Advanced Factorization Models for Recommender Systems

Loni, Babak

DOI
10.4233/uuid:0b91c68f-4da7-4745-8d08-c39c0bb00e81

Publication date
2018

Document Version
Final published version

Citation (APA)
Loni, B. (2018). *Advanced Factorization Models for Recommender Systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:0b91c68f-4da7-4745-8d08-c39c0bb00e81

Advanced Factorization Models for Recommender Systems

Babak Loni

Advanced Factorization Models for Recommender Systems

ADVANCED FACTORIZATION MODELS FOR RECOMMENDER SYSTEMS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen ,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 12 December 2018 om 15:00 uur

door

Babak LONI

Master of Science in Computer Science,
Technische Universiteit Delft, Nederland,
Geboren te Teheran, Iran

Dit proefschrift is goedgekeurd door de promotoren

Prof. dr. A. Hanjalic
Prof. dr. M. A. Larson

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. A. Hanjalic,	Technische Universiteit Delft
Prof. dr. M. A. Larson,	Technische Universiteit Delft
Onafhankelijke leden:	
Prof. dr. G.J. Houben,	Technische Universiteit Delft
Prof. dr. M. Reinders,	Technische Universiteit Delft
Prof. dr. A. De Vries,	Radboud Universiteit Nijmegen
Prof. dr. J. Wang,	University College London
Prof. dr. J. Jose,	University of Glasgow

Keywords: recommender systems, collaborative filtering, factorization machines

Printed by: Ridderprint BV | `www.ridderprint.nl`

Cover Design: Mahboobeh Goudarzi

ISBN 978-94-6375-232-9

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

To my parents, Soodabeh and Mostafa.

Science is a wonderful thing
if one does not have to earn one's living at it.

Albert Einstein

Contents

SUMMARY

Recommender Systems have become a crucial tool to serve personalized content and to promote online products and media, but also to recommend restaurants, events, news and dating profiles. The underlying algorithms have a significant impact on the quality of recommendations and have been the subject of many studies in the last two decades. In this thesis we focus on factorization models, a class of recommender system algorithms that learn user preferences based on a method called factorization. This method is a common approach in Collaborative Filtering (CF), the most successful and widely-used technique in recommender systems, where user preferences are learnt based on the preferences of similar users.

We study factorization models from an algorithmic perspective to be able to extend their applications to a wider range of problems and to improve their effectiveness. The majority of the techniques that are proposed in this thesis are based on state-of-the-art factorization models known as Factorization Machines (FMs).

In recommender systems, factorization is typically applied to a matrix, referred to as the user-item matrix, that reflects the interactions between users and items. Our first proposal is a set of algorithms, based on FMs, that exploits information that is present beyond the user-item matrix and that is not exploited otherwise with conventional matrix factorization. We show that such algorithms are able to improve the efficiency and the accuracy of the recommendations.

Our second proposal is to extend the applicability of FMs to ranking problems in recommender systems. FMs are originally designed to address the rating prediction problem, where the underlying model is optimized to learn from and to predict user ratings. Ranking problems, on the other hand, have a rather different view and approach to generate recommendations. They are optimized to learn a ranking for items and can be trained not only using explicit ratings but also using binary or unary user feedback, making them a favorable approach to create recommendations when explicit user feedback is not available. Our second proposal aims to combine the flexibility and expressiveness of FMs with advantages of ranking models, and to benefit from both approaches. We propose an adapted optimization technique to be able to properly exploit FMs to implement ranking. This proposal is later extended with further adaptation to be able to effectively learn from multiple types of positive feedback. We show that the underlying signal (such as click or share) through which user provides feedback, contains useful information that is not typically exploited by conventional CF models. Our proposal is able to distinguish such signals from each other to learn models that are more accurate representations of user preferences.

Our third proposal turns to the underlying training algorithm in FMs and aims to learn the importance of features with additional weight parameters. This proposal, referred to as Weighted Factorization Machines (WFM), is applied for both scenarios of rating prediction and ranking, and their applications on context-aware and cross-domain recommendations. The ability of WFM to learn weights can avoid a time-consuming search to find optimal weights for features. WFM improves the accuracy of rankings and maintains competitive accuracy for rating prediction, compared to the state-of-the-art approaches.

The last chapter of this thesis proposes a set of ideas based on the insights that are learned in the course of our research on factorization models, and can be further investigated in future studies.

SAMENVATTING

Recommender-systemen, systemen die aanbevelingen doen, zijn een cruciaal hulpmiddel geworden voor het aanbieden van gepersonaliseerde inhoud, en voor het promoten van online producten en media, maar ook om restaurants, evenementen, nieuws en datingsprofielen aan te bevelen. De onderliggende algoritmen hebben een significante invloed op de kwaliteit van aanbevelingen en zijn het onderwerp geweest van vele studies in de afgelopen twee decennia.

In dit proefschrift richten we ons op factorisatiemodellen, een klasse van recommendersysteemalgoritmen die gebruikersvoorkeuren leren door een methode dat *factorization* of factorisatie wordt genoemd. Deze methode is een gebruikelijke aanpak in *Collaborative Filtering* (CF), de meest succesvolle en meestgebruikte techniek in recommendersystemen, waarbij gebruikersvoorkeuren worden geleerd op basis van de voorkeuren van vergelijkbare gebruikers.

We bestuderen factorisatiemodellen vanuit een algoritmisch perspectief om hun toepassing uit te breiden naar een groter aantal problemen en om hun effectiviteit te verbeteren. De meeste technieken die in dit proefschrift worden voorgesteld zijn gebaseerd op state-of-the-art factorisatiemodellen die bekend zijn als *Factorization Machines* (FM's).

In recommendersystemen wordt het factorisatieproces meestal toegepast op een matrix, die de interacties tussen gebruikers en items weergeeft. Deze matrix wordt de *user-item*-matrix genoemd. Ons eerste voorstel is een reeks algoritmen, gebaseerd op FM's, die informatie benut die buiten de user-item-matrix aanwezig is en die anders niet wordt benut met conventionele factorisatiemodellen. We laten zien dat dergelijke algoritmen in staat zijn om de efficiëntie en nauwkeurigheid van de aanbevelingen te verbeteren.

Ons tweede voorstel is om de toepasbaarheid van FM's op rangordeproblemen in recommendersystemen uit te breiden. FM's zijn oorspronkelijk ontworpen om het probleem om beoordelingen te voorspellen aan te pakken, waarbij het onderliggende model geoptimaliseerd is om beoordelingen van gebruikers te leren en te kunnen voorspellen. Rangordeproblemen hebben daarentegen een nogal andere visie en aanpak om aanbevelingen te genereren. Ze zijn geoptimaliseerd om items te rangschikken en kunnen niet alleen worden getraind met behulp van expliciete beoordelingen, maar ook met behulp van binaire of unaire gebruikersfeedback, waardoor ze een gunstige manier zijn om aanbevelingen te genereren wanneer expliciete gebruikersfeedback niet beschikbaar is. Ons tweede voorstel is erop gericht om de flexibiliteit en expressiviteit van FM's te combineren met de voordelen van rangschikkingsmodellen en dus te profiteren van beide aanpakken. We stellen een aangepaste optimalisatietechniek voor om FM's goed te kunnen exploiteren en hiermee te kunnen rangschikken. Dit voorstel wordt later uitgebreid met een verdere aanpassing om effectief te kunnen leren van meerdere soorten positieve feedback. We laten zien dat het onderliggende signaal (zoals muisklikken of deelacties) waarmee de gebruiker impliciet feedback geeft, nuttige informatie bevat die normaal niet door conventionele CF-modellen wordt gebruikt. Ons voorgestelde aanpak is in staat om dergelijke signalen van elkaar te onderscheiden om modellen te leren die accuratere representaties zijn van gebruikersvoorkeuren.

Ons derde voorstel keert zich tot het onderliggende trainingsalgoritme in FM's en heeft als doel om het belang van *features* (kenmerken) met additionele gewichtsparameters te leren. Dit voorstel, *Weighted Factorization Machines* (WFM) genoemd, wordt toegepast voor beide scenario's: het voorspellen en rangschikken van beoordelingen en hun toepassingen op *context-aware* en *cross-domain* aanbevelingen. Het vermogen van WFM om gewichten te leren kan een tijdrovende zoekoperatie om optimale gewichten voor features te vinden voorkomen. WFM verbetert de nauwkeurigheid bij het rangschikken en behoudt een concurrerende nauwkeurigheid bij het voorspellen van beoordelingen in vergelijking met state-of-the-art oplossingen.

Het laatste hoofdstuk van dit proefschrift biedt een reeks ideeën op basis van inzichten die zijn verkregen tijdens ons onderzoek naar factoriseringsmodellen en die in toekomstige studies verder kunnen worden onderzocht.

I

PRELUDE

1

INTRODUCTION

Factorization models are state-of-the-art models for collaborative filtering, a class of recommender systems algorithms that generate recommendations by automatic prediction (filtering) of user's interest based on the interests of other users. In this thesis, we focus on Factorization Machines (FMs), a general factorization framework that can mimic other factorization models by feature engineering.

In this chapter we first provide a short introduction about recommender systems and collaborative filtering. We then briefly describe factorization models and in particular Factorization Machines and further discuss over advantages and limitations of FMs. We further introduce some problems in recommender systems and explain how we address those problems with advanced factorization models, as introduced in the technical chapters of this thesis.

1.1. RECOMMENDER SYSTEMS

Recommender Systems are a subclass of information retrieval systems that seek to generate recommendations that fit users' interest. Recommender systems are essential tools in promoting products and services in a wide range of applications including online shopping, movie and music streaming services, social networks, professional networks and news services. Recommendations in YouTube account for about 60% of clicks in its home page [21]. Netflix reports [32] that 80% of its movies are discovered through recommendations. Spotify [42] and Pandora[1], two giant music streaming services, also report the critical role of recommender systems in the success of their services.

Recommender Systems algorithms are generally classified into two groups: content based methods and Collaborative Filtering (CF). Content-based methods utilize the features of items to recommend similar items to the item that user consumed. CF approaches generally learn user's interest by relying on the behavior of similar users. CF is generally more popular than content-based methods due to its superior performance and accuracy [22]. CF methods however, suffer from the *cold-start* problem, that is, when the system does not have enough information about new users and items. In such cases the content-based methods are superior [105]. In many real-world scenarios, *hybrid* methods, combination of content-based and collaborative filtering methods are used [2, 14].

Recommender system algorithms typically rely on predicting a *utility* for user-item interactions. The items with high utility with respect to a user are recommended to the user. Most of the early recommender systems are addressing the *rating prediction* problem, that is, they are optimized to predict the rating that a user might give to an item. In such cases user rating is considered as the utility of user-item interactions. The predicted utility is then used to rank items and the top items are recommender to the user [69]. Another class of recommender system algorithms are optimized to directly *rank* items based on the preferences of the users. This class of algorithms is particularly useful when explicit user ratings are not available. In such cases implicit user feedback (such as 'click', 'bookmark' or 'add-to-basket') are exploited in order to learn a ranked list.

1.2. COLLABORATIVE FILTERING

Collaborative Filtering (CF) is the most popular and successful method in recommender systems, which is based on the idea that users with common interests in the past are likely to have similar interests in future. Most of the successful and recent recommender systems methods are based on CF methods [114]. Collaborative Filtering is also the winner of the Netflix contest competition [10]. Recently efforts have been done to combine CF methods with deep learning [36, 123] to benefit from the advantages of the two techniques.

In CF techniques, user-item interactions are typically represented by a matrix where rows represent the users, columns represent the items and each cell contains the rating of a user to an item, if available. The objective of a rating prediction problem is to predict the ratings in the missing cells. CF methods can be classified into two categories: memory-based and model-based methods. In the following we provide a brief overview

[1] http://www.theserverside.com/feature/How-Pandora-built-a-better-recommendation-engine

of both techniques.

1.2.1. MEMORY-BASED COLLABORATIVE FILTERING

Memory-based algorithms, also known as neighborhood methods, directly use the stored ratings to predict the utility of user-item interactions. Memory-based methods take into account the *similarities* of users or items to calculate ratings. Recommendations can be generated from a user-centric point of view, where the predictions are calculated as weighted average of ratings given by similar (neighbor) users. In this case the similarities of users are considered as weight parameters. From an item-centric point of view, the similarities of all items to a target item i are calculated and then regarded as weight parameters to calculate the weighted average of user's rating for the target item. Similarities can be calculated using Pearson correlation method or by using the cosine similarity method. Users (items) are represented by feature vectors where features are items (users) and the ratings are the value of the features.

Memory-based methods are simple to implement, and the recommendations are easy to explain. However, theses methods have limited scalability on large datasets [114].

1.2.2. MODEL-BASED COLLABORATIVE FILTERING

Model-based approaches takes another perspective to predict user ratings. In contrast to the memory-based methods where the predictions are calculated based on the stored ratings, model-based approaches use the available ratings to *learn* a model to predict the utility of user-item interactions.

Among several model-based CF methods, factorization models [53] are the most popular model-based CF methods and have attracted majority of CF research due to their superior accuracy and scalability, as witnessed by the model that won the Netflix competition [8]. Other model-based methods include Maximum Entropy models [132], Latent Dirichlet Allocation [56], Singular Value Decomposition [12] and mixture models [52]. In this section we briefly review factorization models.

Factorization models learn a representation of users and items in a shared latent feature space. Predictions are calculated based on how similar the users and items are in this latent space. The features in the latent space are called *factors* and the process of learning those factors is referred as *factorization*. Since user-item interactions are typically represented as a matrix, this technique is also referred as *Matrix Factorization (MF)*. The utility of a user-item interaction is calculated as the inner product of the two vectors. In rating prediction problems, user's rating is considered as the utility of interactions and the model learns to predict the ratings. Figure 1.1 illustrates a user-item matrix and the two factorized matrices of P and Q, which represent the learned factors for users and items.

Early matrix factorization techniques use Singular Value Decomposition (SVD), a popular technique to factorize a matrix to a latent semantic space [12]. The SVD technique however, cannot deal with missing values and thus the missing values are typically filled with 0, resulting to a poor performance. Moreover, the SVD technique is very prone to over-fitting and causes the model to be less general. Recent techniques learn the latent factors only from the observed rating by solving an optimization problem. For the rating prediction problem the latent factors are learned by optimizing a regularized loss

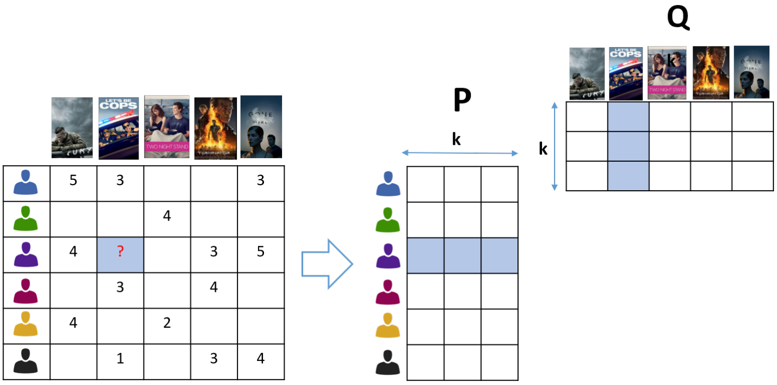


Figure 1.1: Representation of a general matrix factorization model. User and items are factorized (projected) into a joint latent space. A missing rating (shaded cell in this figure) is predicted by calculating the inner product of the learned factors of the corresponding user and item. Parameter *k* is the dimentionality of factorization (number of factors).

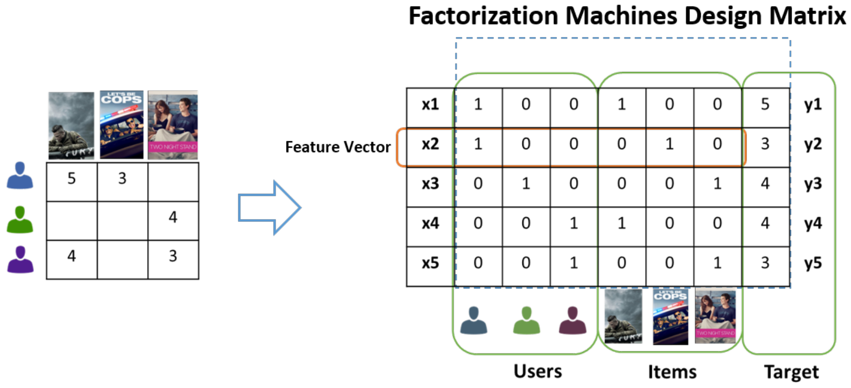
function (typically squared loss) based on the training data. The optimization problem can be solved by different techniques, among which Stochastic Gradient Descent (SGD) and Alternative Least Square (ALS) are two mostly used techniques [55].

1.3. Advanced Factorization Models

Factorization models are not only limited to rating prediction problems nor to the user-item matrix. Real-world scenarios can be more complicated and capabilities of factorization models are also more than predicting users' rating. In this section we present advanced factorization models that can address wider range of problems in recommender systems and can leverage more recent state-of-the-art techniques and learning algorithms. We first introduce Factorization Machines (FMs), an advanced factorization model that is the basis of most of the models that we proposed in this thesis and further discuss about their advantages and limitations. We then present three challenges that we covered in this thesis, review some related work and discuss how they can be addressed with advanced factorization models.

1.3.1. Factorization Machines

Factorization Machines [89], are general factorization models that can mimic other factorization models by *feature engineering*, i.e., using domain knowledge to create features. In contrast to matrix factorization models where user-item interactions are represented by a matrix, in FMs each user-item interaction is represented by a feature vector similar to supervised machine learning approaches. Such representation makes FMs a flexible model since additional information can be encoded to feature vectors as auxiliary features and the underlying model of FMs can seamlessly exploit such features and possibly learn a more accurate model. For example, if context of user-item interactions are available in a dataset, such context can be encoded as auxiliary features in FMs and a context-



5	3	
		4
4		3

x1	1	0	0	1	0	0	5	y1
x2	1	0	0	0	1	0	3	y2
x3	0	1	0	0	0	1	4	y3
x4	0	0	1	1	0	0	4	y4
x5	0	0	1	0	0	1	3	y5

Figure 1.2: In Factorization Machines user-item interactions are represented in terms of feature vectors. Each cell (rating) in the user-item matrix is represented by a single feature vector **x** and the rating is considered as the output of the model (y) corresponding to that feature vector.

aware model can be learned. In [92], Rendle showed how several factorization models such as matrix factorization, SVD++ [53] and attribute-aware matrix factorization [26] can be modeled by factorization machines based on feature engineering. Factorization Machines have also been successfully applied in context-aware recommendation [99], Cross-Domain collaborative filtering [78] and Social Recommendations [70]. In addition to their flexible representation, FMs are accurate and scalable models. In the past few years, FMs became very popular not only in the academic world, but also as a popular solution for industries. Recently, FMs have been offered as a managed machine learning service in Amazon SageMaker [2]. Most of the advanced factorization models that we propose in this thesis are based on FMs.

The standard FMs are designed for data with explicit feedback (such as user ratings). A user rating in FMs is represented by a sparse feature vector where each user and item corresponds to a feature (thus the number of features is the sum of cardinalities of users and items). The target (output) of the FMs model is considered to be the rating, which we want to predict. Figure 4.1 illustrates how a user-item matrix can be represented by feature vectors in FMs. The set of feature vectors with their corresponding model output (rating) is referred as FMs *design matrix*.

The underlying model of FMs learns to predict the user's rating given a user-item feature vector. Similar to the case of matrix factorization, the model parameters in FMs are learned by optimizing an objective function, which is defined based on the training data. The objective function is typically a squared loss over training samples together with regularization terms to prevent over-fitting. The optimization can be done with three different methods [92]: Stochastic Gradient Descent (SGD), Alternating Least Square (ALS) and Markov Chain Monte Carlo (MCMC).

Factorization Machines have several advantages compared to other factorization models. In [72] we listed three advantages for FMs, namely, generalization, expressiveness

[2] `https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html`

and performance. FMs are general model since they learn from interactions between any pairs of features (not only user-item interactions). Furthermore, they can learn the model from higher-order interactions (for example user-item-context interactions). FMs are expressive due to their flexible representation model. Several factorization techniques can be represented by factorization machines by feature engineering. And finally FMs are accurate and scalable models as witnessed by several studies [13, 47, 92, 93, 99].

Despite their advantages, the standard model of FMs are only optimized for rating prediction problems and might not be necessarily an effective model if explicit feedback is not available. Furthermore, FMs offer a possibility to exploit additional information in terms of auxiliary features. However, the extent to which auxiliary features are effective is not thoroughly studied. In Section 1.3.3, we introduce some advance factorization models based on FMs that cover some of the issues we mentioned.

1.3.2. BEYOND THE USER-ITEM MATRIX

Factorization Machines are not the only models that can exploit information beyond the user-item matrix. In the past few years a notable body of research has been dedicated to leveraging additional information that is available beside the user-item matrix. The additional source of information can be related to users and items (e.g., user demographic, item features) or it can be pertained to the user-item interactions, typically reflecting the context of such interactions.

The side information can be used to pre- or post-filter recommendation results or can be exploited within the recommendation algorithms [84]. A comprehensive overview of the collaborative filtering methods that exploit information beyond the user-item matrix can be found in [112]. In their work, Shi et al. introduce three categories of algorithms that exploit side information beyond the user-item matrix: extensions for memory-based techniques, extensions for model-based techniques and graph-based collaborative filtering methods. In the first category, side information is used to better estimate similarities between users and items [14, 122]. The second category exploits side information in the underlying model. An example in this group is Collective Matrix Factorization (CMF) [115] where user-item matrix and side information are jointly factorized. Factorization of side information (such as movie-genre matrix) can alleviate sparsity problem and lead to more accurate latent factors. The third category leverages graph-based algorithms where typically connections between users and items are exploited to deal with the cold-start problem. An example of such approaches is TrustWalker [44], where knowledge of the trust network among users are exploited based on a random-walk algorithm to alleviate sparsity problems and to learn a more accurate model.

As mentioned earlier, Factorization Machines are also able to leverage side information beyond the user-item matrix. The advantage of FMs compared to other models is that the side information can be exploited by embedding it as additional features and in contrary to the other approaches the underlying model remains the same or can be adapted with minor changes. An example of FMs that exploits side information without adapting the underlying model is [99] where FMs have been used for context-aware recommendations.

The side information can be even beyond user, item and context features and can be derived from other domains. In **Chapter 2** we propose a method based on FMs that exploits user-item interactions in auxiliary domains (for example user feedback in an online movie store) to improve recommendations in a target domain (such as a book store). This method and similar approaches are also referred as *cross-domain* collaborative filtering methods. A brief survey of such approaches can be found in [63].

1.3.3. BEYOND RATING PREDICTION

In many practical situations user feedback is not available in terms of explicit ratings and only *implicit feedback* from users such as click or download is collected. Implicit feedback is typically unary and positive-only. Examples of feedback in social media and online shopping that can be considered as positive feedback are 'like', 'add-to-favorites' and 'bookmark'. Examples of implicit feedback, which are typically regarded as positive signal, are 'click', 'share' and 'add-to-basket'. In some systems negative feedback can also be collected [46, 85]. Due to the absence of explicit ratings in such systems, algorithms that are optimized to predict ratings are not effective for generating recommendations. In such scenarios, the recommender model is trained to learn the optimal *ranking* for each user. Such approaches are also referred to as *top-N* recommendation models [24], as they recommend the top-N items in the ranked list. Learning-to-rank methods in recommender systems can be very effective also for datasets with explicit feedback [111].

For datasets with implicit feedback, collaborative filtering can also be implemented with memory-based or model-based approaches. Similar to rating prediction problems, memory-based methods [69] can generate recommendations by estimating similarities of users based on their implicit feedback. Alternatively, item similarities [68] can be estimated to recommend similar items to the items that user consumed.

Most of the model-based approaches for datasets with implicit feedback are based on factorization. Factorization-based approaches can be classified into three groups based on the underlying optimization models. The first group are *point-wise* methods, which learn to predict the utility of user-item interactions (i.e., predicting points) based on a loss function that is optimized for prediction. A successful example from this group is Weighted Regularized Matrix Factorization (WRMF) [39]. In this method user feedback is regarded as binary data and the user and the item factors are leaned with an optimization technique similar to the matrix factorization. The second group are *pairwise* methods. An outstanding example in this group is Bayesian Personalized Ranking (BPR) [97], which learns user and item factors by learning to rank pairs of items with respect to users. The third group are *list-wise* models, where model parameters are learned by directly optimizing a ranking metric such as Mean Reciprocal Rrank (MRR) [109], Expected Reciprocal Rank (ERR) [107] or Mean Average Precision (MAP) [108].

In **Chapter 3,** we introduce model-based approaches for learning from implicit feedback, and in particular our proposed model, which is based on Factorization Machines. In this chapter we borrow the pair-wise optimization idea of BPR and apply it to FMs. We refer to this model as FM-Pair. We further evaluate the effectiveness of this approach on two problems of context-aware and cross-domain recommendation (for datasets with implicit feedback).

Chapter 4, which is built based on the FM-Pair model, adapts the optimization process of FM-Pair in such a way that multiple types of user feedback (such as click, or like) can be exploited simultaneously. The proposed technique in this chapter focuses on the process that samples the training data points. Among the related work in this area, Gantner et al. [29] and Rendle et al. [95] propose two approaches where the underlying algorithm of BPR is extended to sample better items from training set resulting to a faster convergence of the algorithm. A similar direction of research exploits active learning strategies to elicit "useful" user feedback to generate better recommendations. A survey of such techniques can be found in the work of Elahi et al. [25]. In Chapter 4, we exploit the type of user feedback to sample more informative training points in order to learn a more accurate representation of user preferences. We refer to this process as *multi-channel* sampling. The performance of the multi-channel sampling method is also evaluated in terms of item coverage and time-complexity.

1.3.4. ADVANCED LEARNING MODELS

The underlying learning algorithms that are used to learn a collaborative filtering model have significant impact on the quality and scalability of the models. A vast majority of research in collaborative filtering is devoted to improving the underlying algorithms of different CF models or introducing new machine learning techniques for collaborative filtering. A comparative study of several collaborative filtering algorithms can be found in [59].

In the scope of factorization, proposed techniques either introduce extensions that enables the model to exploit additional data, or improve the underlying model of learning. Bayesian Matrix Factorization [86] and TagiCofi [131] are examples of extensions to matrix factorization that are capable of exploiting additional data. An example of algorithms that try to improve the underlying factorization models is Non-negative Matrix Factorization [79], which proposes to solve the optimization in matrix factorization with a non-negativity constraint.

In the line of research on Factorization Machines, some studies propose to improve the underlying model of FMs. Gaussian Process Factorization Machines [80] is an extension of FMs where interaction between features are modeled with Gaussian kernels. Field-aware Factorization Machines [48] is another extension of FMs, where several representations of features are learned depending on the *field* (group) that the feature belongs to. For example, assuming that user, item and context feature each represent a group of features, two set of latent factors are learned for users where the first set is used to interact with items and the second set is used to interact with context. Attentional Factorization Machines [126], employs a neural network model where additional parameters are learned to reflect the importance of interactions between features in FMs. In **Chapter 5** we introduce Weighted Factorization Machines, an extension to FMs model where weight parameters are learned for groups of features. Such weight parameters can control the contribution of different feature and prevent the model to be negatively influenced by noisy data.

1.4. CONTRIBUTIONS OF THIS THESIS

The purpose of this thesis is to study and extend the capabilities of factorization models to a wider range of problems in recommender systems and benefit from their advantages to address real-world recommendation scenarios. The majority of the factorization techniques that we developed in the course of this thesis are based on FMs. In this thesis, we propose solutions, mainly based on FMs, for common scenarios in recommender systems such as cross-domain recommendation, top-N recommendation, and learning from multiple user feedback. Further details about the contribution of this thesis and the research questions that we answered are described below.

1.4.1. OUTLINE

This thesis is divided into seven chapters. These chapters are grouped into several parts, with three core parts (Parts II, III, IV) that represent the technical contribution of this thesis. Each of these three parts consists of one, two or three chapters that study factorization model from one particular aspect. Part II proposes techniques that exploit information that is not limited to the user-item matrix. In part III, we study factorization models that are beyond the rating-prediction problem, and can be applied to ranking problems. Part IV focuses on the learning methods of FMs and proposes improved learning mechanism for rating-prediction and ranking.

Beside the technical parts, in part V we briefly introduce the implementation framework that we developed in the course of this thesis, which is used for the majority of the experiments in this dissertation. Below we outline the scope and contribution of each chapter.

Chapter 1 (this chapter) provides a general overview of recommender systems, outlines the two classes of algorithms for generating recommendations and describes in more details factorization models, a popular and successful model for collaborative filtering. In this chapter we also briefly introduce Factorization Machines and we present the necessary background to the reader so that the story of the thesis becomes clearer.

In **Chapter 2**, we address the problems that can be solved based on the standard model of Factorization Machines for rating prediction. We introduce three different applications of rating prediction with FMs. The first application (Chapter 2A) exploits FMs for the task of cross-domain recommendation. We propose a method to encode information from *auxiliary* domains as additional features in the FM model that is trained on a *target* domain.

The second application (Chapter 2B) exploits additional information that is inherent in the user-item matrix (but not directly exploited by conventional matrix factorizaion) to improve the accuracy of recommendations. We refer to this approach as ‘Free-lunch’ enhancement, since we are not using any additional data beside the user-item interactions.

The third application (Chapter 2C), which is inspired by the first application, proposes an alternative way to train a collaborative filtering model using Factorization Machines. In this approach, instead of training the model with the entire training dataset, the model is trained by directly using a slice of interest from the training data while the rest of data is exploited indirectly by auxiliary features in FMs.

All the above problems are applied for rating prediction problem as the standard model of FMs are designed for rating prediction. In this chapter we answer the following research questions:

- *Can we use Factorization Machines to address the cross-domain recommendation problem? How effective can FMs be for this problem?*

- *Can we use auxiliary features that are extracted from the user-item matrix in FMs and if yes how effective they are?*

- *Can we train a recommender model with FMs more efficiently by training on the 'right' slice of data?*

In **Chapter 3**, we propose to exploit Factorization Machines for ranking problems in collaborative filtering. In this chapter we discuss limitations of the standard FMs model and propose an alternative way to train FMs model when explicit user feedback is not available. This chapter introduces *FM-Pair*, an extension of FMs that enables them to be used for ranking problems (top-N recommendation). We study the effectiveness of this method in two different recommendation scenarios namely, Context-Aware recommendation and Cross-Domain collaborative filtering. Chapter 3 explains the implementation details of our extended FMs model and evaluate this model on several datasets. In this chapter we answer the following research questions:

- *Can we use the standard model of FMs for ranking (instead of rating prediction)? Is that an effective model?*

- *How can we effectively use FMs for learning-to-rank? Can we apply a pairwise optimization model for FMs and use them for datasets with implicit feedback?*

- *If the answer to the previous question is yes, how effective would be such method for problems such as context-aware and cross-domain collaborative filtering?*

In recommender systems different types of user feedback can be collected. **Chapter 4**, which is build based on the previous chapter, goes beyond the conventional training algorithms and propose two methods to exploit multiple types of user feedback (example of positive feedback are 'click', 'like', 'add-to-basket' and so on) to improve recommendations. In this chapter, Factorization Machines have been utilized to effectively learn user preferences from multiple user feedback. We show that conventional integration of auxiliary information as features is not always the best way to exploit additional data with FMs. We then propose an adapted sampling mechanism, to sample data from the training set, in order to better learn the underlying recommender model. This chapter answers the following research questions:

- *Does a recommender model that uses multiple types of feedback generally perform better that a model that does not exploit all feedback channels?*

- *Can we exploit types of feedback as auxiliary features in FMs and how effective such method would be?*

- *Can we use types of feedback to better sample training data from our dataset and if yes, is it more effective than conventional integration of information in FMs (i.e., exploiting such knowledge as auxiliary features)?*

- *What are the different possibilities of sampling? Which method works best on different datasets?*

Chapter 5 turns to the underlying model of Factorization Machines and proposes an extension to the model where the importance of feature can be learned by the model to potentially train a more accurate recommendation model. We propose weighted Factorization Machines (WFM), where the model learns weights for *groups* of features and uses the learned weights to predict the utility of user-item interactions more precisely. In this chapter we propose two adaptations of FMs optimization algorithms for datasets with explicit and implicit feedback to learn the weight parameters that we introduced in the model. Chapter 5 answers the following research questions:

- *Can we learn the importance of features as well as other model parameters in Factorization Machines?*
- *Can we train more accurate recommender models by learning weights for different groups of features? How can it be applied for datasets with implicit feedback?*

In **Chapter 6** we briefly present WrapRec, an evaluation framework for recommender systems, where the majority of the algorithms that we introduce in this dissertation are implemented. In this chapter we also highlight different approaches for evaluation of recommender systems using the WrapRec toolkit and describe capabilities of WrapRec for several types of experiments.

Chapter 7 summarizes this dissertation, draw some conclusions, discusses about the lessons that are learned, and suggests some future directions in this topic.

1.5. How to Read this Thesis

The scientific contribution of this thesis is written in parts II, III and IV. Part V, briefly describes WrapRec, the framework that we developed in the course of our research. Each technical chapter of this thesis (except Chapter 3) is connected to one publication, which is referenced at the beginning of the chapter. In this book, we retain the original form of publications with minor modifications. Each chapter represents an independent work that can be read without necessarily reading previous chapters. As a consequence, there might be similar topics in the introductory and related work sections of the technical chapters and the notation and terminology might vary slightly across them.

1.6. List of Publications

The papers that are published in the course of this thesis is listed below. For the papers that are directly connected to this dissertation, the reference to the corresponding technical chapter is added in parenthesis.

- Loni, Babak, Roberto Pagano, Martha Larson, and Alan Hanjalic. "Top-N Recommendation with Multi-Channel Positive Feedback using Factorization Machines" In *ACM Transactions on Information Systems (TOIS).* Accepted. 2018. (**Chapter 4**)
- Larson, Martha, Alessandro Zito, Babak Loni, and Paolo Cremonesi. "Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms." In *Proceedings of the FATREC Workshop on Responsible Recommendation.* 2017.

- Liang, Yu, Babak Loni, and Martha Larson. "CLEF 2017 NewsREEL Overview: Contextual Bandit News Recommendation.", In *Proceedings of the CLEF 2017 NewsREEL Challenge*. 2017.

- Loni, Babak, Roberto Pagano, Martha Larson, and Alan Hanjalic. "Bayesian personalized ranking with multi-channel user feedback." In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 361-364. ACM, 2016. (**Chapter 4**)

- Loni, Babak, Martha Larson, Alexandros Karatzoglou, and Alan Hanjalic. "Recommendation with the Right Slice: Speeding Up Collaborative Filtering with Factorization Machines." In *RecSys Posters*. 2015. (**Chapter 2C**)

- Loni, Babak, Alan Said, Martha Larson, and Alan Hanjalic. "'Free lunch' enhancement for collaborative filtering with factorization machines." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 281-284. ACM, 2014. (**Chapter 2B**)

- Said, Alan, Babak Loni, Roberto Turrin, and Andreas Lommatzsch. "An Extended Data Model Format for Composite Recommendation." In *RecSys Posters*. 2014.

- Fazeli, Soude, Babak Loni, Alejandro Bellogin, Hendrik Drachsler, and Peter Sloep. "Implicit vs. explicit trust in social matrix factorization." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 317-320. ACM, 2014.

- Loni, Babak, and Alan Said. "WrapRec: an easy extension of recommender system libraries." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 377-378. ACM, 2014. (**Chapter 6**)

- Said, Alan, Simon Dooms, Babak Loni, and Domonkos Tikk. "Recommender systems challenge 2014." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 387-388. ACM, 2014.

- Basak, Debarshi, Babak Loni, and Alessandro Bozzon. "A Platform for Task Recommendation in Human Computation." In *RecSys 2014 CrowdRec Workshop*. ACM. 2014.

- Fazeli, Soude, Babak Loni, Hendrik Drachsler, and Peter Sloep. "Which recommender system can best fit social learning platforms?" In *European Conference on Technology Enhanced Learning*, pp. 84-97. Springer, Cham, 2014.

- Loni, Babak, Yue Shi, Martha Larson, and Alan Hanjalic. "Cross-domain collaborative filtering with factorization machines." In *European conference on information retrieval*, pp. 656-661. Springer, Cham, 2014. (**Chapter 2A**)

II

Beyond the User-Item Matrix

2A

Cross-Domain Collaborative Filtering with Factorization Machines

Factorization machines offer an advantage over other existing collaborative filtering approaches to recommendation. They make it possible to work with any auxiliary information that can be encoded as a real-valued feature vector as a supplement to the information in the user-item matrix. We build on the assumption that different patterns characterize the way that users interact with (i.e., rate or download) items of a certain type (e.g., movies or books). We view interactions with a specific type of item as constituting a particular domain and allow interaction information from an auxiliary domain to inform recommendation in a target domain. Our proposed approach is tested on a data set from Amazon and compared with a state-of-the-art approach that has been proposed for Cross-Domain Collaborative Filtering. Experimental results demonstrate that our approach, which has a lower computational complexity, is able to achieve performance improvements[1].

[1] This chapter is published as Loni, Babak, Yue Shi, Martha Larson, and Alan Hanjalic. "Cross-domain collaborative filtering with factorization machines." In *European conference on information retrieval*, pp. 656-661. Springer, Cham, 2014.

2A.1. Introduction

Cross-domain Collaborative Filtering (CDCF) methods exploit knowledge from *auxiliary* domains (e.g., movies) containing additional user preference data to improve recommendation on a target domain (e.g. books). While relying on a broad scope of existing data in many cases is a key to relieving the problems of sparse user-item data in the target domain, CDCF can also simultaneously benefits different data owners by improving quality of service in different domains.

In most CDCF approaches (e.g., [64], [82]) it is assumed that user behavior in all domains is the same. This assumption is not always true since each user might have different domains of interest, for example, rating items consistently more frequently or higher in one domain than in another. In a recent work, Hu et al. [38] argue that CDCF should consider the full *triadic* relation *user-item-domain* to effectively exploit user preferences on items within different domains. They represent the user-item-domain interaction with a tensor of order three and adopt a tensor factorization model to factorize users, items and domains into latent feature vectors. The rating of a user for an item in a domain is calculated by element-wise product of user, item and domain latent factors. A major problem of tensor factorization however, is that the time complexity of this approach is exponential as it is $O(k^m)$ where k is the number of factors and m is the number of domains.

In this chapter we exploit the insight that user preferences across domains could be deployed more effectively if they are modeled separately on separate domains, and then integrated to generate a recommendation on the target domain. We therefore address the problem with *factorization machines* (FM) [92], which make such modeling possible. In addition, the FMs are more flexible than the tensor representation regarding the ways of capturing the domain-specific user preferences and could lead to more reliable recommendations. Finally, FMs are polynomial in terms of k and m, making them computationally less expensive than tensor factorization models [92].

FMs have already been applied to carry out CF in a single domain, [92, 99], but have yet to be exploited to address the CDCF problem. Here we apply FMs to cross-domain recommendation in a way that allows them to incorporate user interaction patterns that are specific to particular types of items. Note that in this chapter, we define a domain as a type of item. The set of users is not mutually exclusive between domains, but we assume that their interaction patterns differ sufficiently to make it advantageous to model domains separately. The novel contribution of our work is to propose an extension of FMs that incorporates domains in this pattern and to demonstrate its superiority to single domain approaches and to a state-of-the-art CDCF algorithm.

2A.2. Related Work

Cross-Domain Collaborative Filtering: An overview of CDCF approaches is available in Li [114]. Here, we restrict our discussion of related CDCF approaches to mentioning the advantages of our approach compared to the major classes of existing algorithms. *Rating pattern sharing* algorithms, exemplified by [64], groups users and items into clusters and matches cluster-level rating patterns across domains. The success of the approach depends, however, on the level of sparseness of user-item information per domain. *Latent feature sharing* approaches, exemplified by [82], transfer knowledge between domains via a common latent space and are difficult to apply when more than two domains are involved [38]. *Domain correlation* approaches, exemplified by [110], use common information (e.g., user tags) to link domains and fail when such information is lacking.

Factorization Machines: Factorization Machines (FM) [92] are general models that factorize user-item collaborative data into real valued feature vectors. Most factorization models such as Matrix Factorization can be modeled as a special case of FM [92]. Despite typical CF models where collaboration between users and items are represented by a rating matrix, in factorization machines the interaction between user and item is represented by a feature vector and the rating is considered as class label for this vector. More specifically let's assume that the data of a rating prediction problem is represented by a set S of tuples $(\mathbf{x}, y)$ where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a n-dimensional feature vector and y is its corresponding label. Factorization machines model all interactions between features using factorized interaction parameters. In this chapter we adapted a FM model with order $d = 2$ where only *pairwise* interaction between features are considered. This model can be represented as follows:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} w_{j,j'} x_j x_{j'} \tag{2A.1}$$

where w_j are model parameters and $w_{j,j'}$ are factorized interaction parameters and are defined as $w_{j,j'} = \mathbf{v}_j.\mathbf{v}_{j'}$ where $\mathbf{v}_j$ is k-dimensional factorized vector for feature j. For a FM with n as the dimensionality of feature vectors and k as the dimensionality of factorization, the model parameters that need to be learnt are $\Theta = \{w_0, w_1, \ldots, w_n, v_{1,1}, \ldots, v_{n,k}\}$. Three learning approaches have been proposed to learn FMs [92]: Stochastic Gradient Descent (SGD), Alternating Least-Squares (ALS) and Markov Chain Monte Carlo (MCMC) method. We exploit all 3 methods in this chapter.

2A.3. Cross-Domain CF with Factorization Machines

Assume we are given collaborative data of users and items in m different domains $\{\mathbb{D}_1, \ldots, \mathbb{D}_m\}$. The domains are different based on the type of items that exists in them. While rating information for a user might be very sparse in one domain (e.g. Books), he might have rich collaborative data in another domain (e.g. movies). The purpose of cross-domain CF is to transfer knowledge from different auxiliary domains to a target domain to improve rating predictions in the target domain.

To understand our approach, without loss of generality lets assume $\mathbb{D}_1$ is the target domain and $\{\mathbb{D}_2, \ldots, \mathbb{D}_m\}$ are the auxiliary domains. Also consider U_j and I_j as the set of users and items in domain $\mathbb{D}_j$. The standard rating prediction problem in the target domain $\mathbb{D}_1$ can be modeled by a target function $y : U_1 \times I_1 \to \mathbb{R}$. We represent each user-item interaction $(u, i) \in U_1 \times I_1$ with a feature vector $\mathbf{x} \in \mathbb{R}^{|U_1|+|I_1|}$ with binary variables indicating which user rated which item. In other words, if user u rated item i the feature vector $\mathbf{x}$ is represented as:

$$\mathbf{x} = (\underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|U_1|}, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|I_1|}) \tag{2A.2}$$

where non-zero elements are corresponding to user u and item i. The feature vector $\mathbf{x}$ can also be represented by its sparse representation $\mathbf{x}(u,i) = \{(u,1),(i,1)\}$.

Given the feature vector $\mathbf{x}(u,i)$ in the target domain, our cross-domain CF approach extend this vector by adding collaborative information of user u from other domains. Now lets assume that $s_j(u)$ represents all items in domain $\mathbb{D}_j$ which are rated by user u. For each auxiliary domain $\mathbb{D}_j$, $j = 2,\ldots,m$, our method extend $\mathbf{x}(u,i)$ with a vector $\mathbf{z}_j(u)$ with the following sparse representation:

$$\mathbf{z}_j(u) = \{(l,\phi_j(u,l)) : l \in s_j(u)\} \tag{2A.3}$$

where $\phi_j(u,l)$ is a domain-dependent real valued function. We define ϕ_j based on the rating of user u to item l and normalize it based on total number of items which is rated by user u in domain $\mathbb{D}_j$:

$$\phi_j(u,l) = \frac{r_j(u,l)}{|s_j(u)|} \tag{2A.4}$$

where $r_j(u,l)$ specifies the rating of user u to item l in domain $\mathbb{D}_j$. In the above definition ϕ_j is a function of $r_j(u,l)$ which reflects rating patterns of user u in different domains. Furthermore, it is normalized by considering the number of items which are rated by user in an auxiliary domain. This means that if a user is a frequent rater in an auxiliary domain, the contribution of each single rated item in this domain would be less compared to a rated item in an auxiliary domain with smaller number of user ratings. The above definition of ϕ_j prevents the model to be overwhelmed by too much information from auxiliary domains. This is one of the main advantages of factorization machines, namely to allow control of the amount of knowledge that is transferred from auxiliary domains. Note that the function ϕ_j can be also defined in other forms to reflect contribution of various domains in different ways. Based on our experiments we found the above definition of ϕ_j simple yet effective to transfer knowledge from auxiliary domains. Given the above definitions, we can now represent the extended vector $\mathbf{x}$ with the following sparse form:

$$\mathbf{x}(u,i,s_2(u),\ldots,s_m(u)) = \{\underbrace{(u,1),(i,1)}_{\text{target knowledge}}, \underbrace{\mathbf{z}_2(u),\ldots,\mathbf{z}_m(u)}_{\text{auxiliary knowledge}}\} \tag{2A.5}$$

The above feature vector serves as the input into the FM model in Equation (1), while the output variable y is the rating of user u to item i in the target domain. Based on our proposed feature expansion method, the FM will only need to focus on the users in the target domain, resulting in an improvement in terms of computational cost.

2A.4. Experiments

We conducted our experiments on Amazon dataset [62] which consists of rating information of users in 4 different domains: books, music CDs, DVDs and video tapes. The dataset contains 7,593,243 ratings on the scale 1-5 provided by 1,555,170 users over 548,552 different products including 393,558 books, 103,144 music CDs, 19,828 DVDs and 26,132 video tapes.

We build the training and test set in two different ways similar to [38] to be able to compare our approach with them. In the first setup, TR_{75}, 75% of data is considered as training set and the rest as test set, and in the second setup, TR_{20}, only 20% of data is considered as training set and the rest as test set.

We implemented a recommendation framework with C#[2] on top of two open source libraries for recommender systems: MyMediaLite [30] which implements most common CF approaches including Matrix Factorization, and LibFM [92] which implements FM learning algorithms. We first compared FMs with matrix factorization method on two different single domains and then we compare the results of our proposed method with the state-of-the-art CDCF work [38] on the same dataset. We also compare our method with a *blind* combination of all items from all domains to show that the improvement of our results is not only due to additional training data. We used mean absolute error (MAE) and root mean square error (RMSE) as evaluation metrics in our experiments. Table 1 lists the MAE and RMSE scores on the two different setups TR_{75} and TR_{20} and based on the following approaches:

- **MF-SGD (***D***)**: Matrix Factorization method using SGD learning algorithm on single domain *D*.
- **FM-X (D)**: Factorization Machine method on single domain D based on learning algorithm X (SGD, ALS or MCMC).
- **FM-All-X (D)**: Combining all rating data into single domain (blind combination) and testing target domain *D* by using FM with algorithm X. This approach simply increases the size of training data by including the rating data of all domains. In other words, the feature vector **x** is represented as in equation (2B.2) and all items in different domains are treated the same.
- **FM-X (**$D_T, \{D_A\}$**)**: Factorization Machine method on target domain D_T and auxiliary domains $\{D_A\}$ based on algorithm X.
- **PF2-CDCF**: The Cross-Domain CF method which is proposed by Hu et al. [38] on the same dataset.

Comparison of results on single domains in table 1 shows that by using MCMC learning method, FM method performs better than matrix factorization. Comparison of FM-MCMC and FM-All-MCMC methods reveals that simply including the rating data of auxiliary domains into target domain does not cause any improvement on rating prediction and it can also hurt the result since the additional data can be noisy for the target domain. The best results, FM-MCMC (Book, {Music, DVD,Video}) and FM-MCMC (Music, {Book, DVD,Video}), are obtained using our adopted cross-domain method with MCMC learning method and are better than PF2-CDCF on the same dataset.

2A.5. DISCUSSION AND FUTURE DIRECTIONS

In this chapter we adapted a model using factorization machines to exploit additional knowledge from auxiliary domains to achieve performance improvement in cross-domain

[2] `https://github.com/babakx/WrapRec`

Table 2A.1: Performance comparison of different single- and cross-domain factorization models on the Amazon dataset

Method \ Setup	TR_{75}		TR_{20}	
Target: Book	MAE	RMSE	MAE	RMSE
MF-SGD (Book)	0.62	0.86	0.89	1.14
FM-SGD (Book)	0.69	0.92	0.74	0.96
FM-ALS (Book)	0.72	0.99	0.75	1.07
FM-MCMC (Book)	0.60	0.79	0.72	0.94
FM-All-MCMC (Book)	0.60	0.79	0.76	0.99
FM-MCMC (Book, {Music, DVD, Video})	**0.46**	0.64	**0.69**	0.92
PF2-CDCF (Book, {Music, DVD, Video}) [38]	0.50	-	0.76	-
Target: Music				
FM-MCMC (Music)	0.71	0.95	0.77	1.00
FM-MCMC (Music, {Book, DVD, Video})	**0.67**	0.91	**0.74**	0.98
PF2-CDCF (Music, {Book, DVD, Video}) [38]	0.70	-	0.82	-

CF. The success of CDCF is highly dependent on effectively transferring knowledge from auxiliary domains, which can be well exploited with FMs. A key factor of success of our approach is the ability to encode domain-specific knowledge in terms of real-valued feature vector, which became possible with FMs and which enables better exploitation of the interaction patterns in auxiliary domains. The experimental results show that our adopted method can perform better than state-of-the-art CDCF methods while it benefits from low computational cost of FMs.

In the future, we want to apply our method to more complicated CDCF scenarios particularly when the source and target domains are more heterogeneous. Another extension to our approach is to also use *contextual* information from both target and auxiliary domains to investigate whether exploiting context can result in even better CDCF performance.

2B

‘Free-Lunch’ Enhancement with Factorization Machines

The advantage of Factorization Machines over other factorization models is their ability to easily integrate and efficiently exploit auxiliary information to improve Collaborative Filtering. Until now, this auxiliary information has been drawn from external knowledge sources beyond the user-item matrix. In this chapter, we demonstrate that Factorization Machines can exploit additional representations of information inherent in the user-item matrix to improve recommendation performance. We refer to our approach as ‘Free Lunch’ enhancement since it leverages clusters that are based on information that is present in the user-item matrix, but not otherwise directly exploited during matrix factorization. Borrowing clustering concepts from codebook sharing, our approach can also make use of ‘Free Lunch’ information inherent in a user-item matrix from an auxiliary domain that is different from the target domain of the recommender. Our approach improves performance both in the joint case, in which the auxiliary and target domains share users, and in the disjoint case, in which they do not. Although the ‘Free Lunch’ enhancement does not apply equally well to any given domain or domain combination, our overall conclusion is that Factorization Machines present an opportunity to exploit information that is ubiquitously present, but commonly under-appreciated by Collaborative Filtering algorithms[1].

[1] This chapter is published as Loni, Babak, Alan Said, Martha Larson, and Alan Hanjalic. "’Free lunch’ enhancement for collaborative filtering with factorization machines." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 281-284. ACM, 2014.

2B.1. Introduction

Factorization Machines (FMs) [92] are general models that factorize user-item collaborative data into real-valued feature vectors. FMs have recently attracted the attention of the recommender system community because of the ease and effectiveness with which they can integrate information from external sources, for example, context information [99]. However, until now, focus of FM approaches in the recommender system community has been on integrating new sources of information. In this chapter, we investigate the potential of FMs to help us make the most of information that we already have. Because our approach makes use of information that is already present in the user-item matrix, it has the feel of delivering 'something for nothing.' In this spirit, we call our approach 'Free Lunch' enhancement of Factorization Machines.

The specific 'Free Lunch' effect that we focus on in this chapter arises from known, but under-appreciated information inherent in the user-item matrix. In general, conventional Collaborative Filtering (CF) approaches, including memory-based and model-based methods, exploit similarities that are based on sets of rated items. The information in user-item matrixes can, however, be repackaged to create other representations of items and users. Specifically, here, we investigate one such repackaging that views users and items in terms of their overall rating patterns. Overall rating patterns are expressed as rating histograms, which are 'trivial' in the sense that they require simple aggregation of information inherent in the user-item matrix. However, clustering these histograms creates user and item categories that constitute new representations of information that is not otherwise exploited by matrix factorization. In this chapter, we introduce an approach that uses Factorization Machines to integrate these category labels into a Collaborative Filtering algorithms capable of improving recommendation performance. The approach reveals its full potential when used to exploit information not only from the user-item matrix of the target domain, but also information from the user-item matrix of auxiliary domains. In this respect it constitutes a simple, yet effective, method for Cross-Domain Collaborative Filtering.

2B.2. Background and Motivation

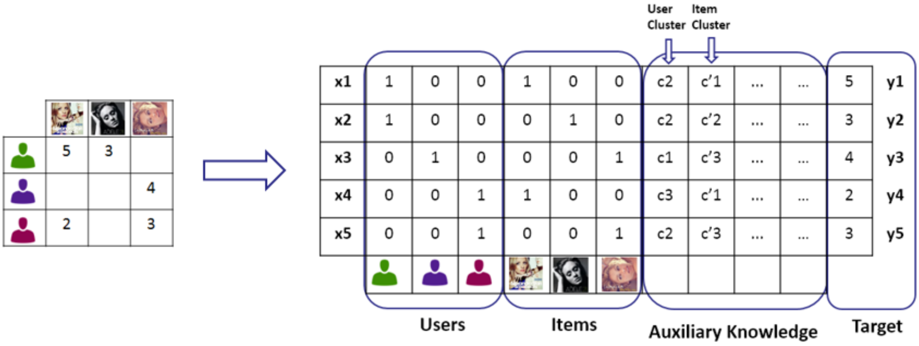
The 'Free Lunch' approach does not literally achieve something for nothing. Rather, as mentioned above, it takes advantage of an underexploited representation of users and items as rating histograms. A rating histogram encodes the normalized frequency with which a user assigns a certain rating or an item is assigned a certain rating. Upon first consideration, rating histograms appear to destroy the very connections between users and items that provide the basis for CF. Worse, they capture bias in user rating habits, e.g., the general tendency of individual users to assign very high or very low scores. It is exactly this bias that many similarity metrics seek to avoid. However, seen from a different perspective, rating habits have the ability to capture something about the 'rating style' of individual users or the 'style' of users who rate specific items. Ultimately, the success of our approach attests to the benefit of taking this perspective.

The perspective is supported by evidence from the literature that similarity of rating style is connected to similarity in taste in populations of consumers. For example, Tan and Neetessine [119] presents a study of the Netflix Prize data set that reveals patterns such as a tendency of people who give higher ratings to watch mainstream movies. In other words, users who are similarly “generous" or “stingy" with high ratings when rating movies, may actually have a tendency to prefer the same kind of movies. The basic insight of our approach is that such relationships already exist implicitly in the user item matrix, but are not exploited by conventional matrix factorization approaches. Note that we are not making a claim that rating histograms are optimally suitable for comparing users. Our ‘Free Lunch’ enhancement still stands to benefit from a weak indicator of similarity. Our position is if we can stand to benefit from this information, we should seize the opportunity. Note that using normalized rating histograms to represent users means that it is possible to calculate a similarity between two users who have not necessarily rated the same items. In cases where recommender systems must confront extremely sparse data conditions, weak indicators of similarity could prove to be particularly useful.

Use of clustering techniques in CF stretches back into early history of recommender systems, and includes many variants. In the earliest work, clustering was used to reduce dimensionality, focusing on standard representations of users as vectors of rated items. Also, clustering has been used as a way to integrate content into CF to create hybrid recommendation algorithms. For example, in Li and Kim’s approach [66], content-based information about items is used to create item clusters, which are then represented in a cluster-based matrix to which CF approaches are applied on. The clustering in our work differs from previous uses of clustering for CF in that it relies on neither standard representations, nor or additional information about items or users, but rather uses new representations of existing information in order to create clusters.

Clustering has proven particularly useful in Cross-Domain Collaborative Filtering (CDCF) algorithms, algorithms that exploit one domain to make recommendations in another. Li et al. [64] proposed a clustering approach known as *codebook transfer* for CDCF. Their model tries to find similar rating patterns in different domains, and then transfer cluster-level patterns to improve recommendation in a target domain. The approach requires additional factorizations on auxiliary domains which makes it computationally more complex compared to a typical matrix factorization model. Our ‘Free Lunch’ enhancement approach also uses domain transfer based on clusters. However, where Li et al. [64] use conventional representations of users and items and co-clustering, our approach represents users and items in terms of rating histograms. Our approach incurs little computational burden with clustering. Rather, in ‘Free Lunch’ Enhancement, clustering is a pre-processing step, since it is performed off-line on individual domains and is not part of building the model.

Our previous work [78], established that FMs achieve state-of-the-art performance in CDCF, and can be straightforwardly applied in the case of joint domains, i.e., cases in which the target domain and the auxiliary domain do not share the same users. That work did not use clusters, but rather exploited individual ratings. In this chapter, we take FM performance as our baseline, and investigate how it can be improved by exploiting user and item clusters. Our proposed approach also applies to disjoint domains, i.e., domains that do not share users.



	5	3	
			4
	2		3

							User Cluster	Item Cluster				
x1	1	0	0	1	0	0	c2	c'1	...	...	5	y1
x2	1	0	0	0	1	0	c2	c'2	...	...	3	y2
x3	0	1	0	0	0	1	c1	c'3	...	...	4	y3
x4	0	0	1	1	0	0	c3	c'1	...	...	2	y4
x5	0	0	1	0	0	1	c2	c'3	...	...	3	y5
	Users			Items			Auxiliary Knowledge				Target	

Figure 2B.1: Encoding of user and item clusters as auxiliary features in Factorization Machines.

2B.3. Enhancement Approach for FMs

This section provides the necessary background on FMs and explains the details of our 'Free Lunch' enhancement approach, that creates clusters using rating histograms for integration into FMs.

2B.3.1. Factorization Machines

In contrast to typical Factorization techniques in which the interaction of users and items are represented by a matrix, in Factorization Machines [92] the interaction of a user and an item (i.e., when a user rates an item) is represented by a feature vector. To understand how FMs work, let us assume that the data of a rating prediction problem is represented by a set S of tuples $(\mathbf{x}, y)$ where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is an n-dimensional feature vector representing user-item interaction and y is the rating value. Factorization machines model all interactions between features using factorized interaction parameters. The interactions can be between a pair of features, or it can even be between larger number of features. In this chapter, we adapted an FM model with order 2 where only the interactions between pairs of features are taken into account. This model can be represented as follows:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} w_{j,j'} x_j x_{j'} \tag{2B.1}$$

where w_j are model parameters and $w_{j,j'}$ are factorized interaction parameters and are defined as $w_{j,j'} = \mathbf{v}_j . \mathbf{v}_{j'}$ where $\mathbf{v}_j$ is k-dimensional factorized vector for feature j. For an FM with n as the dimensionality of feature vectors and k as the dimensionality of factorization, the model parameters that need to be learnt are $\Theta = \{w_0, w_1, \ldots, w_n, v_{1,1}, \ldots, v_{n,k}\}$.

The parameters of the model can be learnt by three different learning algorithms [92]: Stochastic Gradient Descent (SGD), Alternating Least-Squares (ALS) and Markov Chain Monte Carlo (MCMC) method. MCMC learning method proved fastest in our exploratory work and were adopted for our experiments.

2B.3.2. Cluster Encoding

Factorization machines require the user-item interactions to be represented by a feature vector. This characteristic allows us to incorporate any additional knowledge in terms of real-valued features. We take this advantage of FMs into account and extend the user-item feature vectors with cluster level features which potentially can improve the rating prediction task.

In FMs, a standard rating prediction problem is represented by a target function $y : U \times I \to \mathbb{R}$. We represent each user-item interaction $(u, i) \in U \times I$ with a feature vector $\mathbf{x} \in \mathbb{R}^{|U|+|I|}$ with binary variables indicating which user rated which item. In other words, if user u rated item i the feature vector $\mathbf{x}$ is represented as:

$$\mathbf{x} = (\underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|U|}, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|I|}) \tag{2B.2}$$

where non-zero elements correspond to user u and item i. The feature vector $\mathbf{x}$ can also be represented by its sparse representation $\mathbf{x}(u, i) = \{(u, 1), (i, 1)\}$.

Our cluster encoding algorithm extends the above feature vector by user and item clusters. Therefore, the target function would be $y : U \times I \times C_u \times C_i \to \mathbb{R}$ where C_u and C_i are user and item cluster spaces. User and item clusters can also be drawn from other, auxiliary domains. In other words, the clusters that users and items belong to in auxiliary domains can also be identified and added to the feature vector. More specifically, we can represent the sparse form of a cluster-enhanced feature vector as follows:

$$\mathbf{x}(u, i) = \{(u, 1), (i, 1), (c_j(u), 1), (c_j(i), 1) : j = 1 \ldots m\} \tag{2B.3}$$

where $c_j(u)$ and $c_j(i)$ are user and item cluster ids in domain j and m is the total number of domains.

Figure 2C.2 illustrates our proposed approach to building feature vectors from a user-item matrix. For each rating, a binary feature vector is created in which the corresponding user and item are indicated by 1s and the remaining features are 0s. This feature vector is then extended by additional binary features which specify the user and item cluster in different domains. The rating values are specified as the output of each vector. Using FM we try to predict the output of the test samples.

2B.3.3. Cluster Construction

Our clustering approach considers users with similar rating patterns to be similar, assigning them to the same cluster. In the context of our work, rating patterns are based on the histogram of rating values. More specifically a user u_j is represented by the feature vector $u_j = (u_{j1}, \ldots, u_{jp})$ where p is the upper bound of rating values in the dataset domain and u_{jk} is the number of items which are rated with value k by user u_j. Similarly, an item i_j can be represented by $i_j = (i_{j1}, \ldots, i_{jp})$. Given the above representations, we calculate user and item clusters using the K-means clustering algorithm. The distance between vectors is calculated based on Euclidean distance. We choose K-means due to its simplicity and efficiency, which distinguishes it from alternative clustering algorithms [4].

For cross-domain scenarios, we create clusters of users and items in auxiliary domains, and for each user and item in the target domain the most similar clusters are identified by measuring the distance of a user or an item to the center of clusters. The most similar clusters in the auxiliary domain are then used to extend the user-item feature vector in Equation 2B.3.

2B.4. Experiments

In this section we explain the datasets and frameworks that we used and describe the experiments that we did with their results. We further analyze and discuss the results to find out whether the improvements obtained by our method is significant or not.

2B.4.1. Datasets and Framework

We conducted our experiments on a dataset from Amazon [62] consisting of four joint domains, Movielens and Epinions datasets, each having a different set of users and items. The four domains in the Amazon dataset are books, music CDs, DVDs and video tapes. The Amazon dataset contains 7,593,243 ratings on a 1-5 scale provided by 1,555,170 users over 548,552 different products including 393,558 books, 103,144 music CDs, 19,828 DVDs and 26,132 VHS tapes. We use the Movielens 1 million dataset, which consists of 1,000,209 ratings on a 1-5 scale, 6,040 users and 3,883 items. The Epinions dataset contains 664,865 ratings also on a 1-5 scale, 49,289 users and 139,737 items. The data is split into a training set (75%) and a test set (25%).

We implemented our approach within our recommendation framework [77] using C#, which is built on top of two open source libraries for recommender systems: MyMediaLite [30], which implements the most common CF approaches, and LibFM [92], which implements FMs algorithms.

2B.4.2. Results

We evaluated our 'Free Lunch' enhanced FM approach with respect to three different enhancement scenarios, and a range of different settings for the number of clusters, which were created with K-means clustering. Figures 2B.2 and 2B.3 present the performance of our approach on the Amazon Books and and the Amazon Music datasets in terms of Root Mean Square Error (RMSE). Each graph shows the performance of the dataset under our three scenarios: 1) Target domain clusters: user and item clusters from the same domain (i.e., a single target domain) are used to extend the feature vectors (i.e., Equation (3) with $j = 1$). 2) Target and joint auxiliary domain clusters: user and item clusters from a joint auxiliary domain are used (i.e., Equation (3) with $j > 1$). 3) Target and disjoint auxiliary domain clusters: user and item clusters from a disjoint auxiliary domain are used (i.e., users do not overlap). To show the effectiveness of our approach, we compare the performance of each scenario with a baseline in which no user or item clusters are used. In other words, for our baseline, the user-item interactions are simply represented by feature vectors as described in Equation (2). The results in Figure 2B.2 and 2B.3 demonstrate that our method outperforms the baseline on both datasets and in all three scenarios. The best performance is achieved on joint domains, which is not unexpected since the domains share users, and the auxiliary domain brings additional information about items rated by these users into play. What is more surprising is that clustering applied in the single domain case, or applied in the case of joint domains is

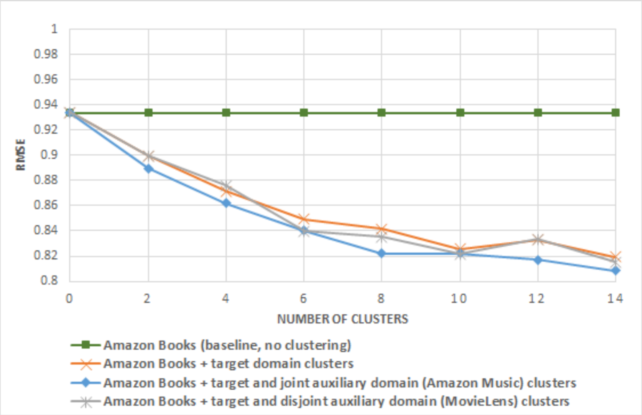


Figure 2B.2: Performance of our proposed method on the Amazon Books dataset based on different number of clusters and different enhancement scenarios. The baseline (no clusters) is represented by the straight line.

able to come very close to the performance in the joint domains. Figures 2B.2 and 2B.3 also reveal that for all three scenarios, the performance also improves as the number of clusters is increased, until it reaches a point where there is no improvement, at $k = 10$.

We investigate the results achieved in the joint domain scenario with a statistical analysis of the results (in terms of RMSE). This analysis reveals that the improvements obtained for the Amazon Books dataset (Figure 2B.2) when extended with clusters from Amazon Music are statistically significant ($p < 0.05$) regardless of the number of clusters used when compared to the results from the baseline (no clusters). However, a similar analysis of the results obtained for the Amazon Music dataset enriched with clusters from Amazon Book (Figure 2B.3) does not point to significance ($p > 0.2$). We note that the performance of the joint and disjoint scenarios on each dataset track the performance of the single domain scenario. This fact suggests that a strongly performing target domain is critical for the approach to benefit from joint domains. Strong performance of the target domain could be related to size, here, 400 thousand books vs.100 thousand music CDs.

Next we turn to investigate the 'target domain cluster' scenario involving a single domain in more detail. Table 2B.1, summarizes the results from the Amazon Books and Amazon Music data set, and also reports further results calculated on the Movielens and Epinions datasets (both RMSE and MAE evaluation metrics). The performance on the different data sets is reported for k=10 clusters, the best condition determined in the previous experiment, and compared with the no-clustering baseline. These results confirm that the performance improvement delivered by 'Free Lunch' enhancement is not limited to Amazon data sets, but is achieved on other data sets as well. The relative improvement on the Amazon Book dataset (12% in terms of RMSE) is the largest. The relative greater improvement on the Amazon data set may be attributable to the fact that it is sparser than the Movielens or Epinions sets.

As the results in Table 2B.1 show, the best improvement is achieved on the Amazon Book dataset (12% in terms of RMSE), while there are less dramatic improvements

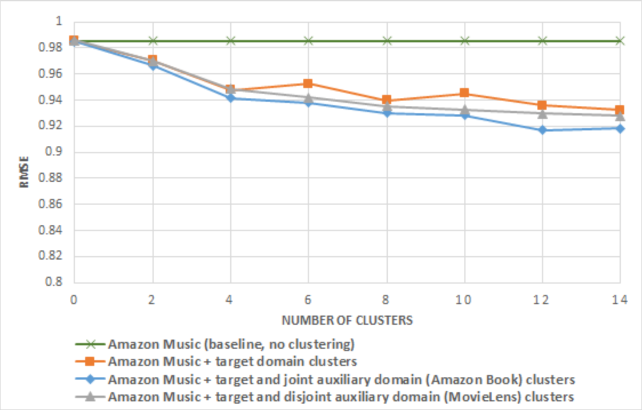


Figure 2B.3: Performance of our proposed method on Amazon Music dataset based on different number of clusters and different enhancement scenarios. The baseline (no clusters) is represented by the straight line.

on the Movielens and Epinions datasets. This can be due to the fact these datasets are denser compared to the Amazon dataset, implying that the underlying similarities between users and items are already well captured, and clustering information is of less importance. 'Free Lunch' enhancement can apparently contribute more to the rating prediction task in cases where datasets are sparse, meaning that additional representations of the information in the user-item matrix have more to contribute.

Table 2B.1: Comparison of our cluster-enhanced approach with the no-cluster baseline

	No clusters		10 clusters	
Dataset	**RMSE**	**MAE**	**RMSE**	**MAE**
Amazon Books	0.933	0.841	0.825	0.778
Amazon Music	0.984	0.861	0.945	0.832
Movielens	0.936	0.862	0.890	0.838
Epinions	1.082	0.920	1.024	0.891

2B.5. CONCLUSION AND OUTLOOK

We have presented a 'Free Lunch' enhancement approach that makes use of the ability of Factorization Machines to easily and effectively integrate additional information. Our approach demonstrates that it is not necessary to turn to outside resources to find additional information useful for improving Factorization Machines, but instead we can make better use of information already at hand. The larger message is that FM approaches that do not first attempt to maximize the benefit they can derive from information at hand, i.e., 'Free Lunch', are missing opportunities.

In this work, we have focused on clusters formed using rating histograms. We show that this information can be used to improve rating prediction in cases where only a single, target domain is available, or in cases where an auxiliary domain (joint or disjoint with the target domain) is also available. Our future work will also investigate other possible types of ‘Free Lunch’ information. The list of sources that we are interested in ranges from popularity information to random assignment of users to user-groups. Further, we will work to make more detailed understanding of what makes a domain particularly a suitable target domain for ‘Free Lunch’ enhancement, and how to best identify useful auxiliary domains.

2C

Speeding Up Collaborative Filtering with Factorization Machines

In this chapter We propose an alternative way to efficiently exploit rating data for collaborative filtering with Factorization Machines. Our approach partitions user-item matrix into ‘slices’ which are mutually exclusive with respect to items. Thanks to advantage of Factorization Machines where user-item interactions can be represented with feature vectors, our approach make use of this advantage and extend user-item feature vectors with complementary information which can improve recommendations. Our ‘Slice and Train’ method consists of two steps: in the first step the slices are created by leveraging existing item categories or by clustering. In the second step the training is done by only using the ratings from the slice of interest (target slice) while the rating from other slices are exploited indirectly as complementary features. We demonstrate, using experiments on two benchmark datasets, that same performance as using the complete dataset can be achieved while the time complexity of training can be reduced significantly[1].

[1] This chapter is an extension of Loni, Babak, Martha Larson, Alexandros Karatzoglou, and Alan Hanjalic. "Recommendation with the Right Slice: Speeding Up Collaborative Filtering with Factorization Machines." In *RecSys Posters.* 2015.

2C.1. Introduction

In this chapter, we investigate the idea that the ‘right’ data, rather than all data, should be exploited to build an efficient recommender system. We introduce an approach called ‘Slice and Train’ that trains a model on the right slice of a dataset (a ’sensible’ subset containing the current items that need to be recommended) and exploits the information in the rest of the dataset indirectly. This approach is particularly interesting in scenarios in which recommendations are needed only for a particular subset of the overall item set. An example is an e-commerce website that does not generate recommendations for out-of-season or discontinued products. The obvious advantage of this approach is that models are trained on a much smaller dataset (only the data in the slice), leading to shorter training times. The magnitude of this reduction is illustrated in Fig. 3.7, which depicts training times on the two datasets that are used in the experimental investigation in this chapter, and shows that training time on the average slice, is much less than on the whole dataset. The ‘Slice and Train’ approach also has, however, a less expected advantage, namely, that it offers highly competitive performance with conventional approaches using the whole dataset, and in some cases even improves the performance. This advantage means that it is advantageous to apply ‘Slice and Train’ even in cases in which predictions are needed for all items in the dataset, by training a series of separate models, one for each slice.

Our approach consists of two steps: first the data is partitioned into a set of slices containing mutually exclusive items. The slices can be formed by grouping items based on their properties (eg. the category of item), availability (eg. whether or not an item is available or it is a historical/discontinued item), or by more advanced slicing methods such as clustering. In the second step the model is trained using the samples in the slice of interest, i.e., *target* slice, while other slices are indirectly being exploited as *auxiliary* slices. To efficiently exploit information from auxiliary slices our approach trains the recommender model using Factorization Machines [92]. Factorization Machines (FMs) generate recommendations by working with vector representations of user-item data. A benefit of these representations is that they can easily be extended with additional features. Such extensions are usually used to leverage additional information [99], or addition domains [78], to improve collaborative filtering.

This chapter makes two contributions: first, it introduces the ‘Slice and Train’ method that offers computational speed up, while at the same time maintaining or improving prediction performance, and, second, it demonstrates the robustness of the improvement by experiments with multiple learning methods, and by comparing the indirect exploitation of information from other slices, with a conventional source of additional information used in Factorization Machines, namely item context (i.e., properties).

2C.2. Related Work

Partitioning data and training different subsets has been studied before, but from other perspectives [1, 23, 43]. These approaches are mainly based on fusion and in this way are different from our approach. The ‘Right Slice’ approach bears a superficial resemblance to fusion or ensemble methods, such as the work of Decoste [23], however, although we train multiple models on subsets of the user-item matrix, the prediction is made with the

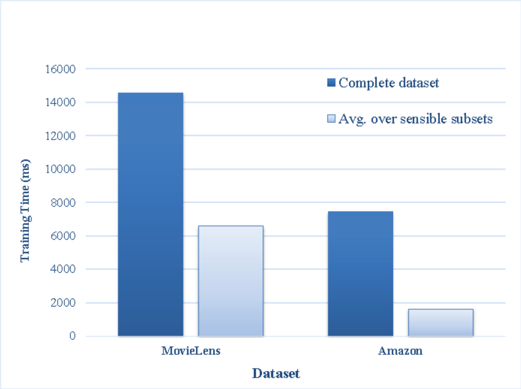


Figure 2C.1: Complexity of training a model when only a sensible subset of data is used versus using the complete dataset.

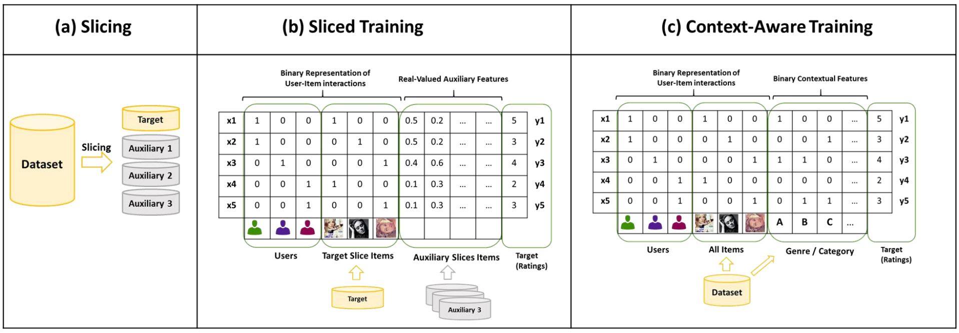
'Right Slice' model only, and the information in the other slices is exploited indirectly. In contrast, in the fusion methods the prediction is mainly done by aggregating the judgment from all the slices. One advantage of our method compared to ensemble methods is that in the cases that we are interested on recommendations on a particular subset of data we do not need to train all slices and we can benefit from lower training complexity. Other studies have been done to increase scalability of recommendation by clustering data [18, 67], but these have not proposed any means of exploiting the information from other clusters to maintain or improve performance.

Another body of work related to this chapter are the studies that use FMs for recommendation. Several research efforts have been carried out that use FMs to efficiently embed additional information for collaborative filtering. Rendle et al. [99] exploited FMs for context-aware recommendation. Other recent studies with FM have been done for Cross-Domain collaborative filtering [78] and exploiting social network information for recommendation [129]. The approach proposed in this chapter is different from other previous work on FMs in the way that the auxiliary features are created and transferred.

2C.3. THE SLICE AND TRAIN METHOD

To convey an understanding of our method, we first introduce the general framework of Factorization Machines, and we then explain our extension to FMs to implement the slice and train method that is used by the 'Right Slice' approach.

FMs are general predictors, working with real-valued feature vectors, that are optimized for sparse data [92]. In FMs, a typical rating prediction problem can be represented by a set of feature vectors with their corresponding labels. The user-item interactions are represented by feature vectors and the ratings are considered as the output. More specifically let us assume that the data of a rating prediction problem is represented by a set D of pairs $(\mathbf{x}, y)$ where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a n-dimensional feature vector and y is its corresponding output. The vector $\mathbf{x} \in \mathbb{R}^{|U|+|I|}$ is defined as a binary vector that indicates which user rated which item, and the user rating is considered to



x1	1	0	0	1	0	0	0.5	0.2	...	...	5	y1
x2	1	0	0	0	1	0	0.5	0.2	...	...	3	y2
x3	0	1	0	0	0	1	0.4	0.6	...	...	4	y3
x4	0	0	1	1	0	0	0.1	0.3	...	...	2	y4
x5	0	0	1	0	0	1	0.1	0.3	...	...	3	y5

x1	1	0	0	1	0	0	1	0	0	...	5	y1
x2	1	0	0	0	1	0	0	0	1	...	3	y2
x3	0	1	0	0	0	1	1	1	0	...	4	y3
x4	0	0	1	1	0	0	0	0	0	...	2	y4
x5	0	0	1	0	0	1	0	1	1	...	3	y5
							A	B	C	...		

Figure 2C.2: Slicing process (a), and feature construction of the 'Slice and Train' method (b) compared with context-aware training (c) using Factorization Machines.

be the corresponding output y. In other words, if user u rated item i the feature vector $\mathbf{x}$ can be represented by its sparse representation as $\mathbf{x}(u,i) = \{(u,1),(i,1)\}$, where non-zero elements correspond to user u and item i.

Representing user-item interaction, with feature vectors makes it possible to engineer features or expand them with additional features to train better models. In this work, we exploit this advantage of FMs to enhance the user-item feature vectors with additional features that are translated from auxiliary slices.

Figure 2C.2-a and 2C.2-b show our proposed 'Slice and Train' method. The left part of the Figure illustrates the slicing step where dataset is divided into a set of slices based on properties of item or based on clustering. In the second step as illustrated in Figure 2C.2-b, the feature vectors are built for training the model. Each row represents one feature vector $\mathbf{x}$ with its corresponding output y (i.e., rating). The feature vectors consist of two parts. The first part consists of binary values that indicate which user rated which item in the target slice. The second part consists of real-valued features which are calculated based on the rating information of other auxiliary slices. We compare our method with a context-aware method [99], illustrated in Figure 2C.2-c, in which item properties are used to extend the feature vectors.

The auxiliary features are build based on connecting the users in different slices. Basically for each binary user-item interaction in the target slice, our approach looks for the ratings that are provided from same user in auxiliary slices and then it normalizes the ratings and add them to the feature vectors. To understand how the auxiliary features are built, assume that the dataset is divided into m slices $\{S_1,\ldots,S_m\}$. Let us also assume that the items that are rated by user u in slice S_j is represented by $s_j(u)$. By extending the feature vector $\mathbf{x}(u,i)$ with auxiliary features, we can represent it with the following sparse representation:

$$\mathbf{x}(u,i) = \{\underbrace{(u,1),(i,1)}_{\text{target slice}},\underbrace{\mathbf{z}_2(u),\ldots,\mathbf{z}_m(u)}_{\text{auxiliary slices}}\} \tag{2C.1}$$

where $\mathbf{z}_j(u)$ is sparse representation of auxiliary features from slice j and is defined as:

$$\mathbf{z}_j(u) = \{(l,\phi_j(u,l)) : l \in s_j(u)\} \tag{2C.2}$$

where $\phi_j(u,l)$ is a normalization function that defines the value of the auxiliary features. Trivial choices for ϕ_j are to just consider it as an indicator of feature (with binary value of 1) or consider the rating as its value. However, we introduce a more advance normalization function which features better performance in our exploratory experiments. We define ϕ_j based on the ratings that user u gave to items in slice j and normalize it based on the average value of user ratings as follows:

$$\phi_j(u,l) = \frac{r_j(u,l) - \bar{r}(u)}{r_{max} - r_{min}} + 1 \tag{2C.3}$$

where $r_j(u,l)$ indicates the rating of user u to item l in slice j, $\bar{r}(u)$ indicates the average value of user ratings, and r_{max} and r_{min} indicate the maximum and minimum possible values for rating items.

For context-aware training with FM [99], as illustrated in Figure 2C.2-c the feature vectors $\mathbf{x}(u,i)$ can be extended with features corresponding to context or attributes of users and items. In this work we used same context features that we use for creating slices. The slices are created based on the genre or category of items. Therefore the additional features can be binary features indicating the genre or category of items and can be represented as $\mathbf{x}(u,i,c(i)) = \{(u,1),(i,1),\mathbf{z}_c(i)\}$ where $z_c(i)$ is sparse representation of context features and similarly as Eq. (2) can be defined as $\mathbf{z}_c(i) = \{(l,1) : l \in s_c(i)\}$ where $s_c(i)$ are feature ids of the category or genres of items. Note that the context-aware training method uses the complete dataset to train the model whereas the 'Slice and Train' method only uses the samples in auxiliary slices and the information of other slices are exploited as additional features. In other word the size of training samples (number of rows in design matrix) for the sliced training is equal to the number of samples in the target slice while for the context-aware training the number of training samples is equal to the total number of samples in the dataset. As the complexity of training in FMs depends on the size of train samples, the sliced training can benefit from a reduced time complexity compared to context-aware training. In the next section we will show that the sliced training can gain competitive performance compared to context-aware training while the time complexity of training and recommendations is significantly lower.

2C.4. Dataset and Experiments

In this chapter, we tested our method on two benchmark datasets of MovieLens 1M movie ratings dataset[2] and Amazon reviews dataset[3] [62]. Table 5.1 lists the statistics of the two datasets that we used in this chapter. The Amazon dataset contains product ratings in four different groups of items namely books, music CDs, DVDs and Video tapes. We use these four groups as natural slices that exists in this dataset. For the MovieLens dataset we build slices based on the genre of movies. As movies in this dataset can have multiple genres, we create slices by clustering movies using their genres. To perform the clustering each movie is represented by a binary feature vector indicating the genres of the movie. The movie vectors are the clustered using a k-means clustering algorithm. Various number of clusters (i.e., slices) can be made for this dataset. Via exploratory experiments we found that two or three slices perform well. Table 5.1 lists the statistics

[2] http://grouplens.org/datasets/movielens/

[3] https://snap.stanford.edu/data/amazon-meta.html/

Table 2C.1: The statistics of the datasets used for the ‘slice and train’ method

Dataset	**No. Users**	**No. Items**	**No. Ratings**
MovieLens	3,706	6,000	1,000,029
Amazon (total)	15,994	84,508	269,947
– Books	15,994	40,884	90,563
– Music CDs	15,994	23,798	62,373
– Video tapes	15,994	9,995	57,590
– DVDs	15,994	9,831	59,421

of the two datasets that we used in this chapter. Note that to be able to test our auxiliary feature enhancement mechanism we choose a subset of Amazon dataset where user population is the same in all four groups, i.e., each user has at least one rating in each group (slice).

In order to test our approach, the data in target slices are divided into 75% training and 25% test data. For every experiment 10% of training data is only used as validation set to tune the hyper-parameters of learning methods.

Factorization Machines can be trained using three different learning methods [92]: Stochastic Gradient Decent (SGD), Alternating Least Square (ALS) and Markov Chain Monte Carlo (MCMC). Each of these three learning method has its own advantage and disadvantages. The SGD and ALS methods are highly dependent to the initialization of hyper-parameters such as learning rate, regularization values and initialization of factorized interactions whereas the MCMC method has an advantage that the regularization values are learned automatically during training. This makes MCMC a favorable choice since with this method there is no need to search for optimal hyper-parameters anymore. On the other hand, MCMC in practice is slower than the other two learning method, although Rendle [92] claims the same order of computational complexity for MCMC as the other two learning methods. To ensure that our proposed sliced training is insensitive to the learning method we tested our approach using all the three methods. As mentioned earlier, ALS and MCMC require searching for optimal hyper-parameters which is a time-consuming pre-processing step. We used a greedy method to search each parameter by fixing all other parameters. Note that these parameters are dependent on dataset and the experimental setups. The sub-optimal hyper-parameters that we found for the three learning methods are listed in Table 2C.2.

The experiments are implemented in the WrapRec open source toolkit[77] and the source code is available online [6]. The underlying optimization algorithms are implemented in LibFM [92]. Furthermore, we compare the performance of FM with Matrix Factorization (MF) to ensure our FM-based solution is competitive with other state-of-the-art factorization methods. We used an SGD-based implementation of MF in My-

[4] Regularization paramters are bias term regularization, 1-way interactions regularization and 2-way interaction regularization.

[5] This is the standard deviation of the normal distribution that is used to generate the initial values for the factorizaed parameters.

[6] http://babakx.github.io/WrapRec/

Table 2C.2: The hyper-parameters and design choices of the three FM learning methods

Parameter \ Dataset	MovieLens	Amazon
Num. Factors (all methods)	5	5
Num. Iterations (all methods)	50	50
SGD - Learning rate	0.005	0.002
SGD - Regular.[4] (bias,1-way,2-way)	0,0,0	0.1,0.1,0.1
SGD - Stdev.[5]	0.1	0.1
ALS - Regular. (bias,1-way,2-way)	1,1,1	0.1,0.1,0.5
ALS - Stdev.	0.1	0.2
MCMC - Stdev.	0.1	0.1

MediaLite[7] toolkit. We also further optimize the training time by using an optimization technique namely *block structure* extensions [94] which save memory and computation effort by finding similar patterns in feature vectors.

Table 2C.3 lists the performance of our sliced training method compared with context-aware and other methods. The experiments are evaluated using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) evaluation metrics. The six different setups that are compared are the followings:

- **FM-ALL**: FM applied on the complete dataset. In this setup no auxiliary features are used.
- **FM-ALL-CTX**: Context-aware FM applied on the complete dataset. Context for Amazon dataset is the category of item and for MovieLens is the genres of items.
- **FM-SL**: FM applied on independent slices. No context or auxiliary features are used for this setup. The reported metrics are average results over all slices.
- **FM-SL-AUX**: FM applied to the independent slices with feature vectors that are extended by auxiliary features derived form auxiliary slices.
- **FM-SL-CTX**: FM applied on independent slices while context features from items in the same slice is being used. In this method no auxiliary information is used.

The first two lines of Table 3 report results when all the data is used. The improvement of performance between FM-ALL and FM-ALL-CTX confirms that FMs are able to exploit additional context information related to items. The effectiveness of FM as our model is also confirmed when we compare FM-ALL baseline with Matrix Factorization (MF). The same setup trained with MF results to RMSEs of 1.2927 and 0.8939 for Amazon and MovieLens datasets which are slightly worse than the FM-ALL results.

Comparing the remaining lines yields four interesting insights. First, the difference between FM-ALL and FM-SLICE is actually quite small. Recall that FM-SLICE trains a FM on only one slice of the data, and discards the rest. It is a surprising result that discarding such a large portion of the data (up to 50% depending on the slice, cf. Table 5.1) does not result in a larger performance deterioration.

[7] http://www.mymedialite.net

Table 2C.3: The performance of the proposed ‘Slice and Train’ method compared to other experimental setups in terms of RMSE and MAE.

Eval. Metric	**RMSE**					
Dataset	**Amazon**			**MovieLens**		
Setup \ Learning	ALS	MCMC	SGD	ALS	MCMC	
FM-ALL	1.0490	1.1610	1.2574	0.8962	0.8894	0.8842
FM-ALL-CTX.	1.0349	1.0755	1.1688	0.8861	0.8613	0.8867
FM-SLICE	1.0659	1.2330	1.2525	0.9046	0.8974	0.9018
FM-SLICE-AUX	1.1011	1.0539	1.1348	0.8958	0.8644	0.8974
FM-SLICE-CTX	1.0932	1.1240	1.2026	0.9045	0.8738	0.9026
	MAE					
Dataset	**Amazon**			**MovieLens**		
Setup \ Learning	ALS	MCMC	SGD	ALS	MCMC	
FM-ALL	0.8169	0.9533	1.0396	0.8325	0.8387	0.8270
FM-ALL-CTX.	0.8088	0.9164	0.9850	0.8286	0.8244	0.8347
FM-SLICE	0.8321	0.9915	1.0451	0.8356	0.8427	0.8369
FM-SLICE-AUX	0.8488	0.9017	0.9116	0.8321	0.8260	0.8414
FM-SLICE-CTX	0.8363	0.9411	1.0025	0.8352	0.8312	0.8380

Second, performance generally improves when moving from FM-SLICE to FM-SLICE-AUX. In other words, using data from the target slice, and adding auxiliary features that are derived from the other slices, performs comparably with the case that the whole dataset is used (FM-ALL). This result is highly desirable, given the computational speed up associated with training a FM model only on one slice of the dataset (cf. Fig. 1) since in such case the number of training points (i.e. number of rows in the FMs design matrix) are significantly smaller.

Third, the performance of FM-SLICE-AUX actually improves in many cases over the performance gained using the whole dataset (i.e., FM-ALL). This result reveals the ability of the ‘Right Slice’ approach to yield benefit both in terms of speed-up and prediction. We note that this effect is highly consistent across datasets and metrics with MCMC. This consistency suggests the robustness of the effect. We highlight MCMC because the regularization values are optimized during training. In contrast, SGD and ALS do not represent fully optimized settings because their parameters were optimized on FM-ALL and used for the rest of the conditions.

Fourth, we note that FM-SLICE-CTX, the condition that trains on only one slice of the data, and adds context information from that slice, performs quite well. It is unable to achieve the performance of FM-ALL-CTX, but actually does approach the performance of FM-ALL. This comparison is interesting since it demonstrates that in cases in which only one slice of the dataset is available, it is still possible to approach the performance of using all the data, if context information on items is added. Overall, however, the ‘Right Slice’ method FM-SLICE-AUX is a better choice than FM-ALL-CTX.

2C.5. CONCLUSION

In this chapter, we have presented an alternative method, referred to as the ‘Slice and Train’ method, to exploit additional information contained in the training data for collaborative filtering based on Factorization Machines. Using two benchmark datasets, we show that the ‘Slice and Train’ method can maintain similar performance as using the complete dataset while speeding up the training time.

III

Beyond Rating Prediction

3

Factorization Machines for Data with Implicit Feedback

Factorization Machines (FMs) are generic factorization models for Collaborative Filtering (CF) that offer several advantages compared to the conventional CF models. They are expressive and general models that can mimic several CF problems, their accuracy is state-of-the-art, and their linear complexity makes them fast and scalable. Factorization Machines however, are optimized for datasets with explicit feedback (such as ratings) and they are not very effective for datasets with *implicit* feedback. Although FMs can also be used for datasets with implicit feedback by a trivial mapping of implicit feedback to explicit values, but we will empirically show that such trivial mapping is not optimized for ranking. In this work, we propose *FM-Pair*, an adaptation of Factorization Machines with a *pairwise* loss function, making them effective for datasets with implicit feedback. The optimization model in FM-Pair is based on the BPR (Bayesian Personalized Ranking) criterion, which is a well-established pairwise optimization model. FM-Pair retains the advantages of FMs on generality, expressiveness and performance and yet it can be used for datasets with implicit feedback. We also propose how to apply FM-Pair effectively on two collaborative filtering problems, namely, context-aware recommendation and cross-domain collaborative filtering. By performing experiments on different datasets with explicit or implicit feedback we empirically show that in most of the tested datasets, FM-Pair beats state-of-the-art learning-to-rank methods such as BPR-MF (BPR with Matrix Factorization model). We also show that FM-Pair is significantly more effective for ranking, compared to the standard FMs model. Moreover, we show that FM-Pair can utilize context or cross-domain information effectively as the accuracy of recommendations would always improve with the right auxiliary features. Finally, we show that FM-Pair has a linear time complexity and scales linearly by exploiting additional features.

3.1. INTRODUCTION

The role of Recommender Systems (RS) as a tool to provide personalized content for users is becoming more and more important. Among different techniques that have been introduced for recommender systems in the past two decades, Collaborative Filtering (CF) has become the most successful and widely-used technique. Early CF techniques were mainly based on neighborhood approaches [100] while recently model-based techniques such as Matrix Factorization (MF) [104] attracted more attention due to their superior performance and scalability [55]. Matrix factorization techniques generally learn a low-dimensional representation of users and items by mapping them into a joint latent space consisting of latent *factors*. Recommendations are then generated based on the similarity of user and item factors.

Factorization Machines (FMs) [88] are general factorization models that not only learn user and item latent factors, but also the relation between users and items with any *auxiliary*[1] features. This is done by also factorizing auxiliary features to the same joint latent space. In contrast to the conventional factorization techniques where the training data is represented by a matrix, the input data for FMs are feature vectors just similar to the input data for the other supervised learning methods such as Support Vector Machines or regression models. This creates a great flexibility for FMs by allowing them to incorporate any additional information in terms of auxiliary features. Thanks to the flexibility of FMs on representing the data, FMs can mimic other factorization models by feature engineering without the need to change the underlying model. In [92] Rendle shows how several factorization model such as Matrix Factorization [55], Attribute-Aware models [27] and SVD++ [54] can be mimic by FMs. Factorization Machines have been successfully applied in different collaborative filtering problems including context-aware recommendation [99], cross-domain collaborative filtering [78] and social recommendation [130]. Factorization Machines have linear time complexity and thus are scalable for large datasets. They have been shown [99] to be significantly faster than tensor factorization [50], a popular context-aware CF method. Moreover, an effective parallel optimization method for FMs has been developed [47], reporting significant speed-up in training time compared to the standard training models.

Despite the great advantages of Factorization Machines, the FMs model is not optimized for data with implicit user feedback. All the aforementioned studies have been developed for datasets with explicit feedback. Implicit user feedback (such as clicks) are typically unary or positive-only feedback, and thus there are no explicit real-valued scores for user-item interactions. A trivial approach to train FMs with such datasets, is to *map* positive feedback to a real-value number such as +1. Negative examples can be sampled from unobserved interactions and can be assigned a real-valued label of 0 or -1. The model can then be trained just like the standard FMs model. However, such mapping methods are associated with two problems: firstly, the sampled interactions with negative labels might not be a real negative feedback as the user might not have the chance to observe the item [97]. Secondly, due to the *point-wise* optimization techniques in FMs, the model learns to correctly predict +1s and -1s, which is not necessarily the optimal model for ranking. We experimentally show that such trivial mapping is not

[1] We use the term “auxiliary" for any additional information that is available next to the user-item matrix. Auxiliary features can be user and item attributes, context, content, information from other domains and so on.

an accurate model for learning from implicit feedback.

The existing work on FMs with implicit feedback is limited and the scope of existing experimental studies is narrow. In a recent work, Guo et al. [34] introduce PRFM (Pairwise Ranking Factorization Machines), where they adapt a pairwise optimization technique to learn from implicit feedback. This work however, has not fully exploited one of the main advantages of FMs, namely the ability to encode additional information as auxiliary features. In PRFM, contextual information has been exploited to adapt the *sampling* stage of the learning method and thus the model needs to be re-adapted for different types of auxiliary information. Furthermore, PRFM was only tested on one explicit dataset. In [34], explicit feedback was mapped to unary positive-only feedback, and thus it is not clear how PRFM performs on datasets with inherent implicit feedback. In another work, Nguyen et al. [80] introduced Gaussian Process Factorization Machines (GPFM), a *non-linear* adaptation of Factorization Machines. In GPFM, interactions between users, items and context are captured with non-linear Gaussian kernels. They also introduced a pairwise optimization model for GPFM for datasets with implicit feedback and used it for context-aware recommendations. However, GPFM is not linearly scalable as the underlying optimization method relies on calculating the inverse of Gaussian kernels, which is a computationally-intensive task. Furthermore, the GPFM model is dependent on the choice of kernel, thus making it more complex compared to the standard FMs model. Nevertheless, we empirically compare the performance of GPFM with our method based on the training time and recommendations accuracy, in Section 5.5.

In this work we consolidate previous work on FMs and introduce a generic Factorization Machines framework for datasets with implicit feedback. Similar to [34], we adapt a pairwise optimization method based on BPR (Bayesian Personalized Ranking) criterion [97]. BPR is an state-of-the-art learning-to-rank method for collaborative filtering that learns to correctly rank *pairs* of items with respect to a user. BPR has been successfully applied for datasets with implicit feedback and it has been shown [97] to be more effective than other learning-to-rank methods such as Weighted Regularized Matrix Factorization (WRMF) [40]. Unlike [34], our proposed training model does not depend on the sampling method. We adapt the optimization model of BPR based on the existing auxiliary information that are represented as additional features. We refer to our implementation as *FM-Pair* since we are using a pair-wise optimization method for FMs. FM-Pair is a linear model that can exploit additional information as auxiliary features just like the standard FMs, without requiring any adaptation to the underlying model. We further propose two applications of FM-Pair on context-aware and cross-domain collaborative filtering problems. We test FM-Pair on four implicit and explicit datasets with different tasks. We find that by using the right data, FM-Pair outperforms state-of-the-art methods for learning from implicit feedback data. We also show that FM-Pair is significantly more effective than the trivial implicit-to-explicit mapping method. Furthermore, we empirically demonstrate the effectiveness of FM-Pair on exploiting auxiliary features (i.e., context or cross-domain information). We also empirically show that FM-Pair scales linearly by increasing dimensionality of factorization or number of auxiliary features.

FM-Pair is publicly available as a part of WrapRec[2], an open-source evaluation frame-

[2] https://github.com/babakx/WrapRec

work for recommender systems and can be used simply on different datasets.

The contributions of this chapter can be summarized as follows:

- An extension to Factorization Machines is introduced that allows the use of FMs for datasets with implicit feedback. Inspired by BPR [97], FM-Pair is implemented with a pairwise loss function without requiring explicit feedback for user-item interactions.

- We propose to apply FM-Pair to exploit context and provide context-aware recommendations for datasets with implicit feedback. Similarly, we propose a method to apply FM-Pair for the task of cross-domain collaborative filtering.

- We release the implementation of FM-Pair as a part of WrapRec, an evaluation framework for recommender systems. The usage of WrapRec framework is briefly described in this chapter. More details about WrapRec can be found in Chapter 6.

In the remainder of this chapter we first provide a brief introduction to FMs and discuss some background and related work. In Section 3.3, we introduce FM-Pair and its pairwise optimization method in detail. In Section 3.4, we propose two applications of FM-Pair for context-aware and cross-domain collaborative filtering. In Section 3.5 we describe the datasets, our evaluation method and the experiments that we performed in this chapter and further elaborate on the results of those experiments. We conclude this chapter by summarizing the contributions and discussing about possible extensions in Section 3.6.

3.2. Background and Related Work

In this section we briefly introduce the model of Factorization Machines for explicit feedback datasets and explain how the input data is represented in FMs. We also review the related work on Factorization Machines in more details.

Factorization Machines represent each user-item interaction by a real-valued feature vector $\mathbf{x}$ with a corresponding output value of y. Typically the feature vectors are binary vectors with two non-zero features corresponding to user and item. In case of explicit user feedback, the output value y would be the actual rating given by the user. Figure 3.1 illustrates how the user-item rating matrix can be modeled by the feature vectors $\mathbf{x}$ and output values y. Each rating in the user-item matrix is represented by a feature vector. The feature vectors indicate which user rated which item and if auxiliary information (such as context) is available for the user-item interactions, it is represented by real-valued auxiliary features.

More specifically, let us assume that the input data is represented by a set S of tuples $(\mathbf{x}, y)$ where $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ is a n-dimensional feature vector and y is its corresponding output value. Factorization Machines learn a model based on the interaction between features. The FM model with the order of 2, where the interactions up to order of 2 (i.e., pairs) are considered, is represented as follows:

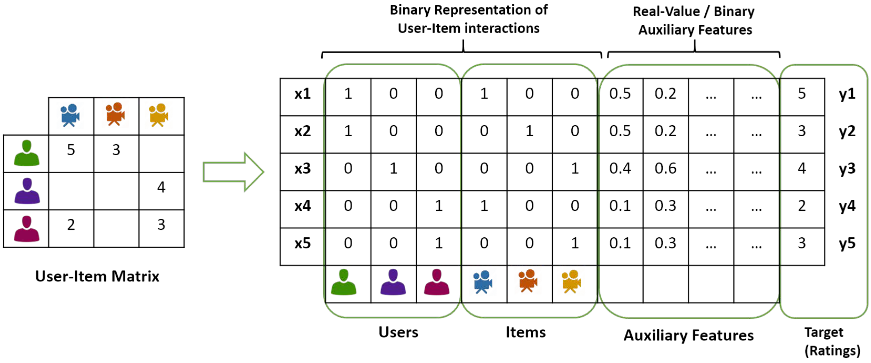
$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} w_{j,j'} x_j x_{j'} \qquad (3.1)$$

where w_j are first order interaction parameters and $w_{j,j'}$ are second order factorized interaction parameters and are defined as $w_{j,j'} = \langle \mathbf{v}_j . \mathbf{v}_{j'} \rangle$ where $\mathbf{v}_j = (v_{j,1}, \ldots, v_{j,k})$ is k-dimensional *factorized* vector for feature j. In fact, FMs factorize any feature that is represented in a feature vector $\mathbf{x}$ and consider the terms $x_j x'_j$ as weight parameters for the pairwise interactions between the factorized parameters. As you might notice, the FM model is similar to a polynomial regression model. However, FMs differ from a standard polynomial regression model by the fact that the parameters $w_{j,j'}$ are not independent parameters as they are inner product of two factorized vectors. This makes the total number of parameters much lower (compared to the regression model) and makes FMs favorable for problems with sparse and high dimensional data such as collaborative filtering. For a FM with n as the dimensionality of feature vectors and k as the dimensionality of factorization, the model parameters that need to be learned are $\Theta = \{w_0, w_1, \ldots, w_n, v_{1,1}, \ldots, v_{n,k}\}$.

Rendle [92] proposes three learning methods to learn the parameters of FMs: Stochastic Gradient Descent (SGD), Alternating Least-Squares (ALS) and Markov Chain Monte Carlo (MCMC) method. In principal all the three methods find the optimal parameters by optimizing the same objective function but they use different techniques to solve the optimization problem. The objective function is defined by summing up the losses of individual samples in the training set. A regularization term is also added to the objective function to prevent over-fitting. The objective function L with square loss over training set S is defined as:

$$L(\Theta, S) = \sum_{(\mathbf{x},y)\in S} (f(\mathbf{x}|\Theta) - y)^2 + \sum_{\theta\in\Theta} \lambda_\theta \theta^2 \tag{3.2}$$

where $\theta \in \Theta$ are model parameters and λ_θ is regularization value for parameter θ. The optimal parameters Θ_{OPT} are found by minimizing the objective function, i.e., $\Theta_{OPT} = \text{argmin}_\Theta L(\Theta, S)$. Rendle [92] showed that all three learning methods has the same time complexity. The advantage of MCMC over the other two optimization techniques is that it is insensitive to hyper-parameters (such as regularization values) which can avoid time-consuming search for hyper-parameters. On the other the advantages of the SGD technique are its simplicity and lower storage complexity. Details about the optimization


Binary Representation of User-Item interactions
Real-Value / Binary Auxiliary Features
User-Item Matrix
Users
Items
Auxiliary Features
Target (Ratings)

5	3	
		4
2		3

	Users			Items			Auxiliary Features				Target (Ratings)	
x1	1	0	0	1	0	0	0.5	0.2	...	...	5	y1
x2	1	0	0	0	1	0	0.5	0.2	...	...	3	y2
x3	0	1	0	0	0	1	0.4	0.6	...	...	4	y3
x4	0	0	1	1	0	0	0.1	0.3	...	...	2	y4
x5	0	0	1	0	0	1	0.1	0.3	...	...	3	y5

Figure 3.1: An overview of data representation in Factorization Machines.

techniques can be found in [92].

Factorization Machines have several advantages compared to other factorization methods:

- **Generalization**: Factorization Machines are *general* factorization models. Despite other factorization models (such as matrix factorization) where specific entities (e.g. user, item) are factorized, in FMs *any* dimension that can be represented in terms of a feature can be factorized to a low-dimensional latent space. In matrix factorization, predictions are generated by taking into account the interaction between user and item, but in FMs the predictions are generated by taking into account the pairwise interaction between *any* pair of features (including user and item). FMs can even take into account higher order interactions between features.

- **Expressiveness**: The fact that the input data in FMs are represented by feature vectors, not only makes FMs easy to use but also makes it possible to mimic several collaborative filtering methods such as Tensor Factorization [50] by feature engineering. This obviate the need to introduce a new prediction and inference methods for such cases. Other example of CF methods that can be represented by FMs are SVD++ [54], Attribute-Aware matrix factorization [27] and Joint Matrix Factorization [114].

- **Performance and Scalability**: The complexity of prediction and inference in FMs are linear in terms of number of latent factors and the number of non-zero features [92] and thus FMs can be scaled for large datasets. Furthermore, a parallel implementation of FMs with shared-memory architecture has been proposed [47] that can achieve noticeable training speed-up on large datasets.

3.3. Learning from Implicit Feedback

In this section we introduce FM-Pair, an adaptation of Factorization Machines with a pairwise optimization methods. Previous studies [80, 97, 113] have reported better performance for pairwise learning-to-rank methods compared to point-wise methods for datasets with implicit feedback. While FM-Pair benefits from the effectiveness of pairwise learning-to-rank, it can also leverage the arbitrary auxiliary features that might be available in the input data.

The optimization technique in FM-Pair is inspired by the BPR [97] optimization criterion. The BPR method comes with an assumption that all observed positive feedback is preferred over the missing preferences. The training data in BPR consist of a user and a pair of items where the first item, referred as the *positive item*, is chosen from the user positive feedback and the second item, referred as the *negative item*, is sampled from the unobserved interactions. More specifically, the training data in BPR would be a set of tuples $S_P = \{(u,i,j) | i \in I_u^+ \wedge j \in I \backslash I_u^+\}$ where I_u^+ is set of all positive feedback from user u, i is an item with positive feedback from u and j is a missing item for that user which is sampled uniformly from the unobserved items. The BPR optimization technique learns to correctly rank the items in any given pair of items, with respect to a given user.

In FM-Pair arbitrary information can be represented by auxiliary features and thus the pairwise learning method should be able to exploit those features. Let us assume that $\mathbf{z}(u,i)$ are the auxiliary features associated with user u and item i. Then the tuple $(u,i,j) \in S_P$ indicates user u prefers item i over item j under the observed auxiliary features $\mathbf{z}(u,i)$. Auxiliary features can be user features, item features, context or additional information about user-item interaction. FM-Pair finds the optimal parameters Θ by maximizing the following likelihood function:

$$\prod_{(u,i,j)\in S_P} p(i >_{u,\mathbf{z}} j|\Theta) \tag{3.3}$$

where $i >_{u,\mathbf{z}} j$ indicates item i is preferred over j by user u under auxiliary features $\mathbf{z} = \mathbf{z}(u,i)$. Similar to the BPR model, the probability $p(i >_{u,\mathbf{z}} j|\Theta)$ is defined by mapping a utility function $g_\mathbf{z}(u,i,j)$ to a value between 0 and 1. This can be done by the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Therefore:

$$p(i >_{u,\mathbf{z}} j|\Theta) = \sigma(g_\mathbf{z}(u,i,j|\Theta)) \tag{3.4}$$

The utility function g captures the interaction between user u, item i and item j with presence of auxiliary features $\mathbf{z}(u,i)$. Similar to BPR, the utility function g is defined by calculating the difference between the utility of individual interactions. FM-Pair calculates the utility of individual interactions by taking into account the auxiliary features $\mathbf{z}(u,i)$. We define the utility function g as:

$$g_\mathbf{z}(u,i,j|\Theta) = f_\mathbf{z}(u,i|\Theta) - f_\mathbf{z}(u,j|\Theta) \tag{3.5}$$

The utility of individual interactions $f_\mathbf{z}(u,i|\Theta)$ can be calculated using equation (4.3). In this case the input feature vector $\mathbf{x}$ is a sparse vector in which features corresponding to user u, item i and $\mathbf{z}(u,i)$ are non-zero. Thus the vector $\mathbf{x}$ can be represented with the following sparse form:

$$\mathbf{x}(u,i,\mathbf{z}) = \mathbf{x}_{u,i,\mathbf{z}} = \{(u,x_u),(i,x_z),\{(z,x_z)|z \in \mathbf{z}\}\} \tag{3.6}$$

where x_z is the value of feature z and can be a real value number. The parameters x_u and x_i are considered to be 1 to indicate the corresponding user and item of a feedback (see Figure 3.1 for clarity). By replacing $\mathbf{x}_{u,i,\mathbf{z}}$ in equation (4.3), and expanding $w_{j,j'}$, the individual utility function $f_\mathbf{z}(u,i|\Theta)$ can be written as:

$$\begin{aligned} f_\mathbf{z}(u,i|\Theta) = f(\mathbf{x}_{u,i,\mathbf{z}}|\Theta) = w_0 + w_u + w_i + \sum_{z\in\mathbf{z}} w_z x_z + \sum_{f=1}^{k} v_{u,f} v_{i,f} \\ + \sum_{z\in\mathbf{z}} x_z \sum_{f=1}^{k} v_{u,f} v_{z,f} + \sum_{z\in\mathbf{z}} x_z \sum_{f=1}^{k} v_{i,f} v_{z,f} \end{aligned} \tag{3.7}$$

By replacing (4.9) in (3.5), the pairwise utility function g can be written as:

$$g_\mathbf{z}(u,i,j|\Theta) = w_i - w_j + \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}) + \sum_{z\in\mathbf{z}} x_z \sum_{f=1}^{k} v_{z,f}(v_{i,f} - v_{j,f}) \tag{3.8}$$

Now we define the FM-Pair objective function by taking the logarithm of the likelihood function in equation (3.3) and adding regularization terms:

$$L(\Theta, S_P) = \sum_{(u,i,j)\in S_P} \ln\sigma(g_{\mathbf{z}}(u,i,j|\Theta)) - \sum_{\theta\in\Theta} \lambda_\theta \theta^2 \tag{3.9}$$

Since the FM-Pair objective function is based on likelihood, the optimal parameters are found by maximizing this function, i.e., $\Theta_{OPT} = \text{argmax}_\Theta L(\Theta, S_p)$.

To find the optimal parameters, optimization is done with Stochastic Gradient Descent (SGD) technique. First the parameters are initialized and then they are updated by iterating over the tuples $(u,i,j) \in S_P$ using the following update rule:

$$\theta \leftarrow \theta - \eta \frac{\partial L(\Theta, S_P)}{\partial \theta} \tag{3.10}$$

where η is the learning rate. By replacing (3.9) in (3.10), the update rule would be:

$$\theta \leftarrow \theta + \eta(\frac{e^{g_{\mathbf{z}}}}{1+e^{g_{\mathbf{z}}}} \frac{\partial g_{\mathbf{z}}}{\partial \theta} + \lambda_\theta \theta) \tag{3.11}$$

Based on equation (4.11), the gradients of $g_{\mathbf{z}}$ with respect to θ is defined as:

$$\frac{\partial g_{\mathbf{z}}}{\partial \theta} = \begin{cases} 1 & \text{if } \theta = w_i \\ -1 & \text{if } \theta = w_j \\ v_{i,f} - v_{j,f} & \text{if } \theta = v_{u,f} \\ v_{u,f} + \sum_{z\in\mathbf{z}} x_z v_{z,f} & \text{if } \theta = v_{i,f} \\ -v_{u,f} - \sum_{z\in\mathbf{z}} x_z v_{z,f} & \text{if } \theta = v_{j,f} \\ x_z(v_{i,f} - v_{j,f}) & \text{if } \theta = v_{z,f} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The parameters w_i are typically initialized by 0 and the factorization parameters $v_{*,f}$ should be initialized by a zero-mean normal distribution with standard deviation σ_0 for a better performance. The parameter σ_0 is one of the hyper-parameters of SGD that typically is tuned by cross-validation or by using a validation set.

The SGD algorithm typically iterates over the entire training data and updates the parameters according to the update rule. [97] suggests to draw the positive feedback from the input data by bootstrapping with replacement to prevent consecutive updates on a same user or item for faster convergence. FM-Pair first draws a positive feedback $(u, i, \mathbf{z}_{u,i})$ from the input dataset D and then samples a negative item j from $I \backslash I^+$ uniformly. In the next step, the utility function g is calculated and then the parameters are updated according to the update rule (3.11). Figure 2 summarizes the FM-Pair learning algorithm.

3.3.1. Computational Complexity

FM-Pair have a linear learning and prediction time complexity. The main effort in the FM-Pair SGD algorithm is to calculate $g_{\mathbf{z}}(u,i,j)$ (line 6 in Figure 2). According to (4.11), this can be done in $\mathbb{O}(k + |\mathbf{z}|k)$. Sampling positive pairs (u,i) and negative items j (lines

Algorithm 1: Learning FM-Pair with Stochastic Gradient Descent.

```
   Input: Training Data $D$
   Output: Model parameters $\Theta$
1  initialize $\Theta$
2  do
3      sample $(u,i)$ from $D$ and create $\mathbf{x}_{u,i,\mathbf{z}}$
4      sample $j$ from $I \backslash I_u^+$ create $\mathbf{x}_{u,j,\mathbf{z}}$
5      let $g_\mathbf{z}(u,i,j|\Theta) = f(\mathbf{x}_{u,i,\mathbf{z}}|\Theta) - f(\mathbf{x}_{u,j,\mathbf{z}}|\Theta)$
6      update $\Theta$ according update rule (3.11)
7  while convergence
8  return $\Theta$
```

4 and 5 in Figure 2) can be done efficiently [95] in $\mathbb{O}(1)$. Updating the parameters are done according to (3.11) and (3.12). For each point $(u,i,j,\mathbf{z})$ in the training data only the parameters corresponding to that point is updated since the gradient of the other parameters are 0. Thus the complexity of updating the parameters for each training point (line 7 in Figure 2) is $\mathbb{O}(|\mathbf{z}|k)$ according to (3.12). Putting it all together, the complexity of one iteration in FM-Pair SGD algorithm is $\mathbb{O}(k(|\mathbf{z}|+1))$ and the complexity of a full iteration on the entire training data D is $\mathbb{O}(k(|\mathbf{z}|+1)|D|)$. Therefore, the computational complexity of FM-Pair is linear in terms of number of latent factors k and number of auxiliary features $\mathbf{z}$. In the experiments section we empirically demonstrate the training time of FM-Pair based on k and $\mathbf{z}$.

3.3.2. Analogy Between FM-Pair and BPR-MF

FM-Pair can mimic other factorization methods with feature engineering, similar to the standard Factforization Machines. A specific model that can be represented with FM-Pair is BPR-MF (BPR with Matrix Factorization utility) [97]. The matrix factorization model calculates the utility of a user-item interaction as the inner product of the user and item factors, that it, $f^{MF}(u,i) = \sum_{f=1}^{k} v_{u,f} v_{i,f}$. By considering f^{MF} as the utility function of matrix factorization model, the utility of triples (u,i,j) in BPR-MF is defined as:

$$g^{MF}(u,i,j) = \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}) \tag{3.13}$$

By comparing the above equation with (4.11), one can notice that g^{MF} is a special case of $g_\mathbf{z}(u,i,j)$ where $\mathbf{z} = \emptyset$ (i.e., no auxiliary features) and the parameters w_i and w_j are 0. In fact when there are no auxiliary features, FM-Pair, compared to BPR-MF, learns two additional parameters w_i and w_j that can be regarded as global item biases for positive and negative items.

3.4. Improved Recommendations with Auxiliary Data

A great advantage of Factorization Machines as mentioned earlier, is that they are capable of exploiting arbitrary information as auxiliary features to improve recommendations. In this section, we propose to apply FM-Pair for the context-aware and cross-domain collaborative filtering for datasets with implicit feedback.

3.4.1. Context-Aware Recommendation with FM-Pair

Context is a dynamic set of parameters describing the state of the user at the moment of experience. Some studies also consider user and item attributes as a type of context [3]. Context-aware recommendation relies on this additional information to better learn from the interactions between users and items. In FM-Pair, context can be considered as one type of auxiliary features for user-item interactions. We represent the context of a user feedback with the following sparse representation:

$$\mathbf{z}(u,i) = \{(z, x_z) | x_z \neq 0\} \tag{3.14}$$

where z is a context and x_z is its corresponding value. For example, if available context of an interaction (u,i) are user mood (e.g. “happy") and movie genre (e.g. “action"), then we can represent the context of interaction with $\mathbf{z}(u,i) = \{(\text{happy}, 1), (\text{action}, 1)\}$. By expanding the feature vector $\mathbf{x}$ with context features, the feature vector $\mathbf{x}$ would have the following sparse form:

$$\mathbf{x}(u,i,\mathbf{z}) = \mathbf{x}_{u,i,\mathbf{z}} = \{(u, x_u), (i, x_i)\} \cup \mathbf{z}(u,i) \tag{3.15}$$

and the following expanded form:

$$\mathbf{x}_{u,i,\mathbf{z}} = (\underbrace{0,\ldots,0,x_u,0,\ldots,0}_{|U|}, \underbrace{0,\ldots,0,x_i,0,\ldots,0}_{|I|}, \underbrace{x_{z_1},\ldots,x_{z_m}}_{|Z|}) \tag{3.16}$$

where Z is the set of contextual feature and $m = |Z|$. The parameters x_u, x_i and x_z are the actual values of features, which be seen as weight parameters specifying the strength of features. The parameters x_u and x_i are typically considered to be 1 to just indicate the presence of features. The parameters x_z can be assigned with any real-values to control the weight of contextual features. For the categorical context such as user mood typically binary values are used similar to x_u and x_i. However, the binary values can also be replaced with real values. If the features are continuous by their nature (e.g. user age, time of the day and number of clicks), a real value can be considered as the value of the feature. According to [99] it is often desirable that the value of auxiliary features sum up to 1. Continuous features can also be mapped to categorical features by forming several bins. With our preliminary experiments we found that mapping continuous context to categorical features results to better accuracy. In Section 5.5, we describe our feature mapping on the two datasets with contextual features.

3.4.2. Cross-Domain Recommendations

Cross-Domain Collaborative Filtering (CDCF) methods exploit additional information from *source*[3] domains to improve recommendations in a *target* domain. The main idea of CDCF is that a user’s taste in one domain (e.g., movie) can be exploited to better learn

[3] The source domains are also referred as “auxiliary" domains. In this chapter we used the term “source" to avoid confusion with auxiliary features.

user's taste on another domain (e.g., music). Different methods for the problem of CDCF have been proposed. A great overview of the existing solutions can be found in [15]. Among different techniques that have been developed for cross-domain CF, in an earlier study [78] we proposed a method for the problem of CDCF with Factorization Machines for the rating prediction problem. In this section, we propose a similar technique that has been adapted for FM-Pair and thus can be applied for datasets with implicit feedback.

Thanks to the flexibility of Factorization Machines in exploiting additional features, the information from source domains can be translated to auxiliary features to expand the feature vectors in the target domain. The expanded feature vectors can be then used as the input data for FM-Pair to train a model. With the same assumption as the case of context-aware recommendation, the extra features can enrich the feature vectors to better learn the user preferences. We refer to our proposed cross-domain CF method with FM-Pair as FM-Pair-CD. The advantage of FM-Pair-CD is that it does not require any adaptation to the underlying FM-Pair model and thus the model can be used out-of-the-box. FM-Pair-CD proposes a simple and yet effective way to exploit features from source domains in order to improve the performance of recommendations in the target domain. Here the domain refers to the type of item that user interacted with. For example, if user provides a feedback for a movie, the domain is “movies".

To understand how FM-Pair-CD represents the auxiliary features, suppose p is the number of source domains and $I_j(u)$ is the set of items in domain j that user u interacted with. FM-Pair-CD proposes to use one auxiliary feature for every item that user interacted with in the source domains. Therefore, the feature vectors $\mathbf{x}$ in the target domain can be represented with the following sparse form consisting of both target and source domain features:

$$\mathbf{x}(u,i) = \{ \underbrace{(u,1),(i,1)}_{\text{target domain features}} \underbrace{,\cup_{j=1}^{p}\{(z, x_z(u,j)) | z \in I_j(u)\}}_{\text{source domains' features}}\} \tag{3.17}$$

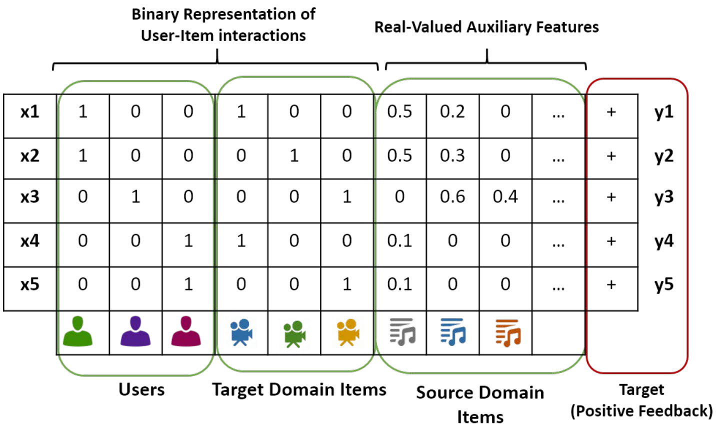
where $x_z(u,j)$ is the value of feature z, i.e., the weight that should be considered for the interaction of user u with item z in the source domain j. We propose the following two approaches to define the feature values $x_z(u,j)$:

- Binary indicator: In this case similar to the representation of user and item, the presence of a source domain feature is specified by indicator value of 1. We denote $x_z^B(u,j)$ as binary representation of features. In other words, in this case, with binary values we indicate which items have been rated by the user in the source domains.

- Normalized by count: In this case the values of source domain features are normalized by the number of feedback that user provided in the source domain. The normalized value $x_z^C(u,j)$ is defined by:

$$x_z^C(u,j) = \frac{1}{|I_j(u)|} \tag{3.18}$$

The auxiliary features in the FM-Pair-CD method, are in fact the items in source domains with feedback from the user. The normalization of auxiliary features ensures that


Binary Representation of User-Item interactions
Real-Valued Auxiliary Features
Users
Target Domain Items
Source Domain Items
Target (Positive Feedback)

x1	1	0	0	1	0	0	0.5	0.2	0	...	+	y1
x2	1	0	0	0	1	0	0.5	0.3	0	...	+	y2
x3	0	1	0	0	0	1	0	0.6	0.4	...	+	y3
x4	0	0	1	1	0	0	0.1	0	0	...	+	y4
x5	0	0	1	0	0	1	0.1	0	0	...	+	y5

Figure 3.2: Representation of feature vectors in FM-Pair-CD.

the target features are not dominated by auxiliary features. Figure 3.2 illustrates how FM-Pair-CD represents feature vectors for input data. In this example the target domain is the “movie" domain and source domain is the “music" domain. The music items that same user has interacted with are considered as auxiliary features for the interactions in the target domain. As you might notice from Figure 3.2, the total number of auxiliary features is the number of items in source domains. We found that using only a fraction of user feedback from source domains results in almost same improvement compared to the case that all user feedback from source domains are used. Such feature selection makes the size of feature vectors smaller and thus the training and prediction become faster. The selected features can be based on most popular items, highly rated items, or just randomly chosen items. In fact, by applying this feature selection method, the auxiliary features in (5.22) can be represented as:

$$\cup_{j=1}^{p} \{(z, x_z(u, j)) | z \in I_j(u) \wedge z \in I^S(u)\} \tag{3.19}$$

where $I^S(u)$ are the selected features. Since finding the most informative items from source domains is not the focus of this work, we just consider $I^S(u)$ to be random items from source domains.

3.5. DATASETS, EXPERIMENTS AND EVALUATION

In this section we first introduce the datasets that we used in this chapter, we then describe our evaluation method, and then we explain our experiments.

3.5.1. DATASETS

We used the following four datasets in this chapter, ranging from the popular MovieLens dataset to a recent industry dataset of XING. These datasets are chosen so that we can test different scenarios with auxiliary features and to cover both implicit and explicit feedback scenarios.

Table 3.1: Statistics of the dataset used in this chapter.

Dataset	**#Users**	**#Items**	**#Feedback**	**Sparsity(%)**	**Scale**
MovieLens 100K	943	1,682	100K	93.74	1-5
Amazon	15,994	84,508	270K	99.98	1-5
Frappe	957	4,082	96K	97.54	Implicit
XING	9,751	9,821	223K	99.76	Implicit

- **MoviLens 100K**: The MovieLens 100K dataset[4] is a popular benchmark dataset for recommender systems with explicit feedback consisting of 100K user rating on the scale of 1 to 5. This dataset also contains several user and item attributes that can be used as auxiliary features in FM-Pair.

- **Amazon**: The Amazon dataset [62] consists of user ratings on the products on the Amazon website. The ratings are on the same scale as the MovieLens dataset. The products belong to four different domains: Books, Music CDs, Video tapes and DVDs. This dataset has been used for some previous work on cross-domain CF [38, 78].

- **Frappe**: Frappe is a context-aware mobile app discovery tool. It logs number of time users run an application on their mobile phone. It also logs several contexts such as time, date, location and weather. The Frappe dataset [7] consists of 100K implicit positive feedback. An observation is considered as a positive feedback if user runs an application at least one time. We used this dataset because of the presence of several contextual features. This dataset has also been used in one of the related work [80].

- **XING**: XING[5] is a professional social network and a job discovery platform. This dataset is a subset of the RecySys 2016 challenge[6] and consists implicit user feedback on job postings. Users can either click on a job posting, bookmark it or apply for it. We consider any user action on a job positing as a positive feedback. Since the original dataset was extremely sparse, we densified the dataset by considering users and items with at least 20 interactions.

Table 5.1 list the statistics of the three datasets that we used in this chapter.

3.5.2. EXPERIMENTS SETUP AND EVALUATION

All experiments in this chapter are done with four-fold cross-validation to make sure the hyper-parameters are not tunned for one particular test set. FM-Pair is implemented as a part of the WrapRec[77] open source project. WrapRec can be used as a command line tool in different platforms. The source code and documentation can be found in `http://wraprec.crowdrec.eu/`.

[4] http://grouplens.org/datasets/movielens/

[5] http://www.xing.com/

[6] http://2016.recsyschallenge.com/

EVALUATION METHOD

The performance of our experiments are evaluated with two ranking metrics namely Recall and NDCG (Normalized Discounted Cumulative Gain). To calculate these metrics on datasets with positive-only feedback, typically for each user in the test set a rank list is created. The metrics are then calculated based on the presence of test feedback on top of the list. However, when auxiliary features such as context are available for the feedback, this strategy is not suitable as it is not clear based on which context the scores (i.e., the utility of a user-item interaction) should be generated. To make an unbiased estimation of performance when auxiliary features are available, we applied the approach known as *One-plus-random* [9, 19] where the following procedure is applied: For every user-item interaction in the test set (which is a positive feedback with possible auxiliary features), 1000 random items which are not observed with that user are scored based on the same user and auxiliary features. Then a ranked list including the target item is created. The position of the target item in the ranked list is used to calculate the metrics. If the targeted test point appears in the top N positions of the list, then it would be considered as a *hit*. In case of a hit, the recall of a single test point would be 1 and otherwise it would be 0. The overall metric is calculated by averaging on all points. That is:

$$\text{Recall@N} = \frac{1}{|T|}\sum_{i=1}^{|T|} \mathbb{I}(r_i \leq N) \tag{3.20}$$

where r_i is the rank of the ith test sample in the generated list, $\mathbb{I}$ is the indicator function and $|T|$ is the size of test set. The above metric can be interpreted as the hit rate of the test samples where a hit is defined as presence of the relevant test point in the top N positions of the ranked list.

Based on the one-plus-random evaluation method, we also adopted the MRR metric as follows:

$$\text{MRR@N} = \frac{1}{|T|}\sum_{i=1}^{|T|} \frac{1}{r_i}\mathbb{I}(r_i \leq N) \tag{3.21}$$

We use the MRR metric since it also takes into account the position of the relevant item in the list. Note that these metrics are not absolute metrics [9] and their value does not reflect the real performance of the system. However, they are reliable metric to *compare* the performance of different experiments.

3.5.3. COMPARISON OF FM-PAIR WITH OTHER METHODS

The proposed FM-Pair algorithm is compared with several methods on the four datasets. In this experiment no auxiliary or context features are used, and the methods are compared only based on the positive feedback in the user-item matrix. The following setups have been tested in this experiment:

- **Most-Popular**: This is a baseline method where the most popular items are recommended to the users.
- **FM-Map**: In this setup the training is done similar to the FMs for rating prediction. For the positive feedback in the training set the output value of +1 is considered. Same number of unobserved interactions are sampled uniformly and they

are considered as negative feedback with output value of -1. The positive and sampled negative feedback is used to train the FMs model. To resemble the experiment for datasets with explicit feedback, the ratings higher than user's average were mapped to +1 and negative feedback were sampled in the same way as the implicit feedback datasets.

- **BPR-MF**: This method is an implementation of BPR method with Matrix Factorization as the utility function [97]. With this method there is no possibility to incorporate auxiliary information.
- **FM-Pair**: This is the proposed method in this chapter. The FM-Pair algorithm is listed in Figure 2. For the two datasets of MovieLens and Amazon with explicit feedback, the ratings above user's average rating is considered as positive feedback.

Hyper-parameters and Experimental Reproducibility

The three datasets of MovieLens, Amazon and Frappe are publicly available. For the experiments in this section, the number of factors (parameter k) is set to 10 and the number of iterations of the SDG algorithm on the training data is set to 300. The standard deviation of the the normal distribution for initialization (parameter σ_0) is set to 0.1. The learning rate of the SGD algorithm (parameter λ) varies per dataset. The following learning rates are used for each dataset: XING: 0.075, Frappe and MovieLens: 0.005, Amazon: 0.001. The two hyper-parameters of σ_0 and λ are found with a grid search with our four-fold cross-validation experiments. That is, the hyper-parameters that result to the best performance on the *average* of all folds are chosen, thus they are not optimized for one particular test set.

Description of the Results

Table 3.2 lists the performance of the above five methods on the four datasets based on Recall@10 metric[7]. As it can be seen from the table, in three out of the four datasets, the FM-Pair is performing better than the other baselines. In the XING dataset, BPR-FM is slightly performing better than FM-Pair. It is worth mentioning that when there are no auxiliary features (such as this experiment) the underlying model of FM-Pair is very similar to BPR-MF (see Section 3.3.2). This can explain the close performance of the two methods. Nevertheless, the additional parameters of FM-Pair contribute to some accuracy gain in three out of the four datasets. Another observation that you can see in this table, is that the FM-Map method is not really effective for ranking compared to the two pairwise methods in our experiments. This can be explained by the fact that the standard FMs optimization method is a pointwise method that in principle learns to correctly predict the mapped scores and it is not optimized for ranking. Note that we also tried to map the sampled unobserved feedback to other values than -1, but in practice the result were very similar. For the two datasets of Amazon and MovieLens with explicit feedback, we did an additional experiment where the model is trained with the standard FM with original ratings. The ranked lists are then generated based on the

[7] Other ranking metrics such as MRR and NDCG are also calculated, but due to the high correlation between the values, we only report Recall@10 for this experiment.

Table 3.2: Comparison of different learning-to-rank methods on the four dataset based on Recall@10. The numbers in parentheses are standard deviations of the results in the four folds.

Method / Dataset	**XING**	**Frappe**	**Amazon**	**MovieLens**
Most-Popular	0.0306 (0.0005)	0.1643 (0.0087)	0.0222 (0.0010)	0.1180 (0.0012)
FM-Map	0.0287 (0.0015)	0.1230 (0.0260)	0.0348 (0.0014)	0.0728 (0.0036)
BPR-MF	0.2925 (0.0030)	0.1428 (0.0167)	0.0962 (0.0056)	0.2278 (0.0024)
FM-Pair (this chapter)	0.2920 (0.0077)	0.1816 (0.0161)	0.0972 (0.0030)	0.2357 (0.0016)

predicted ratings. For this experiment we achieved a Recall of 0 for Amazon and 0.001 for the MovieLens dataset. This shows that even for datasets with explicit feedback, ranking based on predicting ratings is not really effective. Previous studies [6, 19] also showed the ineffectiveness of pointwise methods for ranking.

COMPARISON WITH GPFM

In addition to the methods listed in Table 3.2, we also compare the performance of FM-Pair with the pairwise method of GPFM [80] since it is very close to our work as it also adapted a pairwise optimization technique for FMs. We tested the GPFM method on Frappe and MovieLens datasets. For the Frappe dataset we achieved a Recall@10 of 0.1615 and for the MovieLens a Recall@10 of 0.1562 was achieved, both less than the performance of FM-Pair (See Table 3.2). However, the remarkable advantage of FM-Pair compared to GPFM is that the computational complexity of FM-Pair is significantly lower than GPFM. Figure 3.3 compares the epoch time (the time of a full iteration on dataset) of the three methods of BPR-MF, GPFM and FM-Pair on two datasets of Frappe and MovieLens. The numbers are represented in the log scale due to the significant difference between the epoch time of GPFM with the other two methods. The epoch time of FM-Pair is slightly higher than the epoch time of BPR-MF due to the presence of two additional parameters (see Section 3.3.2). The GPFM method on the other hand, is significantly slower that the other two methods due to the fact that GPFM need to calculate the inverse of the covariance matrix of the preference kernels [80]. This introduces a significant computational complexity in the training of GPFM.

Due to the high space complexity of GPFM, running the experiment on the larger datasets of XING and Amazon was not even possible on our testing machine[8]. Since the complexity of GPFM is significantly higher than the other methods, we did not further investigate on testing GPFM on our larger datasets.

We used the Matlab implementation[9] of GPFM that was released with that work to train the model but the evaluation was done in the same way as other methods, as described in Section 3.5.2, to have a fair comparison between the methods. The kernel of the Gaussian process is chosen to be the RBF kernel, the recommended kernel of the model.

[8] The experiments are run on a machine with 8 GB of memory and an Intel i5 processor with 4 CPU cores.

[9] `http://trungngv.github.io/gpfm/`

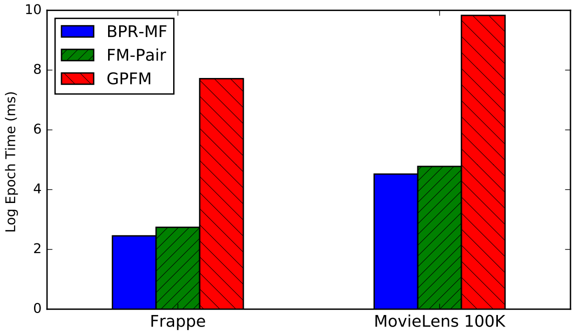


Figure 3.3: Comparison of the epoch time (the time of a full iteration on the dataset in milliseconds) of three pairwise learning-to-rank methods on the log scale.

3.5.4. FM-PAIR WITH AUXILIARY DATA

In the second set of our experiments, we test the performance of FM-Pair with auxiliary information. We use FM-Pair for the two scenarios that we described in Section 3.4: context-aware recommendation and cross-domain collaborative filtering.

FM-PAIR WITH CONTEXT

Among the four datasets that we use in this chapter the two datasets of Frappe and MovieLens have several context and attributes. The final context and attributes that are used in the experiments are found with a naive greedy method, where the top performing features are combined. For the Frappe dataset the following contexts are used: daytime (with seven possibilities of `sunrise, morning, noon, afternoon, sunset, evening, night`), weekday (day of the week), isweekend (with two values of `workday` or `weekend`) and homework (with three values of `unknown, home, work`). For the MovieLens dataset, we used the genre of movies as auxiliary features in FM-Pair. Each movie in this dataset has one or more genres from the 17 genres that exists in this dataset.

Table 3.3 compares the performance of FM-Pair with context or attributes as auxiliary features (FM-Pair-Context), with the original FM-Pair without any auxiliary features. We reported Recall@10 and MRR@10 for the two setups in this experiment. The results show that the FM-Pair can clearly exploit context or auxiliary attributes if they are present in a dataset.

CROSS-DOMAIN RECOMMENDATION WITH FM-PAIR

To test the performance of FM-Pair for cross-domain collaborative filtering, we use the dataset of Amazon where the items come from four different domains. We use the two domain of `books` and `music` as target domains and use other domains as source domains. The experiments are done with four-fold cross-validation and on each fold only the interactions from the target domain are evaluated. The source domains are used to generate auxiliary features, as described in Section 3.4.2. Figure 3.4 illustrates our four-fold cross-validation splitting method on the Amazon dataset with one target domain

Table 3.3: Performance of FM-Pair with context compared to the standard FM-Pair without auxiliary features.

	Recall@10		**MRR@10**	
Method / dataset	Frappe	MovieLens	Frappe	MovieLens
FM-Pair	0.1816	0.2357	0.0745	0.1027
FM-Pair-Context	**0.2064**	**0.2601**	**0.0890**	**0.1191**

and three source domains. The design choices and hyper-parameters of the experiment are the same as the ones described in Section 3.5.3 except that for the `books` domain, the learning rate of the SGD algorithm is set to 0.001 due to its faster convergence.

The following three setups are used to demonstrate the performance of the FM-Pair method for cross-domain recommendations. In all setups, the evaluation is done for the target domain.

- **FM-Pair**: In this setup FM-Pair is only applied on the target domain and source domains are not exploited.

- **FM-Pair-All**: In this setup the source domains are used as additional training samples. Thus, no auxiliary features are generated.

- **FM-Pair-CD**: In this setup source domains are exploited to generate auxiliary features for the training feature vectors in the target domain. In fact, the number of training samples in this setup is the same as the first setup but the feature vectors are expanded with auxiliary features. The value of auxiliary features are defined based on equation (3.18) and for each user we take at most five feedback from each source domain to avoid large feature vectors.

Table 3.4 lists the performance of the above three setups on the Amazon dataset where domains of `books` and `music` are used as target domains. First, as you can see in this table, the FM-Pair-CD outperform the other two methods on the accuracy of recommendations. The second interesting observation is that when recommendations are generated for a target domain, using only interactions from that particular domain (setup FM-Pair) is better than using the entire dataset for training (setup FM-Pair-All). Similar effect has been shown in a previous study as well [73]: using a sensible subset of data can perform better than using the entire dataset. In fact, exploiting the source domains by means of auxiliary features in FM-Pair has a better effect than blindly combining all samples from all domains in one big dataset.

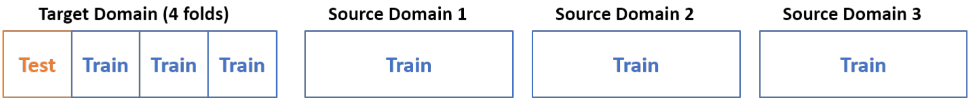


Figure 3.4: Illustration of the cross-validation splitting method for the cross-domain recommendation experiment with FM-Pair on the Amazon dataset.

Table 3.4: Performance of cross-domain recommendation with the FM-Pair-CD method compared to the single-domain training scenarios.

	Recall@10		**MRR@10**	
Method / Target Domain	Books	Music	Books	Music
FM-Pair (Target-only)	0.1058	0.0966	0.0452	0.0356
FM-Pair-All	0.0831	0.0855	0.0357	0.0378
FM-Pair-CD	**0.1238**	**0.1060**	**0.0490**	**0.0405**

3.5.5. CONVERGENCE AND COMPLEXITY OF FM-PAIR

In this section we further analyze the convergence and complexity of FM-Pair by monitoring the performance of FM-Pair with different number of iterations (of the training algorithm) and different dimensions of factorization. Figure 3.5 compares the performance of FM-Pair, FM-Pair-Context and BPR-MF on different number of epochs on the two datasets of Frappe and MovieLens. In Figure 3.6, we illustrate the performance of cross-domain recommendations with FM-Pair-CD compared to the two setups of FM-Pair and FM-Pair-All on different number of epochs. The models are evaluated on every 10 epochs with Recall@10.

As you can see in Figure 3.5, on the Frappe and MovieLens dataset all models converge rather fast due to the density of datasets. However, an interesting observation on the Frappe dataset is that FM-Pair and FM-Pair-Context already achieve a high recall after the first few epochs whereas the BPR-MF algorithm converge later and yet cannot achieve FM-Pair's performance even with higher number of epochs. A closer examination of Section 3.3.2 and Table 3.2 can explain this result. The high recall of the popularity algorithm on the Frappe dataset exhibits a high tendency on popular items in this dataset. On the other hand, the presence of bias parameters w_i and w_j in the FM-Pair model can learn such biases, that turns to be very effective on training the model in popularity-skewed datasets such as Frappe. The effectiveness of such bias parameters on CF models have also been shown in previous studies (e.g. [55]).

On the two datasets of Amazon, as you can see in Figure 3.6, the FM-Pair-All converge faster, most likely due to the larger number of training samples, but fails to reach the performance of FM-Pair and FM-Pair-CD. The FM-Pair-CD performs better that the other two methods even with smaller number of epochs and thus it is an effective model to leverage cross-domain auxiliary feature.

In Section 3.3.1 we showed that the complexity of FM-Pair is linear in dimensionality of factorization (parameter k) and the number of auxiliary features ($|\mathbf{z}|$). Experimental results also confirms the linearity of FM-Pair. Figure 3.7 illustrates the influence of the two parameters k (left chart) and $|\mathbf{z}|$ (right chart) on the epoch time of different datasets (the effect of parameter $|\mathbf{z}|$ is only illustrated on the Frappe datasets since this is the only dataset with *multiple* context features). The reported epoch time is the average epoch time on four-fold cross-validation experiments and the bars indicate the standard deviation of the four folds. As you can see in the two charts for both parameters the epoch time grows linearly (with small errors) and thus the linearity of FM-Pair can be confirmed.

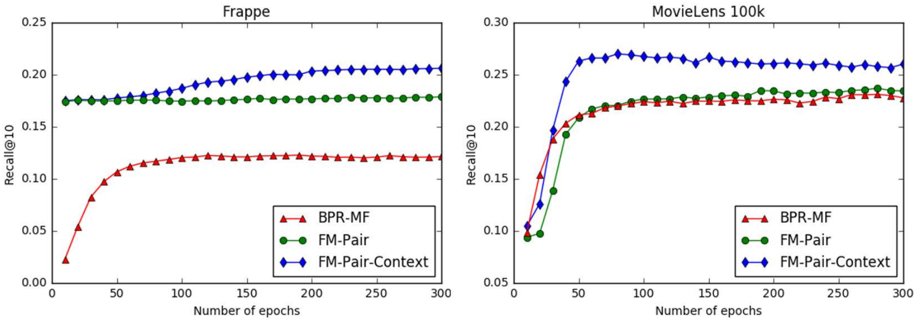


Figure 3.5: Empirical comparison of the convergence of FM-Pair (with or without context) with BPR-MF on the two datastes with auxiliary features.

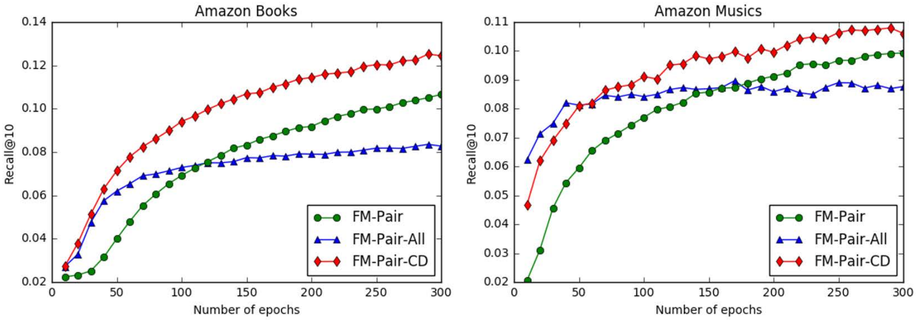


Figure 3.6: Empirical comparison of the convergence of cross-domain CF with FM-Pair compared to the single-domain models on two domains of the Amazon dataset.

3.5.6. Using WrapRec

The implementation of this chapter is published in the WrapRec toolkit. WrapRec is an open source evaluation framework for recommender systems that can wrap algorithms from different frameworks and evaluate them under same setup. WrapRec is written is C# and can be used in multiple platforms. WrapRec can be used as a command-line tool. To use WrapRec all setting need to be defined in one configuration file. The configuration file specifies the model and its parameters, how the dataset should be read and split, and how the evaluation should be done. The command-line tool can be downloaded from the WrapRec website[10] and can be simply used as:

- (Windows): `WrapRec.exe [path-to-config-file]`
- (Linux and Mac): `mono WrapRec.exe [path-to-config-file]`

Details about the format of the configuration file and usage of WrapRec can be found in the WrapRec website. The experiments on this chapter can be reproduced by using the configuration file that is defined for the experiments.

[10] `http://babakx.github.io/WrapRec/`

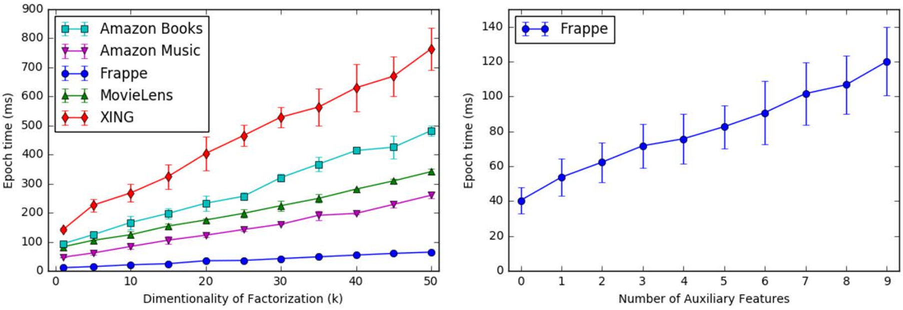


Figure 3.7: Empirical comparison of the the training time of different datasets based on the dimensionality of factorization (left chart) and number of auxiliary features (right chart).

3.6. CONCLUSION AND FUTURE WORK

In this chapter we introduce FM-Pair, an adaptation of Factorization Machines with a pairwise learning-to-rank optimization technique. In contrast to the original model of Factorization Machines, FM-Pair can be used effectively for datasets with implicit feedback, thus addressing a wider range of problems in recommender systems. In this chapter we show that for ranking problems, FM-Pair is more effective than the standard FMs even on datasets with explicit feedback. FM-Pair leverages a pairwise learning-to-rank method inspired by the Bayesian Personalized Ranking (BPR) criterion, which optimizes the model parameters for ranking. Similar to the standard Factorization Machines, FM-Pair can exploit additional features such as context, user and item attributes, cross-domain information and any discrete or real-valued auxiliary features. In this chapter we also propose to apply FM-Pair for context-aware and cross-domain collaborative filtering problems. Experimental results on four datasets with implicit or explicit feedback showed the effectiveness of FM-Pair for datasets with implicit or explicit feedback with or without auxiliary features. We showed that when no auxiliary features are exploited FM-Pair is at least as accurate as state-of-the-art methods such as BPR-MF, if not more. However, FM-Pair shines with its ability to easily exploit additional features without any effort to adapt the underlying model. For the two task of context-aware and cross-domain CF we show that FM-Pair is effective on exploiting such features. The model can be trained without much of overhead on training time while considerable improvement can be achieved by exploiting additional features. Comparison of FM-Pair with GPFM, which is also capable of exploiting context features, exhibits superiority of FM-Pair in terms of accuracy and complexity.

In this chapter we also observed that the trivial implicit-to-explicit mapping is not an effective way of using FMs for learning-to-rank from datasets with implicit feedback. In fact, the standard FMs are not optimized for ranking and even for datasets with explicit feedback, standard FMs are not effective for ranking problems.

We also analyzed the convergence and complexity of FM-Pair in the tested datasets. An interesting observation was the ability of FM-Pair to leverage item biases that turns to be very effective for the Frappe dataset, which is a popularity-skewed dataset. We also empirically show that FM-Pair scales linearly on dimensionality of factorization and number of features.

As a future work, the proposed methods for context-aware and cross-domain CF can be further investigated by studying the effect of selected features on the performance of the recommendations. For example, for the task of cross-domain CF, as we mentioned earlier, the features from source domain can be transferred with several possibilities. In this work the features correspond to items in the source domains. The number and characteristics of the selected items in source domains is subject to further studies. Similarly, for the task of context-aware recommendation, the contribution of different context features can be adjusted by feature engineering.

In this chapter we adapted a pairwise optimization technique for Factorization Machines. Further studies can be done to apply other optimization techniques such as listwise learning-to-rank methods for Factorization Machines.

4

Top-N Recommendation with Multi-Channel Positive Feedback

User interactions can be considered to constitute different feedback channels, e.g., view, click, like or follow, that provide implicit information on users' preferences. Each implicit feedback channel typically carries a unary, positive-only signal that can be exploited by collaborative filtering models to generate lists of personalized recommendations. This chapter investigates how a learning-to-rank recommender system can best take advantage of implicit feedback signals from multiple channels. We focus on Factorization Machines (FM) with Bayesian Personalized Ranking (BPR), a pairwise learning-to-rank method, that allows us to experiment with different forms of exploitation. We perform extensive experiments on three datasets with multiple types of feedback to arrive at a series of insights. We compare conventional, direct integration of feedback types with our proposed method, which exploits multiple feedback channels during the *sampling* process of training. We refer to our method as *multi-channel* sampling. Our results show that multi-channel sampling outperforms conventional integration, and that sampling with the relative "level" of feedback, is always superior to a level-blind sampling approach. We evaluate our method experimentally on three datasets in different domains and observe that with our multi-channel sampler the accuracy of recommendations can be improved considerably compared to the state-of-the-art models. Further experiments reveal that the appropriate sampling method depends on particular properties of datasets such as popularity-skewness[1].

[1] This work was first published as Loni, Babak, Roberto Pagano, Martha Larson, and Alan Hanjalic. "Bayesian personalized ranking with multi-channel user feedback." In *Proceedings of the 10th ACM Conference on Recommender Systems,* pp. 361-364. ACM, 2016. This chapter is extension of the above work and is accepted to be published as Loni, Babak, Roberto Pagano, Martha Larson, and Alan Hanjalic. "Top-N Recommendation with Multi-Channel Positive Feedback using Factorization Machines" In *ACM Transactions on Information Systems (TOIS)*. 2018 [76].

4.1. INTRODUCTION

Recommender systems exploit user interactions in order to generate predictions of user preference. Today's recommenders are not dependent on users' explicit ratings, but rather are able to exploit implicit feedback. The number of types of implicit feedback, here referred to as *channels*, that can be collected from users is large, and arguably growing. These feedback channels often provide positive-only feedback, meaning that they capture preference, but not any dislike or negative feedback. They include views, clicks or likes but also replies, bookmarks, follows, saves, shares, purchases, flags, and mentions. The potential posed by these channels raises the question of whether additional forms of feedback add extra value, and, if so, how this value can best be exploited to improve recommender systems.

In this chapter, we focus on Factorization Machines (FMs) [90], a state-of-the-art factorization method. We choose FMs because they are well suited for the integration of auxiliary data in terms of additional features [78, 99], and, for this reason, allow us to compare different methods of integrating multi-channel feedback. We point out that many context-aware recommendation approaches, since they provide a mechanism to integrate auxiliary data, could possibly be used to exploit multi-channel feedback. However, FMs represent the state-of-the-art of context-aware recommendation [99] and have been shown to outperform their closest competitor, which is Multiverse Recommendation [50]. The newer extended variations of FMs [48, 80, 126] have also been shown to be effective for context-aware collaborative filtering. For these reasons, our experiments focus exclusively on the FM framework.

Standard FMs are optimized for datasets with explicit feedback. In this work, we use an adaptation of FMs [71] with Bayesian Personalized Ranking (BPR) [98], a *pairwise* optimization method that allows us to learn preferences from unary (positive-only) feedback. Note that positive-only feedback is also referred as implicit feedback in some related work as this kind of user feedback is typically provided implicitly (e.g. user click, share, add-to-basket, etc.). We refer to our method as FM-Pair as it uses a pairwise optimization model for FMs. FM-Pair can address scenarios where user feedback is unary and positive-only, but it can also be applied on datasets with explicit feedback since explicit feedback can be always converted to positive-only by thresholding the relevance [51, 121]. Furthermore, FM-Pair can exploit rating information for more effective optimization. In this work, we also perform experiments on a dataset with explicit feedback to show the effectiveness of our method in different scenarios.

In BPR, the learning is done by forming pairs of items consisting of an item with positive feedback and a negative item, which is typically sampled from the set of items that user has not interacted with. BPR trains a model by optimizing a pairwise loss function with Stochastic Gradient Descent (SGD). It has been shown that the performance and convergence of the BPR optimization model is largely dependent on how well the pairs of positive and negative items are sampled [96, 98]. Sampling the “right" items results in faster convergence of the BPR model and makes a contribution to the performance [96].

Our work builds on an initial, short exploratory paper [74], in which we demonstrate the usefulness of differentiating between channels in conventional BPR. We consider channels to constitute different *levels* of feedback, with higher levels reflecting a higher user commitment, and thereby carrying a stronger preference signal, than lower levels.

Briefly, the benefit of leveraging channels in BPR is the advantage offered by a better-informed sampling of positive and negative items.

In this chapter, we move multi-channel feedback to a FM framework to make a comparison between exploiting multiple feedback channels directly (as auxiliary features in the FM model) or via sampling. Building on the ability of FMs to exploit additional information as auxiliary features, we first study how effective it would be if we encode channel types as features in the FM model. Our second approach, which extends our preliminary work [74], takes into account the feedback channels for more effective sampling. In this chapter, we propose a wider range of samplers, introduce an alternative way to sample positive items from a dataset, and use FMs as the scoring function to predict the utility of user-item interactions. We compare the effectiveness of our first and second approach with several baselines on three different datasets with implicit or explicit feedback. Note that multiple types of positive feedback can also be exploited by other methods such as ensemble methods and sequential training [120]. Our proposed method is one possibility to exploit multiple types of feedback by the underlying learning algorithm. This chapter addresses the following research questions (RQs):

- RQ0: As a sanity check, does a recommender that combines multiple channels of feedback improve its performance over a recommender that uses a single channel only?
- RQ1: Does channel-based, level-informed sampling allow for better exploitation of multiple feedback channels than conventional integration of feedback?
- RQ2: How do different sampling strategies perform on different datasets?
- RQ3: What are the advantages of channel-based, level-informed sampling (i.e., performance, coverage, time)?

The main contributions of this chapter are a thorough understanding of the potential and limitations of multi-channel feedback exploitation, supported by extensive experimentation, reproducible because of our choice to include public datasets, and our release of the code for this chapter.

The organization of the remainder of this chapter is as follows. In Section 4.2, we review related work. Section 4.3, provides a background on FMs, Bayesian Personalized Ranking and pairwise Factorization Machines (FM-Pair). In Section 4.4, we introduce the two different approaches to exploit multiple channels with FM-Pair. Section 4.5 describes datasets, our evaluation framework and our experimental setup. In Section 4.6.1, we present the results of the three sets of experiments that we performed in this study, each of which is connected to one of our research questions. Finally, we draw a conclusion and discuss future directions in Section 4.7.

4.2. RELATED WORK

In this section, we briefly review four groups of studies that are related to our work. We first review the state-of-the-art methods based on FMs. We then discuss the studies that propose to use FMs for learning-to-rank. Next, we report the third group of related work that tries to exploit multiple types of user feedback and finally we review the studies that have been done to improve the sampling method of BPR.

Factorization Machines [90] have been mainly used for collaborative filtering and in particular for rating prediction. Some studies try to improve the underlying model of FMs. Field-Aware Factorization Machines (FFM) [48] is an extension of FMs where several representations for a feature are learned, depending on the *fields* (groups of features) of the pairwise interactions in the model. For example, in a model with three groups of features, which corresponds to users, items and context, two representations for a user are learned, one for interacting with items and a second representation for interacting with context. FFM is shown to improve the accuracy of predictions in Click-Through-Rate (CTR) prediction tasks. FFM however, has higher time and space complexity compared to the standard FM model and it is not optimized for ranking problems. In recent work, Xiao et al. [126] propose Attentional Factorization Machines (AFM) where the FM model is represented as a neural network model with additional parameters that are learned using an attention network. The extra parameters learn additional weights for pairwise interactions that can reflect the importance of interactions. This model is also trained with a point-wise loss function, similar to FMs, and is optimized for prediction tasks. Due to the larger number of parameters, this model also has a larger time and space complexity.

Our work shares the goal of using FMs for learning-to-rank with a few recent studies. Nguyen et al. [80] proposed Gaussian Process FMs (GPFM) where interactions between feature vectors are modeled with Gaussian kernels. They also proposed a pairwise ranking model where the GPFM model can be used for datasets with implicit feedback. Another study that uses FMs for ranking is RankFM, work by Qiang et al. [87], where the model uses ordinal regression with an FM kernel for micro-blog ranking. This model is particularly useful when we have explicit negative feedback. Guo et al. [35] proposed a learning-to-rank model for FMs where the categories of items are used for sampling. The effectiveness of their method however, has been proved only on the dataset of MovieLens 100K where the user feedback is explicit. All the above work assumes that user feedback (explicit or implicit) is from a single channel. Our work differs from these contributions in that our model distinguishes different feedback channels and exploits channel information either as auxiliary features, or for more effective sampling.

The idea of using multiple types of feedback has been studied in some previous work. Zanker et al. [128], proposed a neighborhood collaborative filtering method where users' feedback from different channels is combined to calculate similarity of users and items more precisely. They define weights for different channels by measuring each channels' performance when each channel is used individually. This work is based on neighborhood methods, whereas our work is based on low-rank factorization, a model-based approach. While neighborhood methods are simple to implement, they are computationally more expensive and generally less accurate than model-based methods [60, 112]. In this work, we use FMs as a state-of-the-art collaborative filtering method and we do not explore other collaborative filtering approaches. In a recent study, Gai et al. [65] proposed an early fusion method where both implicit and explicit feedback are converted to implicit feedback and the model is trained by an adaptation of the SVD++ algorithm [53]. This method converts explicit ratings to binary relevance data and does differentiate different levels of feedback. However, it has only been evaluated on explicit feedback datasets. In other work, Pan et al. [83] used an extension of BPR, namely BPR with confidence [124], for learning from *heterogeneous* implicit feedback. Their method exploits heterogeneous feedback by modeling feedback confidence. This model can distinguish only two types of feedback namely *transaction* (high confidence) and *examination* (low confidence) feedback. This model was only evaluated on explicit movie ratings where ratings of 5 are considered as transactions and the other rating are considered as examination feedback. However, the effectiveness of this method on real implicit feedback datasets where the confidence of the interactions is not known is not clear.

Combining multiple types of feedback has also been studied using late fusion methods. Da Costa et al. [20] propose an ensemble method to combine models that are learned from individual feedback channels. Tang et al. [120] performed an empirical study on exploiting different feedback channels in LinkedIn to improve recommendations. They proposed to use three methods, late fusion, sequential training and joint training, to exploit different feedback types. The underlying model that is used for training is logistic regression, which is optimized for CTR prediction. To compare our proposed model with late fusion methods, we include an oracle experiment that represents an upper bound on the performance that is achievable with late fusion.

Our proposal is related to other work that tries to improve BPR sampling methods. Ganter et al. [28] proposed a non-uniform sampler for negative items where the probability of sampling negative items is given in advance. Rendle et al. [96] proposed a more advanced sampling method where the probability of sampling negative items is adapted dynamically (Dynamic Oversampling). Lerche et al. [61] proposed BPR with graded relevance. In their method, sampling is partially done using graded feedback. That is, a given ratio of item pairs are sampled from the *observed* interactions using a weighting function. However, calculating the grades of feedback requires additional information such as user ratings or frequency of interactions, which might not be available in a dataset. Furthermore, in our earlier work [75] we empirically showed that sampling a negative item from observed feedback always results in degraded performance presumably because unobserved items are more likely to be negative (and thus better candidates for sampling as negative) compared to observed items [117]. Our method differs from [61] in the sense that it uses multiple channels simultaneously and does not rely on real-valued grades for implicit feedback. Furthermore, we adopt FMs (which are capable of exploiting context or other auxiliary features) as the utility (scoring) function, whereas [61] relies on the standard BPR model with Matrix Factorization utility function.

4.3. Background and Framework

In this section, we provide a brief overview of FMs and Bayesian Personalized Ranking (BPR) to be able to explain FM-Pair, the model that we use in this work.

4.3.1. Factorization Machines (FMs)

Factorization Machines are general factorization models that are capable of modeling different factorization approaches by way of feature engineering [92]. In addition to their superiority with respect to performance and accuracy, FMs offer an advantage because they allow easy integration of additional information in the form of auxiliary features. FMs have been successfully applied in context-aware recommendation [99], cross-domain collaborative filtering [78] and social recommendations [70].

The standard FMs are designed for data with explicit feedback. Each user-item interaction is modeled by a feature vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and the corresponding feedback (rating) is specified by y, a real-value number. The user-item interaction vector $\mathbf{x}$ has two non-zero elements: one corresponding to the specific user and one to the specific item. That is, if user u rates item i, the feature vector $\mathbf{x}$ is specified as:

$$\mathbf{x}_{u,i} = (\underbrace{0, \ldots, 0, x_u, 0, \ldots, 0}_{|U|}, \underbrace{0, \ldots, 0, x_i, 0, \ldots, 0}_{|I|}) \tag{4.1}$$

where U and I represent the set of users and items in the dataset and the cardinalities of them are denoted by $|U|$ and $|I|$. In the above equation, x_u and x_i are the values of the features corresponding to user u and item i. The above feature vector $\mathbf{x}$ can also be represented compactly as: $\mathbf{x}(u,i) = \mathbf{x}_{u,i} = \{(u, x_u), (i, x_i)\}$ where u and i are indices and x_u and x_i are the values of the features. The feature values x_u and x_i are typically considered to be 1 to indicate the involvement of a user and an item in one interaction.

One advantage of FMs, which we build on here, is their ability to exploit additional information (such as user attributes, item attributes and contexts of interactions) in terms of *auxiliary* features. Assuming that a user-item interaction also includes auxiliary features $\mathbf{z}$, the expanded feature vectors $\mathbf{x}$ can be represented with the following compact form:

$$\mathbf{x}(u,i,\mathbf{z}) = \mathbf{x}_{u,i,\mathbf{z}} = \{(u, x_u), (i, x_i), \{(z, x_z) | z \in \mathbf{z}\}\} \tag{4.2}$$

where $z \in \mathbf{z}$ is the index of an auxiliary feature and x_z is the value of that feature. An auxiliary feature z can be binary, categorical or numerical, depending on the type of feature.

Given the feature vectors $\mathbf{x}$ with general representation of $\mathbf{x} = (x_1, \ldots, x_n)$, the standard FMs model with *two-way* interactions is specified as:

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f} \tag{4.3}$$

where n is the number of features, w_j are first order interaction parameters, $v_{*,f}$ are second order factorized interaction parameters and k is the dimensionality of factorization.

The model parameters in FMs according to (4.3) are $\Theta = \{w_0, w_1, \ldots, w_n, v_{1,1}, \ldots, v_{n,k}\}$. Model parameters are learned by optimizing a *point-wise* objective function, which is defined over training data S. The objective function is typically a square loss over training samples plus a regularization term to prevent over-fitting.

In recommendation scenarios involving unary positive feedback (instead of e.g., ratings), the target values y are not available in terms of real value numbers and a point-wise loss function is not suitable. *Pairwise* models on the other hand, learn to correctly *rank* any pair of items for each user. Previous work has reported that pairwise loss functions are better choices for datasets with unary feedback [80, 98, 112]. In this work, we use a pair-wise loss function for FMs based on the BPR model. Before describing such a loss function, we give an overview of BPR in the next subsection.

4.3.2. Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) [98] is a state-of-the-art learning-to-rank method for learning from data with unary (positive-only) feedback. The basic idea of BPR is that for any user u, the items with observed positive feedback from u are preferred over items without any feedback from u. With that assumption, BPR creates training data by forming tuples (u, i, j), where i is an item with positive feedback from u and j is a negative item.

BPR learns the model parameters Θ by maximizing the likelihood function $p(i \succ_u j|\Theta)$ for all training triples ($i \succ_u j$ is read as i is preferred over j by user u). The likelihood function $p(i \succ_u j|\Theta)$ is defined as:

$$p(i \succ_u j|\Theta) = \sigma(\hat{y}_{uij}(\Theta)) \tag{4.4}$$

where σ is the logistic sigmoid function and $\hat{y}_{uij}$ is a real-valued utility function that captures the relationship between user u, item i and item j, given model parameters Θ. In standard BPR, $\hat{y}_{uij}$ is defined as the subtraction of individual user-item utility function:

$$\hat{y}_{uij}(\Theta) = \hat{y}_{ui}(\Theta) - \hat{y}_{uj}(\Theta) \tag{4.5}$$

The utility of user-item pairs $\hat{y}_{ui}(\Theta)$ can be defined by different models such as Matrix Factorization or Nearest Neighbor models [98]. With a simple Matrix Factorization model the utility of user-item pairs $\hat{y}_{ui}$ can be defined as:

$$\hat{y}_{ui}(\Theta) = \sum_{f=1}^{k} v_{u,f} v_{i,f} \tag{4.6}$$

where $v_{u,f}$ and $v_{i,f}$ are the latent factors corresponding to user u and item i and k is the dimensionality of factorization. Similarly, we can define $\hat{y}_{uj}(\Theta)$, the utility of the negative item for the user. Thus, $\hat{y}_{uij}(\Theta)$ can be defined as:

$$\hat{y}_{uij}(\Theta) = \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}) \tag{4.7}$$

Note that with the above utility function, the model parameters Θ are the set of all user and item latent factors. With the above definition, the likelihood function $p(i \succ_u j|\Theta)$ can be represented in terms of parameters $v_{u,f}$ and $v_{i,f}$. The parameters are learned by maximizing $p(i \succ_u j|\Theta)$ for the set of tuples (u, i, j). Rendle et al. [98] showed that the maximum likelihood problem in BPR can be reduced to an optimization problem with the following objective function:

$$L(\Theta, S) = \sum_{(u,j,j)\in D_S} \ln \sigma(\hat{y}_{uij}) - \lambda_\Theta ||\Theta^2|| \tag{4.8}$$

where λ_Θ are the regularization parameters and D_S is the set of tuples (u, i, j). The BPR optimization problem is solved by Stochastic Gradient Descent.

Due to the large number of possibilities to create the tuples (u, i, j), the tuples are typically formed by *sampling* the dataset. The choice of sampling method in BPR is crucial to convergence of SGD algorithm. In [98], it has been suggested that the user-item pairs (u, i) are sampled from S uniformly (with replacement)[2] and the negative items j are uniformly drawn from $I \backslash I_u^+$ where I_u^+ is the set of items that received positive feedback from u. However, in a later study, Rendle et al. [96] proposed a *dynamic* non-uniform sampling method for negative items where the probability of sampling a negative item is dependent on the current model parameters. They showed that for a dataset with tailed item popularity, dynamic sampling has a significant influence on the convergence of SGD.

4.3.3. Pairwise Factorization Machines

Using the BPR optimization criteria, which was described earlier, we can adapt FMs for data with unary feedback. A straightforward approach would be to consider the FM model in Eq. (4.3) as the utility (scoring function) of individual user-item pairs, i.e., $\hat{y}_{ui}$. However, FMs do not have any knowledge about the *type* of feature, i.e., for a given feature vector, FMs do not know which features correspond to user, item or auxiliary information. On the other hand, in the BPR algorithm, for a given user-item pair the user should be known to the algorithm in order to sample an appropriate negative item (recall that negative items are sampled from $I \backslash I_u^+$). We therefore introduce a finer-grained representation of FMs where the algorithm is *aware* of the features (i.e., the input data should specify which feature corresponds to a user[3]).

Let us assume that $\mathbf{x}_{u,i,\mathbf{z}}$ is a feature-aware representation of an input feature vector in FMs such that u corresponds to the user, i corresponds to the item and $\mathbf{z}$ is the set of auxiliary features in $\mathbf{x}$. By replacing the sparse representation $\mathbf{x}_{u,i,\mathbf{z}}$ (Eq. (4.2)) in Eq. (4.3), the utility of individual interactions $\mathbf{x}_{u,i,\mathbf{z}}$ can be written as:

$$\begin{aligned} f(\mathbf{x}_{u,i,\mathbf{z}}) = w_0 + w_u + w_i + \sum_{z \in \mathbf{z}} w_z x_z + \sum_{f=1}^{k} v_{u,f} v_{i,f} + \sum_{z \in \mathbf{z}} x_z \sum_{f=1}^{k} v_{u,f} v_{z,f} \\ + \sum_{z \in \mathbf{z}} x_z \sum_{f=1}^{k} v_{i,f} v_{z,f} + \sum_{z \in \mathbf{z}} \sum_{z' \in \mathbf{z}} x_z x_{z'} \sum_{f=1}^{k} v_{z,f} v_{z's,f} \end{aligned} \tag{4.9}$$

Note that since the only non-zero features in $\mathbf{x}_{u,i,\mathbf{z}}$ are u, i, and $\mathbf{z}$, we were able to expand (4.3) and represent the FM model with finer-grained components. Conceptually, the above equation can be interpreted as a set of bias terms (parameters w_*) and interaction between latent factors of user and item, user and auxiliary features, item and auxiliary features, and interaction between different auxiliary features.

In order to exploit the auxiliary feature $\mathbf{z}$ in our pairwise FM model, we adapt the definition of likelihood and loss functions in BPR to take into account the auxiliary features. Hence, we define a more general likelihood function for BPR as:

$$p(i >_{u,\mathbf{z}} j|\Theta) = \sigma(g_{\mathbf{z}}(u, i, j|\Theta)) \tag{4.10}$$

[2] Sampling with replacement means that the sampled training pair (u, i) is replaced after sampling and can be sampled again.

[3] In the implementation of this work, we support a file format for FM-Pair where the type of feature can be specified.

where $i >_{u,\mathbf{z}} j$ indicates that item i is preferred over j by user u in the presence of auxiliary features $\mathbf{z}$. Here $g_{\mathbf{z}}(u,i,j|\Theta)$ is the utility of pairs of feature vectors in FMs. Similar to Eq. (4.5), $g_{\mathbf{z}}$ can be defined by subtracting the utility function $f(\mathbf{x}_{u,j,\mathbf{z}})$ from $f(\mathbf{x}_{u,i,\mathbf{z}})$. Thus, by using the definition (4.9), $g_{\mathbf{z}}$ can be written as:

$$\begin{aligned} g_{\mathbf{z}}(u,i,j|\Theta) = w_i - w_j + \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}) \\ + \sum_{z\in\mathbf{z}} x_z \sum_{f=1}^{k} v_{z,f}(v_{i,f} - v_{j,f}) \end{aligned} \tag{4.11}$$

Note that the above utility function is very similar to the Matrix Factorization (MF) utility function (Eq. (4.7)). However, it has additional parameters that take into account user and item biases and auxiliary features to score a user-item interaction. From Eq. (4.11) and Eq. (4.9) one can see that the MF utility function $\hat{y}_{ui}$ is a special case of g where the biases parameters are zero and the feature vector $\mathbf{x}_{u,i,\mathbf{z}}$ does not have any auxiliary features.

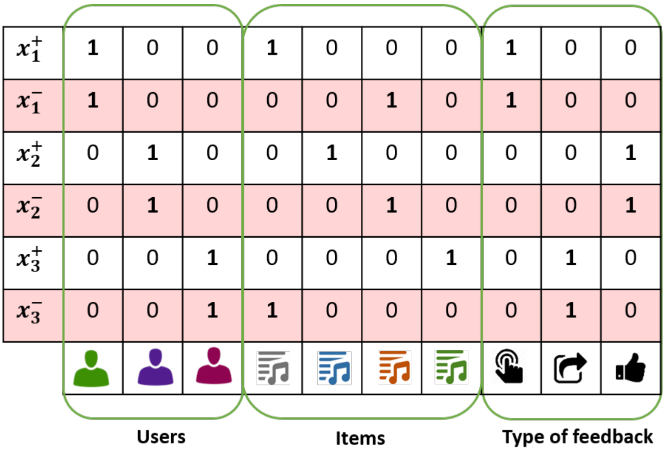
Finally, the objective function $L(\Theta,S)$ can be adapted by replacing $\hat{y}_{uij}$ with $g_{\mathbf{z}}$ in equation (4.8). The optimal parameters are found by minimizing this function, i.e., $\Theta_{OPT} = \text{argmin}_{\Theta} L(\Theta,S)$. This optimization problem can be solved by Stochastic Gradient Descent. Note that the negative items j are sampled from the dataset similarly to how they are sampled in the standard BPR model.

4.4. MULTIPLE CHANNELS IN FM-PAIR

In this section, we introduce two different models to exploit channel information using the FM-Pair model. We first introduce a naïve approach (Section 4.4.1), where the channel information is embedded into the FM-Pair model as auxiliary features. We then propose a more advanced model (Section 4.4.2) where the channel information is used to perform a more effective sampling (multi-channel) for the FM-Pair optimization algorithm.

4.4.1. MULTIPLE CHANNELS AS AUXILLIARY FEATURES

The most straightforward way to exploit multiple types of positive feedback in FM-Pair is to use the types (channels) of the feedback as auxiliary features in the FM model. The basic assumption here is that the type (channel) of feedback contains some information that reflects the commitment level or preference of the user for the item. With the FM-Pair model, the channel of feedback is considered as an additional discrete feature. Figure 4.1 illustrates the design matrix of the FM-Pair model with three positives (white background) and three negative (red background) samples. In this example, items are playlists and the channel types of feedback are ‘click’, ‘share’ and ‘like’. The three types of feedback are considered as discrete features. For each positive example, an artificial negative example is created in such a way that the user and auxiliary features (in this case ‘channel type’) are the same as the positive example and the negative item is a sampled item from $I \backslash I_u^+$ (note that here $\mathbf{x}_1^+$ is an example of $\mathbf{x}_{u,i,\mathbf{z}}$ and $\mathbf{x}_1^-$ is an example $\mathbf{x}_{u,j,\mathbf{z}}$ according to the FM-Pair model). Note here the difference between the design matrix of



	Users			Items				Type of feedback		
x_1^+	1	0	0	1	0	0	0	1	0	0
x_1^-	1	0	0	0	0	1	0	1	0	0
x_2^+	0	1	0	0	1	0	0	0	0	1
x_2^-	0	1	0	0	0	1	0	0	0	1
x_3^+	0	0	1	0	0	0	1	0	1	0
x_3^-	0	0	1	1	0	0	0	0	1	0

Figure 4.1: Embedding feedback channels in Pairwise Factorization Machines (FM-Pair). This matrix shows both positive and sampled negative points. The types of feedback are encoded as binary features.

the FM-Pair model with the design matrix of the standard FM [92]. In FM-Pair, there is no column for labels since we do not have explicit graded feedback. Moreover, since the optimization model in FM-Pair uses positive and negative samples, we have two different types of rows representing positive and negative examples.

The advantage of such representation is that channel types are encoded as additional features in the input data and the FM-Pair model can be used seamlessly. The model learns the underlying latent factors for each of the feedback channels, which is then used for prediction. Note that since the type of feedback is only present on training time (not on the prediction time), the feedback type is only exploited on the training time to better learn user and item factors. The recommendations are generated on a user-basis, that is, for each user a ranked list of items are generated where the candidate items are ranked using Eq. (4.9). It is also worth noting that if other auxiliary or context features are present in a dataset that are available in the prediction time, the ranked list should be created for each combination of user and available context [91].

4.4.2. Multi-Channel Sampling

This section presents our approach, which improves FM-Pair by using level-based sampling exploiting multiple feedback channels. In [74], we introduced the idea of multi-channel sampling for BPR. Here, we provide a fully developed version of the initial idea by integrating multi-channel sampling into FM, decoupling the sampling model for positive and negative items, and carrying out extensive testing.

Assume that $p(u,i,j)$ is the probability distribution from which tuples (u,i,j) are sampled. This distribution can be expanded as:

$$p(u,i,j) = \underbrace{p(u,i)}_{\substack{\text{positive pair}\\\text{sampler}}} \;.\; \underbrace{p(j|u,i)}_{\substack{\text{negative item}\\\text{sampler}}} \tag{4.12}$$

The two components in the above probability distribution correspond to the *positive pair sampler*, by which user-item pair (u,i) is sampled, and the *negative item* sampler, by which j is sampled.

In standard BPR, (u,i) is selected by uniform sampling from the training set[4] S, and j is sampled uniformly from $I \backslash I_u^+$. Thus, the probability distribution of the two samplers in the standard BPR can be denoted by $\frac{1}{|S|}$ for positive pairs and $\frac{1}{|I \backslash I_u^+|}$ for negative items.

For multi-channel positive feedback, we introduce the notion of a feedback *level* that reflects the importance of a feedback channel. The notion of level is also used to define an ordinal scale for different feedback channels. The higher the level of a feedback is, the higher is the user's interest or commitment level to the item. For example, in a music recommendation framework, a user 'like' should have higher level than a 'click', assuming a 'like' is a stronger indication of interest than a 'click'. The assumption of multi-channel sampling is that, given a user, an item at a higher feedback level is preferred to an item at a lower feedback level.

Multi-channel sampling differs from the standard BPR sampling method in two ways: Firstly, the probability of sampling positive pairs and negative items depends on the feedback level. Secondly, the multi-channel method samples additional tuples that would not otherwise be sampled by standard BPR sampling. We point out that the positive and negative items in tuples (u,i,j) can also be interpreted as a 'preferred' and a 'less preferred' item. In our preliminary work [74], we showed that sampling negative items *only* from unobserved interactions is more effective than the negative items that are sampled from observed interactions. Therefore, in this work we propose to sample negative items only from unobserved interactions. Note that when we sample negative items from unobserved interactions, the feedback level still influences sampling probability, since the types of interactions of other users make a contribution, as explained below. In sum, Figure 4.2 illustrates standard versus multi-channel sampling for the FM-Pair model. The arrows show the preferences. For user u the negative item is always drawn from the unobserved items. It is sampled from the overall item pool in a way that is aware of the level of the feedback of other users.

Our multi-channel sampler utilizes feedback channels to over- or under-sample certain items to better learn the user's preference model. Thus, the probability of sampling positive pairs and negative items also depends on the level where they appear.

We denote $\mathbb{L} = (L_1, \ldots, L_p)$ as the ordered set of levels in a dataset such that $L_i \succ L_{i+1}$, that is, feedback in L_i are stronger signals of interest compared to the feedback in L_{i+1}. For example, in Figure 4.2 there are four levels: three positive and one unobserved level. Let L and N denote the level of the positive and the negative item in tuple (u,i,j). The multi-feedback sampler samples tuples (u,i,j) from the following set:

$$D_{MF} = \{(u,i,j) | i \in I_{L,u} \wedge j \in I_{N,u} \wedge L \succ N\} \tag{4.13}$$

such that $I_{L,u}$ represents the set of items in levels L that user u interacted with and $I_{N,u}$ represents the set of items that can be sampled in level N as negative item, which we take to be the set of items with which the user has not interacted.

[4] Note that the user-item pairs can also be selected by iterating over training data, but it has been shown that sampling with replacement (bootstrapping) is more effective than iteration [98].

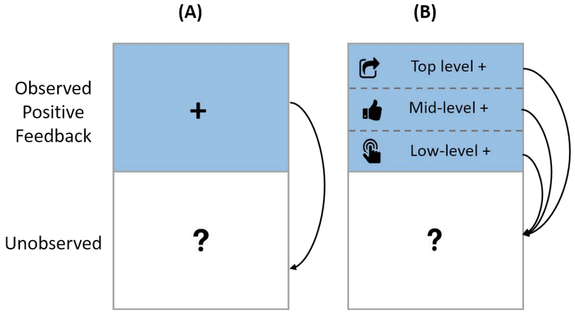


Figure 4.2: Diagram for a single user profile illustrating the difference between standard sampling (A) and multi-channel sampling that is used by the FM-Pair model (B). The arrows indicate the preference relation between positive and negative items. The relative ordering between the different levels of feedback contributes a weight used for sampling positive and negative items in multi-channel sampling.

As already stated, in the multi-channel feedback sampling method the probability of sampling the tuples depends on the levels. We can assume that the tuples are sampled from a joint probability distribution $p(u,i,j,L,N)$, which can be expanded as:

$$p(u,i,j,L,N) = p(u,i,L).p(j,N|u,i,L) \tag{4.14}$$

In the above equation, the positive pair sampler and the negative item sampler are both joint probability distributions with positive or negative levels. We now define different distributions for positive and negative samplers.

Positive Pair Sampler

In multi-channel sampling, the positive feedback sampler can be further expanded as:

$$p(u,i,L) = p(u,i|L)p(L) \tag{4.15}$$

where $p(L)$ is the probability of sampling a positive level. We define $p(L)$ as follows:

$$p(L) = \frac{w_L|S_L|}{\sum_{Q\in\mathbb{L}^+} w_Q|S_Q|} \tag{4.16}$$

where $|S_L|$ is the number of occurrences of feedback in level L and w_L is a weight associated with a level that should reflect the importance of the level. The weight parameters w_L can be either static values that are dependent to the channel types or can be dynamic values that are changed during training (similar to the dynamic sampling method proposed in [96]). In our preliminary experiments, we found that dynamic weights for sampling levels typically results in sub-optimal models. We found that the reciprocal rank of the levels are good candidates as the weights of the levels. Given a level, the positive user-item pair is then sampled uniformly from that level. To summarize, the positive pair sampler first samples a level according to (4.16) and then uniformly samples a user-item pair from that level. With the above sampling method, the positive feedback from higher levels has a higher chance to be sampled. This chance is determined by the size and also the weight of the levels.

NEGATIVE ITEM SAMPLER

The second factor in Eq. (4.14) is the negative item sampler that, similarly to Eq. (4.15), can be expanded to finer-grain distributions:

$$p(j,N|u,i,L) = p(j,N|u,L) = p(j|u,N)p(N|u,L) \tag{4.17}$$

where the first probability distribution is used to sample a negative item given a negative level, and the second distribution is used to sample a negative level given the user and the positive level. Note that sampling an item only depends on the user and the level that the item belongs to. Therefore, we can write $p(j,N|u,i,L) = p(j,N|u,L)$. Similarly, we can set $p(j|u,L,N) = p(j|u,N)$, since given a negative level N, the negative item j is sampled from level N and does not depend on L anymore.

In our preliminary work [74], we showed that using *only* the unobserved level as negative level typically results to a better model. We therefore assume that N is always the unobserved level and thus we can simplify the negative sampler as $p(j|u)$. Note that the standard sampling method of BPR also samples the negative item from unobserved items but it does not consider the level of feedback during sampling.

To have a complete picture on possible sampling methods, we define three different distributions for the negative item sampler where the first one is based on the BPR sampling method [98], the second sampler over-samples popular items, and the third sampler is our proposed multi-channel sampler. In the following, we describe each in turn and describe the underlying intuition.

Uniform Item: Here the negative item is sampled uniformly from the negative level. By considering the unobserved level as the negative level, $p(j|u)$ can be defined in a similar way as it is defined when sampling a negative item in standard BPR:

$$p(j|u) = \begin{cases} \frac{1}{|I \setminus I_u^+|} & j \in I \setminus I_u^+ \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

Uniform Feedback: With this sampling method, first a feedback pair (u', j) is sampled from the set of all positive feedback pairs of other users, and then u' is discarded and j is considered as the negative item. This is comparable to popularity over-sampling since the popular items have higher chance to be sampled. Here $p(j|u)$ can be defined as:

$$p(j|u) = \frac{|(u',j') \in S : u' \neq u \wedge j' = j|}{|S|} \tag{4.19}$$

where $|S|$ is the size of the training data. The idea behind this sampler is to take into account the popularity of items for sampling negative items. This sampling method can be effective if the training dataset has a tailed item distribution. That is, when some items are more popular than the majority of items. During training, when the SGD algorithm iterates over the training data, popular items that a user interacts with are more likely to be sampled as positive and thus the model learns to rank them higher in the list. However, if popular items that the user does not interact with also become more likely to be sampled as negative, such bias for popular items can be reduced, and the overall accuracy of the model can be increased.

Multi-channel: With this method, both the popularity of an item and the level of feedback are taken into account when sampling the negative items. In this approach, first a (u', j) pair is sampled from the joint distribution $p(u', j, L')$, similarly to Eq. (4.15). u' and L' are discarded after the triple (u', j, L') is sampled. The probability distribution of this sampler can be defined as:

$$p(j|u) = \begin{cases} p(u', j, L') & j \in I \setminus I_u^+ \wedge u' \neq u \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

The intuition behind this sampler is that popular and more important items, if not rated by the target user, are better candidates than negative items. Rendle et al. [96] showed that SGD typically converges faster if the model is less certain about the correct order of the sampled items. By considering the more *important* (higher level and more popular) items as negative, the uncertainty of BPR for finding the true rank between the pairs can increase and thus the model can converge faster.

All the above probability distributions can be pre-computed in advance and thus the sampling can be done in $\mathcal{O}(1)$. Therefore, the computational complexity of the multi-channel method is the same as the standard BPR model. Figure 2 summarizes the FM-Pair learning algorithm with adapted positive pair and negative item samplers. p_{pos} refers to the positive pair sampler, which can be according to Eq. (4.15) or can be a uniform distribution. p_{neg} is negative item sampler and can be either (4.18), (4.19), (4.20) or other possible sampling methods.

Algorithm 2: Learning FM-Pair with Stochastic Gradient Descent and adapted samplers.

```
Input: $S$, $p_{pos}$, $p_{neg}$
Output: Model parameters $\Theta$
initialize $\Theta$
do
    sample $\mathbf{x}_{u,i,\mathbf{z}}$ from $S$ according to $p_{pos}$
    sample $j$ according to $p_{neg}$ and create $\mathbf{x}_{u,j,\mathbf{z}}$
    let $g_{\mathbf{z}}(u,i,j|\Theta) = f(\mathbf{x}_{u,i,\mathbf{z}}|\Theta) - f(\mathbf{x}_{u,j,\mathbf{z}}|\Theta)$
    update $\Theta$ according to BPR update rule [98]
while convergence
return $\Theta$
```

4.5. Data and Experimental setup

In this section, we provide a description of our choice of datasets and the design of our experimental setup.

4.5.1. Datasets

We used three different datasets from different domains to evaluate our proposed methods: Kollekt (music recommendation), XING (job recommendation) and MovieLens (movie

Table 4.1: The statistics of the datasets used in this work.

Dataset	#users	#items	#feedback	%sparsity
Kollekt	15,972	34,910	168k	99.96
ML1M	6,040	3,706	1000k	95.53
Xing	9,751	9,821	223k	99.76

recommendation). The first two datasets contain unary feedback with different channels whereas the MovieLens dataset contains explicit user ratings. The three datasets are described below and their statistics can be found in Table 4.1.

- **XING (RecSys challenge 2016)**: XING is a professional social network and a job discovery platform. The XING dataset has been used for the RecSys challenge 2016[5]. Items are job postings and user feedback is collected through three different channels: `click`, `bookmark` and `apply`. In order to reduce data sparsity and the size of dataset, we filtered out all the users and jobs that overall have less than 20 occurrences of feedback in all channels. We choose this dataset because it contains multiple feedback channels, and it also has been made available to the research community.

- **Kollekt**: Kollekt[6] is an online music discovery platform. Users can discover new playlists in the discovery page[7] and can follow them. Following a playlist is considered as a positive feedback on the playlist. Users can also listen to playlist or certain tracks in a playlist without necessarily following it. The ratio of the listening time of a playlist to the total listening time of the user is also considered as user feedback. The listening ratios that are higher than a certain threshold are considered as a positive feedback. We experimentally found that with a threshold of 0.1, the listening time can be considered as positive feedback since it can train accurate models. The dataset consists of user interactions from July 2013 to October 2015. This dataset is not publicly available, but it is useful for our experiments as it involves multiple feedback channels, and has interesting distributional patterns (discussed further below).

- **MovieLens-1M** This is a popular movie recommendation dataset[8] with explicit user feedback containing movie ratings on a 1-5 scale. To apply the multi-channel sampling method to this dataset, we considered each discrete rating as a different level of feedback. For each user, we took the ratings above the user's average rating as positive feedback, both for training and evaluation. We choose this dataset due to its availability and widespread use in the community.

[5] `http://2016.recsyschallenge.com`

[6] `http://kollekt.fm/`

[7] `https://kollekt.fm/discover/playlists`

[8] `http://grouplens.org/datasets/movielens/`

4.5.2. Evaluation Method

The performance of recommendation in all datasets is measured with three ranking metrics of Recall, MRR (Mean Reciprocal Rank) and NDCG (Normalized Discounted Cumulative Gain). Recall@K calculates the ratio of relevant top items (in top K items) to all relevant items, MRR@K measures the average reciprocal rank of the *first* relevant item if it appears in top K positions (otherwise MRR@K is zero), and NDCG@K takes into account the position of all relevant items on top K items of the list [45]. The above three metrics are measured with two cutoffs (K) of 10 and 20. The metrics are calculated per user in the test set and the reported numbers are averages over all users.

All the experiments have been done with four-fold cross-validation. Details about the implementation and the hyper-parameters are described in the next sub-section. The ground truth varies in different datasets depending on the goal and business value of recommendations. In the Kollekt dataset, a playlist is considered relevant for a user only if the user has followed it. On the other hand, for the XING dataset any feedback on a job posting makes it a relevant item according to XING business requirements. For the MovieLens dataset, a movie is considered a relevant recommendation if user has rated it above his average ratings.

4.5.3. Experimental Reproducibility

The experiments are evaluated using WrapRec[9], an open source evaluation framework for recommender systems. The algorithms in this work are implemented by extending the BPR implementation of MyMediaLite[10], a toolkit for recommender system algorithms. The number of latent factors (parameter k) is set to 10 as our preliminary experiments show that for the scale of the datasets that are used in this work, higher values of k do not necessarily improve the accuracy of recommendations. The number of SGD epochs is set to 300. Laster in this section, we study convergence of the FM-Pair algorithm based on the number of SGD epochs.

In our experiments on BPR and FM-Pair, we choose the default values from the MyMediaLite implementation for the hyper-parameters. The latent factors are initialized by a normal distribution with a mean of 0 and standard deviation of 0.1. The SGD algorithm is used with the following hyper-parameters: Learning-rate: 0.05, regularization parameter: 0.002. To check the appropriateness of these parameters for our experiments, we performed four-fold cross-validation on our data for a range of parameters surrounding the default parameters. We observed that the variation in the results for each algorithm was insubstantial, and for this reason we maintain the default parameters for the results reported in the experimental section.

The hyper-parameters of the other baseline algorithms that we used are tuned with the same four-fold cross-validation setup to make sure we are comparing with the best performance that we could achieve for each algorithm. The hyper-parameters for the baseline algorithms are listed as below:

- **AFM**: Batch size of 5000 for all datasets. Learning rate of 0.1 for the MovieLens and 0.05 for the XING and the Kollekt datasets. The dropout ratio is 0.2 for the

[9] `http://babakx.github.io/WrapRec/`
[10] `http://www.mymedialite.net/`

MovieLens and 0.3 for the Kollekt and the XING datasets. We set the attention factor to 1 as the effect of this hyper-parameter is insignificant according to the original paper [126].

- **FFM**: Learning rate of 0.001 for the MovieLens and 0.005 for the Kollekt and the XING datasets. Regularization parameter is 0.00002 for all datasets.
- **WRMF**: Regularization parameter of 0.01 for all datasets.

Furthermore, since some of the baseline algorithms (AFM, FFM and GPFM) were implemented with a different evaluation mechanism, we only used the underlying implementations to generate recommendations and later we performed the evaluation with WrapRec to make sure all experiments are evaluated with the same evaluation method. The two datasets of XING[11] (RecSys Challenge 2016) and MovieLens 1M are publicly available, making it possible to replicate the experiments.

4.6. EXPERIMENTS

In this work we perform three sets of experiments to answer our research questions. We first compare the two approaches to exploit channel information (i.e., integrating it as auxiliary features or exploiting it for sampling) to answer the first research question (RQ1). As a sanity check, we also compare the performance of our baseline model when each feedback channel is used individually. This will answer research question zero (RQ0). In the second set of experiments, we compare five different sampling strategies on all three datasets to understand the effect of different samplers on accuracy and convergence of the model. This will answer our second research question (RQ2). To answer our third research question (RQ3), we evaluate all possible *combinations* of positive and negative samplers in terms of accuracy, time complexity and coverage of items. This experiment will give us a more detailed insight about different samplers based on different criteria.

4.6.1. MULTI-CHANNEL SAMPLING VERSUS CONVENTIONAL INTEGRATION OF FEEDBACK

The purpose of the first set of experiments is three-fold. First, we want to compare the performance of our proposed FM-Pair model when individual feedback types are used as auxiliary features with its performance when all feedback types are used for multi-channel sampling. Secondly, we compare the performance of our proposed method with some baseline and state-of-the-art collaborative filtering models. Finally, as a sanity check we compare the performance of our models with cases that only one type of feedback is used. We also perform an oracle experiment in which recommendations from best channel, on a user-basis, are recommended to users. This experiment can be considered an upper bound for a late fusion approach that dynamically choses the best channel from which recommendations are generated. Table 4.2 compares the performance of different algorithms based on six evaluation metrics on the three datasets that we used in this work. Below, we describe the algorithms that are compared in Table 4.2.

[11] `https://github.com/recsyschallenge/2016`

- **MostPop**: This method represents a baseline method where items are ranked based on their popularity and all users are recommended the top most-popular items.

- **KNN**: This algorithm is user-based K-Nearest Neighbor implementation in MyMediaLite. For this algorithm all types of feedback are exploited, and they are all considered equally relevant. The number of nearest neighbors (parameter K) is set to 10, the default value of the implementation in MyMediaLite.

- **AFM**: Attentional Factorization Machines [12] (AFM) [126] is a recent extension to FMs where additional weight parameters are learned for pairwise interactions. For this model, we used three groups of features, which represent users, items and type of feedback (See Figure 4.1). For the MovieLens dataset, we generate ranked lists based on the predicted ratings and for XING and Kollekt we assigned a label of 1 to observed interactions and sampled random unobserved pairs and assigned a label of -1 to them, similar to [126]. The ranked lists are generated based on the predicted scores. For this algorithm all feedback types are exploited, and the type of feedback is considered as an auxiliary feature.

- **FFM**: Field-Aware Factorization Machines[13] (FFM) [48] is another extension of FMs where several representations of features are learned depending on the fields of the pairwise interactions. For this algorithm, we created three fields corresponding to users, items and feedback type. The ranked lists are created similarly to the previous algorithm.

- **BPR**: This case refers to our experiments with Bayesian Personalized Ranking (BPR) [98] with standard sampling and matrix factorization scoring function. This method does not exploit channel information. All feedback is exploited, and all occurrences of feedback are considered to be equally relevant.

- **WRMF**: This algorithm implements Weighted Regularized Matrix Factorization (WRMF) [39], a factorization-based method for datasets with implicit feedback.

- **FM-Pair (Single)**: In this experiment we use our proposed FM-Pair model where only the feedback from one channel is used. Sampling is done similarly to the standard BPR sampling method. This experiment is done for the XING and the Kollekt datasets as they have different feedback channels.

- **FM-Pair (Prior combine)**: This method implements the idea of *prior combine* [120] where models are trained sequentially. In this experiment, we first train a model using a single feedback channel with FM-Pair (with the hyper-parameters that we listed in Section 5.5.5), and use the trained model as a source of priors for the subsequent model, which is trained with the next feedback type. The subsequent models are initialized with the parameters learned from the preceding models and are trained with the new data points.

[12] We used the implementation in: `https://github.com/hexiangnan/attentional_factorization_machine`

[13] We used the implementation in: `https://github.com/guestwalk/libffm`

- **FM-Pair (Oracle experiment)**: In this method we first train different models for each feedback channel separately. Assuming that a hypothetical late-fusion method is able to pick the best recommendations from each trained model, for each user the recommendations from the model with the best performance is taken. The performance of the final model is then calculated by averaging the best result for each user. The results of this method can be considered an upper-bound for a late fusion approach that can dynamically pick the best channel to generate recommendations.
- **FM-Pair (Std. Sampling)**: In this case we use feedback from all channels using FM-Pair model. Here the channel type is used as an auxiliary feature in the FM-Pair model (see Figure 4.1). The sampling is done similarly to the standard sampling method of BPR.
- **FM-Pair (Multi-Channel)**: In this case we use our proposed FM-Pair model where type of feedback is used to adapt the sampling method. The results in Table 4.2 report the best sampling method for each dataset. In the next two subsections we do finer-grained experiments on different sampling methods.

The results in Table 4.2 leads us to several interesting insights. Firstly, as it can be seen in this table, with the right sampling method **FM-Pair (Multi-channel)** is performing the best compared to other methods. This suggests that the sampling method has a significant role in the quality of recommendations. Secondly, we see that both WRMF and BPR are performing better than the two recent methods of AFM and FFM. We attribute this observation to the fact that these approaches both do not equate the lack of an interaction with a definitive negative feedback. Instead, BPR, with the pairwise optimization, and WRMF, with the use of confidence scores, are incorporating a less naive assumption about the state of the unobserved interactions. We note that FM-Pair also enjoys an advantage in this regard due to its use of pairwise sampling.

To have an insight about the relative accuracy of multi-channel sampling with ensemeble methods, we perfomed two experiments namely **FM-Pair (Prior combine)** and **FM-Pair (Oracle experiment)**. The first method exploits differnet channels to sequantially train the model whereas **FM-Pair (Oracle experiment)** represents an upper bound for a late fusion approach that attempts to select the best recommendation channel for each user. The results show that the oracle improves over all the individual channels, indicating that additional channels have the potential to improve model accuracy (RQ0). Sequential training however, does not outperform the performance of the models trained on individual channels. This result is aligned with the findings in [120], where it has been shown that prior distributions learned by one channel does not necessarily hold for other channels.

The most straightforward way of exploiting feedback type is to add it as an auxiliary feature in the FM-Pair model (see Section 4.4.1 and Figure 4.1). This approach is labeled **FM-Pair (Std. Sampling)** in Table 4.2. This method however, does not always perform better that BPR and it performs worse than the oracle experiment, meaning that it would be possible to train a late fusion approach that could outperform it in a given application scenario. Nevertheless, our proposed multi-channel sampling method, **FM-Pair (Multi-channel)**, beats both the Oracle (representing the best possible late fusion

Table 4.2: Performance of the single-channel and the multi-channel training methods compared to several baselines.

		Recall		MRR		NDCG	
	Method / K	10	20	10	20	10	20
Kollekt	MostPop	0.1259	0.1743	0.0919	0.0966	0.0792	0.0937
	KNN	0.0910	0.1077	0.1095	0.1115	0.0754	0.0802
	AFM	0.0856	0.1383	0.0689	0.0751	0.0559	0.0723
	FFM	0.1178	0.1559	0.1114	0.1154	0.0880	0.0998
	WRMF	0.1600	0.2169	0.1550	0.1610	0.1189	0.1371
	BPR	0.1287	0.1807	0.1002	0.1060	0.0828	0.0989
	FM-Pair (Single: Listening)	0.1141	0.1549	0.0782	0.0823	0.0748	0.0900
	FM-Pair (Single: Following)	0.1316	0.1872	0.0975	0.1111	0.0810	0.0991
	FM-Pair (Prior combine)	0.1035	0.1406	0.0819	0.0858	0.0661	0.0775
	FM-Pair (Oracle experiment)	0.1494	0.2120	0.1289	0.1405	0.0937	0.1125
	FM-Pair (Std. Sampling)	0.1368	0.1955	0.1126	0.1195	0.0907	0.1090
	FM-Pair (Multi-Channel)	**0.1919**	**0.2747**	**0.1568**	**0.1653**	**0.1337**	**0.1598**
XING	MostPop	0.0294	0.0484	0.0648	0.0696	0.0290	0.0362
	KNN	0.1721	0.2408	0.2805	0.2863	0.1750	0.1979
	AFM	0.1403	0.2029	0.2468	0.2541	0.1415	0.1639
	FFM	0.0909	0.1391	0.2003	0.2075	0.1010	0.1173
	WRMF	0.0989	0.1577	0.1771	0.1846	0.1021	0.1229
	BPR	0.1451	0.2342	0.2397	0.2487	0.1428	0.1765
	FM-Pair (Single: Reply)	0.0162	0.0225	0.0334	0.0359	0.0387	0.0339
	FM-Pair (Single: Bookmark)	0.0159	0.0287	0.0223	0.0277	0.0119	0.0132
	FM-Pair (Single: Click)	0.1609	0.2554	0.2686	0.2775	0.1591	0.1928
	FM-Pair (Prior combine)	0.1565	0.2474	0.2472	0.2568	0.1490	0.1850
	FM-Pair (Oracle experiment)	0.1691	0.2620	0.2836	0.2922	0.1603	0.1931
	FM-Pair (Std. Sampling)	0.1429	0.2330	0.2403	0.2494	0.1413	0.1757
	FM-Pair (Multi-Channel)	**0.2010**	**0.3188**	**0.3034**	**0.3119**	**0.1920**	**0.2365**
MovieLens 1M	MostPop	0.1259	0.1743	0.0919	0.0966	0.0792	0.0937
	KNN	0.1697	0.2560	0.4823	0.4874	0.2801	0.2870
	AFM	0.0460	0.0943	0.1640	0.1794	0.0964	0.1173
	FFM	0.0466	0.0884	0.3210	0.3324	0.1491	0.1542
	WRMF	0.1735	0.2716	0.5742	0.5784	0.3587	0.3455
	BPR	0.1744	0.2740	0.5409	0.5468	0.3370	0.3357
	FM-Pair (Std. Sampling)	0.1570	0.2517	0.4924	0.4981	0.3070	0.3067
	FM-Pair (Multi-Channel)	**0.1770**	**0.2770**	**0.5831**	**0.5879**	**0.3685**	**0.3505**

approach) and BPR (representing the best possible approach that uses all channels but does not differentiate them). Furthermore, **FM-Pair (Multi-channel)** performs better than **FM-Pair (Std. Sampling)** meaning that the multi-channel approach can better exploit the underlying channel information. We further performed a t-test significance analysis on the results of the experiments and found that the improvement of the **FM-Pair (Multi-channel)** method compared to the oracle experiment is statistically significant ($p < 0.05$). This experiment allows us to answer RQ1 positively: channel-informed sampling does indeed allow for a better exploitation of multiple feedback channels.

Finally, we compare our adapted method of FM-Pair with Gaussian Process Factorization Machines (GPFM) [80], a recent work that also adapted a pairwise optimization method for FMs using Gaussian kernels. For this setup we used the Matlab implementation[14] of GPFM that was released with that work. The kernel of the Gaussian process is chosen to be the RBF kernel, the recommended kernel from the authors. The GPFM model has much higher time and space complexity, compared to FM-Pair as it requires to calculate the inverse of preference kernels, which is a time-intensive task. With this method we achieved a Recall@10 of 0.1146 for the Kollekt dataset, thus not necessarily better than our adapted FM-Pair method. We only tested this method on the Kollekt dataset and due to the high complexity of this method, we did not further test this method on our larger datasets of XING and MovieLens.

4.6.2. COMPARISON OF SAMPLING STRATEGIES

Next, we turn to methods that exploit channels using channel-based level-informed sampling, described in Section 4.4, and compare them with few baselines. The following sampling strategies were chosen as most interesting and informative methods:

Multi-Channel-Full: In this case the positive user-item pairs and negative items are both sampled informed by levels. The positive pair and the negative item is sampled according to Eq. (4.15) and Eq. (4.20) respectively.

Multi-Channel-Uni: With this sampling method the positive user-item pairs are sampled with the multi-channel method, i.e., Eq. (4.15), but the negative items are sampled uniformly according to the sampling distribution (4.18). With this method the channel information is only exploited for sampling the positive pairs.

Standard BPR: This is the standard BPR sampling method. The positive pairs are sampled uniformly from the dataset and the negative items are sampled uniformly from the set of items with which the sampled user has not yet interacted.

Popularity Oversampling: With this method the popular items are oversampled as negative according to Eq. (4.19). The positive pair sampler is the same as the standard BPR.

Dynamic Oversampling: This is the sampling method that has been introduced in [96]. The chance of an item to be sampled as negative is dynamically updated during the training phase according to the current model parameters. Positive user-item pairs are sampled uniformly the same way as the standard BPR.

The FM-Pair model is trained using the SGD optimization method with 300 iterations and after every 10 iterations the model is evaluated. Figure 4.3 shows the performance of the five sampling methods after an increasing number of iterations. Each point in the curve is the average MRR@10 of the four-fold cross validation experiment. The error bars are the standard deviations of the four folds. The five sampling methods are also compared with the popularity baseline, where the most popular items are recommended.

Consideration of the graphs in Figure 4.3 leads us to several interesting insights. The first insight is that there is always a channel-based (i.e. multi-channel) approach that outperforms standard BPR. This observation confirms again our answer to RQ1 given in Section 4.6.1. Exploiting levels during sampling improves over conventional forms of

[14] `http://trungngv.github.io/gpfm/`

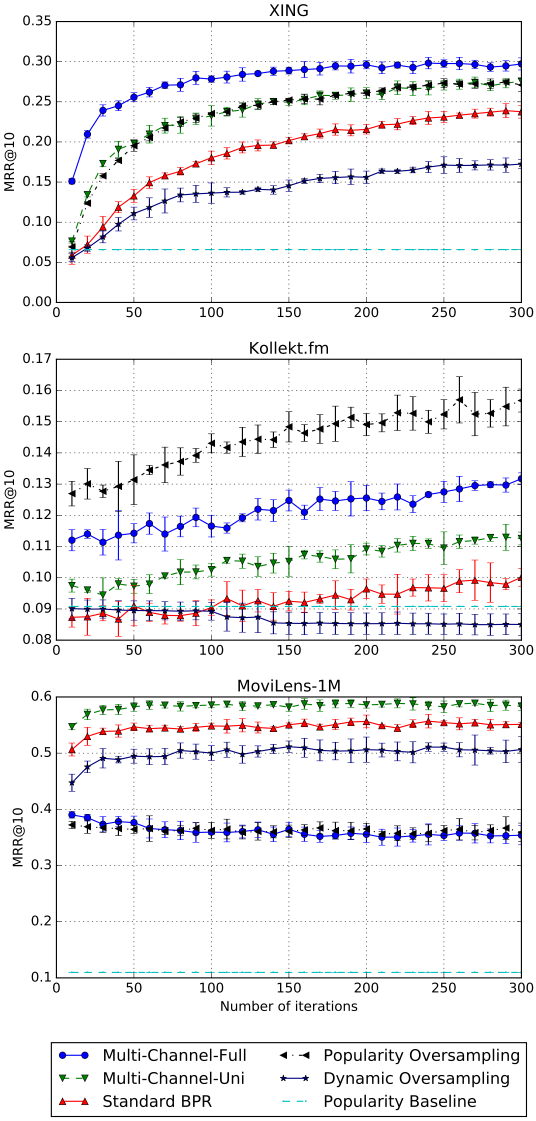


Figure 4.3: Comparison of different sampling method for the FM-Pair model based on the number of iterations in the SGD algorithm.

integration of feedback. Specifically, the improvement over BPR is an improvement over a sampling method that does not exploit channels.

The second insight is that different sampling methods allow maximum performance to be achieved on different datasets. The two multi-channel sampling methods perform the best in the XING dataset. On the MovieLens dataset, Multi-Channel-Uni performs the best, whereas on the Kollekt dataset Popularity-Oversampling achieves the best results.

By increasing the number of iterations, the accuracy of most sampling methods improves except for the Popularity-Oversampling and Multi-Channel-Full methods for the MovieLens dataset. Different aspects of these data contribute to explaining this variation. Looking at the number of items in each dataset reveals that the MovieLens datasets has rather small number of items compared to the other two datasets, and the amount of feedback per item is much larger. With a uniform negative item sampler, in each iteration of the algorithm each item is expected to be sampled approximately 202 time in the MovieLens, 17 times in the Kollekt and 3 times in the XING dataset[15]. Therefore, with a uniform item sampler each item is sampled more frequently compared to the other two datasets. The two sampling methods that perform the worse in the MovieLens datasets (i.e. Popularity-Oversampling and Multi-Channel-Full) are different from the rest in the sense that they oversample popular items as negative, resulting to even more updates for popular items. Note that the Multi-Channel-Uni sampling method still performs better than the standard BPR. This means that oversampling the *positive* user-item pairs from higher levels (i.e. higher ratings in this dataset) achieves better results, whereas oversampling popular items as *negative* worsen the results in this dataset.

In the other two datasets, multi-channel sampling methods as well as popularity oversampling result in better-performing models. In these two datasets, due to larger number of items, choosing the 'right' item becomes more crucial and thus a non-uniform item sampler can outperform a uniform item sampler. In the Kollekt dataset, a less expected result is that the popularity oversampling performs even better than the Multi-Channel samplers. We further examined our datasets [74] and found out that the Kollekt dataset is significantly more popularity-skewed compared to the other two datasets and sampling of non-popular items is less informative, and it might explain that oversampling popular items results in the best model in this dataset. In this dataset 1% of items have 40% of the interactions. This can also explain the rather-high accuracy of the popularity baseline in this dataset.

The Multi-Channel samplers also perform better than the Dynamic-oversampling method [96]. In the Kollekt dataset, Dynamic-oversampling performs even worse when the number of iterations increases. The performance drop of Dynamic-oversampling in this dataset can also be due to popularity-skewness of the data. The effectiveness of Dynamic-oversampling in different datasets requires more in-dept analysis of this sampling method and it falls outside of the scope of this work.

4.6.3. ACCURACY, COMPLEXITY AND COVERAGE OF DIFFERENT COMBINATIONS OF SAMPLERS

We turn to detailed experiments on channel-based, level-informed sampling methods for all three datasets. In this subsection, we study different combinations of positive and negative samplers and compare their performance in terms of accuracy, time-complexity and item coverage.

There are two approaches to choose positive samples: uniform and multi-channel (Eq. (4.16)), and four approaches to choose negative samples: uniform-item (Eq. (4.18)),

[15] In each iteration of SGD on the entire dataset, the number of samplings would be equal to the number of training data points. With a uniform negative item sampler, the above numbers can be calculated by dividing the training data to the total number of items.

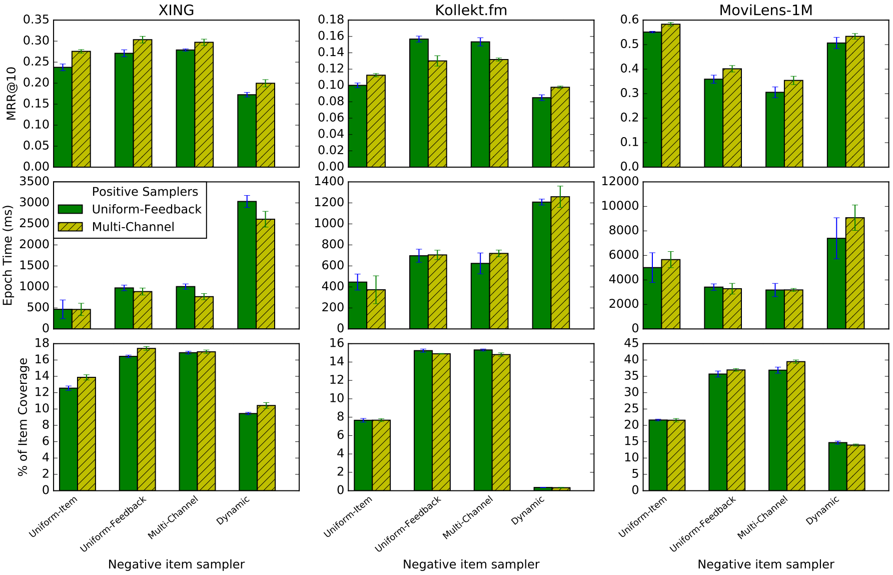


Figure 4.4: Comparison of different sampling methods for the FM-Pair model based on accuracy of recommendations using MRR@10, Epoch time (time of one iteration on the entire dataset) and percentage of covered items in recommendations. The horizontal axis represents different negative item sampling methods and the legends specify the positive sampling methods.

uniform-feedback (Eq. (4.19)), multi-channel sampling (Eq. (4.20)) and dynamic oversampling ([96]). These sampling options create eight different combinations of sampling. In Figure 4.4 we illustrate the performance of these eight combinations in terms of accuracy, time-complexity and item coverage. In this chart, the four types of negative item samplers are represented in the horizontal axis each of which is evaluated with the two types of positive samplers that we described in this work. The MRR@10 achieved by these approaches is illustrated in the top row of Fig. 4.4. We see that there is no universal combination of positive and negative sampling that will lead to the best MRR (check Section 4.6.2 for a more detailed comparison of the five interesting sampling strategies).

Training time is shown in the second row of Fig. 4.4. It is reported in terms of epoch time, which is the time of one iteration on the training data. The time complexity of the multi-channel samplers is in the same range as uniform samplers. The higher time complexity of the Dynamic-oversampling methods can be attributed to the fact that each update has an additional complexity of $\mathcal{O}(k)$.

Coverage is shown in the third row of Fig. 4.4. It reports the total percentage of training items that is recommended. Here the approaches involving Dynamic-oversampling and uniform-item sampling fall short.

In sum, we have found that channel-based, level-informed sampling is able to provide advantages in time complexity and in coverage. It can also improve prediction performance, which is described in more details in Section 4.6.2.

4.7. CONCLUSION AND FUTURE WORK

This chapter uses BPR as a learning-to-rank approach within a FMs framework to fully explore the contribution that multiple channels of user feedback can make to recommendations. We show that multiple feedback channels are useful, that channel-informed, level-based sampling outperforms the more straightforward, but relatively naïve, approaches of late fusion or of integrating channel information as auxiliary features. Multi-channel samplers are shown to outperform standard BPR sampling. Our experiments reveal that the choice of multi-channel sampler depends on the dataset, meaning that the “right" sampling method should be established experimentally for a specific dataset. In this work, we have assumed that the order of the levels is known in advance. Further work can be done on also ‘learning’ the right order of the levels. Our future work will also explore further models such as list-wise learning-to-rank methods for use with multi-channel positive feedback.

IV

Advanced Learning Models

5

Weighted Factorization Machines

Factorization Machines (FMs) are general factorization models for collaborative filtering that are capable of encoding additional information in terms of feature vectors. They have been shown to be effective models for exploiting auxiliary features. FMs are able to learn factorized parameters for all non-zero features in a vector, however, they cannot learn the *importance* of these features. The features' contribution to prediction is influenced by the values that are given to them, which are typically fixed a priori depending on the problem. In this chapter, we first show that the way that samples are encoded as feature vectors has important influence on the accuracy of the model. We then propose adapted optimization models that can learn weights for each *group* of features. Our proposed method that we refer to as Weighted Factorization Machines (WFM) can be applied effectively to both explicit and implicit feedback data. Experimental results on benchmark datasets show that the proposed models can improve the accuracy of recommendations, while maintaining the same computational complexity. [1].

[1] This chapter is currently in preparation to be submitted as Loni, Babak, Mirko Polato, Jaehun Kim, Keki Burjorjee, Martha Larson, and Alan Hanjalic. "Weighted Factorization Machines" In *Proceedings of 27th ACM Conference on User Modeling, Adaptation and Personalization.* 2019.

5.1. INTRODUCTION

Factorization is a tried and true approach to collaborative filtering (CF) for recommender systems. Factorization models learn a low-dimensional representation of users and items by mapping them into a space of *latent* factors [55]. Factorization Machines (FMs) [88] are a general factorization framework for collaborative filtering that can learn latent factors not only for users and items, but also for any *auxiliary* features of users, items or their interactions. In that sense, Factorization Machines can mimic other factorization models, such as Matrix Factorization [55], Tensor Factorization [49], Attribute-Aware factorization [30] and SVD++ [54], by making use of feature engineering. FMs have recently gained momentum in the field of Recommender Systems, due to their superior performance, scalability and simplicity.

In conventional factorization models, the training data is typically represented as a user-item matrix. In Factorization Machines however, the input data is represented as a set of feature vectors each corresponding to a single training sample. A single feature vector indicates which user has interacted with which item. If additional information about user-item interactions is available, it is represented in the form of auxiliary features that extend this vector. This representation gives FMs great flexibility, since additional data can be simply encoded into the model in the form of auxiliary features. The underlying model of FMs learns factorized parameters for any feature that is present in the feature vectors and uses those factorized parameters to predict the score of a given interaction.

Despite the great advantages of FMs for learning the latent factors of auxiliary features, they suffer a disadvantage: They cannot effectively learn the optimal relative contribution of different features in the feature vector for the purpose of calculating the prediction score. This can be more problematic if auxiliary features such as context are present in the feature vectors. In such cases, the model cannot effectively learn how much it should *rely* on the context and how much on the actual user-item interactions. One way to control the effect of different features in FMs is the way that the features are vectorized, that is, how we encode users, items, and any possible auxiliary information in the form of discrete or real-valued features. The feature vectors in FMs are typically built by concatenating *one-hot* vectors of users and items resulting in a binary vector. Auxiliary features however, can be encoded by binary or real-valued features depending on the domain and the type of feature [92]. The encoding method can notably influence the accuracy of the model. Previous studies have not addressed how to *learn* such encodings and they typically require the encodings to be given a priori or they just rely on the naïve binary encoding for each feature.

To demonstrate the importance of feature encoding on the accuracy of recommendations, we performed a simple experiment to observe the performance of FMs when we change the encodings of features by applying a set of weight parameters to the features based on their group[2]. Figure 5.1 illustrate the accuracy of recommendations in terms of two ranking metrics of Precision and MRR on the benchmark dataset of MovieLens 1M. The solid blue bar reflects the accuracy of recommendations when no auxiliary features are used whereas the other three bars show the performance of the model when auxiliary

[2] The group of features can be understood to reflect the feature type. For example user, movie and genre of a movie each represents one group of features. It has also been referred as *field* of a feature in a related work [48].

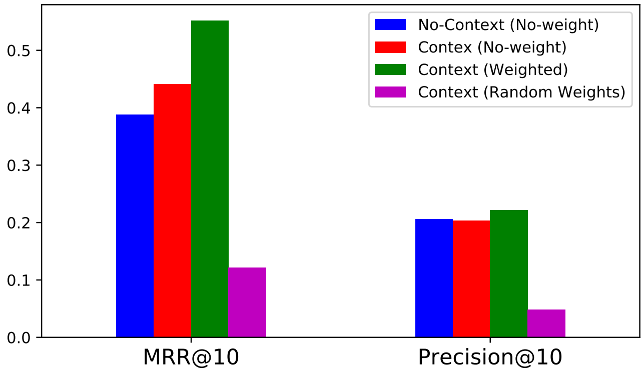


Figure 5.1: The effect of using weights for features in Factorization Machines on the accuracy of recommendations in the dataset of MovieLens1M. With a handcrafted set of weights for each group of features (using prior knowledge) the accuracy of the model can be improved substantially.

features are added with different weighting methods. The red hashed bar show the performance of the model when the auxiliary features are encoded without any weight (i.e. with the standard binary encoding), whereas the green hashed bar applies handcrafted weight parameters for the three groups of features, user, item and genre, such that the weight for genre features are significantly lower than the weights for users and items. The experiment reveals the potential of weighted features for improving the performance of the FMs model. To make sure that such improvement is not random, we also compared the three methods with a scenario where the weight parameters are assigned randomly[3] (the right-most bar).

In this chapter, we build on the initial insight that the performance of FMs could be improved if instead of being handcrafted, optimal weights could be learned. Motivated by the above demonstration, we propose an adaptation of FMs where the importance of each group of features is learned during the optimization process. We refer to our method as Weighted Factorization Machines (WFM) since it learns weight parameters for group of features. Specifically, WFM assigns a weight parameter to each group of features that makes it possible to capture the importance of the group. Optimal weight parameters are learned during the optimization process. This process effectively makes the prediction performance of the FM independent of the initial feature encoding. It is worth mentioning that the weights that we are referring in this work should not be confused with the conventional parameters of the FMs model that are also referred as weights in some related work. Our proposed method can be applied to datasets with both explicit or implicit feedback. For datasets with explicit feedback we adapt a *point-wise* loss function (that is optimized for MSE and can be effective for rating prediction), based on [90], and learns the optimal parameters using gradient descent method. For datasets with implicit feedback we adapt a *pair-wise* loss function (optimized for AUC and effective for ranking) based on [35, 97] and learn the model parameters with gra-

[3] Random weights are generated from a standard normal distribution and the resulting metrics are the average metrics over five times repetition of the experiment.

dient descent. We refer to our models based on point-wise and pair-wise optimization methods as *WFM-Point* and *WFM-Pair*, respectively.

Our proposed method has been tested on four datasets two with explicit and two with implicit feedback. The auxiliary features that are used in the experiments are user and item attributes, context features and features from other domains. Experimental results on different recommendation scenarios show that our proposed method can substantially improve the accuracy of recommendations, while the complexity of the model remains the same.

We implemented two variations of WFM, an implementation with calculated derivatives, and a second implementation with auto differentiation using Tensorflow to exploit the potential of GPU computing for large datasets. The contributions of this chapter can be summarized as follows:

- We propose Weighted FM (WFM), an adaptation of FMs where the importance of features are reflected by weight parameters that are defined for groups of features.

- We adapt the underlying optimization process of FMs, such that WFM can seamlessly utilize both explicit and implicit feedback, and to effectively learn weight parameters.

- We release two implementations of WFM with command-line interfaces, that can be used out-of-the-box.

The remainder of this chapter is organized as follows: In the next section, we give an overview of related work in this area. In Section 5.3, we describe our proposed method in detail. Section 5.4 proposes two applications of WFMs for context-aware and cross-domain collaborative filtering. In Section 5.5, we describe the experiments that we conducted in this chapter. We finally draw a conclusion in Section 5.6 and discuss possible future work.

5.2. RELATED WORK

Despite the existence of an interesting body of research on Factorization Machines, less attention have been given to how features can be encoded in FMs. The standard encoding that is proposed by Rendle in the original paper of FMs [90] suggests the following encoding method: divide the features into different groups (based on the type of feature) and normalize the features in such a way that for each group the features sum up to 1. For groups that have only one feature per group (such as user, item, gender and etc.), this method is essentially a binary representation of feature vectors consisting of concatenation of one-hot encodings. Some studies suggest modified encodings that are applicable in particular situations. For example, [78] propose to normalize auxiliary features based on the number of user ratings (for cross-domain recommendation scenario) and this representation tends to be more effective than the standard encoding method. Recent work [5] considers static handcrafted weights for auxiliary features in FMs and shows that a static value in the range of 0.05 to 0.1 for auxiliary features tend to improve the performance of FMs compared to the standard encoding method. Our preliminary motivating experiment in Section 1 also confirms this observation.

The idea of using weights for features to indicate their importance has been proposed in some earlier work [116, 118, 127] and it has been shown to be effective for classification tasks with Support Vector Machines (SVMs). The above studies are based on the idea of assigning weights to features based on some information-theoretic measures or based on applying a feature selection algorithm to assess the quality of features. A similar idea is applied in kernel methods [33] where weight parameters are assigned for different kernels to learn the importance of kernels.

Along the line of research of optimizing features for Factorization Machines, Cheng et al. [17] proposed Gradient Boosting Factorization Machines (GBFMs) where a greedy feature selection technique is applied to select "good" features and then encode the selected features with the standard FM encoding method. Recently an extension to FMs, namely Attentional Factorization Machines (AFMs) [126] has been proposed with the purpose of discriminating the importance of different feature *interactions*. AFMs resembles FMs with a neural network with additional pooling layer where weight parameters for pairwise interactions between features are learned. In contrast to our work, where weight parameters are learned for groups of features (thus few additional parameters are added to the model), the AFMs model needs to learn a larger number of parameters due to the presence of additional pooling layer. Both AFMs and GBFMs are only optimized for point-wise prediction of interaction scores.

Little related work have been done that notably exploit the potential of feature groups in FMs. An exception is the study of Juan et al. [48], which proposes Field-aware Factorization Machines (FFMs). This model is an extension of FMs where multiple representations of features are learned, depending on the *field* (group) of features. For example in a dataset with three groups of features, two different representations for a user are learned: one for interaction with an item and the other for interaction with context (third group). FFMs have larger time-complexity compared to FMs since multiple representations need to be learned for each feature. Furthermore, the model is optimized for classification and might not be very effective for rating prediction or ranking. Our work share the idea of distinguishing different groups of features with FFMs. However, our model still learns a general representation for each feature but controls the contribution of different features with group-based weight parameters.

5.3. FRAMEWORK

In this section we introduce an adaptation of Factorization Machines that tries to learn the relative importance of different groups of features [92]. Each group of features describes a particular aspect of the model, e.g., users, items, context and so on. As shown in Section 5.1, different approaches to feature encoding can have a substantial impact on the effectiveness of the model. For this reason, we propose, for both explicit and implicit feedback, optimization methods that are able to learn a weight for each group of features.

Let us assume that the input data is represented by the set $\mathscr{D} \equiv \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ where for all $i \in [1, l]$, $\mathbf{x}_i \in \mathbb{R}^n$ is an n-dimensional real-valued feature vector and y_i is its corresponding target value. Let us also assume that $\mathscr{G}$ is the set of groups in which features of $\mathbf{x}_i$ are grouped and let the vectors $\mathbf{x}_i$ be group normalized, i.e., $\forall \mathbf{g} \in \mathscr{G}, \sum_{j \in \mathbf{g}} x_{ij} = 1$, as suggested in [90]. Then, given the function $\pi : [1, n] \mapsto [1, |\mathscr{G}|]$ which maps a feature index to its corresponding group index, we can define the WFM model as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} \alpha_{\pi(j)} w_j x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} \alpha_{\pi(j)} x_j \alpha_{\pi(j')} x_{j'} \langle \mathbf{v}_j, \mathbf{v}_{j'} \rangle, \tag{5.1}$$

where $\boldsymbol{\alpha} \in \mathbb{R}_+^{|\mathcal{G}|}$ obeys the constraint $\|\boldsymbol{\alpha}\|_1 = |\mathcal{G}|$. It is easy to see that this new formulation is a generalization of the standard FM. In fact, by fixing $\alpha_i = 1$ for all $1 \le i \le |\mathcal{G}|$, Equation (5.1) is exactly the 2-way FM model. Similar to the standard FM [92], we can reformulate (5.1) in such a way that it can be computed with a number of operations in the order of $\mathcal{O}(kN_z(\mathbf{x}))$:

$$\begin{aligned} \hat{y}(\mathbf{x}) = w_0 &+ \sum_{j=1}^{n} \alpha_{\pi(j)} w_j x_j \\ &+ \frac{1}{2} \sum_{f=1}^{k} \left[\left(\sum_{j=1}^{n} \alpha_{\pi(j)} x_j v_{j,f} \right)^2 - \sum_{j=1}^{n} \alpha_{\pi(j)}^2 x_j^2 v_{j,f}^2 \right], \end{aligned} \tag{5.2}$$

where $N_z(\mathbf{x})$ is the number of nonzero elements in the vector $\mathbf{x}$.

With the introduction of the group weights we ended up with a model in which the set of parameters that need to be learned is $\Theta \equiv \{w_0, w_q, \ldots, w_n, v_{1,1}, \ldots, v_{n,k}, \alpha_1, \ldots, \alpha_{|\mathcal{G}|}\}$. In order to learn the optimal set of parameters Θ, we use the common regularized objective function, which tries to minimize the sum of the losses over the training data while trying to avoid overfitting:

$$OPT_\Theta(\mathcal{D}) = \underset{\Theta}{\operatorname{argmin}} \sum_{(\mathbf{x},y)\in\mathcal{D}} L(\hat{y}(\mathbf{x}|\Theta), y) + R(\Theta), \tag{5.3}$$

where $L(\hat{y}(\mathbf{x}|\Theta), y)$ is the loss function and $R(\Theta)$ the regularization function. The optimization of (5.3) is done via the Stochastic Gradient Descent (SGD) algorithm using the following generic update rule:

$$\theta \leftarrow \theta - \eta \frac{\partial [L(\hat{y}(\mathbf{x}|\Theta), y) + R(\Theta)]}{\partial \theta}, \tag{5.4}$$

where η is the learning rate.

The standard model of FMs propose to optimize the above loss function with a *point-wise* manner, that is, the prediction error for each training sample (point) is taken into account to calculate the loss. This method however, is only effective for rating prediction problems and might not be necessarily an optimal model for learning from implicit feedback. Most of the methods that extend FMs such as FFM [48] and AFM [126] are also only optimized for rating prediction.

In the following subsections we propose two different loss functions that are optimized for rating prediction and ranking. For ranking problems we propose a *pair-wise* loss function based on the idea of Bayesian Personalized Ranking [97]. We use the same regularization function for both optimization methods, similar to both FMs and BPR methods, which is defined as $R(\Theta) = \sum_{\theta\in\Theta} \lambda_\theta \theta^2$, where λ_θ are regularization hyper-parameters that need to be validated at training time. In both the point-wise and the pair-wise optimization, at each iteration of SGD, after the update of all the model parameters α_g with $g \in [1, |\mathcal{G}|]$ we add a constraint on parameters α_g to control the relative difference of the weight parameters. To do so, we apply a normalization step, inspired by [58], to ensure that $\sum_{i=1}^{|\mathcal{G}|} \alpha_i = |\mathcal{G}|$, and this can be done with the following normalization step:

$$\alpha_g \leftarrow \alpha_g \frac{|\mathcal{G}|}{\sum_{i=1}^{|\mathcal{G}|} \alpha_i}, \tag{5.5}$$

which is applied to all α_g. In what follows, we will refer to this operation with $norm(\boldsymbol{\alpha})$. The above normalization can be seen as a further regularization mechanism. Similar normalization has been done in [58] and it turned out to be an effective approach.

Typically, parameters w_i are initialized with 0 while factorization parameters $v_{*,f}$ should be initialized by a zero-mean normal distribution with standard deviation σ. Finally, for α_g both random and the uniform initializations are possible, but in any case they must be normalized to sum up to $|\mathcal{G}|$.

5.3.1. OPTIMIZATION FOR RATING PREDICTION

For the task of rating prediction with WFM, we define the loss function as a point-wise squared loss [92] over data points $\mathbf{x}$ with target y as:

$$L(\hat{y}(\mathbf{x}|\Theta), y) = (\hat{y}(\mathbf{x}|\Theta) - y)^2. \tag{5.6}$$

We now define the derivatives of this loss function with respect to the parameters $\theta \in \Theta$. By substituting (5.6) into (5.4) we get:

$$\theta \leftarrow \theta - \eta \left(2(\hat{y}(\mathbf{x}|\Theta) - y) \frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \theta} + 2\lambda_\theta \theta \right), \tag{5.7}$$

in which only the gradient of the WFM model w.r.t. to the parameters needs to be calculated. Based on (5.1), the gradient of $\hat{y}(\mathbf{x}|\Theta)$ with respect to w_j for $j \in [0, n]$ is equal to:

$$\frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial w_j} = \begin{cases} 1 & \text{if } j = 0 \\ \alpha_{\pi(j)} x_j & \text{if } 1 \le j \le n \end{cases}, \tag{5.8}$$

which can be computed in $\mathcal{O}(1)$. The gradient w.r.t. the factorization parameters $v_{j,f}$ for $i \in [1, n]$ and $f \in [1, k]$ is defined as:

$$\frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial v_{j,f}} = \alpha_{\pi(j)} x_j \sum_{j' \neq j} \alpha_{\pi(j')} x_{j'} v_{j',f}, \tag{5.9}$$

which can also be computed in $\mathcal{O}(1)$, if we assume to pre-calculate the sum $\sum_j \alpha_{\pi(j')} x_{j'} v_{j',f}$ during the computation of $\hat{y}(\mathbf{x}|\Theta)$. Finally the gradient w.r.t. α_g for $g \in [1, |\mathcal{G}|]$ is equal to

$$\begin{aligned} \frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \alpha_g} = & \sum_{f=1}^{k} \left(\sum_{j|\pi(j)=g} x_j v_{j,f} \right) \left(\sum_{j=1}^{n} \alpha_{\pi(j)} x_j v_{j,f} \right) \\ & + \sum_{j|\pi(j)=g} w_j x_j - \sum_{f=1}^{k} \alpha_g \sum_{j|\pi(j)=g} x_j^2 v_{j,f} \end{aligned} \tag{5.10}$$

The computational complexity of this gradient is $\mathcal{O}(kN_z(\mathbf{x}))$. Algorithm 3 describes the SGD algorithm for the WFM-Point optimization. The complexity of each iteration over a training point $(\mathbf{x}, y)$ is bounded by the cost of updating $\boldsymbol{\alpha}$, which is $\mathcal{O}(|\mathcal{G}|kN_z(\mathbf{x}))$, and since in general $|\mathcal{G}| \ll k$ and $|\mathcal{G}| \ll N_z(\mathbf{x})$ we can approximate it with $\mathcal{O}(kN_z(\mathbf{x}))$.

Algorithm 3: Learning WFM-Point

```
Input: Training data $\mathcal{D}$; Regularization parameters $\lambda_\theta$; Learning rate $\eta$; Std.
       initialization parameter $\sigma$; Set of features groups $\mathcal{G}$; Groups index mapper
       $\pi$
Output: Model parameters $\Theta$
initialize $w_*$, $v_*$ and $\alpha_*$ parameters
$\boldsymbol{\alpha} \leftarrow norm(\boldsymbol{\alpha})$
do
    for $(\mathbf{x}, y) \in \mathcal{D}$ do
        update $w_0$ according to (5.8)
        for $j \in \{1, \dots, n\} \wedge x_j \neq 0$ do
            update $w_i$ according to (5.8)
            for $f \in \{1, \dots, k\}$ do
                update $v_{j,f}$ according to (5.9)
        for $g \in \{1, \dots, |\mathcal{G}|\}$ do
            update $\alpha_g$ according to (5.10)
        $\boldsymbol{\alpha} \leftarrow norm(\boldsymbol{\alpha})$
while convergence
$\Theta \leftarrow (w_0, \dots, w_n, \boldsymbol{\alpha}, v_{1,1}, \dots, v_{n,k})$
return $\Theta$
```

5.3.2. Optimization for Ranking

For datasets with implicit feedback the recommendation task is typically considered as a ranking problem where the items are ranked according to their relevance to a given user and the top items are recommended to the user.

In this section we propose WFM-Pair, a learning algorithm for WFM that is optimized for ranking. The optimization method in WFM-Pair is inspired by [35, 81, 97] where model parameters are learned by pair-wise comparison of items. The common assumption in these studies is that any observed positive feedback is preferred over missing feedback. Given this assumption and the fact that we rely on a pair-wise approach, the training data $\mathcal{S}$ is defined as a set of tuples (u, i, j) such that i is a positive item for the user u, and j is a missing item for u that is sampled uniformly from the unobserved interactions. More specifically, $\mathcal{S} \equiv \{(u, i, j) \mid i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$ where $\mathcal{I}$ is the set of all the items and $\mathcal{I}_u^+$ is the set of *positive* items for user u.

WFM-Pair adapts the BPR optimization model by taking into account the auxiliary features that are present for a given user-item interaction. The model parameters in

WFM-Pair are learned by maximizing the following likelihood function:

$$\prod_{(u,i,j)\in\mathcal{S}} p(i >_{u,\mathbf{z}} j|\Theta), \tag{5.11}$$

where the notation $(i >_{u,\mathbf{z}} j)$ indicates that user u prefers item i over item j under observed auxiliary features $\mathbf{z}$. Similar to BRP, the probability $p(i >_{u,\mathbf{z}} j|\Theta)$ can be modeled by the sigmoid function $\sigma(y) = \frac{1}{1+e^{-y}}$ where y is a scoring function that calculates the utility of training tuples. The utility function $\hat{y}$ is defined as:

$$\hat{y}(u,i,j,\mathbf{z}|\Theta) = \hat{y}(\mathbf{x}_{u,i,\mathbf{z}}|\Theta) - \hat{y}(\mathbf{x}_{u,j,\mathbf{z}}|\Theta), \tag{5.12}$$

where $\mathbf{x}_{u,i,\mathbf{z}}$ are training feature vectors. We represent the feature vectors $\mathbf{x}$ with the sparse form of $\mathbf{x}_{u,i,\mathbf{z}} = \{(u,x_u),(i,x_i),\{(z,x_z) \mid z \in \mathbf{z}\}\}$, where $x_i = x_u = 1$ are used to encode the user and the item, while x_z are values of auxiliary features. By replacing $\mathbf{x}_{u,i,\mathbf{z}}$ in Equation (5.1) and then by applying it in (5.12) we have

$$\begin{aligned}\hat{y}(u,i,j,\mathbf{z}|\Theta) = w_i - w_j &+ \left(\alpha_U \alpha_I \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f})\right)\\ &+ \alpha_I \left(\sum_{z\in\mathbf{z}} \alpha_{\pi(z)} x_z \sum_{f=1}^{k} v_{z,f}(v_{i,f} - v_{j,f})\right),\end{aligned} \tag{5.13}$$

where α_U and α_I are the weights for the user features group and the item features group, respectively.

We now define the objective function $OPT_\Theta(\mathcal{D}_\mathcal{S})$ of WFM-Pair over the set $\mathcal{D}_\mathcal{S} \equiv \{(u,i,j,\mathbf{z}) \mid \forall\ (u,i,j) \in \mathcal{S}, \mathbf{z} = \mathbf{z}(u,i)\}$, as in (5.3) with the loss function defined as the opposite of the logarithm of the likelihood function (5.11):

$$L(\hat{y}(u,i,j,\mathbf{z}|\Theta)) = -\ln(\sigma(\hat{y}(u,i,j,\mathbf{z}|\Theta))), \tag{5.14}$$

Since minimizing (5.3) is equivalent to maximizing its opposite, in order to facilitate the further derivations we consider the maximization version. Similarly to the point-wise method, the optimization is done via SGD. Using the update rule 5.4 and by applying the sigmoid function, we obtain the following update rule for WFM-Pair:

$$\theta \leftarrow \theta + \eta\left(\frac{\hat{y}}{1+e^{\hat{y}}}\frac{\partial\hat{y}}{\partial\theta} + 2\lambda_\theta\theta\right), \tag{5.15}$$

We now define derivatives of $\hat{y}$ with respect to the parameters $\theta \in \Theta$. Based on (5.13), the gradient of $\hat{y}$ with respect to w_q for $q \in [0,n]$ is equal to:

$$\frac{\partial\hat{y}}{\partial w_q} = \begin{cases} 1 & \text{if } q = i \\ -1 & \text{if } q = j \\ 0 & \text{otherwise} \end{cases}, \tag{5.16}$$

which can be clearly computed in $\mathcal{O}(1)$. The gradient of $\hat{y}$ w.r.t. the factorization parameters $v_{q,f}$, with $q \in [1,n]$ and $f \in [1,k]$, are the followings

$$\frac{\partial \hat{y}}{\partial v_{q,f}} = \begin{cases} \alpha_U \alpha_I (v_{i,f} - v_{j,f}) & \text{if } q = u \\ \alpha_I \left(\alpha_U v_{u,f} + \sum_{z \in \mathbf{z}} \alpha_{\pi(z)} x_z v_{z,f} \right) & \text{if } q = i \\ -\alpha_I \left(\alpha_U v_{u,f} + \sum_{z \in \mathbf{z}} \alpha_{\pi(z)} x_z v_{z,f} \right) & \text{if } q = j \\ \alpha_I \alpha_{\pi(z)} x_z (v_{i,f} - v_{j,f}) & \text{if } q = z \in \mathbf{z} \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

According to the above equation the gradients for $v_{u,f}$ and $v_{z,f}$ can be computed in $\mathcal{O}(1)$ whereas the gradients of $v_{i,f}$ and $v_{j,f}$ can be calculated in $\mathcal{O}(N_z(\mathbf{x}))$. However, those gradients are actually opposite to each other so they can be computed only once at each iteration. Finally, the gradient w.r.t to the weights $\boldsymbol{\alpha}$ are defined as:

$$\frac{\partial \hat{y}}{\partial \alpha_U} = \alpha_I \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}), \tag{5.18}$$

$$\frac{\partial \hat{y}}{\partial \alpha_I} = \alpha_U \sum_{f=1}^{k} v_{u,f}(v_{i,f} - v_{j,f}) + \sum_{z \in \mathbf{z}} \alpha_{\pi(z)} x_z \sum_{f=1}^{k} v_{z,f}(v_{i,f} - v_{j,f}), \tag{5.19}$$

and for every $g \in [3, \ldots, |\mathcal{G}|]$, assuming $U = 1$ and $I = 2$, the gradient of $\hat{y}$ w.r.t. α_g is:

$$\frac{\partial \hat{y}}{\partial \alpha_g} = \alpha_I \left(\sum_{z \in \mathbf{z} | \pi(z) = g} x_z \sum_{f=1}^{n} v_{z,f}(v_{i,f} - v_{j,f}) \right). \tag{5.20}$$

The time complexity of (5.18) is linear to the number of factors, so it is $\mathcal{O}(k)$, and both (5.19) and (5.20) can be computed in $\mathcal{O}(kN_z(\mathbf{x}))$.

Algorithm 4 summarizes the WFM-Pair optimization method. The complexity of each iteration over a training point is bounded by the cost of the loop over the auxiliary features, which is $\mathcal{O}(kN_z(\mathbf{z}))$, plus the cost of updating $\boldsymbol{\alpha}$, that is $\mathcal{O}(kN_z(\mathbf{x}))$ (for the same consideration as in Section 5.3.1). Since $N_z(\mathbf{x}) \approx N_z(\mathbf{z})$, we can conclude that the overall complexity of each iteration is $\mathcal{O}(kN_z(\mathbf{x}))$.

5.4. APPLICATIONS OF WFM

An important advantage of Factorization Machines is that, by design, they are able to exploit additional information to improve recommendations. In particular, they are suitable for both context-aware [99] and cross-domain recommendation tasks [78]. In this section we propose two applications of WFM on context-aware and cross-domain collaborative filtering.

5.4.1. CONTEXT-AWARE RECOMMENDATION

Context-aware recommendation aims to take advantage on context information to better model the interactions between users and items. In WFM-Pair, context can be treated

Algorithm 4: LEARN WFM-PAIR

```
Input: Training data $\mathscr{D}$; The item set $\mathscr{I}$; Regularization parameters $\lambda_\theta$; Learning
       rate $\eta$; Std. initialization parameter $\sigma$; Set of features groups $\mathscr{G}$; Groups
       index mapping function $\pi$
Output: Model parameters $\Theta$
initialize $w_*$, $v_*$ and $\alpha_*$ parameters
$\boldsymbol{\alpha} \leftarrow norm(\boldsymbol{\alpha})$
do
    $\mathbf{x}_{u,i,\mathbf{z}} \leftarrow sample(\mathscr{D})$
    $j \leftarrow sample(\mathscr{I} \setminus \mathscr{I}_u^+)$
    $\mathbf{x}_{u,j,\mathbf{z}} \leftarrow \{(u,1),(j,1),\{(z,\mathbf{x}_{u,i,z}) \mid z \in \mathbf{z}\}\}$
    update $w_i$ and $w_j$ according to (5.16)
    for $z \in \mathbf{z}$ do
        $w_z \leftarrow w_z + 2\eta\lambda_\theta\theta$
        for $f \in \{1,\dots,k\}$ do
            update $v_{z,f}$ according to (5.17)
    for $f \in \{1,\dots,k\}$ do
        update $v_{u,f}$, $v_{i,f}$ and $v_{j,f}$ according to (5.17)
    update $\alpha_U$ and $\alpha_I$ according to (5.18) and (5.19)
    for $g \in \mathscr{G}$ do
        update $\alpha_g$ according to (5.20)
    $\boldsymbol{\alpha} \leftarrow norm(\boldsymbol{\alpha})$
while convergence
$\Theta \leftarrow (w_0,\dots,w_n,\boldsymbol{\alpha},v_{1,1},\dots,v_{n,k})$
return $\Theta$
```

as an auxiliary features (i.e, $\mathbf{z}$) for user-item interactions. These auxiliary features can be encoded inside the feature vectors as previously described in (5.1). For example, let the context of an interaction (u,i) be the user mood (e.g., “angry") and the movie genre (e.g., “sci-fy"), then we can represent such context with $\mathbf{z}(u,i) = \{(\text{angry},1),(\text{sci-fy},1)\}$. Consequently, we can represent the expanded form of the feature vector $\mathbf{x}_{u,i,\mathbf{z}}$ as:

$$\mathbf{x}_{u,i,\mathbf{z}} = (\underbrace{0,\ldots,0,x_u,0,\ldots,0}_{|U|},\underbrace{0,\ldots,0,x_i,0,\ldots,0}_{|I|},\underbrace{x_{z_1},\ldots,x_{z_{|Z|}}}_{|Z|}), \tag{5.21}$$

where Z is the set of contextual features. In contrast to the features x_i and x_u that are usually 1, since they represent a one-hot encoding of the user/item identifier, the contextual features (and the auxiliary in general) x_z can assume any real-value. As mentioned previously, according to [90] it is preferable to normalize (i.e., sum up to 1) the auxiliary features, which is a common practice to reduce biases due to different scales. One of the advantages of our proposal is that WFMs can also reduce biases between groups since the algorithm itself manages to find the best weight for each group of features.

5.4.2. Cross-Domain Recommendation

Cross-Domain Collaborative Filtering (CDCF) methods exploit additional information from the so-called *source* (a.k.a. auxiliary) domains to improve recommendations in a *target* domain. The core idea of CDCF is that a user’s preferences in one domain (e.g., music) can be exploited to better learn the user’s taste on another domain (e.g., movies). In the literature, several methods concerning the CDCF problem have been proposed. Cantador et al. [16] provided a good overview of the CDCF methods. In this section we propose how to perform cross-domain collaborative filtering with WFM based on the idea of [78]. The effectiveness of this approach is also tested with the Amazon dataset, which contains user interactions in multiple domains.

Thanks to the flexibility of FMs in exploiting additional features, the information concerning the source domains can be transferred to the target domain by means of auxiliary features. The resulting feature vectors can be used with WFMs to train a model. In order to understand how auxiliary features are represented, let us suppose that p is the number of source domains and $I_j(u)$ is the set of items in domain j that user u interacted with. Then, for every item that the user interacted with in the source domain an auxiliary feature is created. Consequently, the feature vectors $\mathbf{x}$ in the target domain can be represented with the following sparse form consisting of both target and source domain features:

$$\mathbf{x}(u,i) = \{\underbrace{(u,1),(i,1)}_{\text{target domain features}},\underbrace{\cup_{j=1}^{p}\{(z,x_z(u,j))|z \in I_j(u)\}\}}_{\text{source domains' features}} \tag{5.22}$$

where $x_z(u,j)$ is the value of feature z, i.e., the score that should be considered for the interaction between user u and item z in the domain j. Different approaches can be used to define the feature values $x_z(u,j)$. Here we propose to use normalized values for source domain features. That is, the values of source domain features are normalized by the number of interactions that the user made in the source domain. Similar to the context-aware scenario, WFMs learn weights for groups of features. In our implementation, for each source domain we consider a different feature group. For example if the target domain is ‘movies’ and we have user interactions in source domains of ‘books’ and ‘music’, in total we would have four feature groups corresponding to users, items, source domain features from the ‘books’ domain and from the ‘music’ domain.

5.5. Datasets and Experiments

In this section we describe the datasets, evaluation method and experiments that we performed in this chapter.

5.5.1. Datasets

We choose four datasets for our experiments in this chapter. The Amazon dataset (containing explicit user ratings in four domains) has been used in rating prediction experiments to evaluate WFM-Point and also the application of that for cross-domain recommendations. The two datasets of MSD and Frappe (containing implicit user feedback) have been chosen to evaluate WFM-Pair. The benchmark dataset of MovieLens (containing explicit user ratings) has been used for both rating prediction and ranking experiments. Below the four datasets are described. The statistics of the datasets are listed in Table 5.1.

- **Million Song Dataset (MSD)**: Million Song Dataset [11] is a collection of audio features and metadata for one million songs, but also contains user-item interactions. In this chapter, we use a dense subset of the dataset to be used by the collaborative filtering algorithms. All users and items in this subset have at least 50 interactions. Along with the main dataset, there is a set of annotations contatining 13 different genres for music tracks [106], which have been used as auxiliary features. The dataset also contains the play counts of songs for user-song interactions. The play counts have been used as a confidence score for WMF, one of the baseline algorithms that we tested in this dataset.

- **Frappe**: Frappe is a context-aware mobile app discovery tool. It logs the number of times users run an application on their mobile phone. It also logs several contexts such as time, date, location and weather. The Frappe dataset [7] consists of approximately 100K instances of implicit feedback. An observation is considered as positive feedback if the user runs an application at least one time. We choose this dataset due to the presence of several contextual features.

- **MovieLens 1M**: The MovieLens 1M dataset[4] is a benchmark dataset for recommender systems containing 1M movie ratings. We used the movie genres as auxiliary features in FMs, FFMs, and WFMs algorithms. We choose this dataset due to the presence of auxiliary features and widespread use in the community.

- **Amazon**: Amazon [62] is also a dataset with explicit user ratings. The items in this datasets are the products in the Amazon website and they belong to one of the following four domains: books, music CDs, DVDs and video tapes. We use this dataset since it has been used for cross-domain recommendations. We use a densified version of the datasets that have been used in previous work on cross-domain collaborative filtering [38, 78].

[4] https://grouplens.org/datasets/movielens/

Table 5.1: Statistics of the dataset used in this chapter

Dataset	#Users	#Items	#Feedback	Scale
MSD	93,295	29,449	8.3M	Implicit
Frappe	957	4,082	96K	Implicit
ML1M	6,040	3,670	1M	1-5
Amazon	15,994	84,508	270K	1-5

5.5.2. EVALUATION PROTOCOL

The experiments are evaluated with four fold cross-validation. To evaluate the accuracy of recommendations, for WFM-Point we use Root Mean Squared Error (RMSE) [101] to measure the accuracy of predicted ratings. WFM-Pair and other ranking methods are evaluated using Precision@K, Recall@K and Mean Reciprocal Rank (MRR@K) evaluation metrics.

To calculate the above metrics, for each user we first create a set of candidate items containing the user's relevant items and 1000 randomly chosen items [19]. We then create a ranked list by scoring the candidate items. Note that for datasets that have context associated with interactions (specifically for the Frappe dataset), the ranked list is generated for each user-context combination in the test set to make sure all candidate items are evaluated under the same conditions. The final metrics are calculated by averaging the metrics for all user.

5.5.3. WEIGHTED FMS FOR RATING PREDICTION

Rating prediction experiments are done on the MovieLens and Amazon datasets as they contain explicit feedback. On the Amazon dataset, we perform experiments for both single domain and cross-domain scenario. We perform two sets of experiments where the two domains of `books` and `music` are considered as target domains and other domains are considered as source domains. For our rating prediction experiment, the following algorithms are compared:

- **Factorization Machines (FM)**: This model is the standard FMs [90] model for rating prediction. For this setup no auxiliary features are used.

- **FM-Auxiliary**: In this setup, we use the same model as previous setup, but we also take into account the auxiliary features that are available in the datasets. For the two datasets of books and music from Amazon we use cross-domain ratings as auxiliary features based on the method that we described in Section 5.4.2 and for the dataset of MovieLens we use genres of the movies as auxiliary features.

- **WFM-Point**: This is the proposed method in this chapter where the point-wise loss function is used. Here, no auxiliary features are used and the features are divided into two groups corresponding to users and items.

- **WFM-Point-Auxiliary**: In this setup, we use our proposed WFM-Point method and we also use the auxiliary features. For the MovieLens dataset the features are divided into three groups (users, items and genres) and for Amazon they are divided into five

Table 5.2: Comparison of WFM-Point with several methods for rating prediction on MovieLens (ML1M) and the two domains of Music and Books from Amazon datset. The experiments are evaluated using RMSE (lower is better). The auxiliary features are movie genre for the ML1M dataset, and ratings from other domains for the Amazon Book and the Amazon Music datasets.

Method / Dataset	Music	Books	ML1M
Random	2.0343	1.9997	1.7081
User Average	1.0273	0.9516	1.0357
Item Average	1.0646	1.0145	0.9795
BMF	1.0293	0.9088	0.9053
FM	1.0675	0.7956	0.8829
FM-Auxiliary	1.0577	0.7937	0.8727
AFM	0.9769	0.7751	0.8609
WFM-Point	0.9696	0.7907	0.8849
WFM-Point-Auxiliary	0.9602	0.7841	0.8679

groups (users, items and three groups each corresponding to one of the auxiliary domains).

- **AFM**: This is the Tensorflow implementation of Attentional Factorization Machines (AFM) [126], a recent extension to FMs. For this algorithm we used the same auxiliary features as previous setup and we searched several combination of hyper-parameters and reported the best performing results in Table 5.2.

The number of factors for the above methods are set to 10. The above methods are all optimized based on the SGD optimization algorithm and the number of epochs on the training data is set to 100. More details about experimental reproducibility are described in Section 5.5.5.

Table 5.2 compares the performance of the above five methods on the datasets of Amazon and MovieLens. We also listed the accuracy of four baselines namely, Random prediction, User Average, Item Average and Biased Matrix Factorization (BMF) [55].

As can be seen in this table, the WFM-Point method can effectively exploit auxiliary features as it (WFM-Point-Auxiliary) performs better than the case that auxiliary features are not exploited. WFM also results to the lowest RMSE for the Amazon Books dataset and for the other two datasets it performs better than all other methods except AFM. The improved performance of AFM compared to WFM can be attributed the larger number of parameters in AFM and a more effective regularization mechanism (dropout vs. L2 regularization). This improvement however, comes with the cost of significant increase in training and prediction time[5] and a larger number of hyper-parameters that need to be tuned. And as you will see in the next section AFM fall behinds WFM for ranking problems as it is only optimized for rating prediction.

[5] The average epoch time of WFM and AFM on the MovieLens dataset is 0.7 and 3.7 respectively. Both methods are tested on a same machine running on CPU.

Table 5.3: Comparison of WFM-Pair with several methods on datasets with implicit feedback. The methods are compared using Precision, Recall and MRR evaluation metrics with two different cutoffs.

		Precision		**Recall**		**MRR**	
Dataset	**Method \Cutoff**	5	10	5	10	5	10
MSD	MP	0.0297	0.0287	0.0076	0.0146	0.0691	0.0813
	WMF	0.0512	0.0493	0.0111	0.0214	0.1036	0.1207
	BPR-MF	0.1713	0.1730	0.0417	0.0838	0.2739	0.2989
	FFM	0.0585	0.0544	0.0142	0.0260	0.1503	0.1676
	AFM	0.0040	0.0037	0.0063	0.0117	0.0094	0.0113
	FM-Pair	0.1787	0.1789	0.0438	0.0869	0.2806	0.3049
	FM-Pair-Context	0.0505	0.0621	0.0124	0.0304	0.0855	0.1123
	WFM-Pair	**0.2423**	**0.2221**	**0.0615**	**0.1108**	**0.3352**	**0.3552**
Frappe	MP	0.0229	0.0191	0.1143	0.1910	0.0465	0.0586
	WMF	0.0258	0.0184	0.1292	0.1842	0.0776	0.0848
	BPR-MF	0.0373	0.0237	0.1864	0.2367	0.1220	0.1286
	FFM	0.0405	0.0271	0.2023	0.2711	0.1229	0.1320
	AFM	0.0292	0.0234	0.1459	0.2338	0.0753	0.0869
	FM-Pair	0.0334	0.0226	0.1669	0.2256	0.1064	0.1143
	FM-Pair-Context	0.0267	0.0236	0.1337	0.2360	0.0645	0.0782
	WFM-Pair	**0.0440**	**0.0280**	**0.2199**	**0.2802**	**0.1362**	**0.1442**
ML1M	MP	0.1009	0.0977	0.0220	0.0407	0.2271	0.2555
	WMF	0.1189	0.1287	0.0211	0.0431	0.2590	0.2881
	BPR-MF	0.2172	0.2000	0.0464	0.0808	0.3473	0.3663
	FFM	0.1387	0.1378	0.0224	0.0466	0.2960	0.3210
	AFM	0.0329	0.0331	0.0041	0.0086	0.0741	0.0874
	FM-Pair	0.1826	0.2007	0.0382	0.0842	0.4021	0.4361
	FM-Pair-Context	0.2427	0.2298	0.0471	0.0811	0.4729	0.4860
	WFM-Pair	**0.2828**	**0.2414**	**0.0524**	0.0818	**0.4815**	**0.4943**

5.5.4. WEIGHTED FMS FOR RANKING

We use the three datasets of MSD, Frappe and MovieLens 1M for testing the performance of the WFM-Pair method. We use the available context and attributes of the three dataset as auxiliary features for our WFM model.

For the Frappe dataset we use two context groups: weekday (day of the week) and homework (with three values of `unknown, home, work`) creating four feature groups together with user and item groups. For the datasets of MSD and MovieLens the genre of songs and movies are considered as feature groups and thus the total number of feature groups are three.

The following setups are implemented in order to compare the performance of our proposed WFM-Pair method with some baseline and state-of-the-art algorithms:

- **FM-Pair**: In this method, we use Factorization Machines with a pairwise loss function based on the BPR criterion. In this setup we do not use any auxiliary features.
- **FM-Pair-Context**: This is similar to the previous setup but we also use the available

auxiliary features in the dataset.

- **FFM**: FFM [48] is an extension of FMs where different representations for each feature are learned based on the group (field) of features. This method also exploits available auxiliary features in the dataset.

- **WFM-Pair**: This is our proposed weighted FM method with pairwise loss function. The auxiliary features are the same as the previous setup.

- **Other Methods**: In addition to the above methods, we also used the baseline method of Weighted Matrix Factorization (WMF) [41], BPR with Matrix Factorization (BPR-MF) [97] and Most Popular (MP) method.

Table 5.3 compares the performance of the above methods in terms of different ranking metrics on the three datasets of MSD, Frappe and ML1M. The metrics are calculated with two different cutoffs[6]. The three methods of WMF, FFM and AFM perform worse compared to the other models (except MP) most likely due to the fact that they are not optimized for ranking (FFMs is slighty better than FM-Pair in the Frappe dataset). Among the methods that are optimized for ranking, FM-Pair and BPR-MF have rather close performance. Interestingly, when the context features are added to the FM-Pair model (i.e., FM-Pair-Context) with standard normalized encoding the accuracy of the recommendations does not necessarily improve (in two datasets, MSD and Frappe, it declines whereas in the ML1M dataset it improves). Nevertheless, the proposed WFM-Pair method performs better than both FM-Pair and FM-Pair-Context and thus the learned weights can positively influence on calculating the scores of interactions.

5.5.5. EXPERIMENTAL REPRODUCIBILITY

The two implementation of WFM (with regular SGD and with Tensorflow) contains documentation to run WFM and reproduce experiments[7]. The two implementations are slightly different, as the first one use SGD for optimization (i.e., updates are done per sample in the training data), whereas the Tensorflow-based implementation uses Mini-Batch Gradient Descent (MBGD) [102] where the updates are done per mini-batches. Despite this difference, we did not find a noticeable difference between the two implementations in terms of recommendations accuracy.

The hyper-parameters of the algorithms are found via a grid-search over a range of candidate values and vary per dataset. For rating prediction experiments we used SGD implementation. We found regularization coefficients of 0.00025 and 0.0005 for Amazon and MovieLens respectively, and the same learning rate of 0.005 for both datasets. For the ranking problem, we used MBGD implementation with batch sizes of 10K, 100K and 5K for the three datasets of MovieLens, MSD and Frappe, respectively. The learning rate for these datasets was found to be 0.005, 0.01 and 0.001, respectively, and the best regularization coefficient for all three datasets was determined to be 0.05.

[6] Due to the space limitation results with cutoff of 20 is removed. In the paper corresponding to this chapter the results with cutoff of 20 is included.

[7] The first implementation is part of WrapRec and the second implementation can be found in: `https://github.com/babakx/wfm`

5.6. CONCLUSION AND FUTURE WORK

In this chapter we introduced WFMs, an extension to FMs that learns a better representation for features by learning weight parameters for feature groups. We experimentally demonstrated the effectiveness of this approach for two tasks, rating prediction and ranking. WFMs can improve the accuracy of recommendations compared to FMs without introducing additional computational complexity. This improvement can be attributed to the ability of WFMs to model features as groups. The ability of WFMs to learn weights can avoid a time-consuming search to find optimal weights for feature groups.

Our experiments reveal that the standard normalized encoding might not effectively exploit the potential of auxiliary features in FMs. With the weight parameters, the contribution of auxiliary features to prediction can be controlled and thus more accurate models can be learned.

The idea of WFMs can also be applied for classification tasks and further experiments can be done for classification or click-through rate prediction. Another future direction to this chapter is to extend the underlying model of WFM to support wider range of regularization techniques such as dropout. The ability of WFMs to distinguish feature groups can also be applied to neural network models in order to exploit structure of data and learn better models.

V

Implementation Framework

6

WrapRec, An Evaluation Framework for Recommender Systems

WrapRec is an easy-to-use Recommender Systems toolkit which allows users to easily implement or wrap recommendation algorithms from other frameworks. The main goals of WrapRec are to provide a flexible I/O, evaluation mechanism and code reusability. WrapRec provides a rich data model which makes it easy to implement algorithms for different recommender system problems, such as context-aware and cross-domain recommendation. The toolkit is written in C# and the source code is publicly available on GitHub under the GPL license[1].

[1] This chapter is published as Loni, Babak, and Alan Said. "WrapRec: an easy extension of recommender system libraries." In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 377-378. ACM, 2014.

6.1. Introduction

Personalized recommender systems are becoming very popular in online marketing, social networks and mobile applications. RecSys and machine learning communities have developed several successful libraries for recommender systems such as *MyMediaLite*[2], *Apache Mahout*[3] and *LensKit*[4].

Most of existing libraries require specific data formats, usually as text files, and are not flexible enough to support new data formats. Data processing is an important step in recommender systems which are usually neglected in current RecSys libraries. The existing frameworks usually require the users to provide the data in a specific format, and similarly they output results in a specific format. As an example consider that a user has a data source with a particular format which is not supported by the existing toolkits. A common practice to use this data is to convert to a format which is supported by the toolkit and then run the experiment on the converted data. Now, if the user wants to repeat the experiment for many different scenarios, usually the data preprocessing is done by an external application in a semi-manual way which makes the usage of toolkits difficult and error prone.

WrapRec is a toolkit, with defined routines. It allows users to incorporate any data processing steps easily into the experiment, without requiring to change the underlying algorithms. The toolkit is a wrapper around existing libraries, which allows them to be plugged into the system. The main goal of the toolkit is to provide high-level interfaces for low-level services to make it easier to run RecSys algorithms. WrapRec's solution to this goal is to provide a rich data object model to make data access flexible and safe. This feature makes it easier to implement algorithms which rely on multiple data sources such as context-aware and cross-domain recommendation scenarios. Furthermore, WrapRec provides high level interfaces to potential algorithms that can be wrapped into the toolkit. Similarly an abstraction layer for evaluation of algorithms is defined in the toolkit to implement custom evaluators and re-use evaluation logics for different RecSys algorithms.

6.2. Overview of the Toolkit

The WrapRec toolkit consists of three main components: Data Layer, Recommendation Engine and Evaluation Pipeline. The components are made independent of each other, meaning they can be modified and extended without requiring to change other components. Figure 1 illustrates the high level architecture of the toolkit. Below, we briefly describe each component:

- **Data Layer**: This component provides a common data interface, regardless of the underlying format of the data source. If users want to read a custom data format, they need to implement an interface which reads the data and converts it into the common data objects. This feature makes it possible to run different experiments without requiring to change the underlying algorithms or evaluation mechanism. Furthermore, the toolkit is able to represent the data objects within a

[2] `http://mymedialite.net`

[3] `http://mahout.apache.org`

[4] `http://lenskit.grouplens.org`

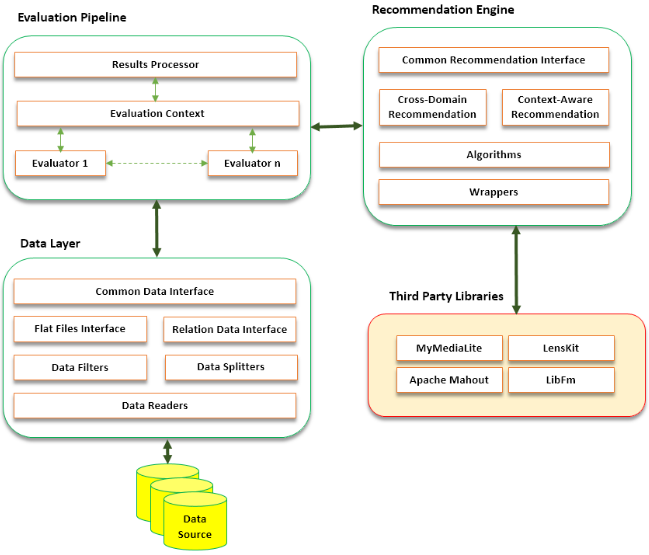


Figure 6.1: The overall architecture of WrapRec.

context which enables the consumer of data to take advantage of the C# language and issue LINQ (Language Integrated Queries) on data. This feature enables users to write an integrated query, select a custom subset of data and pass it to the other components of the system.

- **Recommendation Engine**: This component is the core of the toolkit. It provides common interfaces to RecSys algorithms. The focus of this component is not to implement the RecSys algorithms, but to provide a common interface to different algorithms that are implemented in various libraries. If users want to use an algorithm from existing libraries, they need to wrap the functionality of the third party libraries into the toolkit. Currently, the toolkit provides wrapper classes for MyMediaLite [31] (written in C#) and LibFM [92] (written in C++). This component also provides interfaces to implement Cross-Domain and Context-Aware recommendation algorithms.

- **Evaluation Pipeline**: WrapRec provides evaluation services based on the pipeline design pattern[5]. This enables the evaluator objects to share their results on a context object and thus improve the performance and re-usability of code. Pipeline-based evaluation allows users to re-use the evaluation logic for different RecSys algorithms since the evaluators only talk with high level data objects which are independent of the algorithms.

[5] msdn.microsoft.com/en-us/library/ff963548.aspx

6.3. How to Use the Toolkit

The current version of WrapRec is implemented as a .Net library which can be added to a .Net project and be called through its public APIs. The WrapRec toolkit can be easily installed in a .Net project by installing its package through the *Nuget*[6] package management system, or by adding the library file directly to the references of the project. The source code of the toolkit is publicly available under GPL license on GitHub[7].

A major goal of WrapRec is to be easy to use. Based on the architecture of the system, to run a RecSys experiment users need to define three main objects: a data interface, an algorithm interface and an evaluation pipeline. More advance scenarios can be defined by extending each of the mentioned three objects. Listing 1 lists a simple RecSys experiment which trains a model on the Movielens 1M dataset[8] using MyMediaLite's rating prediction algorithms. The model is then tested using two evaluation metrics on 30% of the data.

Listing 6.1: Sample three step code to run a recommender system experiment

```
// step 1: dataset
var data = new Dataset<ItemRating>(new MovieLensReader("data.dat"), 0.7);

// step 2: recommender
var recommender = new MediaLiteRatingPredictor(new MatrixFactorization());

// step3: evaluation
var ep = new EvaluationPipeline(new EvalutationContext(recommender, data));

ep.Evaluators.Add(new RMSE());
ep.Evaluators.Add(new MAE());

// run the experiments
ep.Run();
```

Each of the three steps in the above example can be extended without modifying other steps. For example, to read a dataset with a different format, a different dataset reader needs to be defined in step 1. More advanced scenarios and samples can be found in the online documentation of the toolkit.

6.4. Updates in WrapRec 2.0

In the new version of WrapRec[9] the specifications of an experiment can be defined in an xml configuration file so that all settings can be defined in one place. The configuration file defines the model and its parameters, specifies how the dataset should be split, and describes how the evaluation should be done[10].

[6] http://www.nuget.org/

[7] https://github.com/babakx/WrapRec

[8] http://grouplens.org/datasets/movielens/

[9] https://github.com/babakx/WrapRec/releases

[10] The schema of the configuration file is described in: http://babakx.github.io/WrapRec/GetStarted.html

6.5. OUTLOOK

In this chapter, we briefly introduced an open source RecSys toolkit that can be used for training and evaluation of recommender system algorithms. The toolkit is currently available on GitHub. The current version of the toolkit is available as a command-line interface as well as a Nuget library. In the current version of the toolkit, the specifications of an experiment can be defined in a configuration file, making it easier to reproduce the experiments that are implemented with this toolkit. In the future, we plan to further integrate this project with existing libraries and frameworks.

VI

OUTLOOK

7

CONCLUSION AND FUTURE WORK

In this thesis, we presented several factorization models for collaborative filtering and proposed how to apply them in different recommender system tasks. We presented our insights and their outcomes in the preceding technical chapters of thesis, supported by several experimental studies. In this chapter, we reflect on the objectives and research questions of this thesis, discuss the concluding points of this thesis, and provide practical points that could be further investigated in future studies.

7.1. DISCUSSION

The objectives of this thesis were to leverage advanced factorization models to effectively exploit the information that is present beyond a user-item matrix, to apply such models in a wider range of problems in recommender systems, and to improve their underlying performance with enhanced algorithms. The majority of the techniques that are presented in this thesis are based on Factorization Machines (FMs) [89], Bayesian Personalized Ranking (BPR) [98] and Matrix Factorization [55].

To address the first aspect of our objective, i.e., exploiting information that are present beyond the user-item matrix, in Chapter 2, we propose a method using Factorization Machines, to exploit auxiliary ratings that are present beyond the target recommendation domain. The proposed technique embeds external domain interactions as auxiliary features in FMs. Experimental results showed the effectiveness of this method for rating prediction problems. Later, in chapters 3 and 5 we adapt the proposed approach for top-N recommendation tasks. The success of this approach depends on how well the cross-domain interactions are translated to auxiliary features. Thanks to the expressiveness of FMs, the underlying model of FMs can learn latent factors for auxiliary features and seamlessly exploit them to better predict the scores of interactions.

In Chapter 2, we further study the expressiveness of FMs by proposing two applications that can leverage information inherent in the user-item matrix. The first application makes use of cluster membership of users and items to build auxiliary features that can be embedded to the training data. The second application, which we refer to as the 'slice and train' approach, propose to slice the training set based on properties of the data, train individual models on each slice, and indirectly use the rest of slices by translating them to auxiliary features. Based on the experiments that were carried out for this method, the 'slice and train' method can outperform the conventional training on the entire dataset.

The main insight that we obtained in Chapter 2 is that FMs provide an opportunity to exploit information that is ubiquitously present, but commonly under-appreciated by collaborative filtering algorithms. Moreover, FMs can seamlessly exploit such information without necessarily requiring the underlying model to be adapted. Furthermore, based on the experimental results, we can conclude that FMs can exploit cross-domain information effectively, they can leverage information that is extracted from the user-item matrix, and can be used to train more efficient models by training on the 'right' slice of data.

The models that are presented in Chapter 2 however, are limited to rating prediction problems since the standard model of FMs are optimized for prediction and not for ranking. To benefit from the advantages of FMs also for top-N recommendation tasks, in Chapter 3 we introduced FM-Pair, an FMs model with pairwise loss function, optimized for ranking. FM-Pair can not only learn a model from explicit ratings, but also from implicit unary feedback. In this chapter we show that FM-Pair is significantly more effective than a naïve implicit-to-explicit-feedback mapping since such mapping creates strong biases about user preferences and the underlying model that is used for training is not optimized for ranking. We also show that FM-Pair can be applied in the scenarios where auxiliary information (such as context or cross-domain interactions) are present, just like the standard FMs models.

The broader message of Chapter 3 is that the standard FM model is not suitable for ranking problems and implicit feedback. However, we can effectively optimize FMs for ranking and learning from implicit feedback with a pairwise optimization technique, while we can benefit from the advantages of FMs such as generalization and expressiveness.

The model that is introduced in Chapter 3, is further adapted for datasets with feedback from multiple channels. This model, which is introduced in Chapter 4, proposes several sampling methods to sample proper training data by exploiting different types of feedback. In this chapter we demonstrate that sampling methods have significant influence on accuracy of recommendations and convergence of the underlying optimization methods. We also show that conventional integration of side information with auxiliary features (in an FM model) is not always the best method to exploit such information. We found that if feedback type is exploited for an adapted sampling (instead of integrating as a feature), the accuracy of the trained FM model can be improved. Moreover, we show that collaborative filtering models that are trained using feedback from multiple channels are generally more accurate than a model that is trained on a single channel only. However, a naïve data aggregation is not an effective method to learn from multiple feedback channels.

The main insight that we gained from Chapter 4, is that, despite the advantages of FMs that we discussed in earlier chapters, a conventional integration of side information with auxiliary features is not always an effective approach to benefit from additional data. Depending on the problem and the recommendation task, integration of side information in the underlying algorithms can be more effective than a general feature encoding method. With this insight, we moved forward toward studying the underlying model of FMs to understand if we can make them smarter in such a way that they can learn the extent to which auxiliary features can be relied on.

In Chapter 5, we propose Weighted Factorization Machines (WFMs), an extension to the FMs model that learn additional weight parameters for each group of features. Such weights can control the contribution of different features and predict the score of interactions more precisely. WFMs are implemented with learning algorithms that are optimized for prediction and ranking. Experimental results show that WFMs outperform FMs in both rating prediction and ranking tasks. This improvement can be attributed to the fact that WFMs are aware of the groups of features and the adapted optimization methods exploit this information to learn the contribution of different groups. The effectiveness of WFMs is also studied in context-aware and cross-domain recommendation tasks.

In Chapter 6, we present WrapRec, a recommender systems library that contains implementation of the algorithms that are introduced in this thesis. WrapRec is also a general evaluation framework for recommender systems that can Wrap algorithms from other implementations and evaluate them under same settings.

In summary, FMs are great models to implement advanced factorization models. Their ease-of-use, expressiveness and generalization make them a desirable model for several recommendation tasks. At the same time, their ability to generalize the factorization process can also be a disadvantage as the underlying model is not aware of the nature of the features that are encoded in the training data. Depending on the problem and the task that is being approached, the proper factorization model needs to be adapted to achieve the best outcome.

7.2. FUTURE WORK

Based on the insights and achievements of this thesis, in this section we propose practical recommendations that can be a basis for future research on recommender systems.

7.2.1. CUSTOM OPTIMIZATION METHODS

The factorization models that are proposed in this thesis learn latent factors based on point-wise (using squared error loss) or pair-wise (using AUC loss) optimization techniques. These two methods learn the model parameters by minimizing or maximizing a loss function that is defined over the training data. The choice of loss function has a crucial role on the performance of the recommendation task. It is important that the loss function is optimized for the task that we want to approach. For example, a loss function that is optimized for prediction is not a proper choice for ranking. Likewise, a loss function that is optimized for ranking, might not be effective if we care about click-through-rate. Sometimes, based on the business requirements, none of the above targets might be interesting for a recommender system. For example, a metric that Spotify uses when it generates recommendations for a playlist, is the number of pages that a user needs to visit until he finds a song to play[1]. With such requirement, relevance of one single song is much more important than a list that is carefully optimized to sort songs based on their inferred relevance. In this case, a list-wise optimization method such as CLiMF [109], which optimizes Mean Reciprocal Rank (MRR), might be more effective than BPR.

An interesting direction of future work on Factorization Machines is to adapt other optimization techniques and study their effect on the metrics that are potentially interesting for a recommendation task. From the implementation point of view, it would be interesting if modeling and optimization tasks of the learning algorithms could be decoupled so that the optimization task could be developed independently. Another possible extension in this respect would be to implement multi-criteria optimization methods for FMs so that their learning algorithm can simultaneously optimize more than one metric.

7.2.2. FACTORIZATION AND CONTENT-BASED FEATURES

Factorization models are fast and effective techniques for collaborative filtering. Factorization Machines exploit any auxiliary features that are present in their training data. However, the choice of features in FMs has a significant influence on accuracy and complexity of models. Content features such as audio signals are not a proper choice of auxiliary features in FMs. Such features are typically dense and significantly increase the training and prediction time of an FM model. The underlying model of FMs attempts to learn latent factors for any feature that is presented in the training data and use the interactions of the trained factors to predict the utility of an item for a user. The interactions between the latent factors of several features of a single item is independent from the target user and is repetitive for all users. The standard model of FMs is not computation-

[1] According to RecSys 2018 Challenge: `http://www.recsyschallenge.com/2018/`

ally optimized to exploit dense content features and the effectiveness of such features on factorization models is not clear.

Future research can be done to optimize the computations of FMs for exploiting dense features. A successful model that exploit content features for factorization is the work of Oord et al. [123] where they propose a deep learning method that learns a representation of items in the same space as the factorized collaborative filtering space. Such techniques can be combined with Factorization Machines to simultaneously exploit content and other auxiliary features for training a hybrid model.

7.2.3. Elicitation of the 'Right' Data

In two different parts of this thesis (Chapters 2 and 4) we observed that by using the 'right' data (with the 'slice and train' algorithm or with the adapted sampling methods), the performance of the factorization models can be improved while the accuracy of recommendations improves or remains the same. Another interesting future research is to study how factorization models can be adapted to use minimum necessary data to learn model parameters. In that direction, we already studied [57] the trade-off between training data volume and accuracy of recommendations using a set of classic recommender system models. Further studies can be done to elicit the 'right' data to train factorization models without reducing the accuracy of recommendations.

7.2.4. Factorization Machines for Other Problems

Factorization Machines have been mainly used in recommender systems. Due to their ability to seamlessly exploit auxiliary features, they could be a proper model to learn latent factors in a shared space for entities that are related to each other. An interesting practical study for future work would be to use FMs in areas such as text classification, latent semantic analysis, and named entity classification. Recent work already applied FMs in sentiment classification [125] and decision prediction in Twitter [37]. Further studies can be done to exploit context or auxiliary information to study the potential of FMs on exploiting such information in other domains.

7.2.5. Unified Evaluation Framework

The vast majority of studies in recommender systems are evaluated using a set of evaluation metrics that are calculated on offline datasets. The technical details of evaluation mechanism such as data split, candidate items, new user/item strategy, and implementation details of metrics are typically disregarded in research papers. Said and Bellogin [103] showed that the implementation details and evaluation model of algorithms have significant role on the resulting metrics and thus, evaluation under different conditions become meaningless. In this thesis, we implemented WrapRec [77], an evaluation framework for recommender systems, to be able to compare different algorithms with the same evaluation framework and details. Further effort in future studies need to be done to adapt new evaluation conditions and metrics to WrapRec or other open source implementations.

BIBLIOGRAPHY

[1] In *Multiple Classifier Systems*, Lecture Notes in Computer Science. 2013.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[4] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M Pujol. Data Mining Methods for Recommender Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 2, pages 39–71. Springer US, 2011.

[5] Bjarni Arnason. Influence of auxiliary features in factorization-based collaborative filtering. 2016.

[6] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM.

[7] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR*, abs/1505.03014, 2015.

[8] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at AT&T Research*, 1, 2008.

[9] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *RecSys '11*, pages 333–336, 2011.

[10] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[12] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[13] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*, pages 3351–3359, 2016.

[14] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov 2002.

[15] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. *Cross-Domain Recommender Systems*, pages 919–959. Springer US, Boston, MA, 2015.

[16] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. *Cross-Domain Recommender Systems*, pages 919–959. Springer US, Boston, MA, 2015.

[17] Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R. Lyu. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 265–272, New York, NY, USA, 2014. ACM.

[18] Mark Connor and Jon Herlocker. Clustering items for collaborative filtering, 2001.

[19] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys '10*, pages 39–46, 2010.

[20] Arthur F. da Costa and Marcelo G. Manzato. Exploiting multimodal interactions in recommender systems with ensemble algorithms. *Inf. Syst.*, 56(C):120–132, March 2016.

[21] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[22] Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[23] Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 2006.

[24] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[25] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Comput. Sci. Rev.*, 20(C):29–50, May 2016.

[26] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *2010 IEEE International Conference on Data Mining*, pages 176–185, Dec 2010.

[27] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 176–185, Dec 2010.

[28] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Bayesian personalized ranking for non-uniformly sampled items. *JMLR W&CP, Jan*, 2012.

[29] Zeno Gantner, Lucas Drumond, Lars Schmidt-thieme, and Christoph Freudenthaler. Bayesian personalized ranking for non-uniformly sampled items, 2012.

[30] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: a free recommender system library. RecSys '11. ACM, 2011.

[31] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[32] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):13:1–13:19, December 2015.

[33] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

[34] Weiyu Guo, Shu Wu, Liang Wang, and Tieniu Tan. Personalized ranking with pairwise factorization machines. *Neurocomputing*, 214:191 – 200, 2016.

[35] Weiyu Guo, Shu Wu, Liang Wang, and Tieniu Tan. Personalized ranking with pairwise factorization machines. *Neurocomput.*, 214(C):191–200, November 2016.

[36] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 144–150. AAAI Press, 2016.

[37] Liangjie Hong, Aziz S. Doumith, and Brian D. Davison. Co-factorization machines: Modeling user interests and predicting individual decisions in twitter. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 557–566, New York, NY, USA, 2013. ACM.

[38] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. Personalized recommendation via cross-domain triadic factorization. WWW '13, 2013.

[39] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[40] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[41] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[42] Kurt Jacobson, Vidhya Murali, Edward Newett, Brian Whitman, and Romain Yon. Music personalization at spotify. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 373–373, New York, NY, USA, 2016. ACM.

[43] Michael Jahrer, Andreas Töscher, and Robert Legenstein. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, 2010.

[44] Mohsen Jamali and Martin Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 397–406, New York, NY, USA, 2009. ACM.

[45] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[46] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, New York, NY, USA, 2010. ACM.

[47] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 43–50, New York, NY, USA, 2016. ACM.

[48] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 43–50, New York, NY, USA, 2016. ACM.

[49] Alexandros Karatzoglou. Collaborative temporal order modeling. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 313–316, New York, NY, USA, 2011. ACM.

[50] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware

collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 79–86, New York, NY, USA, 2010. ACM.

[51] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 493–494, New York, NY, USA, 2013. ACM.

[52] Jon Kleinberg and Mark Sandler. Using mixture models for collaborative filtering. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 569–578, New York, NY, USA, 2004. ACM.

[53] Yehuda 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*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[54] Yehuda 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*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[55] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[56] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA, 2009. ACM.

[57] Martha Larson, Alessandro Zito, Babak Loni, and Paolo Cremonesi. Towards minimal necessary data: The case for analyzing training data requirements of recommender algorithms. 2017.

[58] Ivano Lauriola, Mirko Polato, and Fabio Aiolli. Radius-margin ratio optimization for dot-product boolean kernel learning. In Alessandra Lintas, Stefano Rovetta, Paul F.M.J. Verschure, and Alessandro E.P. Villa, editors, *Artificial Neural Networks and Machine Learning – ICANN 2017*, pages 183–191, Cham, 2017. Springer International Publishing.

[59] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*, 2012.

[60] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *CoRR*, abs/1205.3193, 2012.

[61] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. In *ACM RecSys '14*, pages 353–356, 2014.

[62] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), 2007.

[63] Bin Li. Cross-domain collaborative filtering: A brief survey. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 1085–1086, 2011.

[64] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. IJCAI'09, 2009.

[65] Gai Li and Qiang Chen. Exploiting explicit and implicit feedback for personalized ranking. 2016:1–11, 01 2016.

[66] Qing Li and Byeong Man Kim. Clustering approach for hybrid recommender system. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 33–38, Oct 2003.

[67] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, Jan 2003.

[68] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[69] Nathan N. Liu and Qiang Yang. Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 83–90, New York, NY, USA, 2008. ACM.

[70] Xin Liu. Towards context-aware social recommendation via trust networks. In *International Conference on Web Information Systems Engineering*, pages 121–134. Springer, 2013.

[71] Babak Loni, Martha Larson, and Alan Hanjalic. Collaborative filtering with factorization machines on implicit feedback data. *ACM Trans. Intell. Syst. Technol. (Under review)*, 3(10), September 2017.

[72] Babak Loni, Martha Larson, and Alan Hanjalic. Factorization machines for data with implicit feedback. *Submitted in ACM Trans. Inf. Syst.*, 2018.

[73] Babak Loni, Martha Larson, Alexandros Karatzoglou, and Alan Hanjalic. Recommendation with the right slice: Speeding up collaborative filtering with factorization machines. In *RecSys Posters*, 2015.

[74] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 361–364, New York, NY, USA, 2016. ACM.

[75] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems (To appear)*, RecSys '16, 2016.

[76] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Top-n recommendation with multi-channel positive feedback using factorization machines. *ACM Transactions on Information Systems (TOIS)*, 2018.

[77] Babak Loni and Alan Said. Wraprec: An easy extension of recommender system libraries. In *Proceedings of 8th ACM International Conference of Recommender Systems*, RecSys '14, 2014.

[78] Babak Loni, Yue Shi, Martha Larson, and Alan Hanjalic. Cross-domain collaborative filtering with factorization machines. In *Proceedings of the 36th European Conference on Information Retrieval*, ECIR '14, 2014.

[79] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[80] Trung V. Nguyen, Alexandros Karatzoglou, and Linas Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '14, pages 63–72, New York, NY, USA, 2014. ACM.

[81] Trung V. Nguyen, Alexandros Karatzoglou, and Linas Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '14, pages 63–72, New York, NY, USA, 2014. ACM.

[82] Weike Pan, Evan Xiang, Nathan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction, 2010.

[83] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, 73:173 – 180, 2015.

[84] Umberto Panniello, Michele Gorgoglione, and Cosimo Palmisano. Comparing pre-filtering and post-filtering approach in a collaborative contextual recommender system: An application to e-commerce. In *Proceedings of the 10th International Conference on E-Commerce and Web Technologies*, EC-Web 2009, pages 348–359, Berlin, Heidelberg, 2009. Springer-Verlag.

[85] Ladislav Peska and Peter Vojtas. Negative implicit feedback in e-commerce recommender systems. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*, page 45. ACM, 2013.

[86] Ian Porteous, Arthur Asuncion, and Max Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 563–568. AAAI Press, 2010.

[87] Runwei Qiang, Feng Liang, and Jianwu Yang. Exploiting ranking factorization machines for microblog retrieval. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, CIKM '13, pages 1783–1788, New York, NY, USA, 2013. ACM.

[88] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010.

[89] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[90] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[91] Steffen Rendle. *Context-aware ranking with factorization models*. Springer, 2011.

[92] Steffen Rendle. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.*, 3(3), May 2012.

[93] Steffen Rendle. Scaling factorization machines to relational data. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 337–348. VLDB Endowment, 2013.

[94] Steffen Rendle. Scaling factorization machines to relational data. In *Proceedings of the VLDB Endowment*, volume 6, pages 337–348. VLDB Endowment, 2013.

[95] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *ACM WSDM '14*, WSDM '14, pages 273–282. ACM, 2014.

[96] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM '14*, pages 273–282, 2014.

[97] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[98] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09*, pages 452–461, 2009.

[99] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. SIGIR '11. ACM, 2011.

[100] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[101] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[102] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[103] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: Benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 129–136, New York, NY, USA, 2014. ACM.

[104] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[105] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[106] Hendrik Schreiber. Improving genre annotations for the million song dataset. In *ISMIR*, pages 241–247, 2015.

[107] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. xCLiMF: optimizing expected reciprocal rank for data with multiple levels of relevance. In *ACM RecSys '13*, pages 431–434, 2013.

[108] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 155–164, New York, NY, USA, 2012. ACM.

[109] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 139–146, New York, NY, USA, 2012. ACM.

[110] Yue Shi, Martha Larson, and Alan Hanjalic. Tags as bridges between domains: improving recommendation with tag-induced cross-domain collaborative filtering. UMAP'11, 2011.

[111] Yue Shi, Martha Larson, and Alan Hanjalic. Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation. *Inf. Sci.*, 229:29–39, April 2013.

[112] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, May 2014.

[113] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, May 2014.

[114] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys*, To appear.

[115] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM.

[116] M. H. Soudkhah and R. Janicki. Weighted features classification with pairwise comparisons, support vector machines and feature domain overlapping. In *2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 172–177, June 2013.

[117] Harald Steck. Training and testing of recommender systems on data missing not at random. In *KDD '10*, pages 713–722, 2010.

[118] Yijun Sun. Iterative relief for feature weighting: Algorithms, theories, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1035–1051, June 2007.

[119] T.F. Tan and S. Neetessine. Is Tom Cruise threatend? using Netflix Prize data to examine the long tail of electronic commerce. Technical report, University of Pennsylvania, Wharton Business School, 2009.

[120] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 283–292, New York, NY, USA, 2016. ACM.

[121] Mikhail Trofimov, Sumit Sidana, Oleh Horodnitskii, Charlotte Laclau, Yury Maximov, and Massih-Reza Amini. Representation learning and pairwise ranking for implicit and explicit feedback in recommendation systems. 04 2017.

[122] Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1995–1999, New York, NY, USA, 2008. ACM.

[123] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.

[124] Sheng Wang, Xiaobo Zhou, Ziqi Wang, and Ming Zhang. Please spread: Recommending tweets for retweeting with implicit feedback. In *Proceedings of the 2012*

Workshop on Data-driven User Behavioral Modelling and Mining from Social Media, DUBMMSM '12, pages 19–22, New York, NY, USA, 2012. ACM.

[125] Shuai Wang, Mianwei Zhou, Geli Fei, Yi Chang, and Bing Liu. Contextual and position-aware factorization machines for sentiment classification. *CoRR*, abs/1801.06172, 2018.

[126] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 3119–3125. AAAI Press, 2017.

[127] Hong-jie Xing, Ming-hu Ha, Bao-gang Hu, and Da-zeng Tian. Linear feature-weighted support vector machine. *Fuzzy Information and Engineering*, 1(3):289–305, Sep 2009.

[128] M. Zanker and M. Jessenitschnig. Collaborative feature-combination recommender exploiting explicit and implicit user feedback. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 49–56, July 2009.

[129] Yu Zhang, Xiaomin Zhu, and Qiwei Shen. A recommendation model based on collaborative filtering and factorization machines for social networks. In *Proceedings of 5th IEEE International Conference on*, Nov 2013.

[130] Yu Zhang, Xiaomin Zhu, and Qiwei Shen. A recommendation model based on collaborative filtering and factorization machines for social networks. In *Broadband Network Multimedia Technology (IC-BNMT), 2013 5th IEEE International Conference on*, pages 110–114, Nov 2013.

[131] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: Tag informed collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 69–76, New York, NY, USA, 2009. ACM.

[132] C Lawrence Zitnick and Takeo Kanade. Maximum entropy for collaborative filtering. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 636–643. AUAI Press, 2004.

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

ACKNOWLEDGEMENTS

The last four years and few months that I spent to pursue my PhD was a wonderful journey and an exciting chapter in my life that gave me the opportunity to broaden my knowledge, deepen my understanding of science and technology, develop many soft and hard skills, experience working around the world, and make new friends and colleagues. Looking back to the path that led to this PhD thesis, I am truly indebted to many individuals who accompanied me on this journey, without whom this project would have not been possible.

First and foremost, I would like to express my sincere gratitude to my doctoral advisors Prof. Martha Larson and Prof. Alan Hanjalic who offered me the golden opportunity to work in TU Delft's MMC lab, and who tutored me not only on professional and scientific research but also on personal development. Thanks Martha for your continuous guidance, for your extraordinary support, for always being available to help me and for the challenging and constructive discussions that we had. Thanks Alan for your support, for your positivity and openness, and for your constructive feedback. It was a great pleasure for me to work with the two of you.

I would like to thank CrowdRec, the European FP7 project that financially supported my PhD, and to all project members for the admirable collaboration. I also owe a special thanks to the Dutch national E-Infrastructure and SURFsara for use of their high-performance computing service, so I could perform the many experiments needed for this PhD project.

Thank you to many researchers and professors who helped me with their advices during my PhD. To my external committee members Prof. Gert-Jan Houben, Prof. Marcel Reinders, Prof. Arjen de Vries, Prof. Jun Wang, and Prof. Joemon Jose who took ti me to review my dissertation and for their feedback on this manuscript.

To Saskia Peters and Robbert Eggermont for continuous administrative and technical support.

To Dr. Keki Burjorjee, Dr. Siddharth Patil, Dr. Tao Ye and Dr. Steve Essinger who gave me the great opportunity to work as scientific intern in Pandora Media in summer 2017. To my awesome colleagues and friends in Pandora specially Himan Abdollahpour, Jordi Pons and Jong Wook Kim with whom I really enjoyed the summer internship there. It was a great pleasure and a fabulous learning chance for me to be in such a vibrant environment full of brilliant people.

To Dr. Alexandros Karatzoglou for his advice and support during my visit at Telefonica Research.

To Kollekt.fm, the start-up company in Amsterdam, and to its founders, for the great collaboration and for sharing their data for my research.

To my wonderful colleagues and friends in the MMC lab for the great moments and memories. To Dr. Yue Shi who inspired me with his outstanding research on Recommender Systems. To Dr. Raynor Vliegendhart for his passionate explanations of concepts in computer science, for helping me learning Dutch and for his feedback on the Dutch summary of this dissertation. To Dr. Roberto Pagano for the great collaboration, for the lovely time in Boston and Palermo. To Dr. Alessio Bazzica for initiating the "deep (learning) guys" sessions, and for our memorable trip to Russia. To soon-to-be-doctor Karthik Yadati for the great and memorable discussions. To Soude for her friendly advice and for enthusiastically leading the organization of the RecSysNL (Recommender Systems community of the Netherlands) meetups. To Dr. Alan Said, whom I always call "the Wikipedia of the RecSys community", as I was always amazed by his detailed and up-to-the-point recall of everything that is related to RecSys, who inspired me with his passion about research on recommender systems, and whom I learned a lot from while he was in Delft. To Jay for the nice and inspiring discussions on deep learning and optimization problems. To Ernestasia, Xiuxiu, Manel, Zhe, Xinchao, Yi, Christina, Christoph, Cynthia, Huijuan and Julian for the useful discussions and nice moments.

To my friendly colleagues at De Persgroep with whom I shared the final stage of my PhD and from whom I am still learning. To Anne, Lucas, Jeroen, Vasco and Alen.

To Mahboobeh Goudarzi for making the cover of this book by elegantly visualizing my abstract idea.

I owe a true dept of gratitude to TU Delft. I enjoyed every moment that I spent in this great university. From the first day that I started my master study till today that I am leaving this university, I enjoyed this fantastic environment full of friendly people and was impressed by the excellent support on everything. I deeply appreciate the wonderful organization of this university, graduate school for offering constructive courses, international office for supporting international students, Promood and YoungDelft for extracurricular programs, sport and cultural center for offering various programs and hosting unique events, and many associations, specially DSZ-Wave for organization of the swimming activities where I could refuel my energy and relief my mind. Thanks to all people who made this university such a great place.

To my wonderful friends who helped me to keep the balance between work and life. To Hossein, Farhad, Siamak, Hasti, Alireza, Armaghan, Parsa, Masoud, Abdi, Poolad, Shekofeh, Bahar, Bahram, Ahura, Sara, Elahe, Darya, Fahad Irannejad, Aida, Farzaneh, Javad, Shaghayegh, Reyhan, Roberto, Mirko, Mattia, Nienke, Steven, Sasha, Gamze, Lino, Dorien, Amirali, Negin and Vahid.

To my dear uncle Ehsan for being the continuous source of inspiration and positivity during our little project Tezol.

To my sister Mina, who inspires me with her perseverance and for always being there for me. To my brother-in-law Amir, who makes me proud with his accomplishments. And to my cute nephews Bardia and Barad.

To my brother Afshin: you always make me very happy when you are around, and fascinate me with your endless knowledge about movies, books and technology. I wish I would had read a fraction of books that you have. You are an inspiration for me.

To my love, Mahshid, for her kindness, care and love. Thanks Mahshid, for the blissful moments that you share with me, for supporting me and keeping me calm in stressful moments. For making me a better person. For the happiness that you gave to me. And for many other reasons that are beyond the power of words to be expressed. I am very lucky that I have you.

And finally, thanks to my parents, Sodabeh and Mostafa, for their endless dedication and love. They are responsible for this book more than they can imagine. This book is dedicated to them.

Babak Loni
November 2018
Delft, The Netherlands

Curriculum Vitæ

Babak Loni was born on September 18th 1986, Tehran, Iran. From 2004 to 2009 he pursued a Bachelor of Science in Computer Science in Amirkabir University of Technology, Tehran, Iran. He continued his education by obtaining a MSc. in Computer Science from Delft University of Technology, Delft, the Netherlands in 2011. After a year of working as a Software Engineer in SDL, Amsterdam, he worked a Scientific Programmer at the Multimedia Information Retrieval (MIR) lab from Oct 2012 to Oct 2013. His research was mainly on the topic of "multimedia and crowdsourcing", and carried out in the context of the European FP7 project CubRIK.

In October 2013, he started a PhD in Computer Science in the Multimedia Computing Group (former MIR lab) at Delft University of Technology. His PhD topic was "Recommender Systems", and involved in the FP7 project CrowdRec, which was about crowd-powered recommendation for continuous digital media access. During his PhD, he spent three months as a Data Science intern in Pandora Media, Oakland, USA, and worked on music recommendation. He also visited Telefonica Research lab in Barcelona, Spain, where he worked on factorization models.

Babak has been working as a Machine Learning Engineer in De Persgroep, Amsterdam, Netherlands, since July 2018.