimation based solely on the preference of similar users to the target user. Contrary to the neighborhood-based model, the model-based approach builds user and item latent representations using machine learning algorithms. This thesis research offers the model-based method.

2.1.2 Content-based Methods

The *content-based* systems recommend items to the user, similar to items that he had interactions in the past [21]. The content-based recommender system calculates items' similarities. For example, Amazon suggests products similar to the products that users have in their basket. This similarity can be based on rating correlations among users or item attributes such as item description and location of the user, etc [22][23][24].

Content-based methods are useful for new items with few ratings. They take advantage of the item descriptions, such as movie descriptions (e.g., keywords, genre, directors, actress, and actors). They also use user profiles which can be built by user feedback about different items. The user profile is consists of a set of keywords with weights. These weights show the word strength. The *term frequency-inverse document frequency (TF-IDF)* [25] is most commonly used to assign weights to keywords. The TF-IDF value considers *Term Frequency (TF)* and *Inverse Document Frequency (IDF)*. TF defines the relevance of the term in the item description by considering the term's frequency on the document, and IDF expresses how rare and unique terms are in the collection of items. In the recommendation process, the most similar doc-

uments in the corpus to the user profile are recommended. For documents, similarity could be calculated by a *Euclidean distance* between TF-IDF vectors.

While content-based models can alleviate item cold-start problems (new items), they can not help user cold-start problems (new users). For content-based models to be effective, users should have rated enough number of items. Moreover, content-based models cannot recommend diverse items and mostly recommend expected or obvious items. They cannot recommend items with a specific set of keywords that users have not used in the past. The content-based model is generally combined with CF [26] to solve the mentioned weaknesses.

2.1.3 Hybrid Methods

Hybrid systems combine the CF and content-based methods. They can take advantage of strong points of both CF and content-based methods [27]. There are two subcategories for hybrid methods: *loosely coupled* and *tightly coupled* methods [28]. Loosely coupled methods perform separate collaborative and content-based systems and then combine the outputs into final recommendations. Tightly coupled methods consider auxiliary information (which is processed) as a feature for the collaborative methods. The rating information helps the learning of features. Also, the extracted features can improve the prediction [11]. This thesis contribution can be easily extended to hybrid methods.

2.2 Deep Neural Networks

Deep learning is a sub-field of machine learning. In recent years, deep learning has shown great success in many applications, such as computer vision, speech recognition, and natural language processing. Recently, researchers showed deep learning can improve the performance of the recommender system [18]. We briefly introduce two primary neural networks related to our works: *Multilayer perceptron* and *Autoencoder*.

2.2.1 Multilayer Perceptron

Multilayer perceptron (MLP), also known as feedforward neural network consists of three or more layers: an input layer, one or more hidden layers, and an output layer with non-linear transformations. Formally, we can define MLP with L layers as:

$$\begin{aligned} h_1(x) &= a_1(\mathbf{W}_1 x + \mathbf{b}_1) \\ h_2(x) &= a_2(\mathbf{W}_2 h_1 + \mathbf{b}_2) \\ &... \\ f_{MLP}(x) &= a_L(\mathbf{W}_L h_{L-1} + \mathbf{b}_L) \end{aligned} \tag{2.1}$$

Where $\mathbf{W}_L$ and $\mathbf{b}_L$ are weight matrix and biases associated with layer L, respectively. a_* is the non-linear activation function. Activation function is a non-linear function which can be layer specific. Most commonly used activation functions are *sigmoid* $\sigma(x) = \frac{1}{(1+e^{-x})}$, *hyperbolic tangent* $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and *rectifier unit* $relu(x) = max(0, x)$.

2.2.2 Autoencoder

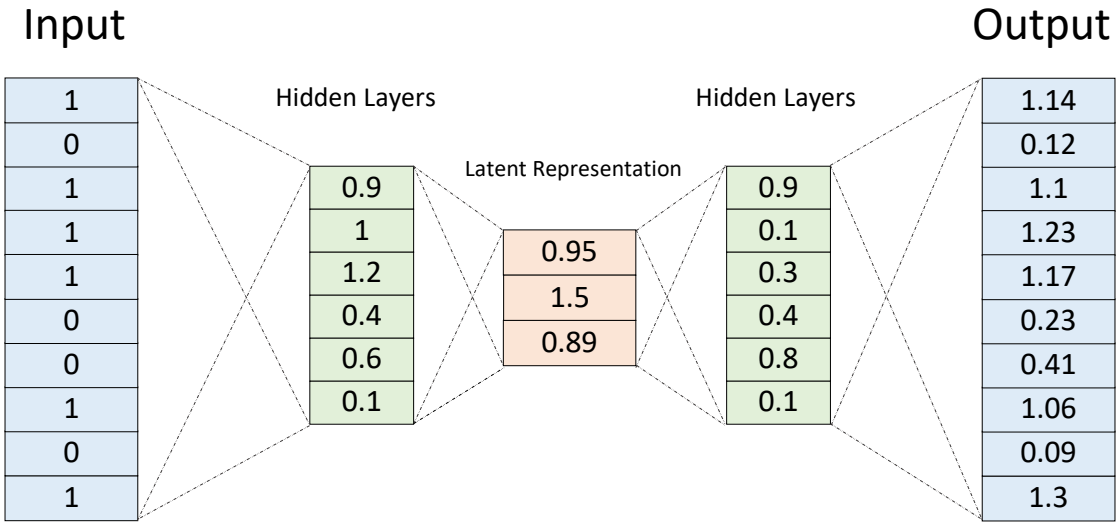


Figure 2.1: Structure of autoencoder. Autoencoder takes input and tries to reconstruct it. Green blocks represent hidden layers. The encoder takes input and converts it to latent representation, then the decoder takes latent representation and converts it to a reconstructed input.

An autoencoder, as shown in Figure 2.1 encodes the input into some representation, such that the input can be reconstructed from that representation. It includes three layers: the input layer, the hidden layer, and the output layer. The encoder compresses the input and generates the latent representation (process from the input layer to the hidden layer). Then the decoder tries to reconstruct the input from the latent representation (process from the hidden layer to output layer). The output layer is an equal size as the input layer. The parameters are learned by minimizing reconstruction error:

$$L(x, \hat{x}) = ||x - \hat{x}||^2, \tag{2.2}$$

where $\hat{x}$ is the model output. The number of neurons in the input layer and the

number of neurons in the output layer are equal. Mean squared error or binary cross-entropy can be used. To make the training phase of autoencoders faster, we can use gradient-based backpropagation methods. There are many variants of autoencoders such as *denoising autoencoder*, *variational autoencoder (VAE)* [29], *sparse autoencoder*, *marginalized denoising autoencoder* [30] and *contractive autoencoder* [31].

2.3 Matrix Factorization

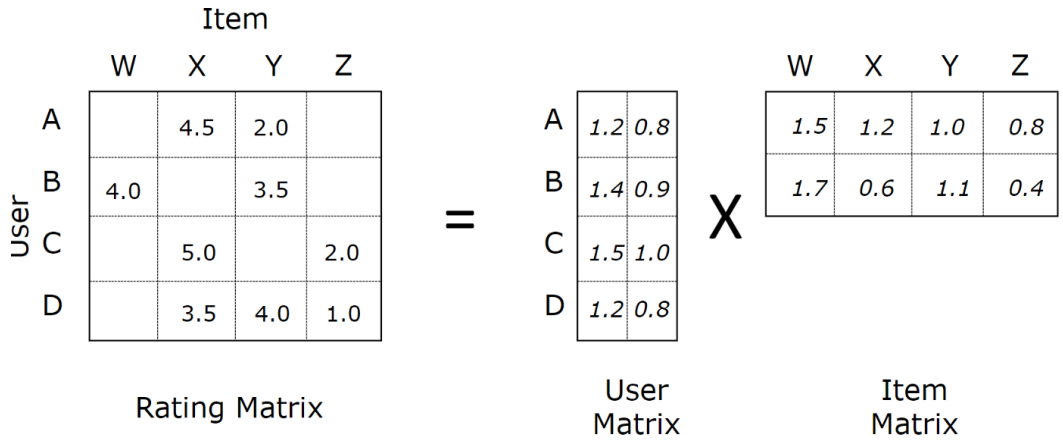


Figure 2.2: Example of Matrix Factorization (MF) taken from [32]. Rating matrix is decomposed to user matrix and item matrix.

Traditional recommendation systems such as collaborative filtering systems, use the basic matrix factorization (MF). MF has been known as the most popular and effective technique in the recommendation [33]–[35]. Users and items are mapped into a shared latent space, and vectors of latent features represent users and items. Figure 2.2 represents example of user and item matrices. Then the user's interaction with the item can be calculated by the inner product of their latent vectors. Vector p_u denotes the latent feature vector for user u, and vector q_i denotes the latent feature

vector for item i. So the interaction between user u and i can be modeled as:

$$\hat{y}_{ui} = p_u^T q_i. \tag{2.3}$$

Weighted regression function can be used to learn the parameters [7]:

$$J = \sum_{u=1}^{M} \sum_{i=1}^{N} w_{ui}(y_{ui} - \hat{y}_{ui})^2 + \lambda(\sum_{u=1}^{M} ||p_u||^2 + \sum_{i=1}^{N} ||q_i||^2), \tag{2.4}$$

where y_{ui} is rating provided by user u for item i, $\hat{y}_{ui}$ is the predicted rating and λ is hyperparameter to control regularization. In the recommendation task with implicit feedback, unobserved ratings are usually assigned to non-zero w_{ui} weight (smaller weight than the weight assigned to observed ratings) [36][7].

One limitation of MF is that the mapping between the representation space and the latent space is assumed to be linear, which is not always true. So, it may not capture the complex (non-linear relation) structure of interaction between users and items.

Chapter 3

Related Work

This chapter reviews the related work on collaborative filtering and deep learning approaches in the recommender system with implicit feedback. We also review some related fields on recommender systems to position this thesis in a broader literature: group recommendations, cross-domain recommendations, social recommender systems, and recommendations with noisy preferences.

3.1 Implicit Feedback Recommendation

In many real-world scenarios, implicit feedback (such as clicking and browsing history) are more common than explicit feedback. The key developments are designing new models for capturing user-item interactions or novel objective functions for model learning. Class of collaborative filtering with implicit feedback is also known as *One-Class Collaborative Filtering (OCCF)* [37].

Matrix factorization (MF) and its variants have been known as the most popular and effective technique in the recommendation [33]–[35]. In MF, users and items are

mapped into a shared latent space, and vectors of latent features represent users and items. Then a user's interaction with an item can be calculated by the inner product of their latent vectors. Weighted regression function can be used to learn the parameters [7]. In the recommendation task with implicit feedback, unobserved ratings are usually assigned to non-zero weight (smaller weight than the weight assigned to observed ratings) [7], [36]. One limitation of MF is that the mapping between user-item interactions and the latent space is assumed to be linear, which is not always true. So, it may not capture the complex structure of interaction between users and items.

Several methods have formulated the recommendation task as a ranking problem. *Bayesian personalized ranking (BPR)* [38] has been widely used for recommendation task. BPR assumes users prefer an interacted item to an un-interacted item and optimizes objective function based on pairwise ranking between items. The BPR optimization objective is based on the maximum posterior estimation, which the *Area Under the ROC Curves (AUC)* be maximized. BPR optimizes the posterior probability $p(\theta| >_u)$ where $>_u$ shows the latent representation for user u. The probability that user u prefers item i over j is calculated by a logistic sigmoid function. Most work focuses on effective negative sampling strategies to improve the performance of BPR. Since BPR was proposed, work has used BPR to optimize their models [36], [39]. For example, Group Bayesian Personalized Ranking (GBPR) [40] is an extension of the BPR framework. GBPR introduces a group preference over items (instead of considering users' preferences for individual items). GBPR can decrease sampling uncertainty by aggregating similar users' features.

CofiRank [41] is designed to directly optimize ranking metrics (NDCG) by fitting a maximum margin matrix factorization model [42]. CofiRank employs an exponen-

tially decaying weight function for ranks, in which more weight is set to the top ranks and less to the bottom. Maximum Margin Matrix Factorization (MMMF) is used to optimize NDCG. *EigenRank* [43] also directly models the recommendation task as a ranking problem and optimizes a preference function by Kendall rank correlation. It extends the neighborhood-based collaborative filtering framework.

Noia et al. [44] proposed an algorithm called *SPrank*, which is a unified hybrid graph-based data model that considers collaborative ranking as a learning-to-rank problem and combines ontological knowledge from the Web of Data with user preferences. It first builds a graph by using user-item interactions and the items' background knowledge. Then, based on the number of paths between a user and an item, features are extracted. Finally, they feed extracted features into learning-to-rank algorithms.

RankALS [45] minimizes a ranking objective function without sampling by adopting a square loss. The objective function directly incorporates ranking optimization. For optimizing, *Alternating Least Squares* (ALS) is used, which is a popular method to optimize the implicit MF model. *CoFiSet* [46] (Collaborative Filtering via Learning Pairwise Preferences over Item-Sets) is a pairwise recommender system which uses pairwise preference on a set of items instead of utilizing a single item. They introduced four variants of their model, CoFiSet(SS), CoFiSet(MOO), CoFiSet(MOS) and CoFiSet(MSO).

These classical methods, despite their success in the recommendation, suffer from some limitations:

- They fail to capture non-linear relationships between users and items.
- They cannot learn diverse users' preferences as they treat each dimension of the latent feature space in the same way.

- They have poor performance on sparse datasets.

3.2 Deep Learning for Recommender Systems

Recently, researchers showed the effectiveness of applying deep learning to the recommendation task [18]. By bringing non-linearity, they capture more enriched representations for users and items. Multilayer perceptron (MLP) can add non-linearity to existing recommender system methods. *Neural collaborative filtering (NCF)* [8] learns interactions between users and items. It uses an MLP to learn the user-item interaction function. NCF is able to express and generalize matrix factorization (GMF). Users and items are represented via a one-hot encoding. Above the input layer is the embedding layer. The obtained user (item) embedding can be considered as the latent vector for the user (item) in the latent factor model. They used binary cross-entropy as a loss function.

Wide & Deep model [10] was introduced for an app recommendation for Google play. It consists of two components: A *wide component* and a *deep component*. The wide component is a generalized linear model that is responsible for cross-product features. At the same time, deep component extracts non-linear relations among features. Item features are learned through a feed-forward neural network with embeddings. The model takes cross features as inputs to a linear model and jointly trains the linear model with a deep neural network model. Combining wide and deep components enable the recommender to capture both memorization and generalization.

Another example of the neural network-based recommendation model with implicit feedback is a *Visual Bayesian Personalized Ranking (VBPR)* [47] framework.

VBPR extends the Bayesian Personalized Ranking (BPR), which includes visual features.

Neural network-based recommendation model (NeuRec) [48] is a non-linear model that understands the relationship between items and users. They stored implicit feedback of users in the interaction matrix. The user-based NeuRec, first, maps each row of the interaction matrix to dense high-level representations with feed-forward neural networks. Then these representations are fed to a multi-layers neural network to learn user-item interactions. Similarly, the item-based NeuRec maps each column of the interaction matrix to a dense representation with a multi-layers neural network. With experiments, they showed that NeuRec performs better with sigmoid as activation function than tanh, relu and identity.

Neural Network-based Collaborative Ranking (NCR) [49] is a general collaborative ranking framework. NCR assumes that the user prefers an observed item to an unobserved item. It consists of three layers, the embedding layer, the hidden layers, and the output prediction layer. And it models a (user, item, item) triplet interaction. They adopt the binary cross-entropy loss for their model because prior work [50][8] showed that binary cross-entropy loss performs well for neural network-based ranking models. Then they proposed an algorithm to find the top-k ranked items.

Elkahky et al. [51] utilized deep learning for content-based multi-domain recommendation. They used neural networks to model the cross-domain behaviors of the user. This model learns user features and item features from different domains jointly and maps them to shared space. They used *Deep Structured Semantic Models* (DSSM) [52], where the first neural network includes a user's query, and the second neural network includes implicit feedback. DSSM is a deep neural network for learning

semantic representations of entities in a common continuous semantic space. Documents and queries are converted to vectors into a common low-dimensional space. Then vectors are fed into MLPs to extract semantic features. They used objective function based on a cosine similarity between the document's semantic features–query pairs extracted by the neural networks.

Of the most relevant to our work are recommender systems built based on autoencoders or their variations. Autoencoders have been a successful approach for deep learning models for recommender systems [28][53][54][55]. *AutoRec* [53] is an autoencoder framework that can be used for collaborative filtering. Each user and each item are represented as a partially observed vector. AutoRec learns an autoencoder that encodes these vectors into lower-dimensional latent space and then decodes them to make missing rating predictions. It is also able to learn a non-linear latent representation. They evaluated their model's performance as the number of hidden units varied and found that AutoRec with 500 hidden units showed the best result (minimum Root Mean Square Error). In their model, weight matrices are shared among the autoencoder.

Collaborative Denoising Auto-Encoder (CDAE) [14] presents a framework for a recommendation models. CDAE is mainly designed for ranking prediction. Users' and items' representations are learned by a Denoising Auto-Encoder. The input of CDAE (user partially observed preferences) is corrupted by Gaussian noise. They took advantage of negative sampling. CDAE parameters increase linearly with both the number of users and the number of items.

Collaborative Deep Ranking (CDR) [56] employs Stacked Denoising Autoencoders to represent the feature of item content (such as the title and abstract of the ar-

ticles and tags) into the pair-wise ranking model's Bayesian framework. CDR is a hybrid pair-wise approach with implicit feedback that jointly implements representation learning and collaborative ranking. They confirmed that using the ranking loss can significantly improve the recommendation performance.

JCA [15] is a joint model of user-based and item-based autoencoders that can capture user-user and item-item correlation. This model consists of two autoencoders, which are called the user component and item component. The model takes the whole rating matrix as a dual input. The user component takes the user rating vector (i.e., one row of rating matrix) as an input, and the item component takes the item rating vector (i.e., one column of the rating matrix) as an input. The user component and item component predict two completed rating matrices independently, and then the final output is computed by combining the two outputs (unweighted average of two predicted rating matrices). They also adopt normalization constant to alleviate the influence of item feedback heterogeneity. JCA is optimized only by a pair-wise hinge-based objective function. They also proposed a novel mini-batch optimization algorithm to train JCA without loading the entire rating matrix, which is more practical for especially massive datasets. New users and items can be trained through the mini-batch optimization. The number of parameters of the JCA increases linearly with both the number of users and the number of items, making it more prone to overfitting.

Another example of autoencoder model for recommendation is *Deep generative ranking (DGR)* Wasserstein autoencoder based model [57]. DGR is a recommender system that jointly models the generation of implicit feedback data and the creation of a pair-wise ranking list. In DGR, implicit feedback data is generated under the

Beta-Bernoulli distribution. DGR also generates the pair-wise ranking list by utilizing both interacted and non-interacted items for each user.

Mult-VAE [16] is a collaborative filtering model for implicit feedback based on variational autoencoders. Mult-VAE assumes the distribution of the latent variables can be estimated from the implicit feedback data. Mult-VAE uses a multinomial log-likelihood instead of the Gaussian likelihood. The generative model of Mult-VAE samples a latent representation for each user, and then the feedback history of each user is assumed to be drawn from the multinomial distribution. Then the latent representation is utilized to reconstruct the input rating matrix. This work also proposed a regularization hyperparameter to control the trade-off between the reconstruction loss and the Kullback-Leibler (KL) loss in the objective function. Mult-VAE showed better performance than CDAE. Recently, *RecVAE* [58] proposed a new approach to optimizing this hyperparameter which leads to better performance than Mult-VAE. In RecVAE, this hyperparameter for the Kullback-Leibler term is user-specific, which depends on the amount of implicit feedback available for each user.

We can model user-item interactions as a bipartite graph. Berg et al. [59] proposed GC-MC for the recommendation task by considering it as a link prediction task with a convolutional neural network graph. Graph autoencoder consists of a graph encoder and a pairwise decoder. The encoder takes a graph adjacency matrix and outputs a node embedding matrix. The decoder takes pairs of node embedding and predicts respective entries in the adjacency matrix. It computes the pairwise distance given network embedding. GC-MC can easily integrate side information into the recommendation model. GC-MC is designed for explicit datasets and uses different weight matrix to decode various types of edges (ratings). Ying et al. [60] proposed a

PinSage a graph CNNs based model for a recommendation on Pinterest. PinSage can handle a billion-scale graph. It combines random walks and graph convolutions to create embedding of nodes. Item embedding incorporates both graph structure and item feature information. For more efficient training, they present a mini-batch by uniformly sampling the neighboring nodes. Multi-modal Graph Convolution Network (MMGCN) [61] combines information from different sources, such as movie and video data. For each modal (visual, acoustic, and textual), MMGCN creates a user-item bipartite graph. After building the bipartite graph, it uses graph convolution networks to train each bipartite graph. Finally, it merges the node information of different modals. Neighbor Interaction Aware Graph Convolution Networks (NIA-GCN) [62] used a pairwise neighborhood aggregation layer to capture relationships between pairs of neighbors. The loss function includes the Euclidean distance between users and items with their neighbors. Low-pass Collaborative Filter (LCF) [63] removes the noise in observed data. LCF is applicable to the large graph as it reduces the time consumption of graph convolution.

3.3 Group Recommendation

Recently, group recommendation becomes helpful in plenty of scenarios. Group recommendations can be useful in different types of groups and items. For example, group recommender systems to find movies for a group of friends to watch or restaurants for a group to eat. Some recent work has addressed the problem of providing recommendations to a group of users. To make efficient group recommendation, Pujahari and Padmanabhan [64] combined both user-user filtering and item-item filtering for predicting items that are common for most of the users in the group. They gener-

ated recommendations by making homogeneous groups. First, the system generates a homogeneous group by finding the degree of similarity between members. Then group recommendations are provided for the group of users. Gorla et al. [65] assumed that the recommendation score for an item depends on its relevance to each group member and its relevance to the group as a whole. They proposed a probabilistic framework to make group recommendations.

Preference-oriented social networks (POSNs) [66] use a social network structure to make better group decisions when some group members' preferences are not observed yet. POSNs exploit ranking networks [67] to capture the preference rankings correlation between users in social networks. They considered user preferences in the form of ranking of a finite set of options (e.g., a set of products, a genre of movies, etc). The generative process for POSNs starts with drawing individual preferences from a ranking distribution; then, POSNs connect individuals with a probability. The probability increases with the similarity of individuals' preferences. Each user preference ranking is drawn independently with *Mallows ϕ-model*. Their empirical results showed the effectiveness of their inference and group recommendation methods. It also confirmed the importance of employing a social network structure to make a better decision for groups with missing preferences of some group members.

Baltrunas et al. [68] studied and compared different aggregation methods (such as Spearman Footrule, Borda Count, Least Misery, and Average) and found no clear winner. They showed that an aggregation method's effectiveness depends on the group size and group similarity between group members.

We can classify most existing group recommendation approaches into two categories: *aggregating individual profiles* and *aggregating individual recommendations*:

- Aggregating individual profiles: aggregating individual members' preferences represent the preferences of the group.

- Aggregating individual recommendations: individual members' recommendations are provided independently, and aggregating individual recommendations generate group recommendations [68].

One of the major issues in group recommendation is the difficulty of the evaluation process. Two evaluation approaches have been mainly used in the field of recommender systems: *live user experiments* and *offline experiments* [69]. In offline evaluation, datasets used for evaluating individual recommender systems are usually used to assess group recommendation systems. Or researchers collect new evaluation datasets from complete, practical systems. If the dataset contains no groups, groups can be generated randomly or based on the similarity between users. For example, users with user-to-user similarity higher than some threshold can form groups [68] where the Pearson correlation coefficient computes similarity.

Sparsity and cold start problems are also among major issues in group recommendations. To address these problems, we can use a cross-domain recommendation. Cross-domain recommender systems (see section 3.4) use the information of two or more different domains to recommend items on one of those domains. For example, suppose we know group members' music preferences, but their movie preferences are not observed. We can exploit their music preferences to recommend movies to them. Using a cross-domain recommendation for groups of users where their preferences are in the form of rankings of items has received little attention.

3.4 Cross-domain Recommendation

Cross-domain recommender systems use the information of two or more different domains to recommend items on one of those domains. The domain in which the recommendations are performed is called *target domain*, and *auxiliary* or *source domain* is the domain from which knowledge and information are transferred to help recommendations in the target domain. Cross-domain recommendations can mitigate the cold start problem by using information acquired from other domains. Collaborative filtering recommenders will not recommend a new item since it has not been rated by someone. Cross-domain systems can solve this problem by recommending items from multiple domains. Moreover, a cross-domain recommendation can reduce sparsity and increase novelty and diversity in the set of the items recommended since considering multiple domains instead of only one domain help get better coverage of the range of user preferences. Besides, it can improve user models. Cross-domain recommenders improve serendipity by using information from multiple domains in which serendipity means the items recommended to users are somewhat unexpected.

Different understandings have been made in the various research area for addressing the cross-domain recommendation problem. A domain is a particular field of thoughts, interest, or activity that can be defined at the four following levels [70].

- *Item level:* recommended items have a different type. In other words, all or at least most of their attributes are different. For example, movies and books belong to different domains, or music and movies.

- *Attribute level*: recommended items have the same type. In other words, two items are considered as belonging to distinct domains if the value of a certain

attribute is different. For example, two books in the same system belong to distinct domains if they have different genres, like comedy and horror books.

- *Type level:* types of recommended items are similar and have some attributes in common. If not all the two items' attributes are not the same, they are considered distinct domains. For example, movies and T.V shows belong to distinct domains.

- *System level:* in this, almost the same items, collected in different ways or from different operators. For example, music listened in the Last.fm, and music listened in the Spotify by users.

Most of the work considers domains at the item and system levels. The most common domains that are used in cross-domain recommender systems are movies, books, and music.

And four scenarios based on the combination of users and items overlapping in the cross-domain recommendation are [71]:

- No overlap: there is no overlap between users or items.

- User overlap: in this case, some shared users exist who have rated for items in both domains.

- Item overlap: there are shared items rated by users from both the domains.

- User and item overlap: there are both overlaps between users and between items.

Codebook transfer algorithm [72] is transfer learning to collaborative filtering problems. They considered domains without any shared users and with no overlap of items.

Their algorithm consists of three steps: First, they applied co-clustering algorithms on users and items simultaneously in the source rating matrix. Then they constructed a cluster-level rating matrix called codebook. The knowledge is transferred through the codebook. After codebook construction, they used the codebook to fill missing ratings in the target rating matrix, which reduces the target domain's sparsity. For reducing the sparsity, users, and items in the target domains are mapped to the co-clusters in the codebook by minimizing the certain loss function. Both two steps are based on tri-matrix factorization. Finally, they used the filled target matrix for predicting the rating. They showed that transferring useful information from a dense source rating matrix is more effective than recommending using only the knowledge in the sparse target rating matrix.

Sahebi [73] proposed a *Canonical Correlation Analysis (CCA)* a cross-domain collaborative filtering method. CCA finds two projection vectors that maximize the correlation coefficient between independent and dependent variable sets. They supposed that users overlapped between the source and target domains and considered the source domain in cross-domain recommender as the independent variable set and the target domain as the dependent variable set. CCA finds the components of each domain that are most similar to each other based on user rating behavior. It determines how much two components are correlated with each other. In order to know how the ratings of a combination of items in the source domain affect the ratings of an item in the target domain, projections vectors can be used. After adding the source domain's user ratings, we can understand how all of a user's ratings in the source domain affect the same user's ratings in the target domain. The ratings of users in the target domain can be estimated by using the projection vectors, the source domain

ratings, and the canonical correlation value.

Feng Yuan proposed a deep domain adaptation model for the cross-domain recommendation [74]. They assumed the target and source domains have the same set of users, but the items are different and considered rating matrices for each domain. Their goal was to predict missing ratings in the target domain by using information from the source rating matrix. They learned a set of embeddings to represent each user's preferences. Embedding representation is used to extract the shared user rating patterns from the two domains. In their model, an autoencoder predicts the missing values in the rating matrix. The input is the partially observed rating vectors for each user (item). They then mapped each vector into a low-dimensional latent space, followed by a reconstruction layer as output to recover the rating vectors. They used one deep feed-forward neural network to reconstruct the rating vectors for the source and the other network for the target domain separately. They showed that their model performs better rating prediction than several state-of-the-art cross-domain recommendation methods.

Collaborative cross networks (CoNet) [75] is a deep learning model with a feed-forward neural network model that models the interactions between users and multiple domains. CoNet's core components are cross-connection units that enable dual knowledge transfer. The target network uses information from the source network and vice versa. Source and target networks are coupled with cross-connections. They defined a joint loss function. They showed their better performance over the state-of-the-art recommendation algorithms. They compared CoNet with *MLP++*. MLP++ is a combination of two multilayer perceptron models by sharing only the user embedding matrix. MLP++ has no cross-connection units. They showed that their model per-

formed better than MLP++, which shows the importance of cross-connection units.

Another approach for transferring knowledge across domains is using deep learning methods by linking cross-domain user latent representation [76]. Variational Autoencoder is used for cross-domain linking. The model incorporates two Vaes for modeling each domain, which is linked at their latent layer. They also modify the optimization criterion to improve the knowledge transfer from the source domain. They compared their model with the current state-of-the-art methods. It performed better than all of the state-of-the-art techniques, including CoNet [75] with metric hit ratio (HR) and NDCG.

Adaptive deep learning strategy (ADC) [77] is an algorithm for the cross-domain recommendation that controls and adjusts the contribution of each domain while optimizing the model parameters. They considered different domains and jointly learned all the different single domain. They compared their method with various single and cross-domain methods, including CoNet [75]. And they showed that ADC could improve recommendation performance comparing to CoNet in the term of evaluation metric NDCG.

Two categories of cross-domain approaches, based on how knowledge from the source domain is used are *aggregating knowledge* and *transferring knowledge* [71]:

- Aggregating knowledge: knowledge of different source domains is aggregated to achieve recommendations in a target domain.
- Transferring knowledge: knowledge is transferred between domains in order to enhance recommendation.

Aggregating knowledge methods fall into 4 categories: *merging user preferences*, *mediating user modeling data*, *combining recommendations* and *linking domains* [71]:

- Merging user preferences: data sources such as ratings, tags, and click-through data from different domains are merged, and a traditional single-domain recommender system is used on the merged data. Therefore, it requires user overlap between the source and target domains. These approaches can enhance recommendation since the users' profiles become richer when multiple sources of personal preferences are combined [71].

- Mediating user modeling data: in this approach, models from different domains are aggregated. The mediation improves the user models of the target recommender. For instance, If there is an overlap of users between the source domain and target domain, a cross-domain recommender based on mediating can extract the list of neighborhoods in the source domain and use it for the target domain's recommendation. This type of method requires either user-overlap or item-overlap between domains [71].

- Combining recommendations: single recommendations are combined where each of the recommenders has a weight that is based on its importance. Combining recommendations can increase diversity, and it is easy to implement; however, it is challenging to tune the weights assigned to each recommendation. Weights can be computed based on different factors, such as the reliability of each recommender [71].

- Linking domains: linking domains by a common knowledge such as item attributes, user attributes, or association rules [71].

Transferring knowledge methods can be divided into two subcategories, *sharing latent features* and *transferring rating patterns*:

- Sharing latent features: source and target domains are related by means of shared latent features.

- Transferring rating patterns: in this type of method, rating patterns are transferred between the source domain and target domain. Rating patterns are correlations between the preferences of groups of users for groups of items.

3.5 Social Recommender Systems

Social networks can be used to improve recommendations. Items can be recommended to users based on the users' ratings that have social relations with the given user. The social recommendation is a task that occurs daily because we always ask friends for recommendations [78]. Thus, to enhance recommender systems and provide more personalized recommendation results and solve sparsity and cold start, we can use social network information among users. Social networks can provide extra information for making recommendations. We can use information from the users who have a connection with the given user. If the user has not rated enough items in systems, her neighbors' information in social networks can help solve sparsity and cold start problems.

Social Poisson factorization (SPF) [79] is a probabilistic model that incorporates social network information into a traditional factorization method. Latent user preferences are used for discovering patterns in user activity, and it also estimates how much each user is influenced by her friends' observed clicks. Then, SPF makes the recommendation. So, the users may like and click items for two reasons, the first reason is that the items attribute match user preferences, and the second reason is

the users' friend likes that item. Each user has both a vector of latent preferences and a vector of influence values in their model. For each of her friends, the user has one vector of influence values. In other words, in SPF, user action and a social network are the observed data. SPF is based on *Poisson factorization (PF)* [80] and models the number of times the user is engaged with an item and finds the latent influence between users in a social network. Accordingly, it matches user preferences with her social friends to produce the top-k recommendations. SPF model does not consider time. When two connected users both like an item, one of them may use it first. In other words, it can not consider the time of users' actions.

Social Recommendation (Sorec) [81] uses a user's social network graph with the user-item rating matrix so to create more accurate and personalized recommendations. Within the social network graph, nodes represent users, and edges represent relations between users. Each relation is associated with a weight that shows how much users trust each other. Social recommender combines social network structure and the rating matrix, based on probabilistic factor analysis. The shared user latent feature space connects the social network structure and the rating matrix. They learned the low-rank user latent feature space and item latent feature space by performing factor analysis. Factor analysis is based on probabilistic matrix factorization. Sorec assumes that the observed data is a linear combination of some latent factors. They used a logistic function that can be improved by employing a Gaussian Kernel or a Polynomial Kernel. To evaluate the effectiveness of their approach, they used the Epinions dataset. The experimental results showed that Sorec is better than other collaborative filtering recommendation system, particularly for cold users. Besides, it can be applied to very large datasets. Their approach can handle the missing value

problem. The disadvantage of this work is that they did not consider the information propagation between users. They did not consider that information from neighbors of the user's neighbor can also be useful. It also does not reflect the real world recommendation because it lacks interpretations.

PsRec [82] is trust-based matrix factorization model. This framework alleviates the sparsity problem and cold start problem. They filled some missing values by information collected from social networks with pseudo ratings. Pseudo ratings are based on an observation that the user likely agrees with her friends' average interest on an item if most of her friends have a similar interest in that item [83]. In order to solve the cold start problem, they produced more pseudo-ratings for cold-start users. They measured the distance between a user's rating and her friends' average rating. Then they used this predicted distance to fill pseudo ratings. They learned the user and item latent features with minimizing the mean square error over the observed ratings. They called the matrix containing both observed ratings and pseudo ratings (for filling missing value) as the merged rating matrix. They used weights that represent confidence in observed rating and pseudo rating for two reasons. First, pseudo-ratings have different importance. Second, treating observed rating and pseudo rating equally in matrix factorization reduces the prediction accuracy because pseudo ratings have a large amount of noise that will destroy the useful information that they can provide. After generating pseudo ratings with corresponding weights, they used an optimization method to find the user-feature matrix and the item-feature matrix. Fusing social information in pseudo ratings and the weight matrix make the recommendation more powerful than other methods that use a similar Probabilistic Matrix Factorization Model. They used Ciao and Epinions datasets to evaluate their framework. Ciao

is a product review website where users can rate and record reviews for different products. Besides, they can build social relations with other users on Cia. The trust relation is described as binary values. If it is one, it represents trust. Otherwise, trust statements are unobserved. PsRec performed the best of all for mean absolute error (MAE) and root mean square error (RMSE) on the two datasets For both all users and cold start users when they compared it with six other methods such as SoRec [81], SocialMF [84], TrustMF [85], and TrustSVD [86].

The authors of the paper [87] proposed a trust-aware recommender system that predicts the rating for items by integrating the trust matrix and the rating matrix. They found users' neighbors and calculated the weights different from traditional collaborative filtering recommender systems. Trust values and similarity measures between users are computed. The trust-aware system replaces the similarity finding with the utilization of a trust metric that can propagate trust over the trust network. Trust metrics algorithms predict, based on the trust network. There are many alternative trust metrics which can be classified to *global* and *local* [87]. Local trust metrics consider the users' very personal views and predict different trust values in other users for every individual user. The trust matrix, which represents all the community trust statements, and the rating matrix, are their model's inputs. They used *MoleTrust* [87] for local trust metric which is a depth-first graph walking algorithm. MoleTrust can control the distance to which trust is propagated. They used PageRank for the global trust metric. For the similarity metric module, they used the Pearson Correlation Coefficient. They tested their framework on the real-world dataset and showed that it worked very effectively for recommending to a new user.

The authors of the paper [88] presented a framework to analyze the interactions

between social influence and selection. They verified that people are similar to their neighbors in a social network. They proposed a mathematical model that both influences and homophily (homophily indicates that neighbors in a social network are similar.) play the role of predicting future behavior in the social network (They tested it over Wikipedia). They addressed that to what extend similarities and social interactions do the work of predictors of future behavior. But their method cannot tell which node has a stronger influence on the other in the social network. Besides, it can not determine the node with the strongest influences in the network. Although their model is more robust than similar frameworks, it requires more parameters. So more data is required to learn the parameters. Their work provides a richer framework for exploring the dynamics of behavior, opinion, and idea change and adoption in networks compared to previous work.

TRUSTMF [89] maps users into two low-dimensional spaces, which are called *truster* space and *trustee* space. They performed mapping by factorizing trust networks. They showed TRUSTMF better predictive accuracy than other trust-based models using four data sets, including Epinions, Douban, and Flixster. They also proposed another model which is called TrustPMF. TrustPMF is a more general and flexible framework and can provide a probabilistic view for understanding the Truster, Trustee, and TrustMF models. TrustPMF performed better than TRUSTMF.

3.6 Recommendation with Noisy Preferences

In recommender systems, it is assumed that the ratings in the datasets have no irregularities or inconsistently. However, users may be inconsistent while rating items [90]. Amatriain et al. [90] proposed an approach to eliminate noisy ratings called

item re-rating. Item re-rating can remove natural noise in user inputs in order to improve accuracy in recommender systems. In this study, they asked users to rate again previously rated items to denoise the dataset. The re-rating was through three trials. The minimum time difference between the first and second trials was 24 hours, and the minimum time difference between the second and third trials was 15 days. The items presented in the first and third trials were in the same random order, but in the second trial, the items were presented in order of popularity. As it is not possible to ask all users to re-rate all items once or twice, they proposed methods to choose which rating to denoise selectively. They selected ratings randomly based on values and based on how noisy the user is. They confirmed that denoising extreme ratings resulted in better performance than denoising mild ratings. One limitation of this work is user participation, which makes it hard to apply in real-world scenarios. It is not practical to ask users to re-rate in every case. Correcting noise instead of removing noise might improve recommendation accuracy.

Another method to handle natural noise in user ratings was presented in [91]. The authors of the paper mentioned two types of noise in the recommender system database. *Natural noise* and *malicious noise*. Natural noise happens when users become inconsistent when they elicit ratings for items. Malicious noise is biased noise being deliberately inserted into a system by attackers. In this paper, they detected and corrected natural noise using only ratings. Their method is based on that users have their own tendency to give ratings and items have their own tendency to receive ratings. So, they classified users, items, and ratings. Each rating and its corresponding user and item are classified as strong, average, or weak. If the user and item behavior are the same and contradict rating classification, then the rating can be

noise. They then find they compute prediction for each noise using a recommendation algorithm for its corresponding user and item. If the difference between the old and new value is higher than a given threshold, then the prediction replaces the original value; otherwise, they keep the old value. They did not consider the relation between users and the correlation between items to detect noise. They also did not consider and manage the inherent uncertainty associated with the ratings in the systems.

Salehi-Abari and Larson [92] proposed a probabilistic framework for modeling noisy subjective preferences. They studied how noise in revealed subjective preferences can impact different voting rules for group recommendations. In this work, they assumed that users revealed or observed preference ranking is a noisy observation of users' true preference. More specifically, they assumed users revealed preference ranking is drawn from a conditional distribution Mallows ϕ-model.

They compared the group preference aggregated from noisy preferences (using an instance of noise model class) with the group preference aggregated from true preferences (drawn independently from a ranking distribution or a real-world preference dataset) to evaluate each aggregation method. Four voting rules: Plurality, Borda, Copeland, and Kemeny [93] were examined. Their empirical results confirmed that each group decision method's robustness differs depending on the underlying noise model and preference distributions.

3.7 Summary

In this chapter we described the related work on collaborative filtering (section 3.1) and deep learning approaches in the recommender system (section 3.2), as well as the relevant fields on recommender systems such as group recommendations (section 3.3),

cross-domain recommendations (section 3.4), social recommender systems (section 3.5), and recommendations with noisy preferences (section 3.6).

Our work is closely related to both JCA and Mult-VAE, as we build on these two strengths. While JCA is jointly optimizing two classical autoencoders, it does not capture the uncertainty of latent representations. Consequently does not benefit from the representation power of variational autoencoders. We deploy two separate variational autoencoders and jointly optimized them by our proposed loss function to address this. Our loss function, by taking into account two variational autoencoders' losses and a pair-wise ranking loss, well tunes our deep learning models for recommendation with implicit feedback. While differentiating from both JCA and Mult-VAE regarding both architecture and loss function, our proposed work can be viewed as the powerful generalization or extension of these two. JCA and our proposed models learn user-user and item-item correlations separately and simultaneously with two separate networks. JCA used autoencoders, while our proposed models consist of variational autoencoders.

Chapter 4

Approach

This chapter presents the main contributions of this thesis. Our goal is to provide personalized item recommendations with the presence of implicit feedback. In this section, we formally define our problem and detail the proposed joint variational autoencoder (JoVA) framework and its variant (JoVA-Hinge) for top-k recommendation with implicit feedback.

4.1 Problem Statement

We assume that a set of n users U can interact with the set of m items I (e.g., users click ads, purchase products, watch movies, or listen to musics). We consider user-item interactions are binary (e.g., a user has watched a specific movie or not), and represent them with the user *implicit feedback matrix* $\mathbf{R} \in \{0,1\}^{m \times n}$ as:

$$R_{ui} = \begin{cases} 1, & \text{if (user u, item i) interaction is observed} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

Where a value of 1 for R_{ui} indicates that there is an interaction between user u and item i and value of 0 for R_{uj} indicates that no interaction between user u and item j is observed. In the implicit feedback study, each entry is one if the corresponding user has interacted with the corresponding item, and 0 otherwise. In the implicit feedback study, the task is to predict the missing entities, i.e., 0 in the implicit feedback matrix [8], [15], [16]. In the implicit feedback study, the unobserved interaction data is a mixture of real negative feedback and missing values (unknown). We can only observe positive feedback signals. It means that we only know which items users like. As we do not know which items users dislike, we can not distinguish between users' real negative feedback and missing values. For this reason, we consider observed interactions as one and all unobserved interactions as 0.

As each column (or row) of the matrix corresponds to a specific item (or user), we let $\mathbf{R}_u$ and $\mathbf{R}_i^T$ denote the user u's and item i's interaction vectors, respectively. We also let $I_u^+ = \{i \in I | \mathbf{R}_{ui} = 1\}$ denote a set of items that user u has interacted with, and $I_u^- = I \setminus I_u^+$ be a set of items that user u has not yet interacted with.

Our goal in top-k recommendation is to recommend k most preferred (or likely) items to user u from I_u^-. To achieve this goal, we predict the likelihood of interaction between user u and I_u^- (or preference of user u over I_u^-), and then select a rank-list of k items with the highest prediction score to recommend to user u. Our learning task is to find a scoring function f that predicts an *interaction score* $\hat{r}_{ui}$ for each user u and an unobserved item $i \in I_u^-$. The scoring function f is formulated as $\hat{r}_{ui} = f(u, i|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents model parameters.

Most of model-based collaborative filtering methods [19] differentiate from each

other on the scoring function f formulation or architecture or the objective functions used for parameter learning. There are various formulation of f such as deep networks [18] and matrix factorization [33], etc. In general, the objective functions fall into two categories. *Point-wise loss* [7], [8], by assuming an unobserved user-item interaction as a negative example, minimizes the error (or distance) between predicted interaction score $\hat{r}_{ui}$ and its actual value r_{ui}. In contrast to point-wise loss, *Pairwise loss* [38][47] directly optimize the ranking of the user-item interaction while assuming that users prefer observed items to unobserved items.

4.2 Preliminaries

In this section, we describe VAE, which serves as a building block for our proposed model.

4.2.1 Variational AutoEncoders (VAEs)

Our model uses the variational autoencoder (VAE) [94] as a building block. The VAE is a popular variant of autoencoders. It is a deep generative model that can learn complex distributions. Each VAE, similar to classical autoencoders, consists of encoder and decoder networks. The encoder first encodes the inputs to latent representations, and then the decoder reconstructs the original inputs from latent representations. However, the VAE differentiates from classical autoencoders by encoding an input as a distribution over latent representations (rather than a single point). This probabilistic representation choice makes VAE a generative model and reduces overfitting by forcing smoother latent representation transitions. Figure 4.1

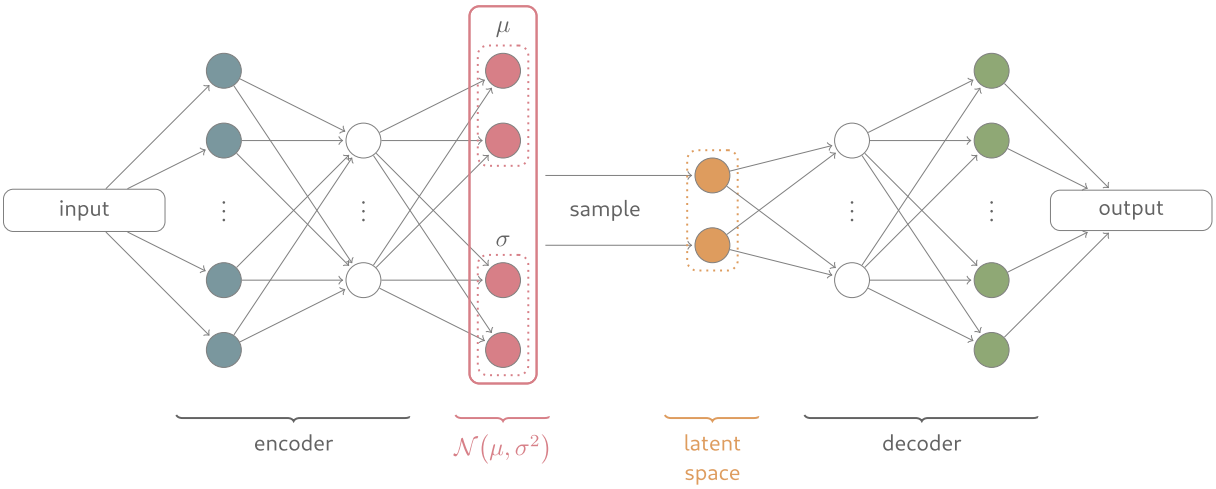


Figure 4.1: Structure of a variational autoencoder. Image is taken from [95]. VAE is used as a building block for our proposed model, for more details see section 4.3.1.

represents the overall structure of the VAE model, including encoder, decoder, and sampling components.

The encoder network of VAE encodes the input $\mathbf{x}$ to a d-dimensional latent representation $\mathbf{z}$, which is a multivariate random variable with a prior distribution $p(\mathbf{z})$. The common practice is to assume that $p(\mathbf{z})$ is a standard multivariate normal distribution:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.2}$$

One can view the encoder as the posterior distribution $p_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ parametrized by $\boldsymbol{\phi}$. Since this posterior distribution is intractable, it is usually approximated by variational distribution [96]:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_{\boldsymbol{\phi}}(\mathbf{x}), \sigma^2_{\boldsymbol{\phi}}(\mathbf{x})\mathbf{I}), \tag{4.3}$$

where two multivariate functions $\mu_{\boldsymbol{\phi}}(\mathbf{x})$ and $\sigma_{\boldsymbol{\phi}}(\mathbf{x})$ map the input $\mathbf{x}$ to the mean and standard deviation vectors, respectively. In VAE, $\mu_{\boldsymbol{\phi}}(\mathbf{x})$ and $\sigma_{\boldsymbol{\phi}}(\mathbf{x})$ are jointly formulated by *inference network*:

$$f_{\boldsymbol{\phi}}(\mathbf{x}) = [\mu_{\boldsymbol{\phi}}(\mathbf{x}), \sigma_{\boldsymbol{\phi}}(\mathbf{x})]. \tag{4.4}$$

The decoder network $p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})$, also known as *generative network*, takes $\mathbf{z}$ and outputs the probability distribution over (reconstructed) input data $\mathbf{x}$. Putting together the encoder and decoder networks, one can lower bound the log-likelihood of the input $\mathbf{x}$ by:

$$\log p(\mathbf{x}) \geq \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} dz = E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})\right] - KLq_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p(\mathbf{z}),$$

where KL is Kullback-Leibler divergence distance measuring the difference between the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the unit Gaussian distribution $p(\mathbf{z})$ This lower bound, known as *evidence lower bound (ELBO)*, is maximized for learning the parameters of encoder and decoder, $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, respectively. Equivalently, for learning VAE parameters, one can minimize the negation of the ELBO as a loss function (see Eq. 4.5) by stochastic gradient decent with the reparameterization trick [94].

$$L_{\texttt{VAE}}(\mathbf{x}|\boldsymbol{\theta}) = -E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})] + KLq_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p(\mathbf{z}), \tag{4.5}$$

where $\boldsymbol{\theta} = [\boldsymbol{\psi}, \boldsymbol{\phi}]$. One can view this loss function as a linear combination of *reconstruction loss* and KL divergence, which serves as a regularization term. KL divergence measures the dissimilarity between two distributions. Our goal is to find

the variational parameters that minimize this divergence. It is common to use Gaussian distribution to approximate the true posterior [17], [96], [97]. It would be easier to compute KL divergence if we use Gaussian distribution [29].

Recent research [16], [58] has introduced regularization hyper-parameter β for controlling the trades of between regularization term and reconstruction loss:

$$L_{\text{VAE}}(\mathbf{x}|\boldsymbol{\theta}, \beta) = -E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z})] + \beta KL q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p(\mathbf{z}), \tag{4.6}$$

In this paper, as our input data $\mathbf{x}$ is a binary vector (i.e., implicit feedback), we consider logistic likelihood for the output of VAE decoder. Defining $f_{\boldsymbol{\psi}}(\mathbf{z}) = [o_i]$ as the output of generative function of the decoder, the logistic log-likelihood for input $\mathbf{x}$ is

$$\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}) = \sum_i x_i \log \sigma(o_i) + (1 - x_i)(1 - \sigma(o_i)). \tag{4.7}$$

Here, $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic function. This logistic likelihood renders the reconstruction loss to the cross-entropy loss.

4.3 Proposed approach

4.3.1 Joint Variational Autonecoder (JoVA)

We here detail the proposed *joint variational autoencoder (JoVA)* framework and its variant for top-k recommendation with implicit feedback. We first discuss the model architecture of JoVA and then discuss various objectives functions used for parameter learning.

4.4 Model

Figure 4.2 illustrates the general structure of JoVA model we propose. Our model consists of two separate variational autoencoders:

- *user VAE*: given the implicit feedback matrix $\mathbf{R}$, the user VAE aims to reconstruct the matrix row-by-row.
- *item VAE*: given the implicit feedback matrix $\mathbf{R}$, the item VAE reconstructs it column-by-column.

In other words, user VAE takes and reconstruct each user vector $\mathbf{R}_u$ (i.e., a row of the matrix). Similarly, item VAE takes and reconstruct each item vector $\mathbf{R}_i^T$ (i.e., a column of the matrix). User vector u indicates the preference of a user u on all the items i in the dataset, whereas item vector i indicates the preferences of all the users to the item i.

User VAE and item VAE independently and simultaneously complete the implicit feedback matrix. The final output of our model is the average of two predicted implicit matrices:

$$\hat{\mathbf{R}} = \frac{1}{2}(\hat{\mathbf{R}}^{user} + \hat{\mathbf{R}}^{item}), \tag{4.8}$$

where $\hat{\mathbf{R}}^{user}$ and $\hat{\mathbf{R}}^{item}$ are implicit matrices predicted (or completed) by user VAE and item VAE respectively. The parameters of user VAE and item VAE are learned jointly with a joint objective function (see below for details).

In Section 4.2.1, we provided details about how to build and train a variational autoencoder. For each user u , the user VAE samples a K-dimensional latent representation z_u from a standard Gaussian prior. Then the latent representation z_u is

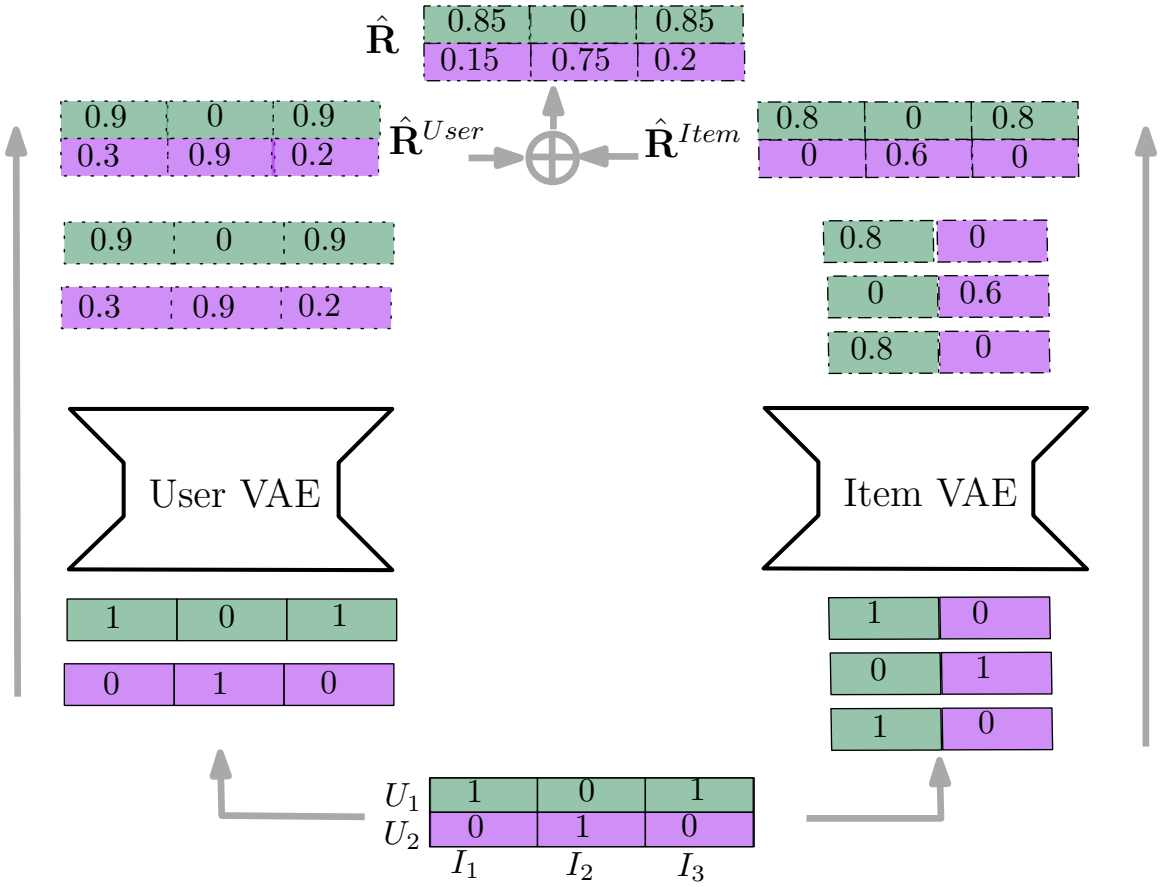


Figure 4.2: Illustration of the proposed model. User VAE and item VAE take the whole rating matrix and recover the whole rating matrix independently. The final rating matrix is the average of rating matrices constructed by user VAE and item VAE.

transformed via a non-linear function f in order to produce a probability distribution over I items $\pi(z_u)$. The decoder is modeled as: $\log P(x_u|z_u)$. Item VAE works similarly.

JoVA model is designed carefully to capture both user-user and item-item correlations at the same time. The item VAE encodes similar items close to each other in its latent representations to preserve their correlations. In contrast, user VAE encodes similar users close to each other in its latent representations to preserve their corre-

lations. The joint optimization of these two VAEs helps their fine-tune calibration. Therefore user VAE and item VAE can complement each other in their predictions. Both item and user VAEs together can learn complementary information from user-item interactions beyond what each could separately learn. This richer learning of user-item interaction is a valuable asset, especially for sparse datasets (as confirmed by our experiments below). In other words, when the dataset is sparse and lacks enough information for recommendation task, considering only user-user correlation or only item-item correlation may not provide enough information to provide a robust recommendation. However, using both user-user correlation and item-item correlation can extract more and complementary information from the data. User VAE can extract information that item VAE is not able to obtain and vise versa. All in all, considering the average of two predicted rating matrices obtained by user VAE and item VAE can yield improved implicit top-k recommendation performance and quality, especially with sparser datasets.

We can see a connection between JoVA and ensemble learning. Ensemble learning combines multiple models into one learning framework. Similar to ensemble learning, JoVA combines user VAE and item VAE into one learning framework for final prediction. From this perspective, each VAE independently predicts the rating matrices, and then the final prediction is the aggregation (unweighted averaging) of VAEs' predictions. Unweighted averaging is shown to be a reliable choice as an aggregation method in the ensemble of deep learning models [98]. Averaging user VAE and item VAE predictions can reduce the expected variance of neural network models and reduce the risk of overfitting. These can result in improving the model accuracy of deep neural networks.

In order to provide the top-K recommended list for each user u, for all the item $i \in \mathbf{I}$, we sort the output scores r_{ui} directly to obtain the top-k ranked items. Then we select the k items that score highest.

4.4.1 Weighted Average

This unweighted averaging in JoVA-Hinge can easily be extended to the weighted averaging at the cost of tuning more hyper-parameters for each dataset, but with the promise of increased accuracy.[1] To study the effectiveness of taking an average user VAE and item VAE, we introduce hyper-parameter γ:

$$\hat{\mathbf{R}} = \gamma \hat{\mathbf{R}}^{item} + (1 - \gamma)\hat{\mathbf{R}}^{user}. \tag{4.9}$$

4.4.2 Model Learning

To learn model parameters of JoVA, we consider two variants of loss functions. We call the proposed model without considering pairwise objective function JoVA and the model with pairwise objective function JoVA-Hinge. One loss function naturally arises from the combination of two user and item VAEs:

$$L_{\texttt{JoVA}}(\mathbf{R}|\boldsymbol{\theta}, \alpha) = \sum_{u \in U} L_{\texttt{VAE}}(\mathbf{R}_u|\boldsymbol{\theta}_U, \alpha) + \sum_{i \in I} L_{\texttt{VAE}}(\mathbf{R}_i^T|\boldsymbol{\theta}_I, \alpha) \tag{4.10}$$

[1] We have confirmed this in experiments (see Section 5.7).

Here, $\boldsymbol{\theta}_U$ and $\boldsymbol{\theta}_I$ represent the model parameters of user and item VAEs repsectively, and $L_{\texttt{VAE}}$ is computed by following equations:

$$L_{\texttt{VAE}}(\mathbf{R}_u|\boldsymbol{\theta}_U, \alpha) = -E_{q_{\boldsymbol{\phi}_U}}[\log p_{\boldsymbol{\psi}_U}(\mathbf{R}_u|\mathbf{z})] + \alpha KLq_{\boldsymbol{\phi}_U}(\mathbf{z}|\mathbf{R}_u)p(\mathbf{z}), \tag{4.11}$$

$$L_{\texttt{VAE}}(\mathbf{R}_i|\boldsymbol{\theta}_I, \alpha) = -E_{q_{\boldsymbol{\phi}_I}}[\log p_{\boldsymbol{\psi}_I}(\mathbf{R}_i|\mathbf{z})] + \alpha KLq_{\boldsymbol{\phi}_I}(\mathbf{z}|\mathbf{R}_i)p(\mathbf{z}), \tag{4.12}$$

with the logistic likelihood of:

$$\log p_{\boldsymbol{\psi}_U}(\mathbf{x_u}|\mathbf{z}) = \sum_u x_u \log \sigma(o_u) + (1 - x_u)(1 - \sigma(o_u)), \tag{4.13}$$

$$\log p_{\boldsymbol{\psi}_I}(\mathbf{x_i}|\mathbf{z}) = \sum_i x_i \log \sigma(o_i) + (1 - x_i)(1 - \sigma(o_i)), \tag{4.14}$$

Where $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic function.

To further specialize and improve accuracy of JoVA model for top-k recommendation, we incorporate a pairwise ranking loss in its loss function. Specifically, we introduce *JoVA-Hinge* loss function:

$$L_{\texttt{JoVA}-\texttt{H}}(\mathbf{R}|\boldsymbol{\theta}, \alpha, \beta, \lambda) = L_{\texttt{JoVA}}(\mathbf{R}|\boldsymbol{\theta}, \alpha) + \beta L_{\texttt{H}}(\mathbf{R}|\boldsymbol{\theta}, \lambda) \tag{4.15}$$

where

$$L_{\texttt{H}}(\mathbf{R}|\boldsymbol{\theta}, \lambda) = \sum_{u \in U} \sum_{i \in I_u^+} \sum_{j \in \tilde{I}_u^-} \max(0, \hat{r}_{uj} - \hat{r}_{ui} + \lambda)$$

is *hinge loss function*, widely and successfully used as a pairwise ranking loss [15], [99], [100]. Here, $\hat{r}_{ui}$ is the predicted ratings of user u for item i, and λ is the margin hyper-

parameter for the hinge loss. The hinge loss is built upon the assumption that user u prefers his interacted item $i \in I_u^+$ over an uninteracted item (or negative example) $j \in I_u^-$ with the margin error of λ. In practice, the hinge loss is usually computed over the sample of negative examples. We have introduced the hyper-parameter β for controlling the influence of hinge loss to the JoVA's objective function.

4.4.3 Variational Autoencoder Model Architecture

In this section, we explain the architecture of user VAE and item VAE. We build the encoder and the decoder of user VAE and item VAE based on MLP (only fully connected layers). In a user VAE and item VAE, the encoder includes one (or more) hidden layer(s) connecting the input layer to the latent representation. Then the decoder consists of one (or more) hidden layer(s) connecting the latent representation to the output layer of user VAE (item VAE). The neural network takes an input vector $\mathbf{x}_u$ (or $\mathbf{x}_i$) and maps it to an output vector: $\mathbf{y}_u$ (or $\mathbf{y}_i$) with:

$$y_{ui} = a(\mathbf{W}_u x_u + \mathbf{b}_u) \tag{4.16}$$

where a is a non-linear activation function. $\mathbf{W}_u$ is a weight matrix, and b is a bias. Both user VAE and item VAE are symmetric VAEs. It means that the dimensions of the decoder's hidden layers are the same as dimensions of the encoder but in reversed order. We explained the choice of activation functions and numbers of hidden layers and dimensions in the experiments chapter.

4.5 Mini-batch Optimization Algorithm

In each training iteration, if we feed the whole user-item-rating matrix for training, it would not be practical, particularly for massive datasets. Following previous work [15], we decomposed the whole rating matrix into several small matrices. Then each of the small matrices is considered as one mini-batch.

In each training iteration, user VAE takes n user rating vectors represented by $\mathbf{B}^U = \mathbf{R}_p \in R^{n \times M}$ where p is a list of n randomly sampled row indexes (user vectors). And in each training iteration, item VAE takes m item rating vectors represented by $\mathbf{B}^I = \mathbf{R}_q \in R^{m \times N}$ where q is a list of m randomly sampled column indexes (item vectors). The user VAE outputs $\hat{\mathbf{B}^U}$ and the item VAE outputs $\hat{\mathbf{B}^I}$. There are $n \times m$ common entries covered by both $\hat{\mathbf{B}^U}$ and $\hat{\mathbf{B}^I}$. Matrix $\mathbf{C} \in \mathbf{R}^{n \times m}$ represents commonly covered parts of $\hat{\mathbf{B}^U}$ and $\hat{\mathbf{B}^I}$. Then in each training iteration, matrix $\mathbf{C}$ can be predicted by:

$$\hat{\mathbf{C}} = \frac{1}{2}(\hat{\mathbf{B}}^U_q + \hat{\mathbf{B}}^I_p). \tag{4.17}$$

Chapter 5

Experiments

Our empirical experiments intend to assess the effectiveness of our proposed methods JoVA and JoVA-Hinge, for top-k recommendation with implicit feedback. We compare our methods' accuracy under various evaluation metrics with an extensive subset of state-of-the-art methods on real-world datasets. We further study the effectiveness of our methods in handling cold-start users. We also investigate the training and inference time of the proposed model. We show the promise of increased accuracy of weighed averaging at the cost of tuning more hyper-parameters for each dataset. Finally, we also investigate the impact of the hyper-parameter choice on recommendation performance. The source code is available on Github[1].

In summary, our experiments aim to address the following questions:

- (Q1) Does JoVA-Hinge outperform state-of-the-art implicit CF methods (section 5.2)?
- (Q2) Are incorporating pairwise ranking loss helpful for learning user represen-

[1] `https://github.com/bahareAskari/JoVA-Hinge.git`

tations (section 5.3)?

- (Q3) How does the proposed model perform for cold-start users comparing other methods (section 5.5)?

- (Q4) Is using an unweighted average reasonable choice comparing to the weighted average for JoVA and JoVA-Hinge (section 5.7)? What is the impact of hyper-parameter choice (section 5.8)?

5.1 Evaluation Datasets

We report results obtained on four real-world recommendation system datasets: MovieLens-1M (ML1M) [2], Yelp [3], Pinterest [4], and Netflix.[5] Table 5.1 provides the statistics of these datasets after pre-processing.

Dataset	**#User**	**#Item**	**#interaction**	**Sparsity(%)**
MovieLens	6,027	3,062	574,026	96.89
Yelp	12,705	9,245	318,314	99.729
Pinterest	55,187	9,911	1,500,806	99.726
Netflix	70,000	17,769	86,23,831	99.31

Table 5.1: Summary statistics of the four datasets.

MovieLens (ML1M) [101] dataset has been widely used for evaluating collaborative filtering models. MovieLens consists of user-movie ratings obtained from a movie recommendation service. ML1M originally includes five-star user-item ratings. As previous work[8], [15], we used the version of the ML1M dataset that every user has

[2]http://files.grouplens.org/datasets/movielens/ml-1m.zip.
[3]https://www.yelp.com/dataset/challenge.
[4]https://sites.google.com/site/xueatalphabeta/academic-projects.
[5]https://www.kaggle.com/netflix-inc/netflix-prize-data

given at least 20 ratings. The Yelp dataset is a subset of Yelp businesses, reviews, and user data. Yelp originally consists of five-star user-item ratings. Pinterest is a visual discovery website for gathering, arranging, and organizing content. Users on Pinterest can pin images on their boards. This dataset is an implicit feedback dataset originally gathered by [102] for investigating the image recommender system performance.

5.1.1 Preprocessing

In ML1M, Yelp and Netflix following previous work [15], [16], [57], a user-item rating was converted to 1 if it is greater than or equal to 4 and 0 otherwise. In Pinterest, following the previous work [8], we kept only users with at least 20 interactions (pins). Interaction is 1 if the user has pinned the image to her board. For each dataset, we randomly selected 20% of all user-item values as the test set and 80% as the training set. In the testing set, we further set 10% as a validation set. For Netflix, we have randomly selected 70,000 users (with all their user-item interactions) from the original dataset.

5.1.2 Evaluation Metrics

In terms of evaluation metrics, we use Precision@k, Recall@k, F1@k and Normalized Discounted Cumulative Gain@k (NDCG@k) to assess the quality of ranked list ω_u predicted for user u with the ground-truth items. We report the score averaged by all test interactions for all metrics.

Precision@k (P@K) quantifies which fraction of u's recommended ranked list ω_u

which are u's true preferred items:

$$P@K(\omega_u, I_u^*) = \frac{1}{k} \sum_{i=1}^{k} 1\left[\omega_u(i) \in I_u^*\right], \tag{5.1}$$

where $1\left[.\right]$ is the indicator function, $\omega_u(i)$ is the i^{th} ranked item in ω_u, and I_u^* is u's true preferred items in held-out data.

Similarly, *Recall@k (R@K)* measures which fraction of u's true preferred items I_u^* are present in u's recommended ranked list ω_u:

$$R@K(\omega_u, I_u^*) = \frac{1}{|I_u^*|} \sum_{i=1}^{k} 1\left[\omega_u(i) \in I_u^*\right]. \tag{5.2}$$

F1-score@k (F1@k) captures both of these metrics by computing the harmonic mean of the precision and recall. It reaches its maximum of 1 if both precision and recall are perfect (i.e., have value of 1):

$$F1@k(\omega_u, I_u^*) = \frac{2 \cdot P@K(\omega_u, I_u^*) \cdot R@K(\omega_u, I_u^*)}{P@K(\omega_u, I_u^*) + R@K(\omega_u, I_u^*)} \tag{5.3}$$

One criticism of P@K, R@K, and F1@K is giving the same importance to all items ranked within the first k. To address this, NDCG@k gives higher weight to the higher ranked items:

$$NDCG@k(\omega_u, I_u^*) = \frac{1}{IDCG@k} \sum_{i=1}^{k} \frac{2^{1[\omega_u(i) \in I_u^*]} - 1}{\log_2(i+1)}, \tag{5.4}$$

where $IDCG@k = \sum_{i=1}^{k} (1/\log_2(i+1))$ normalizes NDCG with the maximum of 1.

We report the average of these metrics in our experiments, when the average is

taken over all testing users.

5.1.3 Baselines

To evaluate the effectiveness of our model, we compare with the following methods:

- **BPR** [38]. This model optimizes the MF model with a pairwise ranking loss. We evaluated the number of latent factors of [8; 16; 32; 64], reporting the best performance. It utilizes (Stochastic gradient descent) SGD to learn user preferences from implicit feedback data. We used a fixed learning rate.

- **CDAE** [6][14]. This model assumes that observed ratings are a corrupted user's preferences. It extends the Denoising Auto-Encoder. It improves the denoising autoencoder by adding a latent user factor to the input. We used sigmoid function as activation function and chose the hinge-based pairwise loss presented in CDAE.

- **Mult-VAE** [7] [16]. This model is a multi-layer VAE model. Mult-VAE used the multinomial probabilistic instead of Gaussian and Bernoulli distributions usually used in VAE. It used Bayesian inference for parameter estimation.

- **NCF** [8] [8]. It is state-of-the-art collaborative filtering method with implicit feedback. It learns user-item interaction function using neural networks. It combines MF and multi-layer perceptrons (MLP) and employs binary cross-entropy loss.

[6] https://github.com/gtshs2/Collaborative-Denoising-Auto-Encoder
[7] https://github.com/dawenl/vae$_c f$
[8] https://github.com/hexiangnan/neural$_c ollaborative_f iltering$

- **JCA** [9] [15]. JCA deploys two classical autoencoders for modeling users and items, and only uses hinge pairwise loss function.

- **FAWMF** [103]. It is an adaptive weighted matrix factorization method based on a variational autoencoder. FAWMF models data confidence weights with a community-based inference neural network.

We have used the implementations and optimal parameter settings reported by the original papers for all these baselines.

5.1.4 Hyperparameters Setting

For learning all the models, we used Adam [104] as the optimizer. We set the learning rate of 0.003. We decomposed the whole rating matrix into several small matrices. Then each of the small matrices is considered as one mini-batch. We set the size of the mini-batch to 1500. So, each mini-batch size is set to encompass 1500 rows and 1500 columns. We use a validation set to find the optimal hyper-parameters for JoVA-Hinge. We set $\lambda = 0.15$ (margin parameter) and $\alpha = 0.01$ for all experiments, but picked β individually for each dataset: $\beta = 0.001$ for Yelp, and $\beta = 0.01$ for both MovieLens, Pinterest and Netflix.

We randomly sampled one negative instance per positive instance, so the ratio of negative sampling is 1. Following previous work [15], in each epoch, we re-sampled negative samples. We had two hidden layers for each encoder and decoder, each with 320 dimensions and tanh activation functions, while the sigmoid activation function was used for the output layers. For both VAEs[10] , we set the dimension of the latent

[9] https://github.com/Zziwei/Joint-Collaborative-Autoencoder

[10] See section 4.4.3 for description of VAE architecture, including its inputs and outputs.

representation to 80. The number of hidden neurons was selected by grid search. Adding more layers did not improve recommendation accuracy.

5.2 Performance Comparison

In this section, we report the performance of our model and the baselines and discuss the results. We compared the top-k recommendations' performance with various $k \in \{1, 5, 10\}$. We report the results F1-score@k and NDCG@k of JoVA-Hinge and baselines on four datasets in Table 5.2 and Table 5.3 [11].

From the results, we can notice that methods using neural networks are superior to traditional ranking approach BPR, which indicates the effectiveness of non-linear features for recommendations task and in learning the user-item interaction function. As we can see, our model achieves the best performance in the term of F1 measure on four datasets and outperforms the state-of-the-art methods. Compared with the best baseline (JCA), F1-score@k is improved by up to 3.65% in ML1M, 25.62% in Yelp, 33.33% in Pinterest, and 50% in Netflix.

For NDCG, JoVA-Hinge also outperforms others significantly in three datasets of Yelp, Pinterest and Netflix. In Yelp, the mimimum improvement is 8.19% (for $k = 10$) and the maximum improvement is 10.86% (for $k = 1$). The JoVA-Hinge has even higher improvement for Pinterest with the mimimum of 21.72% (for $k = 10$) and the maximum of 34.82% (for $k = 1$). In Netflix, the mimimum improvement is 5.26% (for $k = 1$) and the maximum improvement is 11.76% (for $k = 5$). For ML1M and NDCG, the performance of JoVA-Hinge is comparable to the performance of best baseline FAWMF. Higher NDCG determines that the model can efficiently rank

[11] The purple shows the best results and the gray shows the second best results.

	ML1M					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
BPR	0.0410	0.1285	0.1698	0.2843	0.2549	0.2434
NCF	0.0513	0.1487	0.1883	0.2955	0.2727	0.2709
CDAE	0.0518	0.1474	0.1873	0.3428	0.2896	0.2728
Mult-VAE	0.0518	0.1420	0.1801	0.3428	0.2886	0.2695
FAWMF	0.0595	0.1661	0.2068	0.3775	0.3176	0.2991
JCA	0.0602	0.1634	0.2080	0.3699	0.3125	0.2976
JoVA-Hinge	0.0624	0.1665	0.2115	0.3718	0.3143	0.3013
% improve	3.65	0.24	1.68	-1.53	-1.04	0.73

	Yelp					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
BPR	0.0065	0.0180	0.0219	0.01660	0.0223	0.0301
NCF	0.0153	0.0325	0.0350	0.0367	0.0392	0.0497
CDAE	0.0159	0.0315	0.0356	0.0378	0.0390	0.0471
Mult-VAE	0.0148	0.0317	0.0344	0.0350	0.0381	0.0465
FAWMF	0.0152	0.0290	0.0305	0.0358	0.0358	0.0425
JCA	0.0160	0.0350	0.0376	0.0405	0.0440	0.0537
JoVA-Hinge	0.0201	0.0391	0.0401	0.0449	0.0483	0.0581
% improve	25.62	11.71	6.64	10.86	9.77	8.19

Table 5.2: Performance of the baselines and JoVA-Hinge on MovieLens and Yelp under F1@k and NDCG@k metrics. The purple shows the best results and the gray shows the second best results.

	Pinterest					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
BPR	0.0120	0.0292	0.0333	0.0328	0.0312	0.0414
NCF	0.0123	0.0306	0.0375	0.0375	0.0348	0.0479
CDAE	0.0154	0.0349	0.0401	0.0415	0.0387	0.0506
Mult-VAE	0.0153	0.0349	0.0402	0.0466	0.0397	0.0504
FAWMF	0.0131	0.0310	0.0360	0.0416	0.0359	0.0450
JCA	0.0150	0.0383	0.0456	0.0448	0.0424	0.0557
JoVA-Hinge	0.0200	0.0471	0.0542	0.0604	0.0532	0.0678
% improve	33.33	22.97	18.85	34.82	25.47	21.72

	Netflix					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
BPR	0.001	0.003	0.005	0.009	0.010	0.010
NCF	0.001	0.006	0.008	0.013	0.012	0.013
CDAE	0.001	0.005	0.007	0.011	0.011	0.013
Mult-VAE	0.001	0.004	0.010	0.011	0.011	0.011
FAWMF	0.002	0.006	0.009	0.019	0.017	0.017
JCA	0.002	0.007	0.011	0.017	0.016	0.017
JoVA-Hinge	0.003	0.008	0.012	0.020	0.019	0.019
% improve	50	14.29	9.10	5.26	11.76	11.76

Table 5.3: Performance of the baselines and JoVA-Hinge on Pinterest and Netflix datasets under F1@k and NDCG@k metrics. The purple shows the best results and the gray shows the second best results.

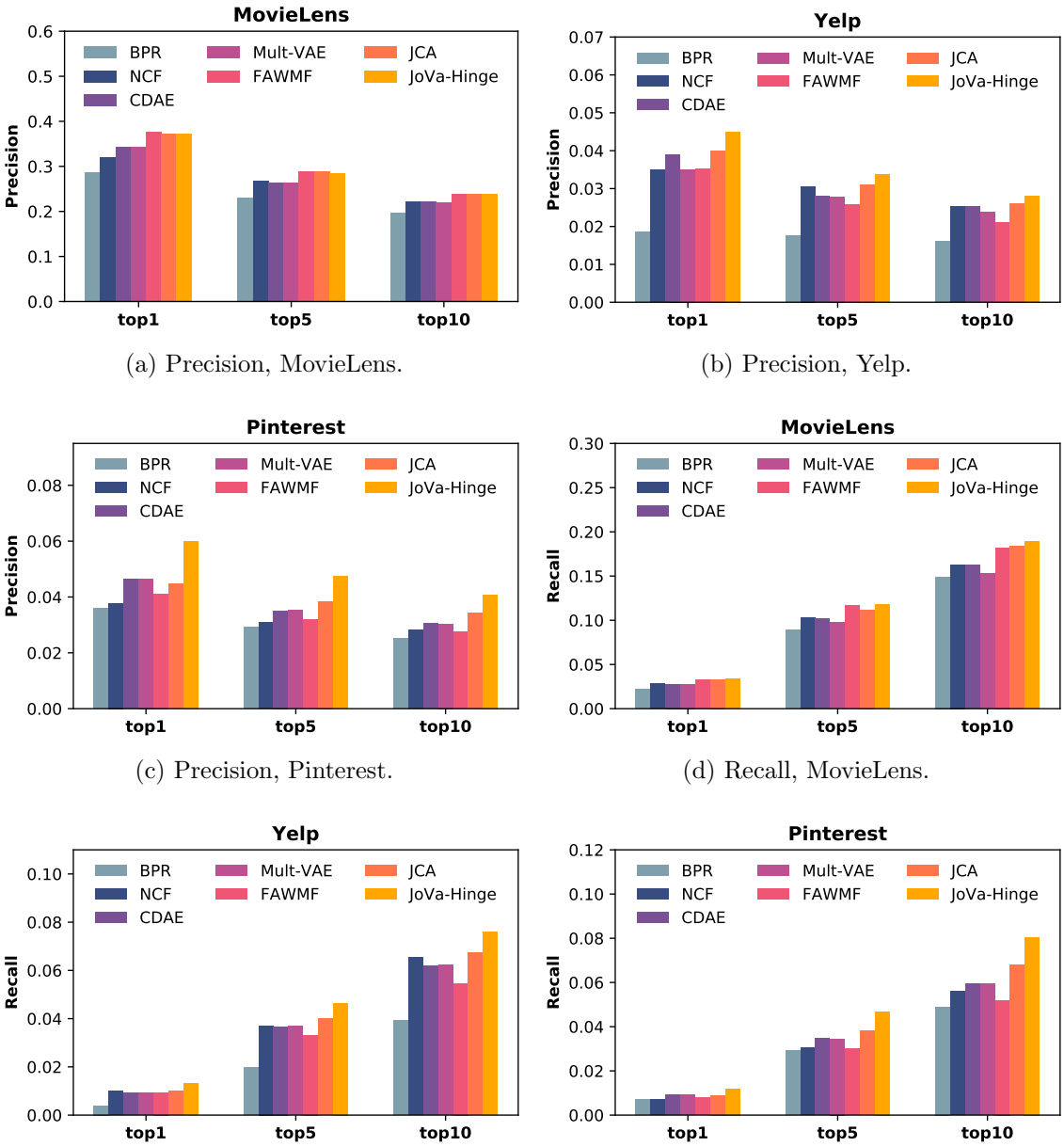
the item's user preferred in top ranks. Cross-examination of F1-score and NDCG results suggests that our JoVA-Hinge model significantly improves the state-of-the-art methods in terms of both F1-score and NDCG for sparse datasets (i.e., Yelp, Netflix and Pinterest).

As we can see, the performance difference between the JoVA-Hinge model and other methods is larger on Yelp, Pinterest and Netflix, which are sparser than MovieLens. It may be due to considering user-user correlations or item-item correlations separately is not enough to provide effective recommendations.

We also find that the performance improvement is more significant for smaller k. It can be considered as advantageous for recommendation tasks. Because although recommendations are presented as an ordered list, users usually pay more attention to the first item. As the position of the item in the ranking gets higher, users may pay less attention. Therefore, our results also suggest that JoVA-Hinge offers a more significant improvement for smaller k (e.g., $k = 1$ or $k = 5$), which is of particular practical interest for reducing cognitive burden on users, when the recommendation slate is small.

Figure 5.1 illustrates the performance of all methods under precision@k and recall@k for various k and datasets. We noticed that the performance with Precision@k and Recall@k trends were similar to trends with F1-score@k and NDCG@k. JoVA-Hinges achieves the best results in most cases. The only exception is on MovieLens datasets, where FAWMF performs better than it by a small margin in Presion@1 and Presion@5. Recall can be considered an essential metric in applications when the purpose is to present the greatest possible number of relevant items to the user.

On sparser datasets (Yelp and Pinterest), the JoVA-Hinge can perform better by



(a) Precision, MovieLens.

(b) Precision, Yelp.

(c) Precision, Pinterest.

(d) Recall, MovieLens.

(e) Recall, Yelp.

(f) Recall, Pinterest.

Figure 5.1: Precision (a–c) and Recall (d–f) for all methods and three datasets of MovieLens, Yelp, and Pinterest.

more significant margin under Precision and Recall. This observation suggests the importance of considering both user-user and item-item correlations on sparse datasets. Similar to previous metrics, all neural-based methods outperform BPR under Precision and Recall by a large margin, which confirms non-linear features' success in the recommendation task. Overall, based on the results, our model improves the recommendation performance.

In the real-world scenario abundant implicit feedback data is available, whereas explicit data feedback is scarce. For example, on Amazon, all users' purchasing history can be considered as a source of implicit feedback data. However, small percentages of users rate the products they bought or provide reviews about them. But for academic research, accessing implicit feedback data is a challenging factor as there are few public implicit datasets (e.g., purchase transactions) available. So it is crucial to pre-process the explicit datasets and convert them to implicit feedback data.

In this thesis, for the MovieLens dataset, following previous work [15], [16], [57] we converted the rating to 1 if it is higher or equal to 4. But some previous work [8] considered rating as 1 if user has rated the item. Table 5.4 shows the results of MovieLens where explicit data is binarized similar to previous work [8] [12]. We converted rating to 1 if the user has rated the item and to 0 otherwise. We can observe that JoVA-Hinge outperforms other models (Mult-VAE, CDAE, FAWMF, and JCA) with some exceptions. Precision was increased by up to 4.56%, F1-score by up to 4.31%, and NDCG by up to 4.63%. Mult-VAE performs better than other models under recall (up to 65.93%) and F1-score@1 (48.55%). Recall is an important metric when we want to recommend as many relevant items as possible to users. Mult-

[12]The purple shows the best results and the gray shows the second best results.

VAE is more successful than other models under recall showing considering user-user correlations using one variational autoencoder can find more relevant items in this dataset. Except for Mult-VAE, JoVA-Hinge performs better than other models under recall.

	ML1M			
	P@1	**R@1**	**F1@1**	**NDCG@1**
Mult-VAE	0.0863	0.0151	0.0257	0.0863
CDAE	0.1266	0.0077	0.0145	0.1267
FAWMF	0.1514	0.0081	0.0154	0.1513
JCA	0.1457	0.0084	0.0159	0.1457
JoVA-Hinge	0.1583	0.0091	0.0173	0.1583
% improve	4.56	-65.93	-48.55	4.63

	P@5	**R@5**	**F1@5**	**NDCG@5**
Mult-VAE	0.0645	0.0547	0.0591	0.0710
CDAE	0.1130	0.0345	0.0529	0.1165
FAWMF	0.1325	0.0366	0.0574	0.1371
JCA	0.1274	0.0349	0.0548	0.1319
JoVA-Hinge	0.1349	0.0388	0.0603	0.1406
% improve	1.81	-40.98	2.03	2.55

	P@10	**R@10**	**F1@10**	**NDCG@10**
Mult-VAE	0.0524	0.0883	0.0658	0.0811
CDAE	0.1071	0.0632	0.0795	0.1194
FAWMF	0.1222	0.0661	0.0858	0.1358
JCA	0.1213	0.0660	0.0855	0.1337
JoVA-Hinge	0.1232	0.0702	0.0895	0.1395
% improve	0.82	-25.78	4.31	2.72

Table 5.4: Performance of the baselines and JoVA-Hinge on MovieLens dataset with different processing under all four metrics. The purple shows the best results and the gray shows the second best results.

5.3 Effect of Pairwise Ranking Loss

To confirm the effectiveness of incorporating pairwise ranking loss in the loss function of the proposed model, we compare the result of JoVA-Hinge with JoVA. In other words, we aim to understand whether both variational encoders and pairwise loss function have contributed to the success of JoVA-Hinge. Table 5.8, 5.6 and 5.7 present the performance of JoVA-Hinge and JoVA in terms of four evaluation metrics.

As we can see, using pairwise ranking loss in conjunction with VAE loss improves accuracy on almost every case (except for P@1, P@10, and NDCG@10 on MovieLens). So we empirically show that the hinge objective function can be effective for the implicit top-k recommendation. We believe this successful combination of VAEs and pairwise loss functions can be extended to other models built based on VAEs building blocks even in other applications (e.g., vision, speech, etc.).

	ML1M			
	P@1	**R@1**	**F1@1**	**NDCG@1**
JoVA	0.3730	0.0329	0.0605	0.3730
JoVA-Hinge	0.3718	0.0340	0.0624	0.3718
	P@5	**R@5**	**F1@5**	**NDCG@5**
JoVA	0.2845	0.1169	0.1657	0.3135
JoVA-Hinge	0.2853	0.1176	0.1665	0.3143
	P@10	**R@10**	**F1@10**	**NDCG@10**
JoVA	0.2382	0.1864	0.2092	0.2990
JoVA-Hinge	0.2340	0.1890	0.2115	0.3013

Table 5.5: Performance of JoVA and JoVA-Hinge for various k and metrics on MovieLense. The purple shows the best values.

	Yelp			
	P@1	**R@1**	**F1@1**	**NDCG@1**
JoVA	0.0420	0.0120	0.0180	0.0433
JoVA-Hinge	0.0449	0.0130	0.0201	0.0449
	P@5	**R@5**	**F1@5**	**NDCG@5**
JoVA	0.0320	0.0430	0.0360	0.0449
JoVA-Hinge	0.0337	0.0464	0.0391	0.0483
	P@10	**R@10**	**F1@10**	**NDCG@10**
JoVA	0.0272	0.0722	0.0395	0.0553
JoVA-Hinge	0.0281	0.0758	0.0401	0.0581

Table 5.6: Performance of JoVA and JoVA-Hinge for various k and metrics on Yelp. The purple shows the best values.

	Pinterest			
	P@1	**R@1**	**F1@1**	**NDCG@1**
JoVA	0.0571	0.0113	0.0189	0.0571
JoVA-Hinge	0.0604	0.0120	0.0200	0.0604
	P@5	**R@5**	**F1@5**	**NDCG@5**
JoVA	0.0464	0.0459	0.0461	0.0516
JoVA-Hinge	0.0474	0.0468	0.0471	0.0532
	P@10	**R@10**	**F1@10**	**NDCG@10**
JoVA	0.0406	0.0799	0.0538	0.0666
JoVA-Hinge	0.0409	0.0805	0.0542	0.0678

Table 5.7: Performance of JoVA and JoVA-Hinge for various k and metrics on Pinterest. The purple shows the best values.

5.4 Ablation Study

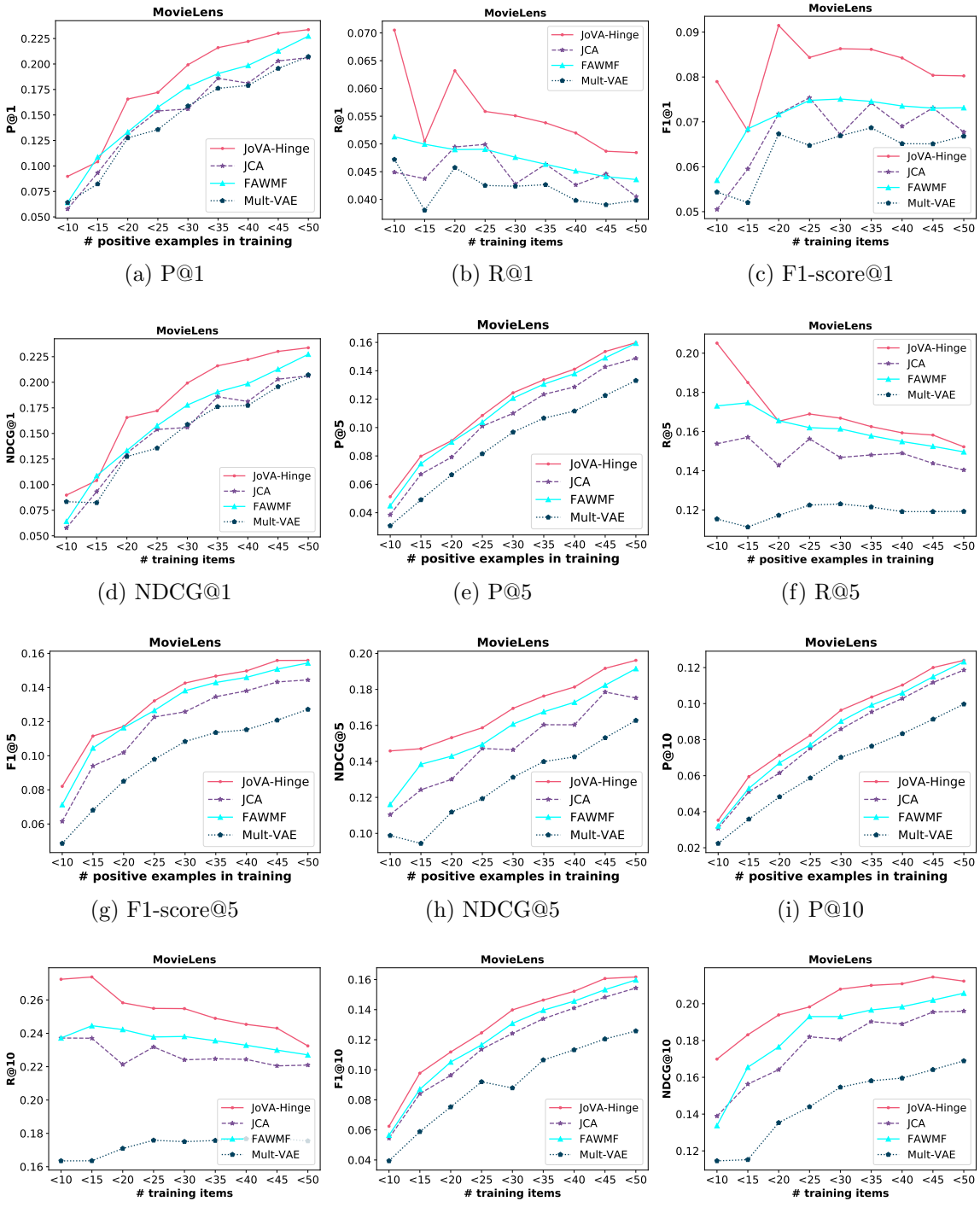
This section reports the results of an extensive ablation study on JoVA-Hinge by removing some of its components and evaluating the resulting models. Table 5.8 shows our ablation studies' results on three datasets under F1-score and NDCG. The results show that JoVA performs better both user VAE and item VAE for all datasets and metrics. This finding means that the ensemble of VAEs is more effective than individual VAEs. We can also observe that hinge loss can improve item VAE, but can not improve user VAE. However, except for one case, JoVA-Hinge improves JoVA for all datasets and metrics. The results suggest that incorporating hinge loss improves the performance of the ensemble of VAEs.

	ML1M		Yelp		Pinterest	
	F1@1	**NDCG@1**	**F1@1**	**NDCG@1**	**F1@1**	**NDCG@1**
User VAE	.0510	.3191	.0150	.0352	.0168	.0508
User VAE-Hinge	.0486	.3043	.0154	.0344	.0127	.0383
Item VAE	.0555	.3423	.0156	.0352	.0178	.0538
Item VAE-Hinge	.0573	.3479	.0181	.0407	.0200	.0597
JoVA	.0605	.3730	.0180	.0433	.0189	.0571
JoVA-Hinge	.0624	.3718	.0201	.0449	.0200	.0604
	F1@5	**NDCG@5**	**F1@5**	**NDCG@5**	**F1@5**	**NDCG@5**
User VAE	.1379	.2683	.0328	.0397	.0430	.0477
User VAE-Hinge	.1360	.2596	.0323	.0388	.0330	.0364
Item VAE	.1556	.2933	.0308	.0376	.0435	.0489
Item VAE-Hinge	.1558	.2932	.0362	.0442	.0469	.0520
JoVA	.1657	.3135	.0360	.0449	.0461	.0516
JoVA-Hinge	.1665	.3143	.0391	.0483	.0471	.0532
	F1@10	**NDCG@10**	**F1@10**	**NDCG@10**	**F1@10**	**NDCG@10**
User VAE	.1750	.254	.0365	.0495	.0512	.0625
User VAE-Hinge	.1728	.2482	.0346	.0474	.0401	.0486
Item VAE	.1980	.2816	.0340	.0472	.0498	.0621
Item VAE-Hinge	.1984	.2816	.0385	.0540	.0538	.0663
JoVA	.2092	.2990	.0395	.0553	.0538	.0666
JoVA-Hinge	.2115	.3013	.0401	.0581	.0542	.0678

Table 5.8: Ablation study of JoVA-Hinge for various k, datasets, and metrics. The purple and grey shows the best and second best results, respectively.

5.5 Impact of Sparsity

Data sparsity and cold-start problems are among the challenges in recommender systems. The cold-start problem refers to users or items that have no or few interactions in systems. We aim to understand how the accuracy of recommendation changes for users with a different number of user-item interactions (i.e., positive examples). We study the average accuracy of users with at most L user-item interactions in training data while increasing L. This setting allows us to study not only users with small L (e.g., L= 10) but also how more availability of user-item interactions affect the accuracy of recommendation.



(a) P@1 (b) R@1 (c) F1-score@1

(d) NDCG@1 (e) P@5 (f) R@5

(g) F1-score@5 (h) NDCG@5 (i) P@10

(j) R@10 (k) F1-score@10 (l) NDCG@10

Figure 5.2: Performance of Mult-VAE, FAWMF, JCA, and JoVA-Hinge on cold start users on the MovieLens.

We compare the performance of models Mult-VAE, FAWMF, JCA, and JoVA-Hinge. From Figure 5.2, we can observe that generally increasing the number of training items enhances the performance of the recommendations. In general, results under different metrics have similar patterns. Unsurprisingly, the performance of roughly all methods increases with more availability of user-item interactions. However, JoVA-Hinge outperforms other methods (with a few small exceptions) for users with a low number of user-item interactions and for well-established users. For R@1, F1-score@1, R@5, surprisingly, JoVA-Hinge works better for users with less than 10 interactions than users with less than 15. These few contradictions in results do not prevent us from concluding that recommendation accuracy increases for users with more observed interactions. We have enough evidence from experiments to confirm our findings.

These results suggest that the overall success of JoVA-Hinge is not limited to a specific class of users, and all users with various numbers of user-item interactions can benefit from its prediction power. Therefore the experiments verify that JoVA-Hinge can achieve a better performance than other methods for users with sparse interactions.

5.6 Runtime Analysis

We conduct further experiments to explore the runtime of JoVA-Hinge, JCA, Mult-VAE, and CDAE. All the models are trained on the same GPU. The training time of different models is shown in Table 5.9. On all the datasets, the training time of Mult-VAE, which uses one VAE, is less than other models. Comparing JoVA-Hinge and Multi-VAE, we can see that using two VAE increases the training time. The training

time of JoVA-Hinge is comparable to JCA. Classical autoencoder learns quicker and needs less computational resources than VAE. Table 5.10 represents inference time of different models. The inference is the step that a trained model is used to infer (predict) the testing examples and contains a similar forward pass as a training process to predict the outputs. In contrast, training does not include a backward pass to compute the loss and update weights.

	ML1M		**Yelp**		**Pinterest**	
	training time	**#epoch**	**training time**	**epoch**	**training time**	**#epoch**
CDAE	1188	54	2698	80	16790	35
JCA	968	50	1750	35	12600	25
Mult-VAE	158	160	216	110	5880	200
JoVA-Hinge	2215	90	2028	40	19500	30

Table 5.9: Comparisons of training time (second[s]) and number of epochs.

Model	**MovieLens**	**Yelp**	**Pinterest**
CDAE	2.5	10	60
JCA	1.75	7.6	28
Mult-VAE	1.5	6	41
JoVA-Hinge	2	5.5	48

Table 5.10: Comparisons of inference time (second[s]) for each epoch.

5.7 Weighted Average

In this section, we investigate the effect of different values of γ. We want to study the effectiveness of the weighted average in JoVA-Hinge. Table 5.11 shows the performance of JoVA-Hinge under F1-score@k and NDCG@k for various k on three datasets with different values of γ (by holding other hyper-parameters at their optimal settings). We increased γ from 0 to 1 with increments of 0.1. In MovieLens and Yelp, JoVA-Hinge with $\gamma = 0.5$ outperforms other values of γ (with few exceptions). In

	ML1M					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0486	0.1360	0.1728	0.3043	0.2596	0.2482
0.1	0.0486	0.1985	0.1843	0.3186	0.2752	0.2638
0.2	0.0539	0.1530	0.1951	0.3411	0.2929	0.2792
0.3	0.0585	0.1568	0.2021	0.3520	0.2995	0.2874
0.4	0.0630	0.1623	0.205	0.3754	0.3094	0.295
0.5	0.0624	0.1665	0.2115	0.3718	0.3143	0.3013
0.6	0.0614	0.1629	0.2096	0.3669	0.3077	0.2971
0.7	0.0616	0.1661	0.2099	0.3740	0.3125	0.2987
0.8	0.0579	0.1619	0.204	0.3581	0.3056	0.2916
0.9	0.0569	0.1608	0.2041	0.3545	0.3033	0.2902
1	0.0573	0.1559	0.1984	0.3479	0.2932	0.2817

	Yelp					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0154	0.0323	0.0346	0.0340	0.0388	0.0474
0.1	0.0161	0.0344	0.0364	0.0352	0.0414	0.0506
0.2	0.0169	0.0345	0.0376	0.0376	0.0418	0.0517
0.3	0.0161	0.0349	0.0381	0.0374	0.0423	0.0525
0.4	0.0193	0.0377	0.0401	0.0412	0.0463	0.0562
0.5	0.0201	0.0391	0.0401	0.0449	0.0483	0.0581
0.6	0.0201	0.0388	0.0411	0.0455	0.0478	0.0579
0.7	0.0187	0.0380	0.0406	0.0431	0.0465	0.0566
0.8	0.0188	0.0381	0.0406	0.0420	0.0462	0.0565
0.9	0.0189	0.0367	0.0395	0.0423	0.0454	0.0557
1	0.0181	0.0362	0.0385	0.0407	0.0442	0.0539

	Pinterest					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0127	0.0331	0.0401	0.0383	0.0364	0.0485
0.1	0.0151	0.0373	0.0443	0.0450	0.0415	0.0545
0.2	0.0171	0.0396	0.0471	0.0510	0.0446	0.0584
0.3	0.0175	0.0420	0.0493	0.0525	0.0471	0.0611
0.4	0.0182	0.0438	0.0514	0.0550	0.0492	0.0637
0.5	0.0200	0.0471	0.0542	0.0604	0.0532	0.0678
0.6	0.0200	0.0474	0.0542	0.0604	0.0532	0.0663
0.7	0.0205	0.0494	0.0564	0.0616	0.0556	0.0706
0.8	0.0210	0.0496	0.0571	0.0638	0.0561	0.0714
0.9	0.0189	0.0473	0.0553	0.0576	0.0529	0.0681
1	0.0200	0.0469	0.0538	0.0597	0.0520	0.0663

Table 5.11: Performance of the JoVA-Hinge under F1@k and NDCG@k metrics with different γ values. The purple shows the best values.

Pinterest, JoVA-Hinge with $\gamma = 0.8$ achieved the best results. JoVA-Hinge with $\gamma = 0.8$ outperforms JoVA-Hinge with $\gamma = 0.5$ up to 5.63%. The best value for γ depends on the dataset, the number of users, the number of items, the number of ratings provided for each item, and the number of ratings given by users. If the implicit feedback matrix includes more enriched item vectors than user vectors, lager γ is a more suitable choice and vice versa.

In general, the results of JoVA-Hinge with $\gamma = 0.5$ are the best or comparable to other γ values' performance. Introducing a hyper-parameter to a model increases the training cost (training time), so it is reasonable to use the unweighted average for JoVA. In conclusion, extending unweighted averaging in JoVA-Hinge to the weighted averaging can increase model accuracy with the cost of tuning more hyper-parameters for each dataset.

5.8 Hyper-parameter Sensitivity

In this section, we investigate how we set the margin parameters of λ. Hyper-parameter λ defines how big the margin between positive feedback and random negative samples is possible. Results on different datasets under F1-score and NDCG with various k are reported in Table 5.12. We can see that on three datasets, JoVA-Hinge with $\lambda = 0.15$ either outperforms other values or is comparable. On MovieLens, $\lambda = 0.1$ can improve up to 1.08% (NDCG@1), on Yelp $\lambda = 0$ improves F1-score by 1% and on Pinterest $\lambda = 0.1$ can improve up to 1.15% (NDCG@1). Therefore, this experiment confirms that $\lambda = 0.15$ is an appropriate choice for the margin parameter.

	ML1M					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0622	0.1679	0.2102	0.3707	0.3139	0.300
0.1	0.0629	0.166	0.2107	0.3758	0.3166	0.3012
0.15	0.0624	0.1665	0.2115	0.3718	0.3143	0.3013
0.2	0.0614	0.1645	0.2069	0.3662	0.3106	0.2965
0.3	0.0588	0.1594	0.2037	0.3561	0.3036	0.2910
0.4	0.0550	0.1572	0.2019	0.3396	0.2981	0.2868
0.5	0.0522	0.1515	0.1964	0.3351	0.2908	0.2802
0.6	0.0510	0.1483	0.1938	0.3290	0.2861	0.2758
0.7	0.0504	0.1478	0.1937	0.3335	0.2891	0.2774
0.8	0.0496	0.1447	0.1896	0.3333	0.2857	0.2731
0.9	0.0456	0.1427	0.1912	0.3182	0.2785	0.2702
1	0.0453	0.1449	0.1911	0.3139	0.2806	0.2700

	Yelp					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0190	0.0377	0.0399	0.0420	0.0460	0.0559
0.1	0.0189	0.0376	0.0405	0.0430	0.0460	0.0565
0.15	0.0201	0.0391	0.0401	0.0449	0.0483	0.0581
0.2	0.0190	0.0380	0.0399	0.0431	0.0463	0.0559
0.3	0.0182	0.0361	0.0395	0.0409	0.0445	0.0550
0.4	0.0189	0.0366	0.0401	0.0414	0.0450	0.0558
0.5	0.0182	0.0384	0.0402	0.0409	0.0464	0.0559
0.6	0.018	0.379	0.0402	0.0405	0.0458	0.0558
0.7	0.0186	0.0377	0.0402	0.0420	0.0458	0.0561
0.8	0.0176	0.0375	0.0402	0.0392	0.0452	0.0553
0.9	0.0178	0.0374	0.0402	0.0408	0.0456	0.0559
1	0.0193	0.0370	0.0401	0.0435	0.04590	0.0562

	Pinterest					
	F1-score			NDCG		
	@1	@5	@10	@1	@5	@10
0	0.0202	0.0468	0.0541	0.0611	0.0530	0.0674
0.1	0.0202	0.0469	0.0541	0.0609	0.0532	0.0674
0.15	0.0200	0.0471	0.0542	0.0604	0.0532	0.0678
0.2	0.0199	0.0465	0.0534	0.0601	0.0525	0.0671
0.3	0.0196	0.0461	0.0538	0.0589	0.0520	0.0670
0.4	0.0193	0.0460	0.0534	0.0584	0.0519	0.0665
0.5	0.0192	0.0454	0.0531	0.0581	0.0511	0.0657
0.6	0.0186	0.0454	0.0533	0.0563	0.0508	0.0657
0.7	0.0188	0.0454	0.0531	0.0568	0.0509	0.0657
0.8	0.0186	0.0449	0.0531	0.0561	0.0504	0.0654
0.9	0.0173	0.0450	0.0530	0.0528	0.0498	0.0648
1	0.0186	0.0452	0.0531	0.0565	0.0507	0.0655

Table 5.12: Performance of the JoVA-Hinge under F1@k and NDCG@k metrics with different λ values. The purple shows the best values.

5.9 Summary

In this chapter, we showed that JoVA-Hinge could perform better than a broad set of state-of-the-art collaborative filtering methods through extensive experiments. Our proposed models have two main strong points: (1) they take advantage of variational autoencoders' representation learning power. (2) they can learn both user-user and item-item correlations. User and item VAEs together can extract more and complementary information from the user/item interactions data. The generative process of VAEs can also help the JoVA and JoVA-Hinge perform better predictions than their similar model with classic autoencoders (JCA). Our experimental results on the three real-world datasets show that JoVA-Hinge with hinge loss usually can perform better than JoVA (see section 5.3). Another important point from our experiments is that the ensemble of VAEs improves the performance of the ensemble of autoencoders (JCA), especially in the sparse dataset (see section 5.2). For this, we need to use VAEs for sparse data. The ensemble of autoencoders is often not powerful enough to learn good representations from sparse data.

Chapter 6

Conclusions

In this chapter, we present conclusions and directions for future work. The section 6.1 describes the conclusions, and the section 6.2 illustrates some directions for future work.

6.1 Overall Summary

In this thesis, we have introduced a joint variational autoencoder (JoVA) for top-k recommendation with implicit feedback. JoVA, as an ensemble of two VAEs, jointly learns user-user and item-item correlations simultaneously. A variant of JoVA, referred to as JoVA-Hinge, includes pairwise ranking loss in addition to VAE's loss to specialize JoVA further for recommendation with implicit feedback. We have investigated a wide variety of hyperparameter settings on different datasets. Our empirical experiments on multiple real-world datasets show that JoVA-Hinge advances the recommendation accuracy compared to a broad set of state-of-the-art methods, under various evaluation metrics. Additionally, our experiments confirm that JoVA-Hinge

outperforms other baselines across all users regardless of their numbers of observed interactions.

6.2 Future Directions

Our JoVA (JoVA-Hinge) model provides a robust framework for the broader investigation of the ensemble of VAEs equipped with pairwise ranking loss in recommender systems or even possibly in other applications (e.g., vision, robotics, etc.). Our models (JoVA and JoVA-Hinge) are generic, which can be extended to various types of recommendation tasks. Some problems that the research of this thesis can continue are presented in the following subsections.

6.2.1 Incorporating Side information

In future work, we are interested in incorporating external information into the framework for improving the recommendation quality. External information includes user and item features (e.g., descriptions, demographic information, etc.), side information (e.g., social networks), context (e.g., time, location, etc.) [21].

6.2.2 Dynamic Environments

In this thesis, we focus on modeling user interests' in stable (non-dynamic) environments. However, in real-world scenarios, users' interests and items' characteristics change over time. In the future, we are interested in extending JoVA to model non-stationary user preferences. In the future, we can adopt reinforcement learning [105]. Reinforcement learning is a powerful approach for sequential decision-making prob-

lems in dynamic environments.

6.2.3 Scalability

Improving the scalability of the JoVA framework is also an important future direction. While our system's time performance is comparable to the other prior work systems, we plan further to improve the scalability of JoVA through distributed computing techniques. The main reason for the long training time is the massive number of model parameters. In future work, we can work on variants of models that reduce the number of parameters.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[3] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, Springer, 2007, pp. 548–562.

[4] X. Amatriain, J. M. Pujol, and N. Oliver, "I like it... I like it not: Evaluating user ratings noise in recommender systems," in *Seventeenth International Conference on User Modeling, Adaptation, and Personalization*, Trento, Italy, 2009, pp. 247–258.

[5] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 143–152.

[6] S. Hacker and L. Von Ahn, "Matchin: Eliciting user preferences with an online game," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1207–1216.

[7] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*, Ieee, 2008, pp. 263–272.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[9] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in neural information processing systems*, 2013, pp. 2643–2651.

[10] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[11] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

[12] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[13] B. Song, X. Yang, Y. Cao, and C. Xu, "Neural collaborative ranking," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1353–1362.

[14] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 153–162.

[15] Z. Zhu, J. Wang, and J. Caverlee, "Improving top-k recommendation via jointcollaborative autoencoders," in *The World Wide Web Conference*, 2019, pp. 3483–3482.

[16] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 689–698.

[17] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 305–314.

[18] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[19] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[20] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*, Springer, 2015, pp. 77–118.

[21] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016, vol. 1.

[22] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*, Elsevier, 1995, pp. 331–339.

[23] X. Li, M. Cheung, and J. She, "Connection discovery using shared images by gaussian relational topic model," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 931–936.

[24] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.

[25] R. Li and X. Guo, "An improved algorithm to term weighting in text classification," in *2010 International Conference on Multimedia Technology*, IEEE, 2010, pp. 1–3.

[26] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[27] N. X. Bach, N. Do Hai, and T. M. Phuong, "Personalized recommendation of stories for commenting in forum-based social media," *Information Sciences*, vol. 352, pp. 48–60, 2016.

[28] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

[29] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[30] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[31] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Icml*, 2011.

[32] S. Ghosh, *Simple matrix factorization example on the movielens dataset using pyspark*.

[33] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[34] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 325–334.

[35] D. D. Le and H. W. Lauw, "Indexable bayesian personalized ranking for efficient top-k recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1389–1398.

[36] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 549–558.

[37] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *2008 Eighth IEEE International Conference on Data Mining*, IEEE, 2008, pp. 502–511.

[38] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[39] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang, "Lambdafm: Learning optimal ranking with factorization machines using lambda surrogates," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 227–236.

[40] W. Pan and L. Chen, "Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[41] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "Cofi rank-maximum margin matrix factorization for collaborative ranking," in *Advances in neural information processing systems*, 2008, pp. 1593–1600.

[42] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Advances in neural information processing systems*, 2005, pp. 1329–1336.

[43] N. N. Liu and Q. Yang, "Eigenrank: A ranking-oriented approach to collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 83–90.

[44] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio, "Sprank: Semantic path-based ranking for top-n recommendations using linked open data," *ACM*

Transactions on Intelligent Systems and Technology (TIST), vol. 8, no. 1, pp. 1–34, 2016.

[45] G. Takács and D. Tikk, "Alternating least squares for personalized ranking," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 83–90.

[46] W. Pan, L. Chen, and Z. Ming, "Personalized recommendation with implicit feedback via learning pairwise preferences over item-sets," *Knowledge and Information Systems*, vol. 58, no. 2, pp. 295–318, 2019.

[47] R. He and J. McAuley, "Vbpr: Visual bayesian personalized ranking from implicit feedback," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[48] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, "Neurec: On nonlinear transformation for personalized ranking," *arXiv preprint arXiv:1805.03002*, 2018.

[49] B. Song, X. Yang, Y. Cao, and C. Xu, "Neural collaborative ranking," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1353–1362.

[50] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 89–96.

[51] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 278–288.

[52] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2333–2338.

[53] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

[54] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 811–820.

[55] S. Zhang, L. Yao, and X. Xu, "Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 957–960.

[56] H. Ying, L. Chen, Y. Xiong, and J. Wu, "Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback," in *Pacific-asia conference on knowledge discovery and data mining*, Springer, 2016, pp. 555–567.

[57] H. Liu, J. Wen, L. Jing, and J. Yu, "Deep generative ranking for personalized recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 34–42.

[58] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko, "Recvae: A new variational autoencoder for top-n recommendations with implicit feedback," *arXiv preprint arXiv:1912.11160*, 2019.

[59] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[60] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[61] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T.-S. Chua, "Mmgcn: Multimodal graph convolution network for personalized recommendation of micro-video," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1437–1445.

[62] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1289–1298.

[63] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," *arXiv preprint arXiv:2006.15516*, 2020.

[64] A. Pujahari and V. Padmanabhan, "Group recommender systems: Combining user-user and item-item collaborative filtering techniques," in *2015 International Conference on Information Technology (ICIT)*, IEEE, 2015, pp. 148–152.

[65] J. Gorla, N. Lathia, S. Robertson, and J. Wang, "Probabilistic group recommendation via information matching," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 495–504.

[66] A. Salehi-Abari and C. Boutilier, "Preference-oriented social networks: Group recommendation and inference," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 35–42.

[67] ——, "Ranking networks," in *NIPS-13 Workshop on Frontiers of Network Analysis: Methods, Models, and Applications*, 2013.

[68] L. Baltrunas, T. Makcinskas, and F. Ricci, "Group recommendations with rank aggregation and collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 119–126.

[69] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*, Springer, 2011, pp. 257–297.

[70] I. Cantador, I. Fernández-Tobıas, S. Berkovsky, and P. Cremonesi, "Cross-domain recommender systems," in *Recommender Systems Handbook*, Springer, 2015, pp. 919–959.

[71] I. Cantador and P. Cremonesi, "Tutorial on cross-domain recommender systems," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ACM, 2014, pp. 401–402.

[72] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction.," in *IJCAI*, vol. 9, 2009, pp. 2052–2057.

[73] S. Sahebi and P. Brusilovsky, "It takes two to tango: An exploration of domain pairs for cross-domain collaborative filtering," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 131–138.

[74] F. Yuan, L. Yao, and B. Benatallah, "Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns," *arXiv preprint arXiv:1905.10760*, 2019.

[75] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 667–676.

[76] S. Ahangama and D. C.-C. Poo, "Latent user linking for collaborative cross domain recommendation," *arXiv preprint arXiv:1908.06583*, 2019.

[77] D. Rafailidis and G. Weiss, "Adaptive deep learning of cross-domain loss in collaborative filtering," *arXiv preprint arXiv:1907.01645*, 2019.

[78] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, ACM, 2011, pp. 287–296.

[79] A. J. Chaney, D. M. Blei, and T. Eliassi-Rad, "A probabilistic model for using social networks in personalized item recommendation," in *Proceedings of the 9th ACM Conference on Recommender Systems*, ACM, 2015, pp. 43–50.

[80] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical poisson factorization.," in *UAI*, 2015, pp. 326–335.

[81] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: Social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 931–940.

[82] Y. Meng, G. Chen, J. Li, and S. Zhang, "Psrec: Social recommendation with pseudo ratings," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ACM, 2018, pp. 397–401.

[83] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[84] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 135–142.

[85] B. Y. Y. L. D. Liu and J. Liu, "Social collaborative filtering by trust," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IEEE*.

[86] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings.," in *Aaai*, vol. 15, 2015, pp. 123–125.

[87] P. Massa and P. Avesani, "Trust metrics on controversial users: Balancing between tyranny of the majority," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 3, no. 1, pp. 39–64, 2007.

[88] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri, "Feedback effects between similarity and social influence in online communities," in *Pro-*

ceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 160–168.

[89] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999.

[90] X. Amatriain, J. M. Pujol, N. Tintarev, and N. Oliver, "Rate it again: Increasing recommendation accuracy by user re-rating," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 173–180.

[91] R. Y. Toledo, Y. C. Mota, and L. Martınez, "Correcting noisy ratings in collaborative recommender systems," *Knowledge-Based Systems*, vol. 76, pp. 96–108, 2015.

[92] A. Salehi-Abari and K. Larson, "Group recommendation with noisy subjective preferences," *Computational Intelligence*, 2020.

[93] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, *Handbook of computational social choice*. Cambridge University Press, 2016.

[94] P. K. Diederik, M. Welling, *et al.*, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 1, 2014.

[95] T. Spinner, J. Körner, J. Görtler, and O. Deussen, "Towards an interpretable latent space: An intuitive comparison of autoencoders with variational autoencoders," in *IEEE VIS 2018*, 2018.

[96] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[97] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[98] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.

[99] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[100] T. Yao, T. Mei, and Y. Rui, "Highlight detection with pairwise deep ranking for first-person video summarization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 982–990.

[101] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[102] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4274–4282.

[103] J. Chen, C. Wang, S. Zhou, Q. Shi, J. Chen, Y. Feng, and C. Chen, "Fast adaptively weighted matrix factorization for recommendation with implicit feedback," *arXiv preprint arXiv:2003.01892*, 2020.

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

[105] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.