SEPARATING-PLANE FACTORIZATION MODELS: SCALABLE RECOMMENDATION FROM ONE-CLASS IMPLICIT FEEDBACK

by

Haolan Chen

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

We study the large-scale video recommendation problem based on user viewing logs instead of explicit ratings. As viewing records are implicitly positive samples, existing matrix factorization methods fail to generate discriminative recommendations based on such one-class data. We propose a scalable approach called separating-plane matrix factorization (SPMF) to make effective recommendations based on one-class implicit feedback, with a learning complexity only comparable to matrix factorization. With extensive offline evaluation in Tencent Data Warehouse (TDW) based on big data, we show that our approach outperforms a wide range of state-of-the-art methods. We also deployed our system online to test with real users in Tencent QQ Browser mobile app. Results show that our approach can increase the video click through rate by 23% over implicit-feedback collaborative filtering (IFCF), a scheme implemented in Spark's MLlib.

Acknowledgments

I would like to thank all the people who contributed in some way to the work described in this thesis. First and foremost, I am deeply indebted to my academic advisors, Dr. Di Niu and Dr. Masoud Ardakani. Their guidance and supervision help me to learn how to be a good researcher and pursue an academic career. Their willingness to discussion helped me through two important years of my life. Additionally, I would like to thank my committee members Dr. Scott Dick and Dr. Linglong Kong for their interest in my work.

I would also like to thank Dr. Kunfeng Lai. He is one of the cooperator in my research work on video recommendation. Thanks for the discussions and advices during the project, and I learned a lot through this research work and I greatly benefited from his keen scientific insight and creativity.

Finally, I would like to acknowledge friends and family who supported me during my time here. First and foremost I would like to thank my father and mother and other family members for their constant love and support. Additionally, I am lucky to have met lots of good friends here and have a happy life. Thanks for their friendship and unyielding support!

Table of Contents

1 Introduction **1**

2 Preliminaries and Background **4**

2.1 Collaborative Filtering 5

2.2 Matrix Factorization 5

2.3 Implicit Feedback and One-Class Problem 7

2.3.1 Imputation-based Matrix Factorization 8

2.3.2 BPR and Pairwise Methods 10

2.4 Restricted Boltzmann Machine 11

3 Separating-Plane Factorization Model **14**

3.1 Separating-Plane Matrix Factorization 15

3.2 Insights and Discussions 18

3.3 Separating-Plane Factorization Machine 20

4 Implementation **22**

4.1 Practical Performance Optimizations 22

4.1.1 Empirical Preference Measures 22

4.1.2 Weighted SPMF and SPFM 24

4.1.3 Time-Decayed Confidence Levels 26

4.2 Implementation and Deployment 26

5 Performance Evaluation **32**

5.1 Single-Machine Tests 33

5.2 Offline Tests on Distributed Clusters 36
5.3 Online A/B Tests 39

6 Concluding Remarks **47**

References **48**

List of Tables

5.1 MPR for Tencent small dataset and Movielens dataset in single-machine tests. 33
5.2 Recall@20 of algorithms using factorization with 20 and 40 features on Movielens dataset in single-machine tests. 35
5.3 Effect of different empirical preference measures in large-scale offline evaluation with SPFM, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. 38
5.4 Results of Online A/B Tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 39
5.5 Analysis of variance (ANOVA) of the 4 methods in Online A/B Tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. . . 45
5.6 Tukey's multiple comparison test for the Online A/B Tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 46

5.7 The results of Online A/B Tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 46

List of Figures

2.1 Restricted Boltzmann Machine: a stochastic neural network 11

3.1 Traditional matrix factorization: $\hat{p}_{ui}$ (and p_{ui} for training samples) vs. $\mathbf{v}_i$ for a certain user u. $\sum_{i=1}^{n} \mathbf{v}_i \neq 0$. 18

3.2 SPMF: $\hat{p}_{ui}$ (and p_{ui} for training samples) vs. $\mathbf{v}'_i$ for a certain user u. Zero hyperplane enlarges the range of $\hat{p}_{ui}$. $\sum_{i=1}^{n} \mathbf{v}'_i = 0$. 20

4.1 Average watching times of videos in the sampled dataset from Tencent, containing 165724 users and 46578 videos. 23

4.2 Average $t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$ vs. t^{i}_{total} in the sampled dataset from Tencent, containing 165724 users and 46578 videos. 23

4.3 CDF of video watching ratio in the sampled dataset from Tencent, containing 165724 users and 46578 videos. 24

4.4 Total numbers of clicks of different videos in the small dataset collected from Tencent, containing 165,724 users and 46,578 videos. . 25

4.5 Estimated preference nodes receive messages from connected feature nodes and calculate residual errors by comparing with sample rating nodes in parallel. 28

4.6 Feature nodes receive residual errors from connected estimated preference nodes and update the weights in parallel. 29

4.7 Workflow of our recommender system running on TDW platform. . 30

4.8 Density of known samples in the small dataset collected from Tencent, containing 165,724 users and 46,578 videos. 30

5.1 Receiver operating characteristic (ROC) curve for MovieLens data in single-machine tests, the publicly available MovieLens dataset contains explicit ratings (from 0.5 to 5) between 668 users and 10325 videos. 35
5.2 MPR of different optimization options as the dimension of feature vector $\mathbf{v}$ varies @ Tencent's small dataset, The Tencent dataset contains 2604572 watching behavior records of the form (3.1) collected in 40 days, involving 165724 users and 46578 videos. 36
5.3 Large-scale offline test: MPR vs. the dimension of latent factor vectors, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. 37
5.4 Large-scale offline test: MPR vs. the number of training iterations, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. 37
5.5 Effect of different empirical preference measures in large-scale offline evaluation with SPFM, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. 38
5.6 Effect of different Time-Decayed Confidence Level measures in large-scale offline evaluation with SPFM, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. 39
5.7 The total number of clicks on Sunday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 40

5.8 The total number of clicks on Monday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 40
5.9 Number of clicks on the recommended videos on Sunday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 42
5.10 The number of clicks on the recommended videos on Monday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 42
5.11 The number of clicks on the recommended videos in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. . . 43
5.12 The number of users with different numbers of clicks on Sunday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 43
5.13 Number of users with different numbers of clicks on Monday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 44

5.14 Reflectance (numbers of clicks per hour) of 4 groups in online A/B tests and the corresponding boxplot. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users. 45

List of Abbreviations

Acronyms	Definition
ALS	Alternative Least Square
AMAN	All Missing As Negative
AMAU	All Missing As Unknown
BPR	Bayesian Personalized Ranking
CF	Collaborative Filtering
CTR	Click-Through Rate
FM	Factorizing Machines
IBCF	Item-Based Collaborative Filtering
IFCF	Implicit-Feedback Collaborative Filtering
MCMC	Markov Chain Monte Carlo
MF	Matrix Factorization
MPR	Mean Percentile Ranking
MSE	Mean Squared Error
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SPFM	Separating-Plane Factorization Machine
SPMF	Separating-Plane Matrix Factorization
TDW	Tencent distributed Data Warehouse

Chapter 1

Introduction

Collaborative filtering (CF) [1]–[3] has become the *de facto* standard in video recommendation. While collaborative filtering techniques are designed to handle explicit ratings, in reality, applications of recommendations made from explicit feedback are limited due to several reasons.

First, although many systems allow users to rate videos in a 0-5 scale, yet the rating data are often extremely sparse, simply because users are reluctant to take further actions to rate videos they have watched. The videos that users have rated are typically those for which users had a strong positive or negative feeling. And for an average video, users may not even bother to give any feedback. This fact implies that there is a survivor bias in the explicit rating data and it is hard to collect non-extreme feedbacks. *Second*, sometimes explicit ratings cannot reflect a user's true preference. The assumption that human users know themselves perfectly is not always true. During an interview in 2013, an algorithm scientist at Netflix has pointed out [4] that their users tend to give high ratings to movies of good quality, while in daily life, they are more likely to watch an average popcorn movie. In other words, there is a gap between true preferences and feedback collected from the artificially designed yet overly simplified 0-5 rating system.

To overcome the limitation of explicit ratings, we can leverage a much larger amount of *implicit* data for recommendation, such as viewing history, recording each video that a user has watched, for how long it was watched, the time that

the action took place, as well as other context information. Such implicit data are usually largely available in system logs, and collecting them does not require any further actions from users.

However, recommendation based on implicit feedback has led to the one-class problem [5]–[8], since each viewing record represents a positive sample of some kind. The reason is that a watching action itself implies some degree of preference by the user over other videos that are available to her. Therefore, all the implicit feedback records collected are biased towards positive samples. When applying collaborative filtering to such one-class data, the predictions will concentrate in the range defined by positive samples and thus have no discriminative power. In addition, unlike numerical rating scores which are used to indicate different levels of preferences, implicit feedback data is less clean and often ill-shaped, i.e., there is no direct correspondence between a watching record and the empirical measure of preference. For example, watching a video for a longer period of time than another video does not necessarily imply a preference of the first video.

Existing methods for collaborative filtering based on one-class implicit data are either based on imputation [5], [6], [8] or Bayesian Personalized Ranking (BPR) [7]. However, imputation-based methods rely on selecting some unknown samples to impute with negative values, and thus may introduce artificial errors. On the other hand, BPR relies on minimizing a pairwise preference loss function, whose complexity could be n times the complexity of the original matrix factorization, n being the number of videos in the system. This prevents BPR from being deployed at a large scale involving a large number of videos.

In this thesis, we propose a novel model called Separating-Plane Matrix Factorization (SPMF) to approach the recommendation problem with implicit positive feedback at a large scale, with several unique strengths. *First*, SPMF does not rely on imputation to generate negative samples. Yet its discriminative power automatically comes from a separating hyperplane that we have introduced into the matrix factorization model. *Second*, we propose a simple yet effective projected stochastic gradient descent (SGD) algorithm to solve SPMF, which has a similar learning

complexity as the original matrix factorization [2]. Thus, SPMF is scalable to the huge problem sizes in reality. *Third*, SPMF can easily be extended to incorporate context information, confidence levels and any other optimizing utilities that are applicable to matrix factorization.

We implemented SPMF in Spark GraphX and deployed the system on Tencent distributed Data Warehouse (TDW) (Tencent's big data processing platform) at a large scale with various performance optimizations. Our recommender system is tested in the Tencent QQ Browser mobile app, which provides a video content search and aggregation service in its app front page. We performed large-scale offline evaluation in comparison to a number of state-of-the-art algorithms, based on the video viewing logs collected from the video aggregation service of QQ Browser mobile app with user consent in a 40-day period, involving 12.5 million users and $100,000$ videos. To show the discriminative power of our method on one-class data, we have also conducted fair comparisons to existing methods based on a public dataset, the MovieLens data, retaining only positive ratings. In all cases, SPMF and its extension Separating-Plane Factorization Machine (SPFM) incorporating context information significantly outperform both imputation-based methods and BPR while incurring a low computation cost.

We also performed online A/B tests on more than $50,000$ real users of Tencent QQ Browser mobile app, split into 4 groups, each receiving recommendations made from SPMF, SPFM, item-based CF [9] or implicit-feedback CF [5] (an imputation-based matrix factorization scheme). The results reveal that our models lead to a higher number of clicks, click-through rate (CTR) and recommendation precision in the real-world scenario.

Chapter 2

Preliminaries and Background

Collaborative filtering (CF) infers a user's preference for each item based on observed interactions including explicit ratings and implicit feedback, e.g., views, purchases, etc. CF has become the state of the art in recommendation systems [3], since it learns from past user-item interactions and can address data aspects that are often hard to model via explicit profiles and content filtering.

We introduce some preliminaries to formally define the collaborative filtering problem and will use them throughout the thesis. Suppose there are m users and n items. Let $P \in \mathbb{R}^{m \times n}$ denote the *preference* matrix between all the users and items, where p_{ui} represents the preference of user u for item i. Let Ω denote the *observed set*, which contains all (u, i) pairs that have interacted and thus have an observed empirical preference value p_{ui}, which may be determined in a number of ways. For example, if explicit ratings are available, the observed p_{ui} is indicated by the numerical rating of user u on item i; if only implicit feedback is available, the observed p_{ui} can be indirectly measured from user behavior logs, e.g., the time that user u has spent watching video i, the number of clicks user u has made on an item i, etc. The collaborative filtering (CF) problem is to infer all unobserved preference values p_{ui} based on the training samples $\{p_{ui} | (u, i) \in \Omega\}$. Recommendations can then be made to each user u based on the estimated $\hat{p}_{ui}$.

2.1 Collaborative Filtering

Collaborative filtering can be divided into two classes, the memory-based CF and model-based CF [3], [10]. Memory-based CF techniques such as item-based CF [9] and user-based CF [11] are based on the idea that similar entities share similar behavior models and thus users or items can be grouped by checking the past interactions among users and items. Item-based CF, for example, simply makes recommendation based on the similarity between consumed items and candidate items. Such a similarity is measured by the number of users that have consumed both items. And the same applies to other memory-based algorithms, such as user-based CF and content-based CF [3].

For example, consider the movie Saving Private Ryan. Its neighbors might include war movies, Spielberg movies, and Tom Hanks movies, among others. To predict a particular user's rating for Saving Private Ryan, we would look for its neighbors, namely, other movies that are similar to Saving Private Ryan and that this user has ever rated. Then the user's rating for Saving Private Ryan is predicted as a weighted sum of the user's ratings for its neighboring movies. Likewise, the user-oriented approach identifies like-minded users who can complement each other's ratings.

On other hand, model-based CF techniques mainly include matrix factorization [2] and other higher-order factorization models, which are also the state-of-the-art techniques to solve collaborative filtering problems. We will describe factorization models in detail in the following.

2.2 Matrix Factorization

Matrix Factorization attempts to characterize both items and users via vectors of latent factors, which can not only measure obvious dimensions such as genre of a movie, but can also capture hidden and even uninterpretable attributes [3]. In fact, MF has become a dominant methodology for collaborative filtering. Experience

with the highly visible Netflix Prize data has shown that MF models deliver higher accuracy than earlier neighborhood methods [12], such as item-based CF [9] and user-based CF [13], which make recommendations based on the preferences of similar items or users. For example, the winning team of Netflix Prize used hundreds of different predictor sets, the majority of which were variants of matrix factorization models [14].

MF estimates the preference of user u for item i as [2]

$$\hat{p}_{ui} = b + w_u + w_i + \mathbf{u}_u^\mathsf{T}\mathbf{v}_i \tag{2.1}$$

where $\mathbf{u}_u \in \mathbb{R}^k$ and $\mathbf{v}_i \in \mathbb{R}^k$ are k-dimensional vectors of user and item latent factors, respectively. The term b is a global bias, while w_u represents the user bias and w_i represents the item bias. The inner product $\mathbf{u}_u^\mathsf{T}\mathbf{v}_i$ measures the match between the latent factors of user u and item i.

MF minimizes the mean squared error (MSE) of the estimation from (2.1) over the observed set, i.e.,

$$\underset{\{\mathbf{u}_u\},\{\mathbf{v}_i\}}{\text{minimize}} \sum_{(u,i)\in\Omega} (p_{ui} - \mathbf{u}_u^\mathsf{T}\mathbf{v}_i)^2 + \lambda\bigg(\sum_u \|\mathbf{u}_u\|^2 + \sum_i \|\mathbf{v}_i\|^2\bigg), \tag{2.2}$$

where the bias terms are omitted for brevity. The MF problem (2.2) can be conveniently solved by stochastic gradient descent (SGD) [2] or alternating least squares (ALS) [2]. Many efforts have been made on the further development of matrix factorization algorithms to improve performance, e.g., Markov chain Monte Carlo (MCMC) methods [15], as well as to incorporate more context information from diverse data sources [13].

Factorization Machine (FM) [16] has moved one step further by allowing the factorization of n-way interactions, e.g., between user and item, user and time, and item and time. Moreover, FM could be solved with a complexity of $O(k|\Omega|l)$ instead of $O(k|\Omega|l^2)$, in which k is the length of latent factor vectors and l is the number of features used (e.g., user, video and time) [16]. FM also performed well

in competitions like KDD Cup [17], [18].

2.3 Implicit Feedback and One-Class Problem

Although MF-based recommenders yield good performance for explicit ratings in a number of real-world systems and competitions, explicit numerical ratings are often unavailable in reality [5], [19], either because users are reluctant to take actions to rate products, or because the setup of the system has not enabled collection of explicit feedback. In these cases, the system can still leverage vast amounts of implicit user behavior logs for recommendation, such as views, clicks, purchases, likes, shares etc. For example, in the video aggregation service of Tencent QQ Browser mobile app, most users seldom rate any video they have watched. But we may still make recommendations based on user watching history through collaborative filtering.

However, a major challenge with implicit feedback is that nearly all types of user actions performed on an item indicate a symbol of preference; if a user has watched a video, it means she/he has at least shown some kind of interest or curiosity in it, since otherwise she/he would have never clicked that video at all. In other words, the implicit feedback data is most likely one-class and contains positive samples only. If the MF model above were applied with only positive training samples, the estimated preference matrix would be filled with positive values in the range of training samples, and recommendations made this way do not have discriminative power.

Another challenge posed by implicit feedback is that unlike explicit ratings, the numerical values of implicit feedback, e.g., the time length a user has viewed a video, the number of clicks on a product by a user, etc., do not directly translate into preference values p_{ui} [5]. And a direct comparison of such values across different items or different users is usually meaningless.

Existing methods that can handle implicit feedback mainly fall into two categories, i.e., *imputation-based methods* and *bayesian personalized ranking* (BPR).

2.3.1 Imputation-based Matrix Factorization

The main idea of imputation is to convert the one-class data to a balanced dataset by artificially assigning values to some unobserved preferences. In [6], two extreme cases of imputation strategies are described, namely, *all missing as negative* (AMAN) and *all missing as unknown* (AMAU), followed by other imputation strategies [8], [20] proposed subsequently.

Furthermore, the notion of *confidence levels* has been introduced in [5] to strike a balance between AMAN and AMAU, which brings additional benefits. We will call this method [5] *implicit-feedback collaborative filtering* (IFCF). Specifically, in IFCF [5], for all observed pairs $(u, i) \in \Omega$ where user u has ever watched video i, $p_{ui} = 1$, while all unobserved p_{ui} are regarded as *negative samples* with an imputed value $p_{ui} = 0$. Then, the following regression is applied onto the entire imputed dataset:

$$\underset{\{\mathbf{u}_u\},\{\mathbf{v}_i\}}{\text{minimize}} \sum_{(u,i)} c_{ui}(p_{ui} - \mathbf{u}_u^{\mathsf{T}}\mathbf{v}_i)^2 + \lambda\left(\sum_u \|\mathbf{u}_u\|^2 + \sum_i \|\mathbf{v}_i\|^2\right) \tag{2.3}$$

where a weight $c_{ui} > 0$ is used to measure the confidence we have in both observed and imputed p_{ui}. For example, c_{ui} may be determined from the number of times user u has watched video i; the lower the frequency, the lower the confidence c_{ui}. Naturally, the observed positive samples are given a relatively high confidence c_{ui}, whereas the unknown imputed entries (that are only suspected to be negative) are given a low confidence c_{ui}. In fact, IFCF [5] has been highly cited and implemented in the official Spark MLlib [21].

Various other imputation-based methods similar to [5] have been proposed for recommendation systems based on implicit feedback and have been applied in different applications. The imputation-based method [5] has been applied to cross-platform news and event recommendation in [22]. The same method to impute p_{ui} and to set confidence levels c_{ui} as that in [5] has been adopted in [23], except that [23] applies this factorization method to different user actions, e.g., watching,

clicks, etc., separately to improve the performance. The personality attributes of users are incorporated into the original user latent factors in [24], where the recommendation problem based on implicit feedback is solved with the same imputation-based method as in [5].

Instead of incorporating the confidence level as a multiplicative term, [25] defines a standalone additive term to model the confidence level that evaluates the confidence of labeling an unknown sample as a truly negative sample, while the specific values of the confidence levels are set according to the domain knowledge of operators. Furthermore, in [26], [27], the authors have defined a set of more complex weighting schemes to measure the confidence levels of different types of user actions. And [28] proposes a hybrid model combining user-item interactions with other user and item attributes including geographical neighborhood characteristics, and performs location recommendation also based on the imputation-based methods. Moreover, the GeoMF model is proposed in [29] to incorporate geographical influence into the imputation-based matrix factorization model for point-of-interest recommendation.

However, imputation suffers from some fundamental drawbacks. First, artificially assigning an unobserved (u, i) pair a fixed preference value (e.g., $p_{ui} = 0$) may introduce errors. Although an action of user u on item i indicates preference, yet not taking any action on an item does not necessarily imply dislike; for example, the user might not even be aware of the existence of the item [5] simply because there are so many items in the service. Artificially assigning a low p_{ui} to an unobserved (u, i) pair may introduce errors. Even though there is an adjustable confidence parameter c_{ui}, it still cannot completely offset the negative impact of artificial imputation. Second, it is also hard to fine-tune the large amount of hyper-parameters in imputation models, e.g., the setup of c_{ui} and p_{ui}, and which data to impute.

2.3.2 BPR and Pairwise Methods

Bayesian Personalized Ranking (BPR) [30] utilizes Bayesian inference to train a pairwise ranking model from only positive feedback. This method is based on the assumption that any observed action of a user u on an item i is a symbol of preference and that user u should prefer item i to any other item j on which she/he has not performed an action. Thus, for each user u, BPR creates binary pairwise samples $p_{u,i,j}$ to indicate if user u prefers item i to item j or not, forming a training set involving both positive and negative entries. This idea leads to the minimization of the aggregate estimation loss of pairwise preferences over all triples (u, i, j) in the training set, to discriminate between a small set of preferred items and a large set of remaining items. Leveraging the pairwise relationships between known samples and unknown samples, BPR introduces less imputation errors and has inspired a number of continued studies that use similar pairwise methods [7], [31]–[35].

However, a major drawback of any pairwise methods including BPR is its computational complexity. For example, assume there are m users, n videos, and $|\Omega|$ observed user-video interactions (i.e., with a known preference value p_{ui}), where each user has watched $\omega = |\Omega|/m$ videos *on average*. In traditional MF that minimizes the MSE, the learning complexity is $O(|\Omega|)$ [36] (assuming the dimension k of latent factors is a constant), whereas the complexity of a pairwise method is $O(m\omega(n-\omega)) = O(|\Omega|(n-\omega))$ on average. In our video recommendation task at Tencent, the number of videos watched by each user on average, ω, is typically no more than 10, while the total number of videos n is typically around $100,000$. Thus, the computational time of BPR will be $100,000$ times that of MF on average.

In fact, theoretically speaking, as a pairwise objective sums over all triples (u, i, j) in the training set, its worst-case complexity is $O(|\Omega|^2)$ [36], which is much larger than the worst-case complexity $O(|\Omega|)$ of MF [36]. Due to the overwhelming complexity, learning algorithms for pairwise methods usually rely on sampling, which may lead to slow convergence [7]. Therefore, even if big data parallel computing clusters are used, as the numbers of users and videos scale up to *billions* and

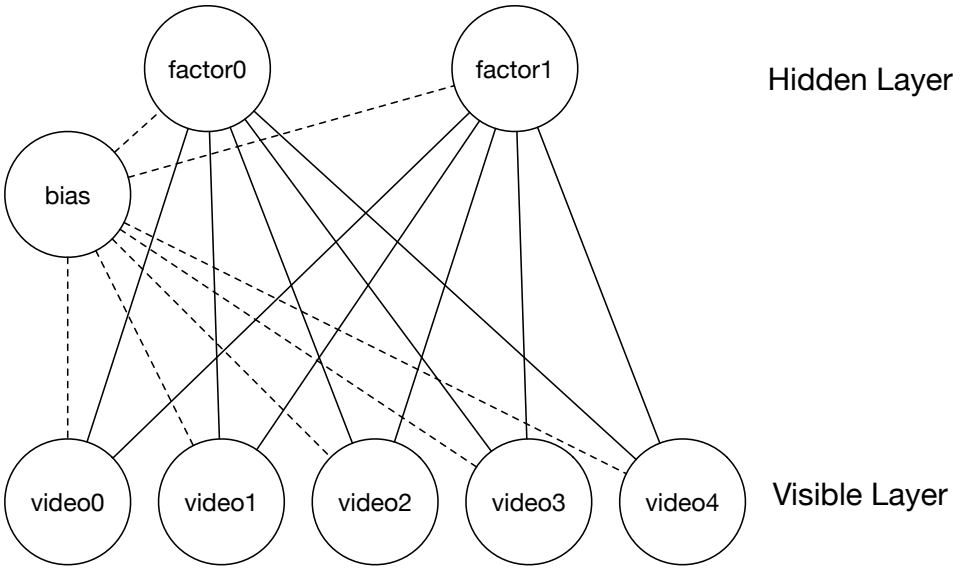

Fig. 2.1. Restricted Boltzmann Machine: a stochastic neural network

millions in a large-scale Internet media service, it is very hard if not infeasible to adopt BPR or other pairwise methods at scale.

2.4 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) is a stochastic neural network which can also be used to solve collaborative filtering problems. RBM was used by one of the winning teams in Netflix Prize Competition [14], and can also handle implicit feedback, which we will show in the evaluation.

An RBM characterizes users and videos with a set of vectors of latent factors. For example, videos such as Harry Potter and Lord of the Rings are very likely to have strong associations with a latent fantasy factor and magic factor, and users who enjoy Madagascar and Kungfu Panda series will probably have associations with a latent animation factor.

As shown in Fig. 2.1, an RBM neural network consists of:

- One layer of visible nodes, which stand for videos ever rated by the users under consideration;
- One layer of hidden nodes, which stand for latent factors that can be learnt from the training dataset;
- Weights between the visible layer and the hidden layer, which stand for the relationship among videos and latent factors;
- A bias node, which stands for the bias of movie popularity.

RBM is technically a binary version of factor analysis. RBM regards hidden nodes as latent factors of interests. By learning from each user's ratings to the videos she has watched, the RBM will obtain a set of weights between videos and latent factors to fit the choices of users. Each pair of a visible node and a hidden node are connected by a corresponding weight. Besides, the bias node is connected to all the visible nodes and hidden nodes to represent the inherent bias of users and videos. For example, even though a user may not be interested in feature films, he/she is very likely to take a look at The Shawshank Redemption, as it is simply too popular. Moreover, giving a rating of 4 out of 5 doesn't necessarily imply a positive attitude of the user, if the user tends to give high ratings to all watched videos.

In general, an RBM works in the way that a user's watched video nodes activate connected hidden nodes which stand for interests. The activated hidden nodes then activate other video nodes in return. In order to illustrate the details, we here assume the neural network has been trained already and show how the nodes are activated for a particular user:

- For each hidden node i, we sum up activation energy collected from all visible nodes (videos) j that node i is connected to to obtain $a_i = \sum_j w_{ij} x_j$, where x_j is the status of visible nodes. Intuitively speaking, as all the connected

nodes j vote to the hidden node i, we sum up their votes and decide whether or not to activate the hidden node i following the procedure shown below.

- We then use a logistic function $\rho_i = \frac{1}{1+e^{-x}}$ to calculate the probability ρ_i that this user has shown interest in hidden node i.
- Next, we update the status of hidden node i based on the computed probability ρ_i, and turn it on with probability ρ_i while turning it off with probability $1-\rho_i$.
- Finally, we activate the visible nodes in return in the same way. The activated visible nodes (videos) represent the videos that the user might be interested in.

In the training procedure, we iterate through the watching/rating records of each user one by one and train the weights with Contrastive Divergence (CD) Gradient learning [37]. It is worth mentioning that there are many variants of RBM, such as multi-categorical RBM [38] and multi-layer RBM [39]. Moreover, instead of training merely the item-wise RBM network, [40], [41] propose to train the item-wise RBM, user-wise RBM and other neighborhood-based RBM networks, combining them to generate recommendations. Furthermore, [42] proposes the Linked RBM for social network link prediction, which is an extended RBM model to learn the preferences of users from both implicit user behavior data and social network data. In [43], the author employs both Deep Bayesian Network (DBN) and RBM to construct a collective DBN which is able to model user preferences in groups. Besides, [44] introduces Social-RBM (SRBM), which incorporates a historical layer into the original visible and hidden layers of the RMB, to learn the correlations between an individual's historical and current features.

There are also other literature about making recommendations with neural networks. A deep neural network is proposed in [45] to extract hidden representations of music and to find similarity among different songs for music recommendation. Similarly, deep factorization models have been used to obtain hidden representations of videos [46]. Moreover, a collaborative topic regression model based on deep neural networks is also used in news recommender systems [47].

Chapter 3

Separating-Plane Factorization Model

In this section, we describe our video recommendation task at Tencent based only on user viewing history, and present our separating-plane factorization models which enable *scalable* recommendations made from positive implicit feedback. Our model has a similar complexity as the traditional matrix factorization and does not need to resort to artificial negative sample imputation.

In our video recommender, each record in the collected logs represents a click-and-watch action of a user u for a video i, in the form of

$$R_{ui} = (u, i, t^{ui}_{\text{watch}}, t^{i}_{\text{total}}, \text{day}, \text{time}, \ldots), \tag{3.1}$$

where t^{ui}_{watch} is the time length (seconds) for which user u watched video i, t^{i}_{total} is the total length of video i, and "day" stands for day of the week when the R_{ui} was collected. Other context information like hour of the day, devices used, video types, can also be easily incorporated into the record.

Recall that p_{ui} is a quantitative preference value we want to estimate between user u and video i, and the observed set of (u, i) pairs is Ω. Note that with explicit feedback, p_{ui} for an observed interaction $(u, i) \in \Omega$ is directly given by the numerical rating of user u on video i. However, with implicit data such as a viewing

record in (3.1), to generate a training sample, we need to empirically convert the record into a quantitative measure of preference p_{ui}. For example, for simplicity, we can let $p_{ui} = t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$ for each observed record (3.1), which represents the fraction of the video i that user u has watched. A similar empirical measure has been used in IFCF [5] for a TV program recommendation task. In Sec. 4.1, we will present a better empirical measure of p_{ui} to generate training samples from observed logs, leading to better recommendation performance in experiments.

Given all the training samples $\{p_{ui}|(u,i) \in \Omega\}$ empirically derived from the collected records, our objective is to estimate all p_{ui}, where an interaction between (u,i) has never been observed yet. As has been mentioned in Sec. **??**, if we directly apply MF to this problem, the estimated preferences $\hat{p}_{ui}$ will all be similar to the training samples (which are all positive since $t^{ui}_{\text{watch}} > 0$), and thus the recommendation made this way will have no discriminative power.

3.1 Separating-Plane Matrix Factorization

Unlike traditional MF which factorizes the preference $\hat{p}_{ui}$ as $\mathbf{u}_u^\mathsf{T}\mathbf{v}_i$, in our separating-plane matrix factorization (SPMF), we model each p_{ui} by

$$\hat{p}_{ui} = \mathbf{u}_u^\mathsf{T}(\mathbf{v}_i - \overline{\mathbf{v}}), \tag{3.2}$$

where $\overline{\mathbf{v}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i$ is the average of all item latent factors. Accordingly, we can learn all the latent factors by solving

$$\begin{aligned}\underset{\{\mathbf{u}_u\},\{\mathbf{v}_i\}}{\text{minimize}} \quad & \sum_{(u,i)\in\Omega}(p_{ui} - \mathbf{u}_u^\mathsf{T}(\mathbf{v}_i - \overline{\mathbf{v}}))^2 \\ & + \lambda\bigg(\sum_u \|\mathbf{u}_u\|^2 + \sum_i \|\mathbf{v}_i - \overline{\mathbf{v}}\|^2\bigg).\end{aligned} \tag{3.3}$$

The rationale of model (3.2) is the following. We assume there is a hidden *absolute interest* value I_{ui} of user u toward video i that is factorized as the inner product between user and item latent factors, i.e., $\hat{I}_{ui} = \mathbf{u}_u^\mathsf{T}\mathbf{v}_i$. However, such absolute in-

terests are better captured by explicit ratings, which are not available in our implicit data of watching history, which only contains watching time information. The key assumption in SPMF is that the observed watching time of a user u on video i can predict her *relative interest* in video i. That is, $p_{ui} = t_{watch}/t_{total}$ is not directly linked to the absolute interest. Rather, it corresponds to the preference of user u for video i *relative to an average video*. Therefore, having learned all the latent factors $\{\mathbf{u}_u\}$ and $\{\mathbf{v}_i\}$, we can estimate any missing preference value p_{ui} as

$$\hat{p}_{ui} = \hat{I}_{ui} - \frac{1}{n}\sum_{i=1}^{n} \hat{I}_{ui} = \mathbf{u}_u^\mathsf{T}\mathbf{v}_i - \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}_u^\mathsf{T}\mathbf{v}_i = \mathbf{u}_u^\mathsf{T}(\mathbf{v}_i - \overline{\mathbf{v}}),$$

which is exactly (3.2), with bias terms omitted for brevity.

Although a user may rate many videos positively (thus with a positive I_{ui}), our assumption is that she will only watch those videos that are *better than average* for her, just because there are too many videos out there in the system (e.g., 100,000 in our training data). For example, for an *average video* i with latent factors $\mathbf{v}_i = \overline{\mathbf{v}}$, SPMF generates a preference value $\hat{p}_{ui} = 0$ according to (3.2), which actually predicts that the user will *never* spend time on this video.

However, applying the stochastic gradient descent (SGD) algorithm directly to problem (3.3) would incur excessive computational complexity. Recall that k is the dimension of latent factor vectors $\mathbf{u}_u$ and $\mathbf{v}_i$. Then, in each iteration, the complexity of SGD is $O(k|\Omega|n)$ [48] in general, since there is a summation of n inner products in the objective function of (3.3) due to the term $\overline{\mathbf{v}}$. As n is the number of all videos and could be around 100,000 in our problem at Tencent, solving (3.3) directly with SGD will introduce tremendous complexity.

We now convert problem (3.3) into an equivalent formulation which we will solve efficiently. Let $\mathbf{v}_i' = \mathbf{v}_i - \overline{\mathbf{v}}$. Substituting $\mathbf{v}_i'$ into (3.3), we obtain the so-called

SPMF problem:

$$\begin{aligned} &\underset{\{\mathbf{u}_u\},\{\mathbf{v'}_i\}}{\text{minimize}} \sum_{(u,i)\in\Omega} (p_{ui} - \mathbf{u}_u^\mathsf{T}\mathbf{v}'_i)^2 + \lambda\bigg(\sum_u \|\mathbf{u}_u\|^2 + \sum_i \|\mathbf{v}'_i\|^2\bigg) \\ &\text{subject to } \sum_{i=1}^{n} \mathbf{v}'_i = 0. \end{aligned} \tag{3.4}$$

Note that problem (3.4) is almost the same as the original matrix factorization problem (2.2), except for the additional constraint $\sum_{i=1}^{n} \mathbf{v}'_i = 0$.

Algorithm 3.1 Projected SGD Algorithm for Problem (3.4)

1: **Input**: The observed set $\{p_{ui}|(u,i) \in \Omega\}$.
2: **Output**: $\{\mathbf{u}_u\}, \{\mathbf{v}'_i\}$ as an approximate solution to (3.4).
3: Initially, $\{\mathbf{u}_u\}, \{\mathbf{v}'_i\}$ are randomly assigned.
4: **for** $k = 1$ to maxIter **do**
5: **for all** observed $(u,i) \in \Omega$ **do**
6: $e_{ui} \leftarrow p_{ui} - \mathbf{u}_u^\mathsf{T}\mathbf{v}'_i$
7: $\mathbf{u}_u \leftarrow \mathbf{u}_u + \eta(e_{ui}\mathbf{v}'_i + \lambda\mathbf{u}_u)$
8: $\mathbf{v}'_i \leftarrow \mathbf{v}'_i + \eta(e_{ui}\mathbf{u}_u + \lambda\mathbf{v}'_i)$
9: Mapping Phase:
10: $\overline{\mathbf{v'}} \leftarrow \sum_{i=1}^{n} \mathbf{v}'_i \;/\; n$
11: **for all** $i = 1, \ldots, n$ **do**
12: $\mathbf{v}'_i \leftarrow \mathbf{v}'_i - \overline{\mathbf{v'}}$

We propose a variant of the *projected SGD* algorithm, as shown in Algorithm 3.1, to solve problem (3.4) efficiently. The only addition to the original SGD are steps 11-14, which map the computed $\mathbf{v}'_i$ in each iteration back to the zero hyperplane $\sum_{i=1}^{n} \mathbf{v}'_i = 0$. To do this, we subtract the average of all $\mathbf{v}'_i$ from each $\mathbf{v}'_i$ in each iteration, which incurs an additional complexity of $O(kn)$ to calculate the average. Therefore, Algorithm 3.1 has a complexity of $O(k(|\Omega| + n))$, which is much lower than applying SGD directly to problem (3.3). Algorithm 3.1 is similar to the projected SGD in [49] which tries to solve a different problem of nonnegative matrix factorization. The difference here is that the projection phase is replaced by mean subtraction.

Solving SPMF has a low complexity $O(|\Omega| + n)$ (when the latent factor dimension k is fixed), which is comparable to the complexity of traditional MF $O(|\Omega|)$

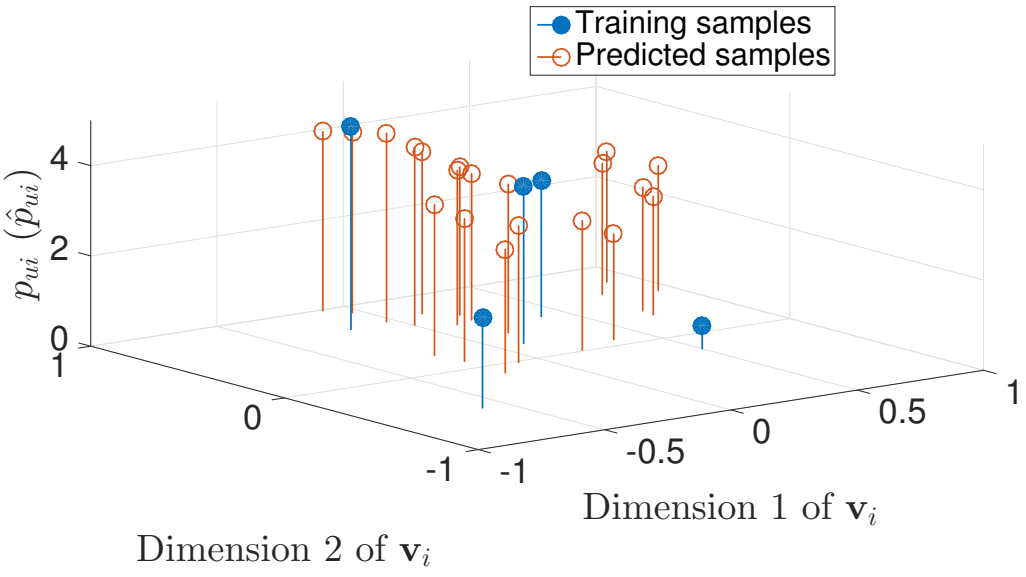

Fig. 3.1. Traditional matrix factorization: $\hat{p}_{ui}$ (and p_{ui} for training samples) vs. $\mathbf{v}_i$ for a certain user u. $\sum_{i=1}^{n} \mathbf{v}_i \neq 0$.

[36]. Note that n is usually no more than $|\Omega|$, which means SPMF at most doubles the complexity of MF. This fact makes SPMF suitable for big dataset such as the video watching history data at Tencent involving a large number of users and videos. In contrast, as has been described in Chapter 2, BPR and pairwise methods suffer from an overly high complexity $O(|\Omega|(n-\omega))$, and thus are not scalable to the problem size at Tencent.

3.2 Insights and Discussions

We provide some insights on why SPMF has a discriminative power for one-class data (containing all positive training samples $p_{ui} = t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$).

Unlike implicit-feedback CF [5], we do not perform any artificial imputation in SPMF to create negative samples to be used in training, since the imputed values are hard to decide, e.g., we cannot say p_{ui} is zero or negative just because user u has never watched video i. Furthermore, mistakenly choosing an unwatched (yet favourite) video as a negative training sample for a user will prevent the video from appearing in the recommendation list.

Rather, the discriminative power of SPMF automatically comes from modelling p_{ui} as relative interests, which finally leads to a zero-hyperplane constraint in (3.4). We illustrate this point in an toy example in Fig. 3.1 and Fig. 3.2. Fig. 3.1 plots the relationship between predicted $\hat{p}_{ui}$ (p_{ui} for training samples) and video latent factors $\mathbf{v}_i$ for a certain user u, in original MF. For simplicity, only the first two dimensions of $\mathbf{v}_i$ are plotted. Similarly, Fig. 3.2 plots $\hat{p}_{ui}$ (p_{ui} for training samples) versus $\mathbf{v}'_i$ for a certain user u in SPMF.

First, SPMF will generate $\hat{p}_{ui}$ in a larger range than MF. In MF, without imputation, all the training samples have a $p_{ui} > 0$, and thus after obtaining latent factors $\{\mathbf{u}_u\}, \{\mathbf{v}_i\}$ from (2.2), the predicted $\hat{p}_{ui} = \mathbf{u}_u^\mathsf{T}\mathbf{v}_i$ will roughly fall in the same range as training samples p_{ui} do, as shown in Fig. 3.1. In contrast, in SPMF, with constraint $\sum_{i=1}^{n} \mathbf{v}'_i = 0$, we have

$$\sum_{(u,i)} \hat{p}_{ui} = \sum_{u=1}^{m}\sum_{i=1}^{n} \mathbf{u}_u^\mathsf{T}\mathbf{v}'_i = \sum_{u=1}^{m} \mathbf{u}_u^\mathsf{T} \sum_{i=1}^{n} \mathbf{v}'_i = 0,$$

which means the predicted $\hat{p}_{ui}$ will have *both positive and negative* values, even if only positive training samples are available. In order to enforce the zero-plane constraint, this means $\hat{p}_{ui}$ will be distributed in a much wider range than training samples p_{ui}. As a result, videos that are very different from those in the training set will be pushed to have a low negative $\hat{p}_{ui}$, making the truly preferred videos stand out.

Second, SPMF always ensures that the training samples p_{ui} and predicted $\hat{p}_{ui}$ for similar videos are all above average, i.e., the videos ever watched by a user indicate some preference from that user. In contrast, in MF, since predictions $\hat{p}_{ui}$ and training samples p_{ui} are distributed roughly in the same range, a video i with the smallest p_{ui} in the training set is almost the least preferred video of user u. Suppose another video j is similar to video i. Then the predicted $\hat{p}_{uj}$ will also be close to p_{ui} and thus video j is also among the least preferred. However, this is apparently not true, because the fact that user u has ever watched video i (i.e., $p_{ui} > 0$) means that video i has at least attracted user u's attention, while most other videos have not.

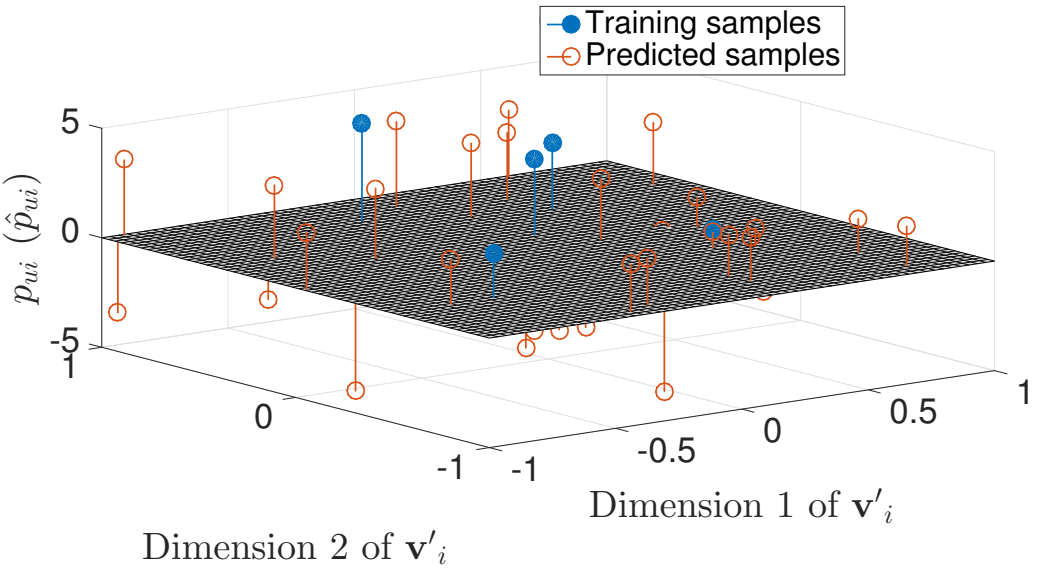

Fig. 3.2. SPMF: $\hat{p}_{ui}$ (and p_{ui} for training samples) vs. $\mathbf{v}'_i$ for a certain user u. Zero hyperplane enlarges the range of $\hat{p}_{ui}$. $\sum_{i=1}^{n} \mathbf{v}'_i = 0$.

3.3 Separating-Plane Factorization Machine

Context information can be used to extend user-video matrix factorization to multi-way factorizations, in a model called the *factorization machine* (FM) [16]. Specifically, we have incorporated the day of the week information available in each record to improve recommendation. The pairwise interactions between the latent factors of users, videos, and day of the week, could all be useful. Just like the user-video interaction reflects the interest of each user in a video, the user-day interaction indicates how likely a user will watch videos on a certain day of the week: some users prefer to watch videos on Friday while others tend to watch more videos on Sunday, depending on individual schedules. Moreover, the video-day interaction models the possibility that a video is watched on a certain day of the week: some videos are uploaded in weekends regularly, while others could be popular in weekdays.

We assume there is a latent factor vector $\mathbf{s}_d \in \mathbb{R}^k$ for each day of the week $d = 1, \ldots, 7$. We also generate each training sample $p_{uid} = t^{uid}_{\text{watch}}/t^{i}_{\text{total}}$ for each (u, i, d) triple in the observed set Ω. Incorporating the three-way interactions among $\mathbf{u}_u$, $\mathbf{v}_i$, and $\mathbf{s}_d$, we can extend SPMF (3.4) into the following *separating-plane factorization*

machine (SPFM) model:

$$
\begin{aligned}
\text{minimize} \sum_{(u,i,d)\in\Omega} & \big(p_{uid} - (b + w_u^{(1)} + w_i^{(2)} + w_d^{(3)} \\
& + \mathbf{u}_u^\mathsf{T}\mathbf{v}_i + \mathbf{v}_i^\mathsf{T}\mathbf{s}_d + \mathbf{u}_u^\mathsf{T}\mathbf{s}_d)\big)^2 \\
& + \Lambda\big(b, \{w_u^{(1)}\}, \{w_i^{(2)}\}, \{w_d^{(3)}\}, \{\mathbf{u}_u\}, \{\mathbf{v}_i\}, \{\mathbf{s}_d\}\big) \\
\text{subject to } & \sum_{u=1}^{m} \mathbf{u}_u = 0,\ \sum_{u=1}^{m} w_u^{(1)} = 0, \\
& \sum_{i=1}^{n} \mathbf{v}_i = 0,\ \sum_{i=1}^{n} w_i^{(2)} = 0, \\
& \sum_{d=1}^{7} \mathbf{s}_d = 0,\ \sum_{d=1}^{7} w_d^{(3)} = 0,
\end{aligned}
\tag{3.5}
$$

where $\Lambda(\cdot)$ is the regularization term. And the biases for each user, video and day of the week, and the global bias are taken into account. For example, the user bias $w_u^{(1)}$ is related to the average preference value of a user u, the video bias $w_i^{(2)}$ is associated with the average preference value of a video i, and the time bias $w_d^{(3)}$ models how likely people will watch videos in a certain day of the week d.

If there were no constraint in Problem (3.5), it could be solved using SGD with a complexity of $O(3k|\Omega|) = O(k|\Omega|)$ in each iteration, since the objective function involves three-way interactions only. With the zero-hyperplane constraints added, we can extend our projected SGD in Algorithm 3.1 to the case of SPFM, such that in the mapping phase, we let each $\mathbf{u}_u$ subtract the mean $\sum_{u=1}^{m} \mathbf{u}_u/m$, let each $\mathbf{v}_i$ subtract the mean $\sum_{i=1}^{n} \mathbf{v}_i/n$, and let each $\mathbf{s}_d$ subtract the mean $\sum_{d=1}^{7} \mathbf{s}_d/7$. And the number of additional operations for the mapping phase is $k(m+n+7)$, leading to an overall complexity of $O(k(3|\Omega| + m + n + 7)) = O(k|\Omega|)$ in each iteration, since $|\Omega| \geq m, n$.

Other discrete features such as hour of the day, user's device type, etc., can easily be incorporated into the SPFM model. In fact, there is some trick in training FM models [16] such that the computational cost of an SPFM with l features is only l times that of SPMF with only user and video features.

Chapter 4

Implementation

In this section, we describe the implementation of the proposed models in Tencent data warehouse (TDW) to handle big data with various performance optimizations. The measurement figures in this section are plotted based on a small sampled dataset collected from Tencent, containing 165724 users and 46578 videos.

4.1 Practical Performance Optimizations

4.1.1 Empirical Preference Measures

To generate training samples from viewing logs, in Chapter 3, we have empirically set $p_{ui} = t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$ for each observed record (3.1) in the observed set Ω. We do not use t^{ui}_{watch} directly, because the watching times (as well as the lengths) of different videos are highly skewed, as shown in Fig. 4.1. Yet, setting $p_{ui} = t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$ for $(u, i) \in \Omega$ is still heavily biased and tends to yield a higher preference value for shorter videos, as shown in Fig. 4.2. Intuitively speaking, it is much easier to finish watching a short video than a long movie. To overcome this bias toward shorter videos, we could set $p_{ui} = t^{ui}_{\text{watch}} \ / \ t^{i}_{\text{avg}}$, where t^{i}_{avg} is the average time that this video in the record is watched by all users.

However, as shown in Fig. 4.3, the distribution of $t^{ui}_{\text{watch}} \ / \ t^{i}_{\text{avg}}$ is still heavy-tailed. In order to normalize p_{ui} down to $[0, 1]$ and reduce the impact of outliers

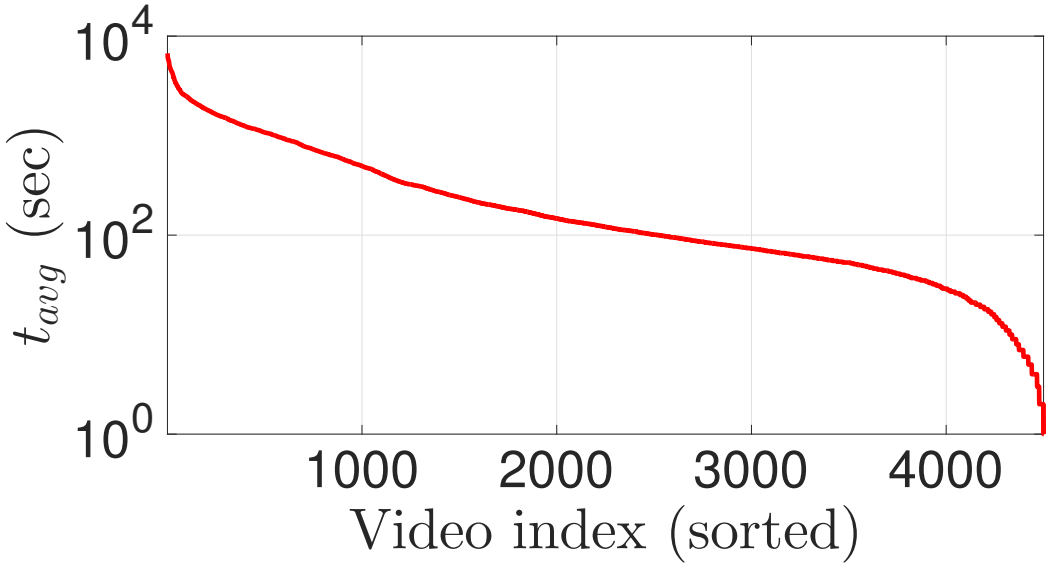

Fig. 4.1. Average watching times of videos in the sampled dataset from Tencent, containing 165724 users and 46578 videos.

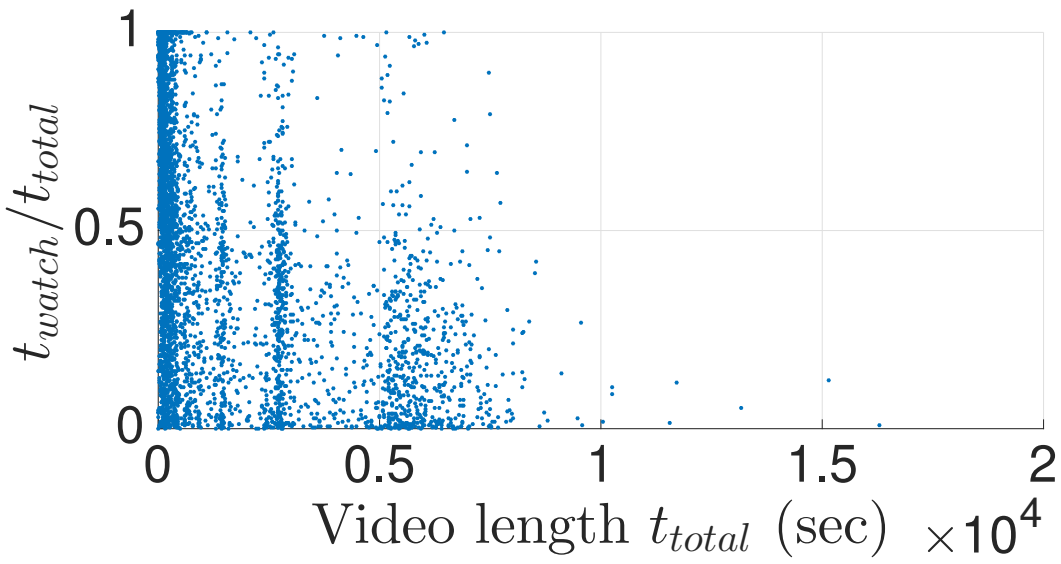

Fig. 4.2. Average $t^{ui}_{\text{watch}}/t^{i}_{\text{total}}$ vs. t^{i}_{total} in the sampled dataset from Tencent, containing 165724 users and 46578 videos.

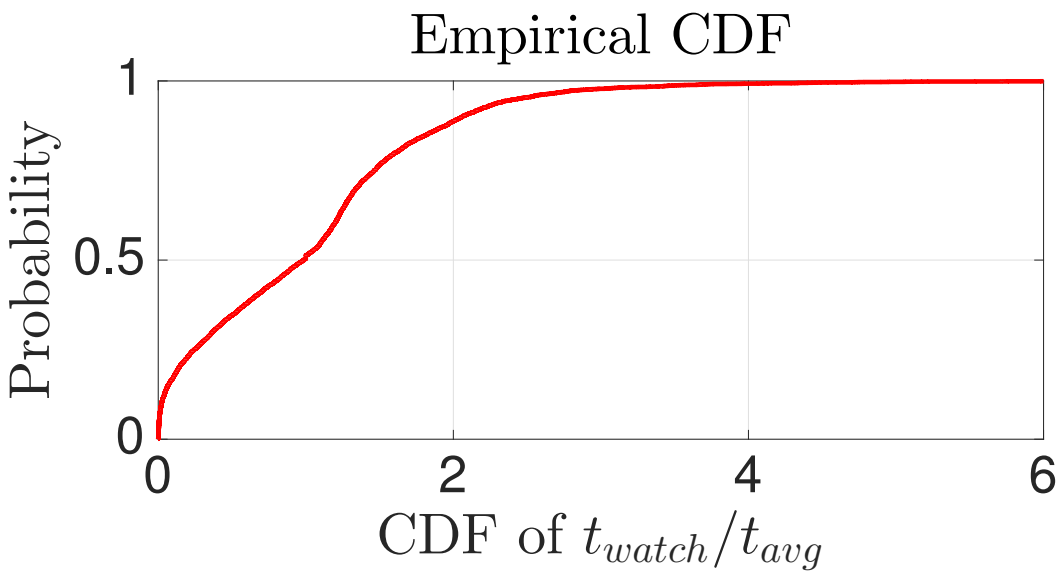

Fig. 4.3. CDF of video watching ratio in the sampled dataset from Tencent, containing 165724 users and 46578 videos.

with an extremely high watching time (as compared to the average watching time), we use the following empirical measure of p_{ui} for each observed record in our implementation:

$$p_{ui} = 0.5\text{tanh}((t^{ui}_{\text{watch}}/t^{i}_{\text{avg}} - 1)\pi) + 0.5, \tag{4.1}$$

We use a hyperbolic tangent function to do the normalization, because the tanh is almost linear near the mean, and thus can differentiate t^{ui}_{watch} around t^{i}_{avg}, and weaken the differences in extreme cases, e.g., we cannot say a user who has watched a movie for 10 times likes the more more than a user who watched it for 5 times. And it does achieve better performance in our evaluation at Tencent on big data to be presented in Chapter 5.

4.1.2 Weighted SPMF and SPFM

Recall that in SPMF, $\hat{p}_{ui}$ is estimated by the *relative interest* of user u for video i as compared to an *average* video, i.e., $\hat{p}_{ui} = \mathbf{u}_u^{\mathsf{T}}(\mathbf{v}_i - \overline{\mathbf{v}})$, where $\overline{\mathbf{v}} := \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i$. However, in reality, not all the videos play an equal role in defining the so-called "average video." In fact, as shown in Fig. 4.4, the video popularity is highly skewed. For

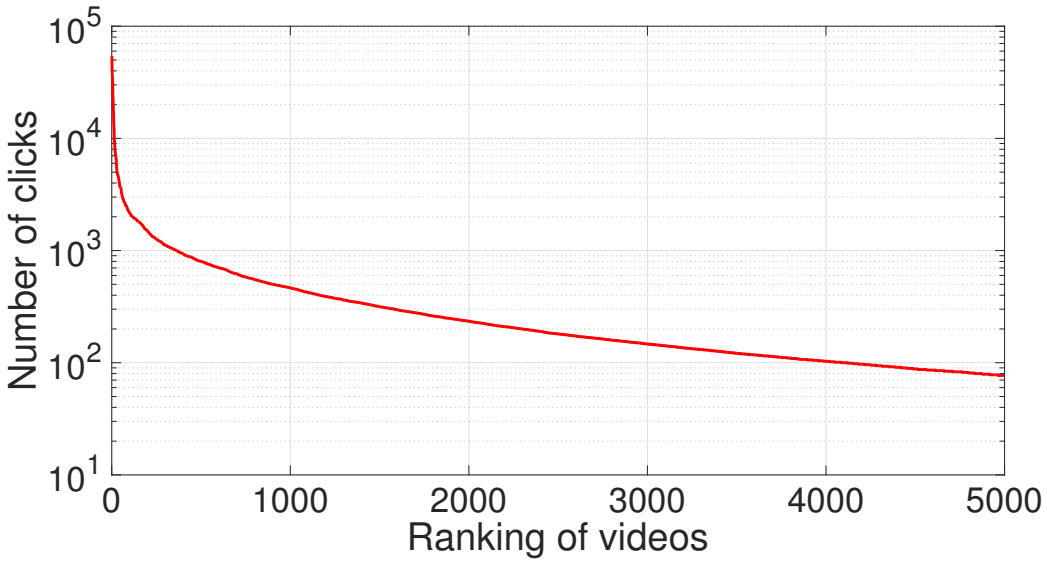

Fig. 4.4. Total numbers of clicks of different videos in the small dataset collected from Tencent, containing 165,724 users and 46,578 videos.

example, in the sampled data from Tencent, 1% of all the videos each receive more than 3600 clicks, whereas 95% of the videos each receives fewer than 800 clicks. Intuitively speaking, popular videos should contribute more in terms of defining the "average video." On the other hand, a cold video seldom viewed by users should only plays a trivial part in the defining $\overline{\mathbf{v}}$. Therefore, we adopt weighted SPMF/SPFM in our implementation, by redefining $\overline{\mathbf{v}}$ as follows:

$$\overline{\mathbf{v}} := \frac{\sum_{i=1}^{n} h(c_i)\mathbf{v}_i}{\sum_{i=1}^{n} h(c_i)}, \tag{4.2}$$

where $h(c_i)$ is the hotness of video i, depending on the number of clicks c_i video i has received. We have tried various types of weights $h(c_i)$ and found that exponential weights $h(c_i) = \alpha^{c_i}$ (with α greater than yet close to 1) yield the best performance. The idea is that the "average video" can be better defined by a small number of extremely hot videos that are highly visible to everyone, appearing in the front page of QQ Browser mobile app or its push notifications.

In weighted SPFM (SPMF), we replace the constraint $\sum_{i=1}^{n} \mathbf{v}_i = 0$ by $\overline{\mathbf{v}} = 0$, with $\overline{\mathbf{v}}$ defined by (4.2). Then, in each iteration of the projected SGD algorithm,

we just need to subtract the newly defined $\overline{\mathbf{v}}$ from each $\mathbf{v}_i$ in the mapping phase. A similar weighting procedure is applied to the video bias terms $w_i^{(2)}$ as well.

4.1.3 Time-Decayed Confidence Levels

Since user interests may change, records collected in recent days can better predict user interests at the present. Let τ_{ui} be the number of days between the time when the record R_{ui} was collected and the present. We further introduce a confidence parameter $c(\tau)$ to weight the loss function as follows:

$$\begin{aligned} &\underset{\{\mathbf{u}_u\},\{\mathbf{v}_i\}}{\text{minimize}} \sum_{(u,i)\in\Omega} c(\tau_{ui})(p_{ui} - \mathbf{u}_u^{\mathsf{T}}\mathbf{v}_i)^2 + \Lambda\big(\{\mathbf{u}_u\},\{\mathbf{v}_i\}\big) \\ &\text{subject to } \overline{\mathbf{v}} = 0, \end{aligned} \tag{4.3}$$

where $\Lambda(\cdot)$ is the regularizer, and $c(\tau) = \beta^{-\tau}$, in which β controls the speed at which $c(\tau)$ decays as τ increases. We find that a mix of β depending on video types leads to the best performance. It is natural that user preferences for short videos are more inclined to vary over time, while their tastes for movies may last longer.

The use of $c(\tau)$ is similar to Weighted Regularized Matrix Factorization (WRMF) [5], [6]. The difference is that WRMF tries to lower the impact of (imputed) negative samples. Since SPMF has already addressed the implicit feedback issue via a separating hyperplane, here $c(\tau)$ is used to emphasize the impact of more recent feedback.

4.2 Implementation and Deployment

In the video recommender system we implemented at Tencent, the feedback data (up to hundreds of GB) contains the user watching history in the past 40 days, involving more than 12,500,000 users and more than 100,000 videos. To scale up to such a huge problem size, we have implemented our algorithms in *Spark* to run in a parallel manner on clusters. Since SPMF is a special case of SPFM, we have implemented the projected SGD algorithm for SPFM using GraphX (Spark's API

for graph-parallel computation) by extending distributed SGD [50] for factorization machines to include the mapping phase.

In particular, we construct a bipartite graph as shown in Fig. 4.5 and Fig. 4.6, where nodes in the leftmost column are called feature nodes representing all latent factor vectors including $\{\mathbf{u}_u\}$, $\{\mathbf{v}_i\}$ and $\{\mathbf{s}_d\}$, while nodes in the middle column are called estimated preference nodes representing $\hat{p}_{uid}$ and nodes in the rightmost are called sample nodes representing p_{uid}. Edges connecting both sides stand for the relationship between features and samples. As shown in Fig. 4.5, a forward-backward procedure is used in training. In the forward phase, feature nodes concurrently calculate the estimated values and forward to their connected estimated preference nodes. By comparing with sample nodes, we can obtain the residual error. In the backward phase, as shown in Fig. 4.6, sample nodes concurrently compare their labeled values with received estimated values and obtain the residual errors, then back-propagate the errors to feature nodes for updating.

We have deployed our Spark-based program on Tencent distributed Data Warehouse (TDW), an open-source big data processing platform developed by Tencent. TDW is used to store data, manage computation resources and perform data mining jobs at Tencent [51]. Specifically, as shown in Fig. 4.7, we execute the following tasks sequentially on TDW: 1) fetch all the view logs of the past 40 days from the data store in TDW, distribute data to a computing cluster in TDW (consisting of 100-300 virtual instances each with 5 GB memory and a 2.2 GHz single core) and preprocess data in parallel; 2) train the SPFM model with the developed Spark program based on all the observed records, and store parameters; 3) roughly select *candidate* videos to be recommended to each user, based on a combination of item-based CF and video popularity, to avoid the overly high complexity to calculate $\hat{p}_{ui}$ for all (u, i) pairs; 4) calculate preference values $\hat{p}_{ui}$ between each user u and the selected candidate videos i for user u, and pick the videos with top $\hat{p}_{ui}$ values to be recommended to user u; 5) fetch the recommendation results and present them to users via the online backend server; 6) record user behavior and store the logs in TDW to be used in tasks 1)-5) again.

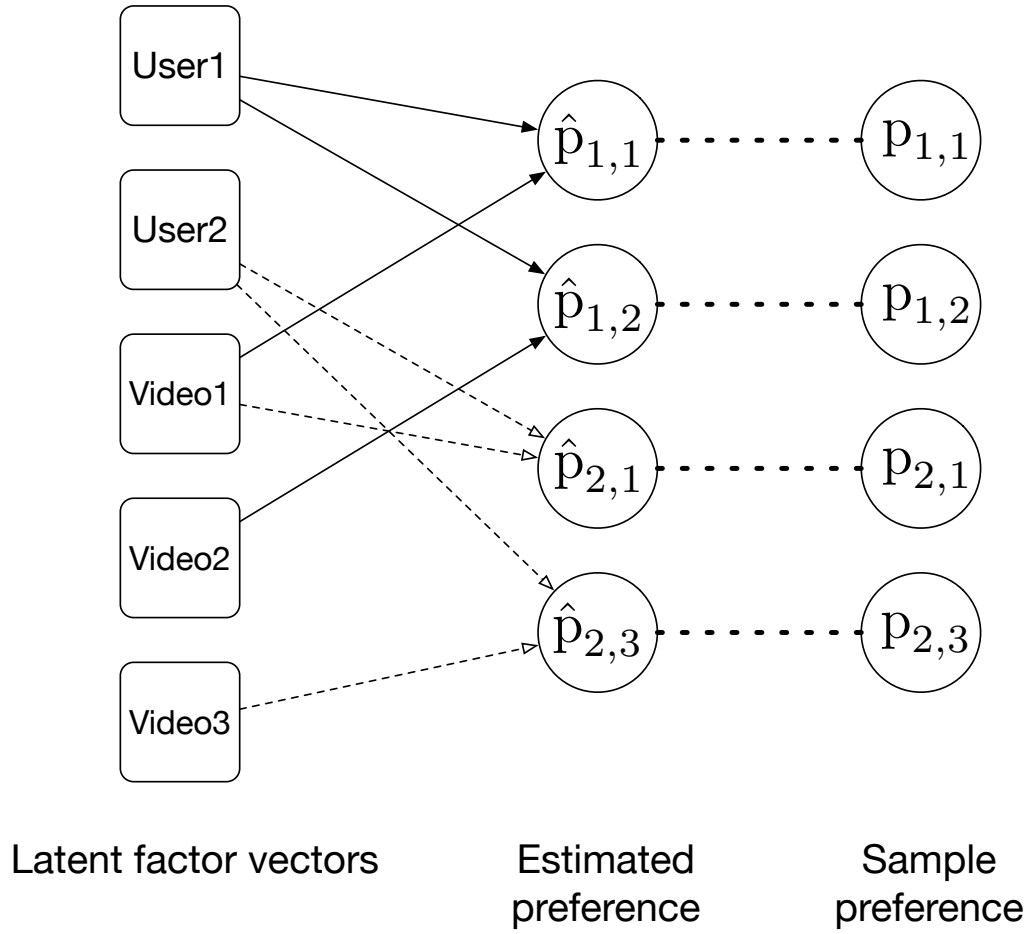

Fig. 4.5. Estimated preference nodes receive messages from connected feature nodes and calculate residual errors by comparing with sample rating nodes in parallel.

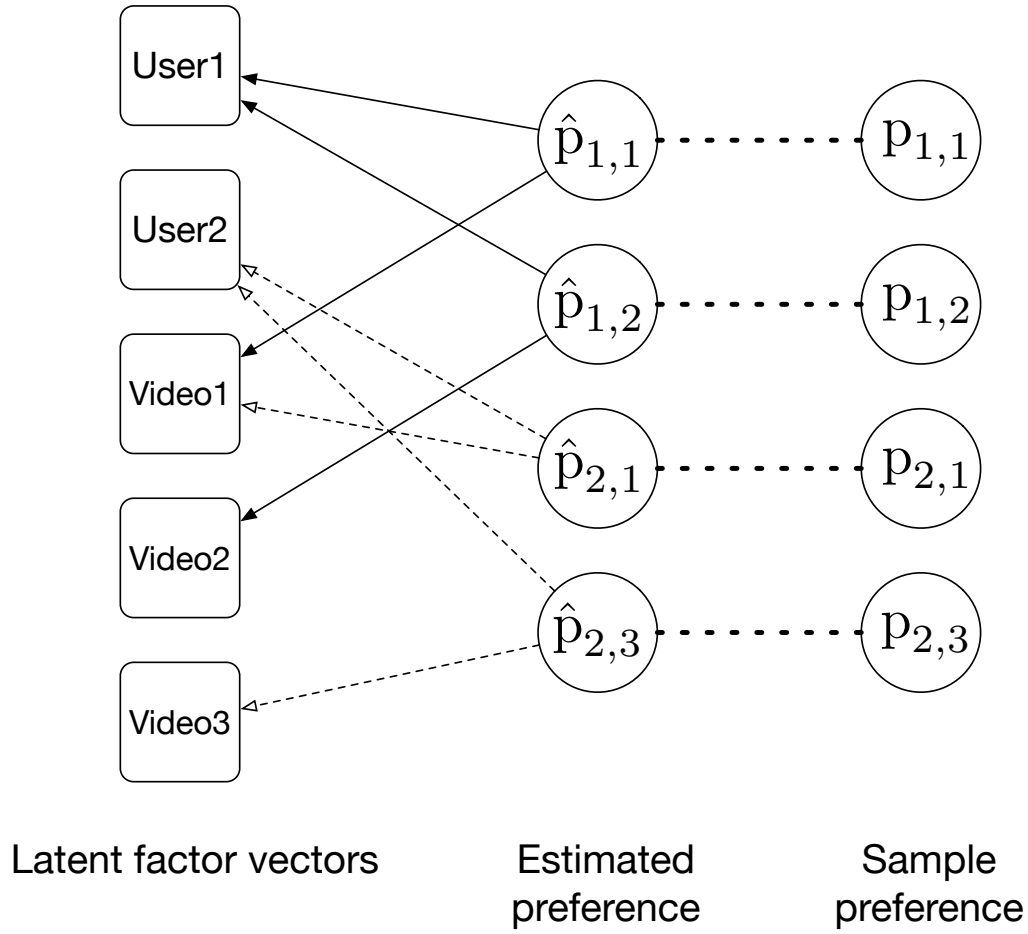

Fig. 4.6. Feature nodes receive residual errors from connected estimated preference nodes and update the weights in parallel.

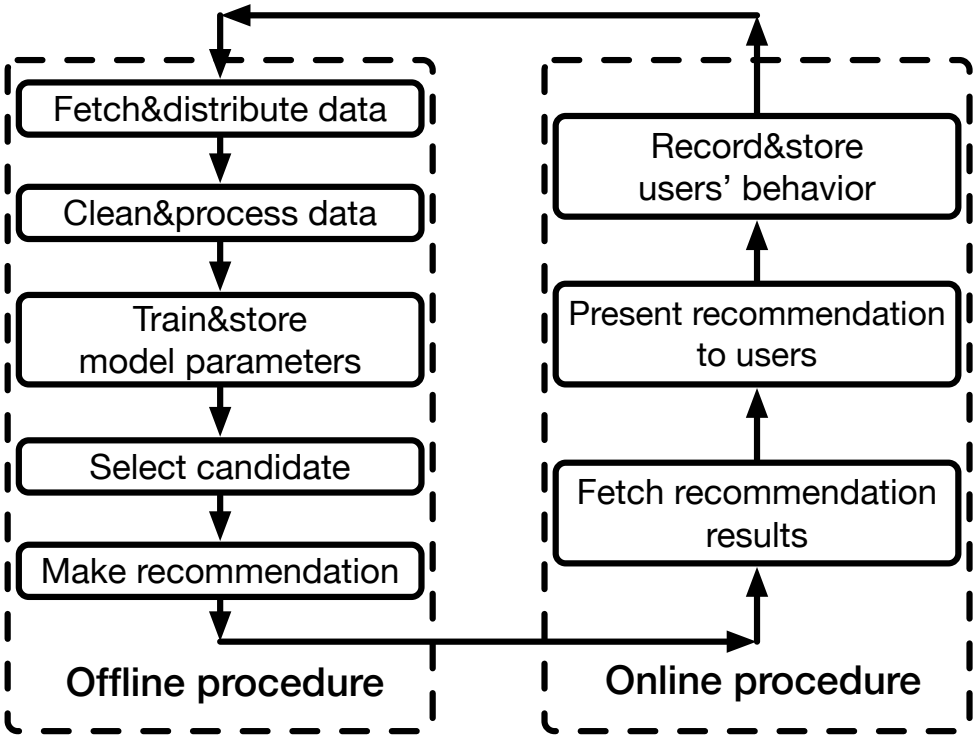

Fig. 4.7. Workflow of our recommender system running on TDW platform.

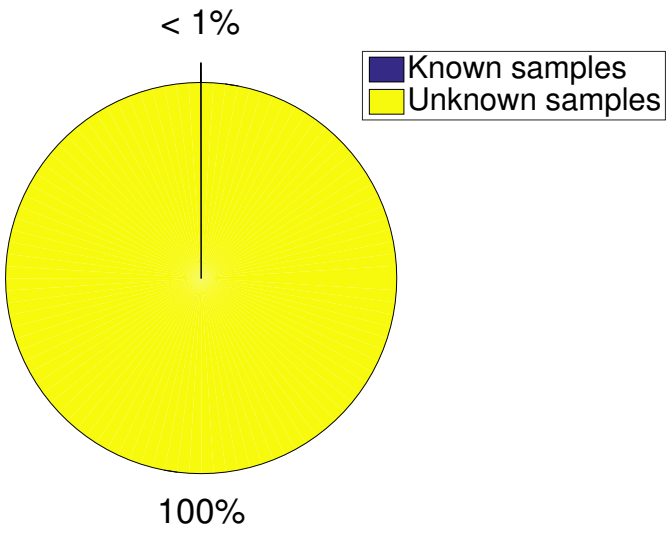

Fig. 4.8. Density of known samples in the small dataset collected from Tencent, containing 165,724 users and 46,578 videos.

In model training in task 2), as shown in Fig. 4.8, only 0.034% of all (u, i) pairs are observed. Thus, the training cost is affordable on a cluster of 100–300 instances. But it is impossible to calculate $\hat{p}_{ui}$ for all (u, i), even if we have obtained all $\{\mathbf{u}_u\}$ and $\{\mathbf{v}_i\}$, simply because there are more than 10^{13} such (u, i) pairs in total. Therefore, task 3) is performed to find popular videos or videos similar to those previously watched by each user via item-based CF to serve as the candidate set for that user. Then, $\hat{p}_{ui}$ is only computed for the videos in the candidate set of each user. Since the number of candidate videos could still be large, which may cause out-of-memory issues in Spark, we serialize the candidate sets of all the users into batches, for which predictions are made sequentially.

Chapter 5

Performance Evaluation

We conduct performance evaluation of our new models in comparison to state-of-the-art recommender schemes that can handle implicit feedback in three scenarios: 1) offline single-machine tests based on two small datasets from Tencent and Movielens, repsectively, 2) offline tests on Tencent data warehouse (TDW) on big data from Tencent, 3) online A/B tests, with training conducted in TDW based on big data and testing on real users.

We evaluate the following algorithms and have individually fine-tuned the performance of all of them for a fair comparison:

- Popularity based recommendation (POP): always recommend the most popular videos;
- Item-based collaborative filtering (IBCF) [9];
- Traditional matrix factorization (MF) (2.2) [2];
- Implicit-feedback collaborative filtering (IFCF) (2.3) [5];
- Restricted Boltzmann Machine (RBM) [38];
- Separating-Plane Matrix Factorization (SPMF) (3.4);
- Separating-Plane Factorization Machine (SPFM) (3.5), with the day of the week feature incorporated;

TABLE 5.1
MPR FOR TENCENT SMALL DATASET AND MOVIELENS DATASET IN SINGLE-MACHINE TESTS.

Model	MF	IFCF	BPR	SPMF
MPR (Tencent's small dataset)	17.5%	15.4%	14.9%	14.6%
MPR (MovieLens dataset)	20.1%	28.4%	18.5%	13.1%

- Variations of SPFM with practical optimizations in Sec 4.1.

The algorithms are compared in terms of precision and click through rate (CTR) in online A/B tests, and in terms of Mean Percentile Ranking (MPR) and ROC curves in offline evaluation. Percentile ranking was introduced in [5] and [52] to evaluate how close a test video is to the top of the ranked list, which is independent of the lengths of recommendation lists.

Formally, MPR is defined as follows. Denote by rank_{ui} the *percentile ranking* of video i within the ordered list of all videos for user u (sorted by predicted preference values), such that rank_{ui} = 0% if video i is predicted to be the most preferred by user u, thus placed at the top of the list, while rank_{ui} = 100% if video i is predicted to be the least preferred for user u, thus placed at the end of the list. Then, we define mean percentile ranking (MPR) as the mean value of $rank_{\text{ui}}$ for all $(u, i) \in \Phi$, where Φ is a test set of actually watched videos. A lower MPR is desirable, as it indicates that the actually watched videos in the test set are closer to the top of the ranked video lists. Note that with random predictions for a large number of users, MPR is 50%, i.e., each test video for a user is ranked in the middle of the sorted list for the user on average.

5.1 Single-Machine Tests

We first present results from single-machine test based on two small datasets from Tencent and MovieLens, respectively. The Tencent dataset contains 2604572 watching behavior records of the form (3.1) collected in 40 days, involving 165724 users and 46578 videos. The test samples consist of the last watched video of each user,

and the rest of the records serve as training samples. On the other hand, the publicly available MovieLens dataset contains explicit ratings (from 0.5 to 5) between 668 users and 10325 videos. To evaluate the scenario of one-class feedback, we only keep the ratings that are above 3, and remove all the ratings no more than 3. Without any time information of ratings, for each user, we choose the rightmost video in the video list that she has positively rated (> 3) to serve as the test sample and the rest as training data.

For both datasets, we generate all $\hat{p}_{ui}$ from training data, sort the preferred video list for each user based on $\hat{p}_{ui}$ and calculate the MPR of testing samples. From Table. 5.1 we can see SPMF outperforms other algorithms and has a lower MPR for both the Tencent data and MovieLens data.

For MovieLens data, we also compare different models in terms of receiver operating characteristic (ROC) curve in Fig. 5.1. The testing samples are *actual positive samples* and the other unwatched videos are *actual negative samples*. Let x be a threshold, such that the *predicted positive samples* for each user u are top-x videos in the ranked list for user u, and the rest in the list are *predicted negative samples* for user u. Then, we can move x from 1 to n, the total number of videos, and obtain an ROC curve in Fig. 5.1, which shows that SPMF outperforms all other algorithms in terms of area under the curve (AUC). SPMF also significantly outperforms BPR, although BPR is a more complex pairwise model.

Moreover, we compare our SPMF with iTALS time bands [53], which also used the MovieLens dataset to evaluate the recommendation performance assuming only positive feedback is available. For a fair comparison, we follow the same experimental setup as in [53]: we use all the 5-star ratings as training samples, based on which top 20 movies are selected for each user by SPMF and iTALS time bands, respectively. Then, following the same evaluation criterion as in [53], we report Recall@20, which is defined as the ratio of the selected top 20 movies for each user falling in the movies rated 4.5 or higher by this user. Recall@20 is an important metric as in practice, users usually view the top 20 items. The movies rated 4.5 or higher by each user are deemed in [53] as the movies truly preferred by the user,

TABLE 5.2
RECALL@20 OF ALGORITHMS USING FACTORIZATION WITH 20 AND 40 FEATURES ON MOVIELENS DATASET IN SINGLE-MACHINE TESTS.

Model	iALS	iCA time bands	iTALS time bands	SPMF
Movielens(20)	0.0494	0.0553	0.0896	0.0910
Movielens(40)	0.0535	0.0494	0.0937	0.0984

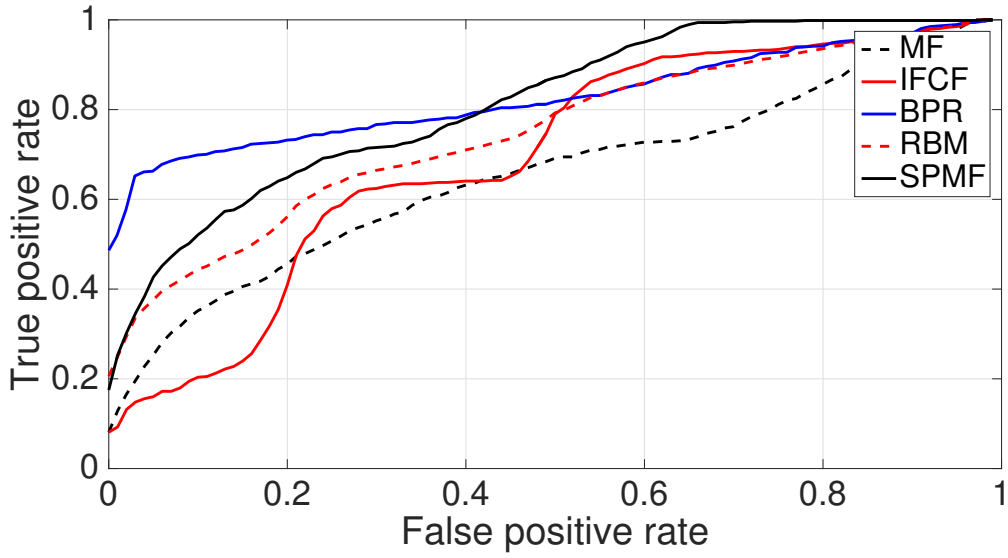

Fig. 5.1. Receiver operating characteristic (ROC) curve for MovieLens data in single-machine tests, the publicly available MovieLens dataset contains explicit ratings (from 0.5 to 5) between 668 users and 10325 videos.

according to some empirical measurements.

Both algorithms are run with the same number of features (the same dimension of latent factor vectors) and the same number of iterations. We set the number of features to 20 and 40, which is used in [53] and commonly used in literature [52], [54]. We report the results in Table. 5.2, where the results of iTALS time bands together with two other baseline schemes that appeared in [53] are authentic values reported by [53], while the results of SPMF are generated from our experiments following the same setup. We can observe that the performance of SPMF outperforms the other algorithms in the same metric adopted in [53].

We then evaluate the performance of SPFM (with day-of-the-week feature) and

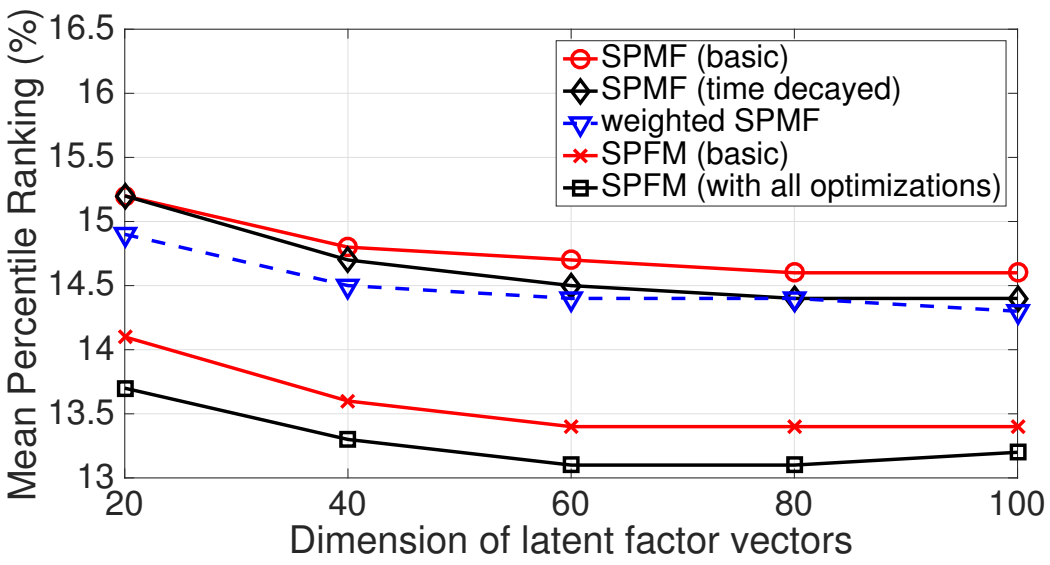

Fig. 5.2. MPR of different optimization options as the dimension of feature vector $\mathbf{v}$ varies @ Tencent's small dataset, The Tencent dataset contains 2604572 watching behavior records of the form (3.1) collected in 40 days, involving 165724 users and 46578 videos.

the effect of practical optimizations described in Sec. 4.1 on Tencent data. As shown in Fig. 5.2, we make comparisons among basic SPMF, SPMF with time-decayed confidence levels (Sec. 4.1.3), weighted SPMF (Sec. 4.1.2), basic SPFM (3.5), and SPFM with all the optimizations implemented. From Fig. 5.2, we can see that as the dimension of latent factor vectors varies, models with optimizations can always achieve lower MPRs than their original versions. Moreover, SPFM with optimizations achieves the best performance among all algorithms.

5.2 Offline Tests on Distributed Clusters

We now present results from offline evaluation based on a big dataset containing about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos. To scale up to such big data, we implemented all the compared algorithms in Spark (see Chapter 4.2 for implementation details of our scheme) and run the programs in TDW.

Similarly, we take the last watched video of each user as the testing sample.

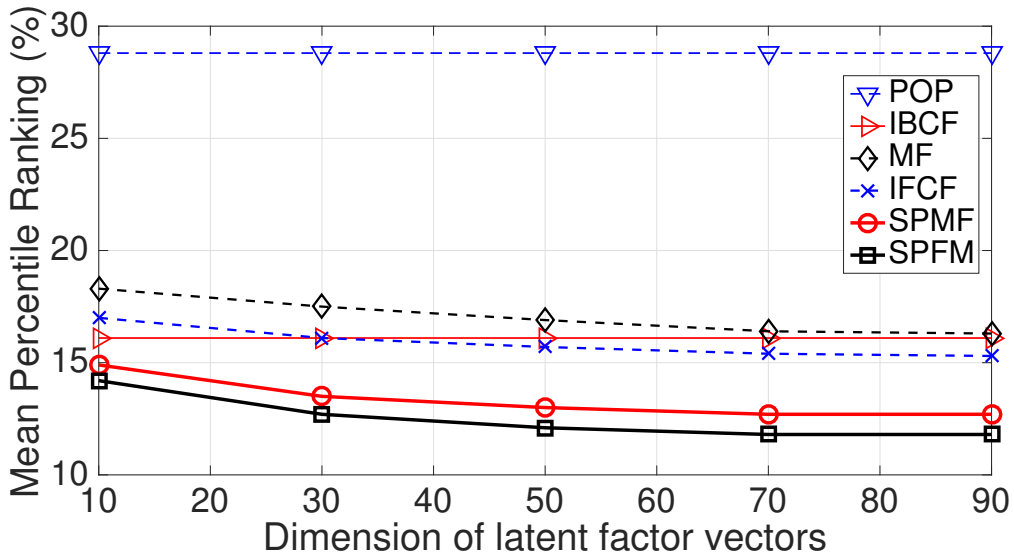

Fig. 5.3. Large-scale offline test: MPR vs. the dimension of latent factor vectors, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos.

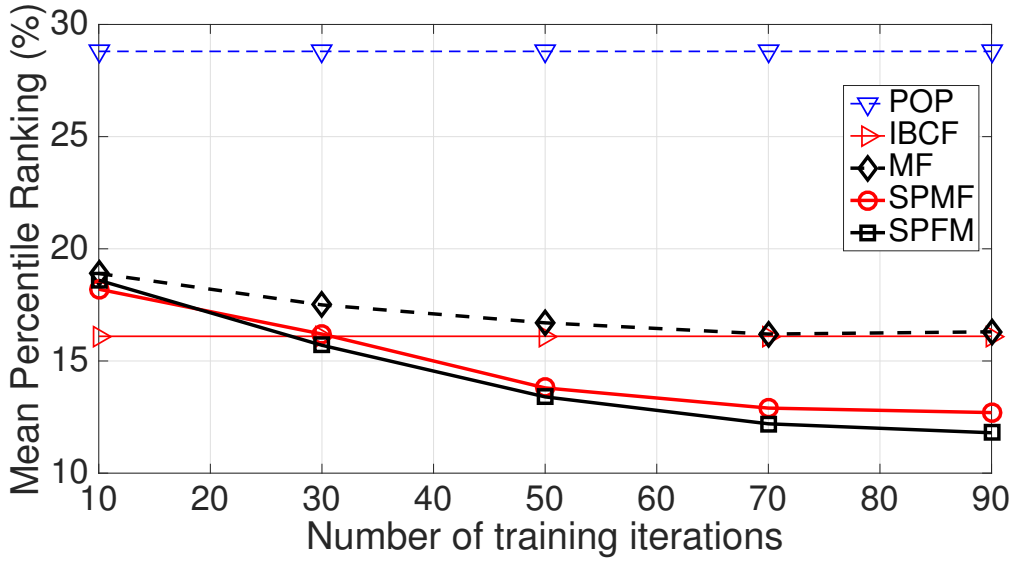

Fig. 5.4. Large-scale offline test: MPR vs. the number of training iterations, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos.

TABLE 5.3
EFFECT OF DIFFERENT EMPIRICAL PREFERENCE MEASURES IN LARGE-SCALE OFFLINE EVALUATION WITH SPFM, THE DATASET CONTAINS ABOUT 400,000,000 VIDEO VIEWING RECORDS OF THE FORM (3.1) COLLECTED FROM TENCENT IN 40 DAYS, INVOLVING MORE THAN 12,500,000 USERS AND 100,000 VIDEOS.

Empirical p_{ui}	1.0	$t_{\text{watch}}/t_{\text{total}}$	$t_{\text{watch}}/t_{\text{avg}}$	tanh
MPR (%)	14.3	13.9	12.0	11.8

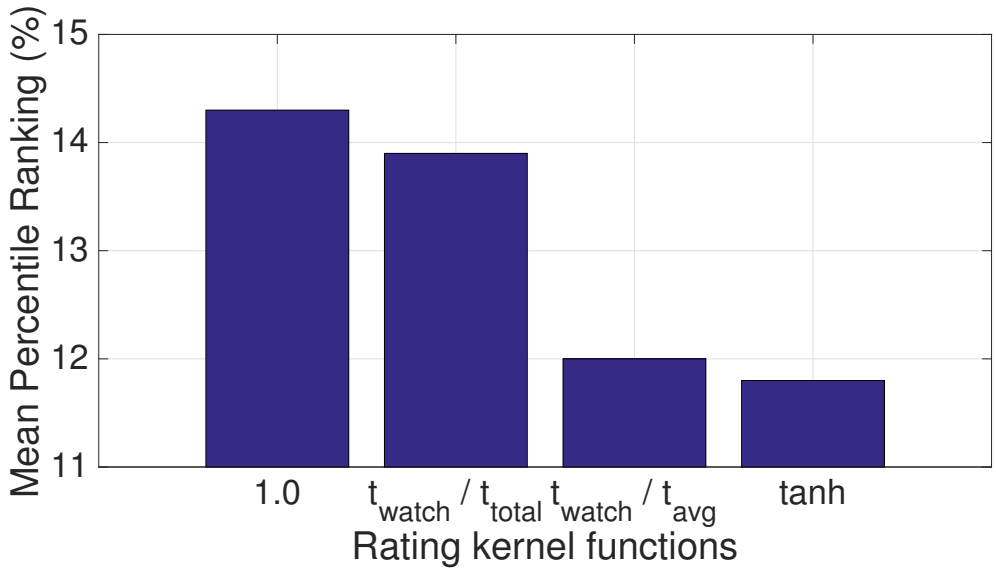

Fig. 5.5. Effect of different empirical preference measures in large-scale offline evaluation with SPFM, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos.

And the rest are training samples. From Fig. 5.3 and Fig. 5.4, we can see that SPMF and SPFM (both with practical optimizations) outperform other state-of-the-art algorithms in terms of MPR. Fig. 5.4 further shows that as more iterations are used in the projected SGD algorithm, the benefit of our methods over traditional MF becomes more significant.

We compare different empirical measures of preference values in Table. 5.3, while keeping other parameter settings unchanged in SPFM. Table. 5.3 and Fig. 5.5 show that setting the preference value p_{ui} for each observed record according to (4.1) outperforms all other variants. Similarly, we kept other parameter settings and changed the time decay parameter β in (4.3). As shown in Fig. 5.6, a blend of β

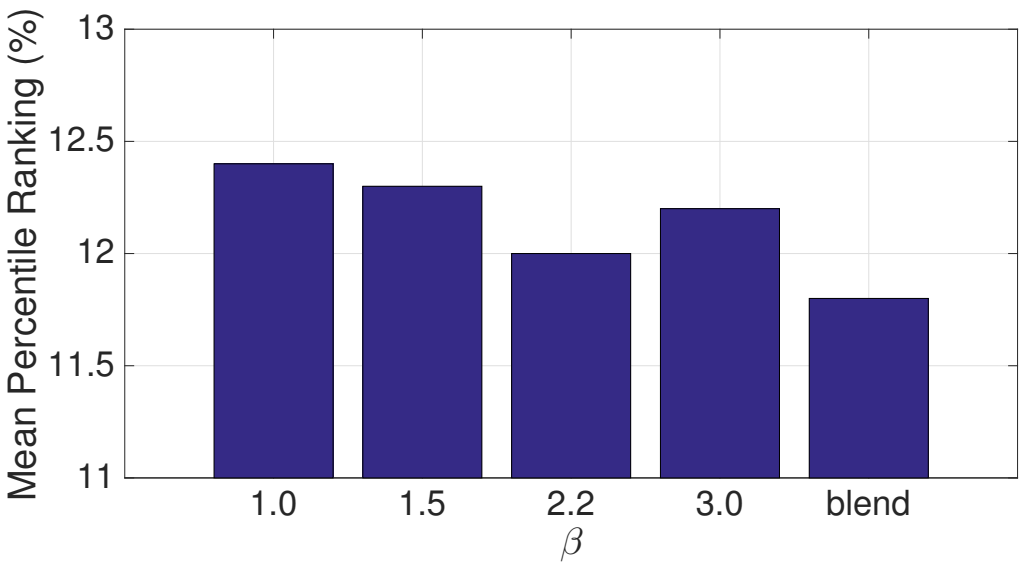

Fig. 5.6. Effect of different Time-Decayed Confidence Level measures in large-scale offline evaluation with SPFM, the dataset contains about 400,000,000 video viewing records of the form (3.1) collected from Tencent in 40 days, involving more than 12,500,000 users and 100,000 videos.

TABLE 5.4
RESULTS OF ONLINE A/B TESTS, THE TRAINING DATASET IS COLLECTED IN THE PAST 40 DAYS, CONTAINING 400 MILLION VIEWING RECORDS, INVOLVING ABOUT 12,500,000 USERS AND 100,000 VIDEOS, AND THE A/B TESTS ARE RUN ON 50865 ACTUAL USERS.

Model	IBCF	IFCF	SPMF	SPFM
# Clicks (Sunday)	14248	14739	17137	17627
# Clicks (Monday)	10273	10531	12825	13421
CTR (%)	14.1	14.5	17.2	17.9
Precision @10 (%)	19.2	20.2	23.3	24.4

values shows better MPR than other fixed values. Similarly we tune the time-decay parameter β and find a mix of β is the best; it can be big for news videos and variety shows while small for series and movies.

5.3 Online A/B Tests

We ran online tests of SPMF and SPFM in comparison to IBCF and IFCF, with training conducted in TDW based on data collected from QQ Browser mobile app in the past 40 days, containing 400 million viewing records, involving about

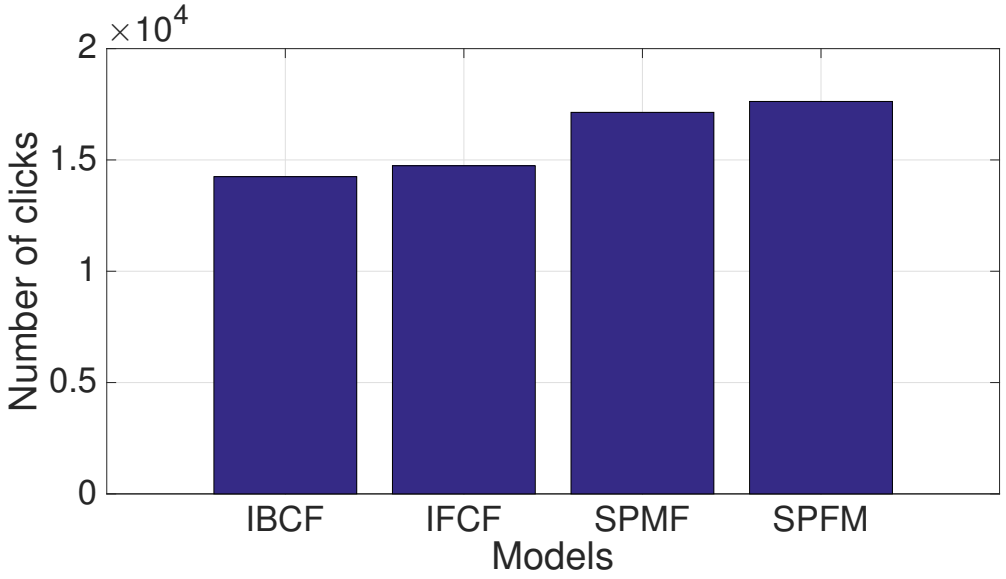

Fig. 5.7. The total number of clicks on Sunday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

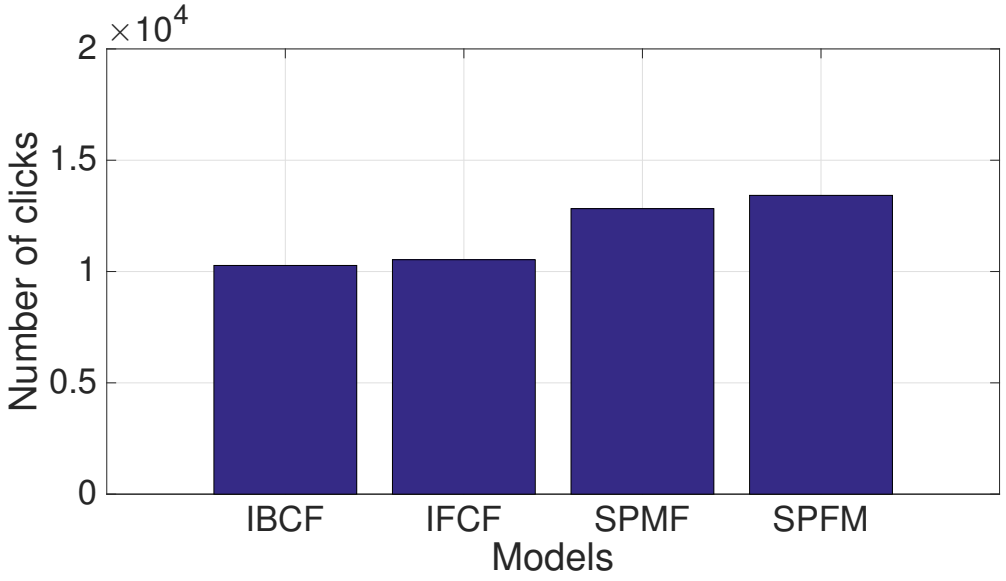

Fig. 5.8. The total number of clicks on Monday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

12,500,000 users and 100,000 videos. See Chapter 4.2 for deployment details. For matrix-factorization-based schemes including SPMF, SPFM and IFCF, due to the complexity to calculate all $\hat{p}_{ui}$ after obtaining all latent factors in the training, we use POP and IBCF to generate a set of 200 candidate videos for each user, and then calculate $\hat{p}_{ui}$ only for these candidate videos for each user and recommend the top 10 videos to each user. We have computed recommendations to all users to verify the system's capability to produce recommendation results for all the users, in fact within 2-3 hours for IFCF, SPMF, and SPFM, and about 30 minutes for IBCF. Therefore, our algorithms are fast enough, since in the production environment at Tencent, training is only performed daily.

When performing online split tests, the recommendation results are pushed to 50865 real users in QQ Browser mobile app from whom consent has been acquired. These users are split randomly and evenly into 4 groups, each group taking the recommendations generated from one of the four schemes. After 2 days (a Sunday and a Monday), we collect the number of clicks on our recommended videos in each of the 4 groups.

Table 5.7 shows that SPFM and SPMF attracted a larger number of user clicks on the recommended videos, with SPFM being the best. Fig. 5.7 and Fig. 5.8 can also help illustrate the difference of total numbers of clicks which is a direct indicator of their performance in A/B tests. Since users are randomly selected, more clicks on recommended videos in a group indicate better recommendation results than other groups. Fig. 5.9, Fig. 5.10 and Fig. 5.11 show how the number of clicks varies along time, in which SPFM and SPMF obviously outperform the baseline schemes most of the time. Fig. 5.12 and Fig. 5.13 show how numbers of clicks are distributed among users, which also shows that SPFM and SPMF outperform the baseline schemes in most cases.

Furthermore, we design a set of statistical experiments to assess the significance of differences among different groups. First, the distribution of the number of clicks per hour for each of the 4 methods is plotted in Fig. 5.14 as reflectance and boxplot. Then, we assume the reflectance data (number of clicks per hour) and

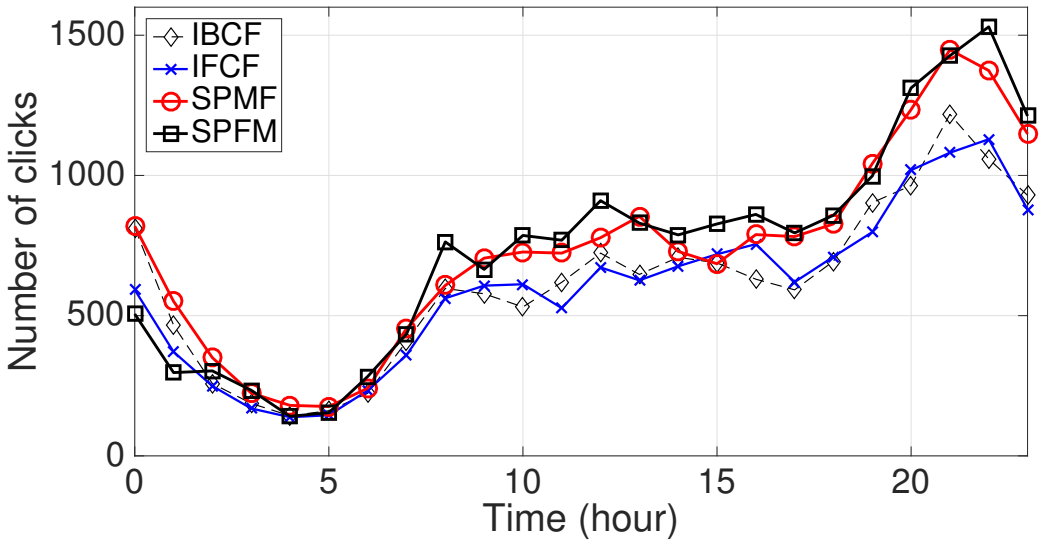

Fig. 5.9. Number of clicks on the recommended videos on Sunday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

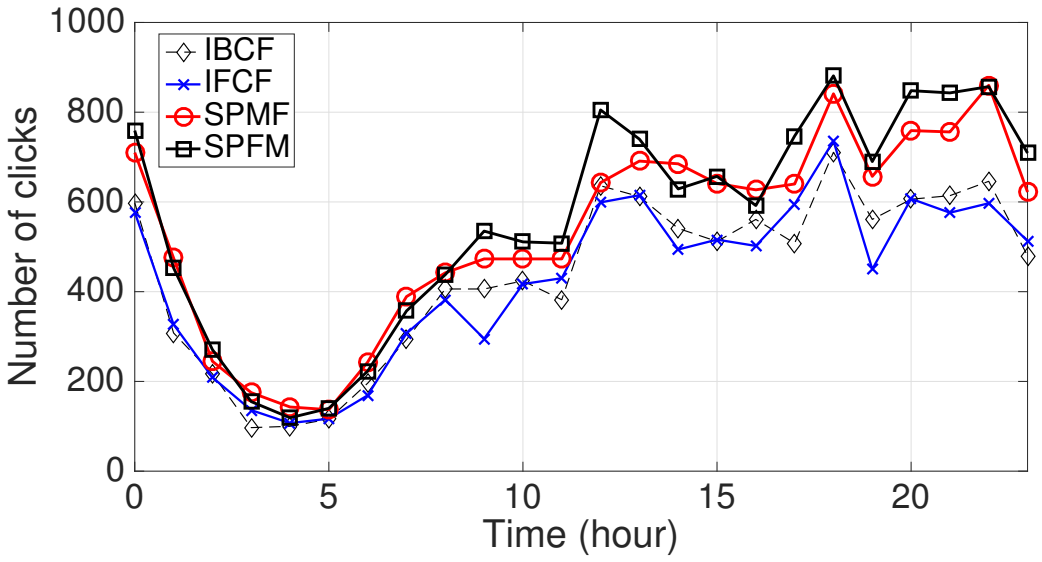

Fig. 5.10. The number of clicks on the recommended videos on Monday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

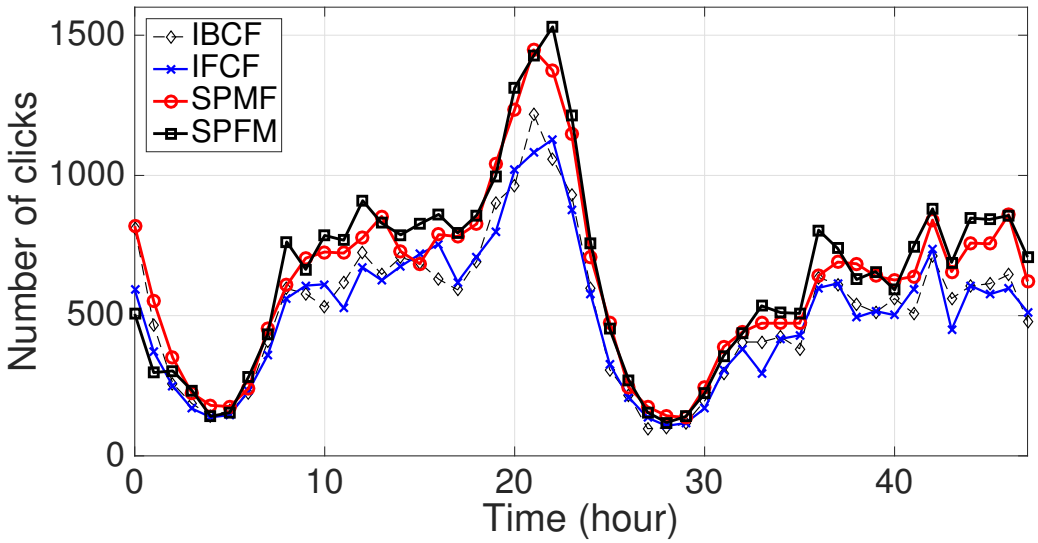

Fig. 5.11. The number of clicks on the recommended videos in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

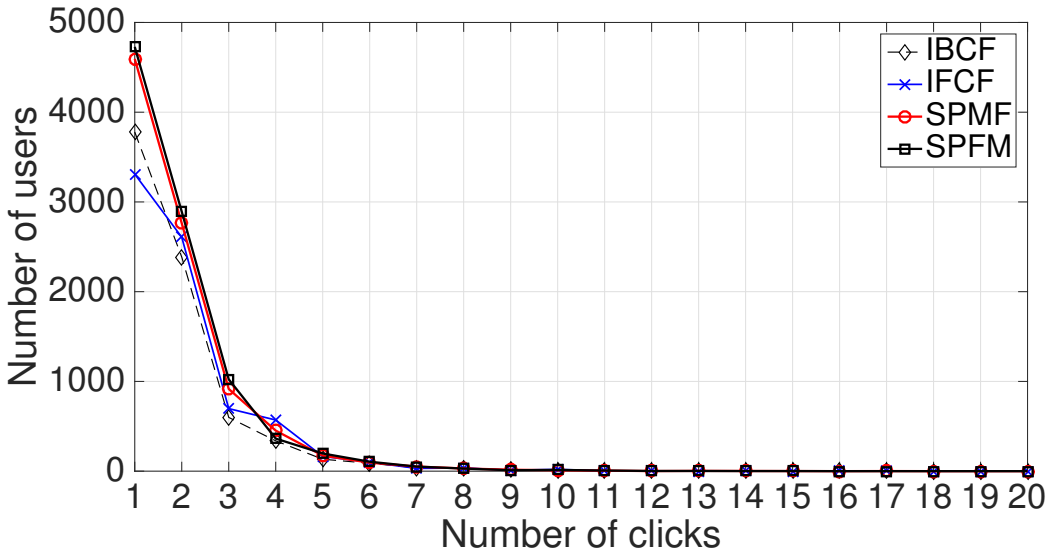

Fig. 5.12. The number of users with different numbers of clicks on Sunday in online A/B tests. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

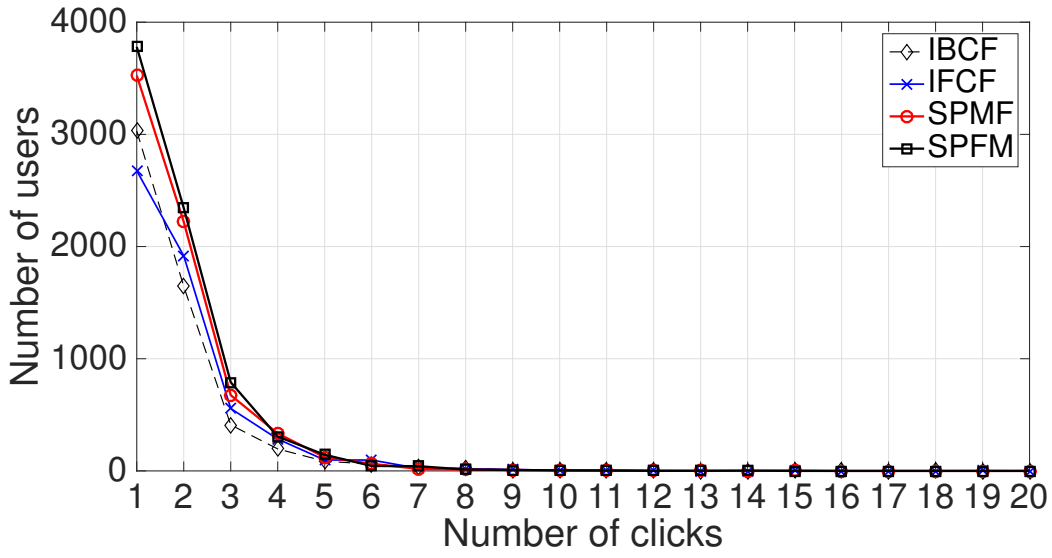

Fig. 5.13. Number of users with different numbers of clicks on Monday in online A/B tests, the training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

operators (the 4 methods) $i = 1, 2, 3, 4$ comply with the *Random Effects Model*: $y_{ij} = \eta + \tau_i + \epsilon_{ij}$, in which η stands for the average reflectance across all the cases, and τ_i is a operator-specific random effect, which measures the difference between the average number of clicks in method i and the average number of clicks across all the cases. The ϵ_{ij} means an individual-specific error, representing the number of clicks from a specific hour j in method i. As a standard approach, we apply the zero-sum constraint $\sum_{i=1}^{4} \tau_i = 0$. After running the one-way analysis of variance (ANOVA), with results shown in Table. 5.5, the F-value is 2.859 and the p-value $\Pr(> F)$ is 0.0383, which is smaller than 0.05 and thus declares a significant operator-to-operator difference. This fact means that there do exist sufficiently significant difference between the experiment groups and control groups. And Table. 5.6 shows that according to the p-adj value, the most significant pairwise difference exists between the SPFM and IFCF, which means our algorithm SPFM is significantly better than the baseline algorithm IFCF.

We also compare different groups in terms of the click through rate (CTR) and Precision@10. CTR is the ratio of the number of videos clicked through from our

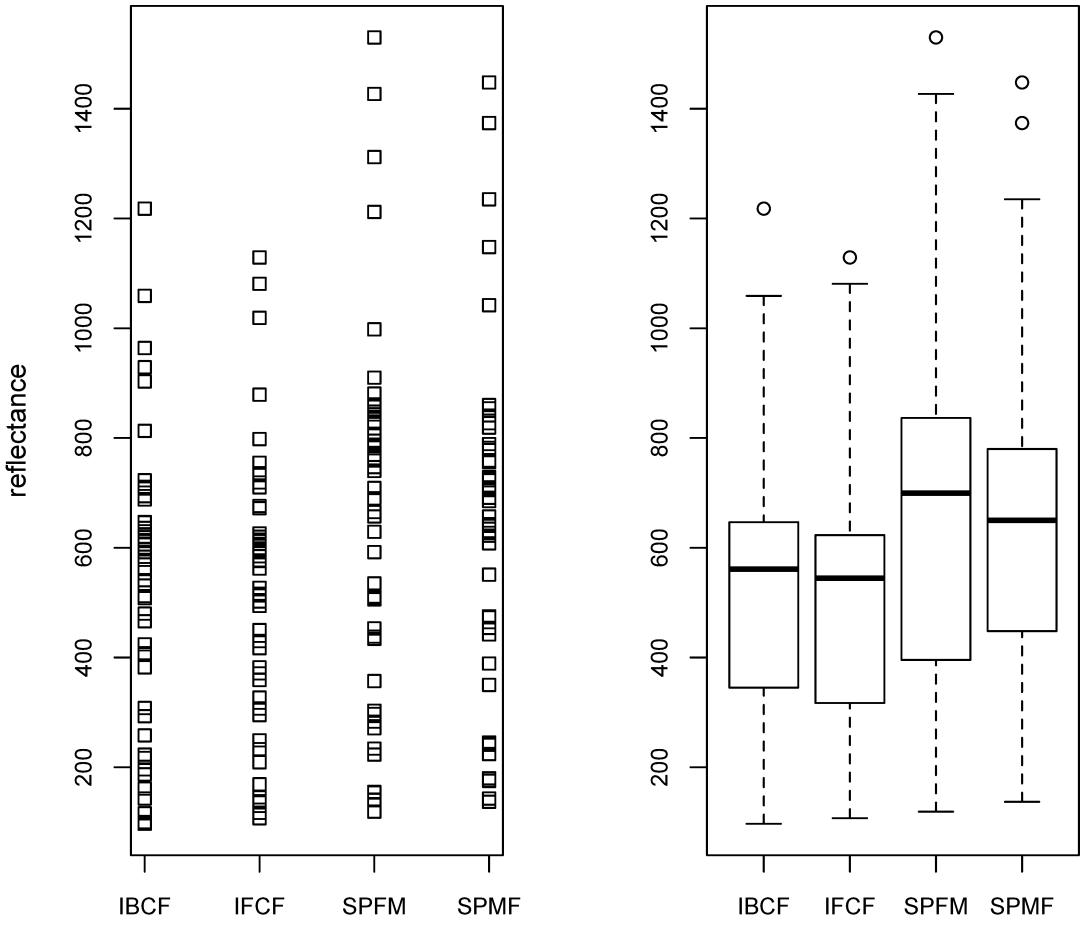

Fig. 5.14. Reflectance (numbers of clicks per hour) of 4 groups in online A/B tests and the corresponding boxplot. The training dataset is collected in the past 40 days, containing 400 million viewing records, involving about 12,500,000 users and 100,000 videos, and the A/B Tests are run on 50865 actual users.

TABLE 5.5
ANALYSIS OF VARIANCE (ANOVA) OF THE 4 METHODS IN ONLINE A/B TESTS, THE TRAINING DATASET IS COLLECTED IN THE PAST 40 DAYS, CONTAINING 400 MILLION VIEWING RECORDS, INVOLVING ABOUT 12,500,000 USERS AND 100,000 VIDEOS, AND THE A/B TESTS ARE RUN ON 50865 ACTUAL USERS.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Operator	3	716100	238700	2.859	0.0383
Residuals	188	15694419	83481		

TABLE 5.6
TUKEY'S MULTIPLE COMPARISON TEST FOR THE ONLINE A/B TESTS. THE TRAINING DATASET IS COLLECTED IN THE PAST 40 DAYS, CONTAINING 400 MILLION VIEWING RECORDS, INVOLVING ABOUT 12,500,000 USERS AND 100,000 VIDEOS, AND THE A/B TESTS ARE RUN ON 50865 ACTUAL USERS.

Operator	diff	lwr	upr	p adj
IFCF-IBCF	-15.60417	-168.48373	137.2754	0.9934966
SPFM-IBCF	122.50000	-30.37957	275.3796	0.1644060
SPMF-IBCF	103.72917	-49.15040	256.6087	0.2965946
SPFM-IFCF	138.10417	-14.77540	290.9837	0.0924009
SPMF-IFCF	119.33333	-33.54623	272.2129	0.1830766
SPMF-SPFM	-18.77083	-171.65040	134.1087	0.9888125

TABLE 5.7
THE RESULTS OF ONLINE A/B TESTS, THE TRAINING DATASET IS COLLECTED IN THE PAST 40 DAYS, CONTAINING 400 MILLION VIEWING RECORDS, INVOLVING ABOUT 12,500,000 USERS AND 100,000 VIDEOS, AND THE A/B TESTS ARE RUN ON 50865 ACTUAL USERS.

Model	IBCF	IFCF	SPMF	SPFM
# Clicks (Sunday)	14248	14739	17137	17627
# Clicks (Monday)	10273	10531	12825	13421
CTR (%)	14.1	14.5	17.2	17.9
Precision @10 (%)	19.2	20.2	23.3	24.4

10-video recommendation lists over the total number of videos clicked by users via all methods, such as via front pages and outside links. Precision@10 is the average ratio of clicked videos out of the 10 recommended videos for all users. Note that here CTR and Precision@10 are only calculated for those users who clicked at least one recommended video. As shown in Table. 5.7, the two groups taking recommendations from SPMF and SPFM have both higher CTR and Precision@10, which means they are more likely to watch a recommended video. In the meantime, from all of the above results, we observe that IFCF [5], as a method dedicated to handle implicit feedback, only slightly outperforms item-based collaborative filtering.

Chapter 6

Concluding Remarks

In this thesis, we propose separating-plane factorization models, an scalable approach to large-scale recommendation problems based on implicit data such as video viewing history. Our model does not rely on artificial imputation, can generate discriminative preference predictions in a wider range than traditional matrix factorization, yet without increasing complexity . We implemented the algorithms on Tencent's TDW cluster with various practical optimizations. Offline evaluations based on training sets involving 12.5 millions users and 100,000 videos show that our scheme outperform various state-of-the-art methods and can accomplish the learning task in 2-3 hours. Further online A/B split tests on 50,865 real users in a 2-day period reveal that our scheme increased CTR by 25% over item-based collaborative filtering and by 23% over implicit-feedback collaborative filtering (IFCF), a scheme available in Spark's MLlib.

References

[1] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 2, 2011.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[3] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[4] T. Vanderbilt, "The science behind the netflix algorithms that decide what you will watch next," http://www.wired.com/2013/08/qq_netflix-algorithm/, 2013.

[5] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 2008, pp. 263–272.

[6] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 502–511.

[7] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 273–282.

[8] H.-F. Yu, M. Bilenko, and C.-J. Lin, "Selection of negative samples for one-class matrix factorization."

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[10] C. C. Aggarwal, "Model-based collaborative filtering," in *Recommender Systems*. Springer, 2016, pp. 71–138.

[11] A. Bellogin and J. Parapar, "Using graph partitioning techniques for neighbour selection in user-based collaborative filtering," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 213–216.

[12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*. Springer, 2007, pp. 291–324.

[13] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

[14] Y. Koren, "The bellkor solution to the netflix grand prize," *Netflix prize documentation*, vol. 81, pp. 1–10, 2009.

[15] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 880–887.

[16] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[17] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, "The yahoo! music dataset and kdd-cup'11." in *KDD Cup*, 2012, pp. 8–18.

[18] S. Rendle, "Social network and click-through prediction with factorization machines," in *KDD-Cup Workshop*, 2012.

[19] D. Kelly and J. Teevan, "Implicit feedback for inferring user preference: a bibliography," in *ACM SIGIR Forum*, vol. 37, no. 2. ACM, 2003, pp. 18–28.

[20] D. Lim, J. McAuley, and G. Lanckriet, "Top-n recommendation with missing implicit feedback," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 309–312.

[21] "Collaborative filtering - spark.mllib," http://spark.apache.org/docs/latest/mllib-collaborative-filtering.htm, 2014.

[22] C.-K. Hsieh, L. Yang, H. Wei, M. Naaman, and D. Estrin, "Immersive recommendation: News and event recommendations using personal digital traces," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 51–62.

[23] Z. Zhao, Z. Cheng, L. Hong, and E. H. Chi, "Improving user topic interest profiles by behavior factorization," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1406–1416.

[24] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador, "Alleviating the new user problem in collaborative filtering by exploiting personality information," *User Modeling and User-Adapted Interaction*, pp. 1–35, 2015.

[25] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. of SIGIR*, vol. 16, 2016.

[26] B. Wang, M. Ester, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "Which to view: Personalized prioritization for broadcast emails," in *Proceedings of the 25th*

International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016, pp. 1181–1190.

[27] C. H. Lin, E. Kamar, and E. Horvitz, "Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing." in *AAAI*, 2014, pp. 908–915.

[28] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 739–748.

[29] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 831–840.

[30] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[31] W.-T. Kuo, Y.-C. Wang, R. T.-H. Tsai, and J. Y.-j. Hsu, "Contextual restaurant recommendation utilizing implicit feedback," in *Wireless and Optical Communication Conference (WOCC), 2015 24th*. IEEE, 2015, pp. 170–174.

[32] W. Pan, H. Zhong, C. Xu, and Z. Ming, "Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks," *Knowledge-Based Systems*, vol. 73, pp. 173–180, 2015.

[33] W. Pan, Z. Liu, Z. Ming, H. Zhong, X. Wang, and C. Xu, "Compressed knowledge transfer via factorization machine for heterogeneous collaborative recommendation," *Knowledge-Based Systems*, vol. 85, pp. 234–244, 2015.

[34] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 283–292.

[35] R. Das, A. Neelakantan, D. Belanger, and A. McCallum, "Chains of reasoning over entities, relations, and text using recurrent neural networks," *arXiv preprint arXiv:1607.01426*, 2016.

[36] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 85–96.

[37] M. A. Carreira-Perpinan and G. Hinton, "On contrastive divergence learning." in *AISTATS*, vol. 10. Citeseer, 2005, pp. 33–40.

[38] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[39] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines." in *AISTATS*, vol. 1, 2009, p. 3.

[40] Y. Liu, Q. Tong, Z. Du, and L. Hu, "Content-boosted restricted boltzmann machine for recommendation," in *International Conference on Artificial Neural Networks*. Springer, 2014, pp. 773–780.

[41] K. Georgiev and P. Nakov, "A non-iid framework for collaborative filtering with restricted boltzmann machines." in *ICML (3)*, 2013, pp. 1148–1156.

[42] K. Li, J. Gao, S. Guo, N. Du, X. Li, and A. Zhang, "Lrbm: A restricted boltzmann machine based approach for representation learning on linked data," in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 300–309.

[43] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and W. Cao, "Deep modeling of group preferences for group-based recommendation." in *AAAI*, 2014, pp. 1861–1867.

[44] N. Phan, D. Dou, B. Piniewski, and D. Kil, "Social restricted boltzmann machine: Human behavior prediction in health social networks," in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2015, pp. 424–431.

[45] 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.

[46] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-nmf model for learning hidden representations." in *ICML*, 2014, pp. 1692–1700.

[47] 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*. ACM, 2015, pp. 1235–1244.

[48] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Parallel matrix factorization for recommender systems," *Knowledge and Information Systems*, vol. 41, no. 3, pp. 793–819, 2014.

[49] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[50] R. S. Xin, D. Crankshaw, A. Dave, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "Graphx: Unifying data-parallel and graph-parallel analytics," February 2014.

[51] "Tdw: Tencent open source distributed data warehouse," http://prog3.com/article/1970-01-01/2819892, 2014.

[52] 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*. ACM, 2008, pp. 426–434.

[53] B. Hidasi and D. Tikk, "Fast als-based tensor factorization for context-aware recommendation from implicit feedback," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 67–82.

[54] I. Pilászy and D. Tikk, "Recommending new movies: even a few ratings are more valuable than metadata," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 93–100.