A CONTENT-BOOSTED MATRIX FACTORIZATION TECHNIQUE VIA USER-ITEM SUBGROUPS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EVIN ASLAN OĞUZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2014

Approval of the thesis:

A CONTENT-BOOSTED MATRIX FACTORIZATION TECHNIQUE VIA USER-ITEM SUBGROUPS

submitted by **EVIN ASLAN OĞUZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. Nihan Kesim Çiçekli _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Department, METU _____

Dr. Ayşenur Birtürk
Computer Engineering Department, METU _____

Dr. Ahmet Kara
TÜBİTAK BİLGEM, ILTAREN _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: EVIN ASLAN OĞUZ

Signature :

ABSTRACT

A CONTENT-BOOSTED MATRIX FACTORIZATION TECHNIQUE VIA USER-ITEM SUBGROUPS

Oğuz, Evin Aslan
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Nihan Kesim Çiçekli

December 2014, 67 pages

This thesis mainly focuses on improving the recommendation accuracy of collaborative filtering (CF) algorithm via merging two successful approaches. Since CF algorithms group like-minded users, a technique called Multiclass Co-Clustering (MCoC) is used in order to group like-minded users more effectively. Since, CF approaches lack incorporating content information, a content-boosted CF approach that embeds content information into recommendation process is used. In the MCoC, a user or an item can belong to zero, one or more subgroups. Thus, it is possible to predict the rating scores of users to items present in the same subgroup. However the prediction results for all users and items are not obtained by MCoC, since a user or an item may belong to zero subgroups. Therefore, content-boosted CF algorithm is applied to the whole set of users and items besides subgroups and finally the results are merged. The content-boosted approach, on the other hand, considers content information in the recommendation process. As content, the genres of movies are embedded into the item latent factor vector in the matrix factorization technique. To sum up, the content-boosted algorithm is applied to the subgroups and the whole set, and the obtained results are merged. Hence the recommendation accuracy is improved.

Keywords: Recommender Systems, User-item Subgroups, Content-boosted MF

ÖZ

İÇERİKLE ZENGİNLEŞTİRİLMİŞ GRUPLAMA İLE UYGULANAN MATRİS AYIRMA TEKNİĞİ

Oğuz, Evin Aslan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Nihan Kesim Çiçekli

Aralık 2014 , 67 sayfa

Bu tezin asıl amacı, işbirlikçi filtreleme algortimasıyla elde edilen önerilerin doğruluk derecesini, başarılı iki algoritmayı birleştirerek arttırmaktır. İşbirlikçi algoritmalar benzer kullanıcı grupları keşfedip, bu gruplar üzerinden öneri yaparlar. Bu tezde gruplar MCoC adı verilen bir gruplama tekniği ile keşfedilmektedir. Ayrıca, nesnelerin içeriğinin öneri sürecine katılması için, içerikle zenginleştirilmiş bir teknik kullanılmaktadır. MCoC tekniğinde, bir kullanıcı veya nesne sıfır, bir veya daha fazla altgruba dahil olabilir. Böylelikle, aynı altgruplara ait kullanıcı ve nesneler için o kullanıcıların sözkonusu nesnelere yapacağı tahmini değerlendirmeler elde edilmektedir. Ancak bir kullanıcı ve nesne hiç bir altgruba dahil olmadığında, işbirlikçi algoritma bütün kullanıcı ve nesneler için tekrar koşturulup elde edilen tahmini değerlendirmeler MCoC yöntemiyle elde edilemeyen değerlendirmeler için kullanılmaktadır. Diğer taraftan, içerikle zenginleştirilmiş işbirlikçi algoritma ise öneri sürecinde nesnelerin içeriğini göz önünde bulundurur. Film öneri sistemleri üzerinde çalışıldığından, içerik bilgisi olarak filmlerin türleri kullanılmaktadır. Özetle, içerikle zenginleştirilmiş işbirlikçi algoritma altgruplar ve bütün kullanıcı nesne seti üzerinde uygulanmakta, ve sonuçlar birleştirilmektedir. Böylece öneri sisteminin doğruluk derecesi arttırılmaktadır.

Anahtar Kelimeler: Öneri Sistemleri, Gruplama, İçerikle Zenginleştirilen matris ayırma

To my dear husband and my parents

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Prof. Dr. Nihan Kesim Çiçekli for her expert guidance and supportive, encouraging approach throughout my masters study and her effort and patience during the supervision of the thesis.

I would like to express my sincere appreciation to the jury members, Prof. Dr. Ahmet Coşar, Assoc. Prof Dr. Halit Oğuztüzün, Dr. Ayşenur Birtürk and Dr. Ahmet Kara for reviewing and evaluating my thesis.

I would like to thank my colleagues in TÜBİTAK BİLGEM / İLTAREN, who always encouraged me and shared their experiences with me.

I would like to thank TÜBİTAK BİLGEM / İLTAREN for supporting my academic studies.

I would like to thank my whole family, for making me who I am now with their love, reinforcement, teachings and encouragement throughout my life. Especially, I would like to thank my dad, who has been one step ahead of me to light my way; and my brother, Suat Serdar Aslan, who made my life easier with his experience in Mathematics during my masters study.

Finally my deepest thanks are to my husband, Hüseyin, for his endless patience, encouragement, support and delicious meals during this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

RS	Recommender System
CF	Collaborative Filtering
SVD	Singular Value Decomposition
MCoC	Multiclass Co-clustering
MF	Matrix Factorization
TMF	Traditional Stochastic Gradient Based Matrix Factorization
GMF	Traditional Stochastic Gradient Based Matrix Factorization via Subgroups
CMF	Content-boosted Stochastic Gradient Based Matrix Factorization
GCMF	Content-boosted Stochastic Gradient Based Matrix Factorization via Subgroups
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
MAP	Mean Avarage Precision

CHAPTER 1

INTRODUCTION

As the penetration of the Internet into human life and the technologies of managing, accessing and distributing the information change dramatically, the difficulty of locating the right information at the right time is increasing. In order to solve this problem, new research areas within computer science have emerged. The hierarchy illustrated in Figure 1.1 indicates the relationship between these proposed solutions [17]. Recommender systems is one of these solutions, predicting user preferences based on the analysis of the past behaviour of the user. They can be referred as information processing tools that extract relevant or significant aspects.

The main reason why the research interest in this area is still very high is, the practical significance of the problem of dealing with the information overload and providing them with personalized recommendations, content and the service. Hence, a number of online companies merged recommender systems with their services. Some examples of such systems include, product recommendation Amazon, product advertisement provided by Google based on the search history, movie recommendations by MovieLens, Netflix and Yahoo! Movies.

The recommendation problem, in its most common formulation, is reduced to the problem of predicting ratings for the items that a user has not seen before. Once a user's ratings for all unrated items are estimated, the items predicted to receive the highest ratings could be recommended.

Broadly speaking, recommender systems are designed based on the three different techniques, namely, collaborative filtering (CF), content-based filtering and the

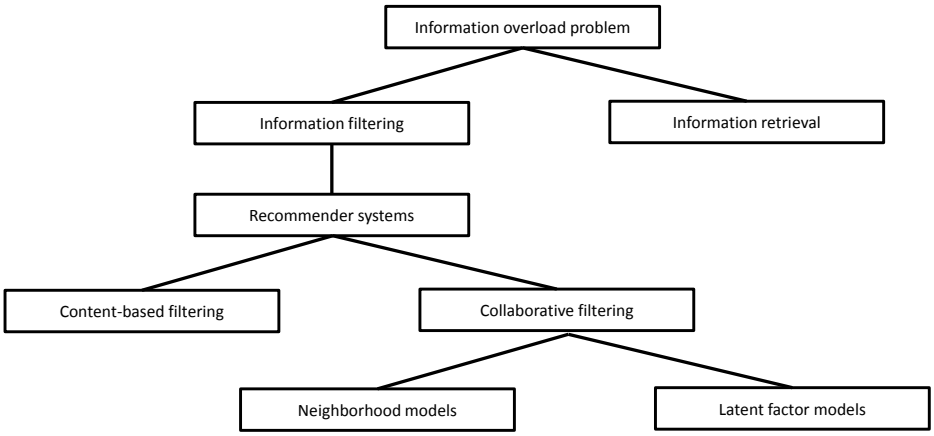


Figure 1.1: A hierarchy of proposed solutions to information overload problem.

hybrid approaches combining content-based and CF. The mostly preferred technique is the CF since, it offers a higher recommendation accuracy compared to content-based approaches, yet it lacks contributing content information into recommendation process that could raise recommendation success. The main idea of CF is generating recommendations based on the users' past behaviours such as, previous transactions and product ratings. Namely, a user is associated with a group of like-minded users and unseen items that are enjoyed by other users in the group are recommended to the user. The group of like-minded users usually consists of users that have rated items similarly. However, two users that belong to the same group, may have totally different tastes when some other item set is considered.

The neighborhood approach and latent factor models are the main branches of the CF. Having highly allusive ability of describing various aspects of data, latent factor models attract attention. Some of the successful implementations of latent factor models use Matrix Factorization (MF) technique [14]. The main idea of matrix factorization is to map both users and items to a shared low dimensional space. In that space high correspondence of users and items lead to recommendations. Yet, most recent matrix factorization techniques neither consider that users that are present in a group of like-minded users may not be in the same group if another item set is available, nor take the content information into account while generating recommendations.

1.1 Thesis Contributions

In this thesis we study recommendation in movie domain and address the following two problems:

1. Obtaining user-item subgroups
2. Embedding content information into recommendation process

Following the idea that users may have varying tastes when different item sets are available, we employ Multiclass Co-Clustering (MCoC) technique that is proposed by Xu et al. [30]. MCoC is a clustering technique, that clusters users and items into a predetermined number of subgroups. The user or item may belong to at most k subgroups which is lower than the total number of subgroups say c, i.e. $k < c$.

In this clustering technique a user or an item can belong to zero, one or more subgroups. These subgroups are used in order to predict a user's rating to items that share the same subgroup. The predictions are obtained by some CF algorithm and then they are merged by a framework that handles cases where a user or an item belong to zero, one or more subgroups. By this technique some, yet not all, prediction scores are obtained. An activity diagram that summarizes the basic flow of this technique is presented below in Fig. 1.2.

Secondly, since CF algorithms lack in terms of incorporating content information in the recommendation process we use the idea that is proposed by Chang et al. [11]. The idea is to embed content information into MF process. The attributes of items are utilized as the content information. More specifically, they use movies' genre information in order to get more accurate results. They embed this information into item latent factor vectors and and rearrange user latent factor vectors accordingly.

In our approach we merge these two solutions in order to get more accurate results. In other words, after discovering user-item subgroups via MCoC we employ the content-boosted matrix factorization, in order to get prediction results for these user-item subgroups. Since we do not get all predictions by using MCoC technique we propose to get prediction results for the whole user-item matrix without using

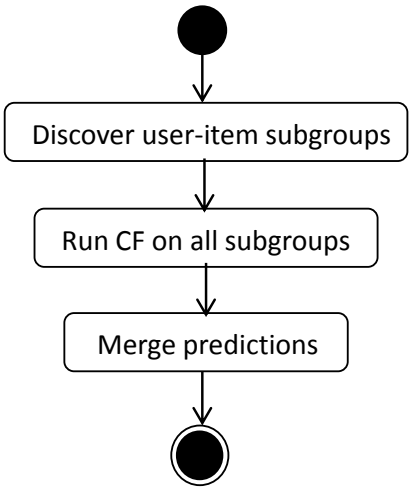


Figure 1.2: A basic activity diagram of MCoC.

user item subgroups, and obtain missing predictions that are not found by the MCoC technique. After that we merge all predictions that are obtained from subgroups and content-boosted CF for whole user-item rating matrix. An activity diagram that summarizes the basic flow of our approach is presented in Fig. 1.3.

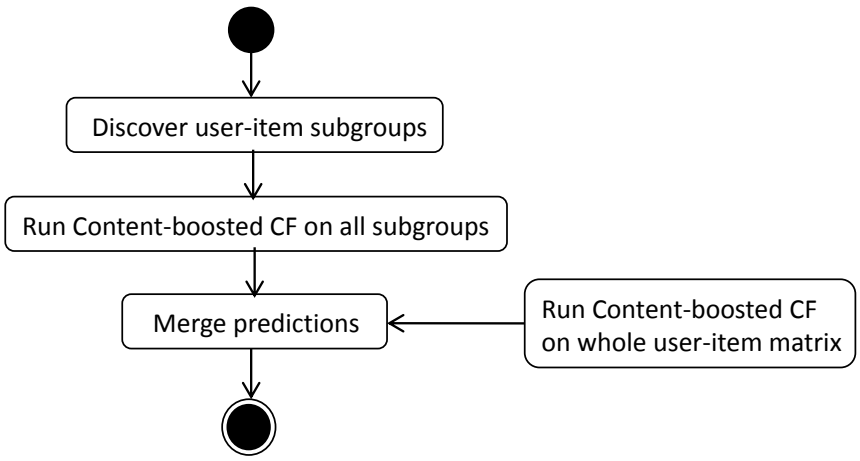


Figure 1.3: A basic activity diagram of our approach.

1.2 Outline

The rest of the thesis is organized as follows:

Chapter 2, gives background information about recommender systems. Gives detailed information about two widely used CF techniques, i.e. content-based filtering and collaborative filtering. Then MF technique, widely used latent factor based CF technique, is described. Three most widely used MF techniques are presented.

Chapter 3, presents user-item subgroups and content-boosted matrix factorization techniques. First these two algorithms are described in detail. Then we describe how the two algorithms are merged in this thesis.

Chapter 4, focuses on the evaluation and experiments. First, the dataset that is used throughout experiments is described. Then, used evaluation metrics are presented. After that, performed experiments are explained in detail. Finally, the results are presented.

Chapter 5, concludes our thesis. First the summary of our approach is presented. Then possible extensions that could be done in the future are explained.

CHAPTER 2

RELATED WORK AND BACKGROUND

A recommender system, in its most common formulation, can be referred as the estimation of the item ratings that have not been seen by a user. In a recommender system an item refers to an object that is recommended; and the user refers to individual to whom recommendation results are addressed [24]. The estimation of *ratings* usually depends on the previous ratings given to items by this user and some other information such as user features and context variables. Once ratings of yet unseen items are estimated, the item(s) with the highest estimated rating(s) can be recommended to the user.[2]

Shortly, the recommender systems can be described as software tools and techniques that suggest items that appeal users [24, 16, 23, 4]. There are a number of possibilities why recommender systems should be studied. Some of them could be stated as follows: [24]

- Increasing the number of items sold
- Selling more diverse items
- Increasing the user satisfaction
- Increasing user fidelity
- Understanding of what the user wants more clearly

The recommendation task could be accomplished in various ways, since recommendation applies many distinct areas, such as e-commerce, music, movie etc.

Apparently recommendation needs of these areas are diverse. Hence there are many approaches to deliver recommendations such as finding some good items, finding all good items, recommending a sequence and recommending a bundle (a group of items that fits well together are suggested) [24].

In order to solve this specific problem of recommendation; certain techniques are proposed. F. Ricci et al. [24] categorize these recommendation techniques into six classes, via a taxonomy provided by [24] and [4]:

1. Content Based
2. Collaborative Filtering
3. Demographic
4. Knowledge Based
5. Community Based
6. Hybrid Recommender Systems

Content Based: This system recommends items depending on the past activities of the user. In fact recommending new interesting items to the user is accomplished via a basic process, that mainly focuses on matching up the features of a user profile, in which interests and preferences are stored, with the features of an item. The system tries to recommend items which are "similar" to the ones a user liked in the past. The term "similar" can be quantitatively derived according to the domain of the items. For instance if recommendation is done in book domain since the author of Anna Karenina and War and Peace are same, a user who likes the first one 'may' like the second one also and vice versa. [21]

Collaborative Filtering: Schafer et al. [25] refers to collaborative filtering as "people-to-people correlation". As it can be deduced from the phrase this type of system recommends a particular user items that other users with similar tastes liked in the past. The similarity in the rating history of the users are considered, while determining the similarity in taste of users. In other words, this approach tries to find 'peers' of users that share similar tastes in the specified domain. After that, items that are mostly liked by the 'peers' of the particular user are recommended [20].

Demographic: This system mainly focuses on the demographic information of the users. The key idea in this system is to generate different recommendations for different demographic niches. That is to say, demographic profile of the user is utilized to recommend items. For instance, F. Ricci et al. [24] states that effective personalization solutions are offered based on demographic profiles of the users in many web sites, such as dispatching web sites according to the language or country of a particular user. Also customizing recommendations according to age can be shown as another example. This approach is more popular in marketing literature compared to recommender systems, therefore there are relatively little research in RS.

Knowledge Based: In this approach specific domain knowledge is used while recommending items. The concepts of how certain items features meet user needs preferences and, how the item is useful for the user are examined carefully in this kind of system. Knowledge-based systems are likely to work better than other recommender systems at the beginning of their deployment, if equipped with learning components. However, they may be overshadowed by other shallow methods that can take advantage of the logs of the human/computer interaction (as in CF).

Community Based: This kind of system is mainly interested in preferences of the friends of the users while recommending items. This technique is closely related with the epigram "Tell me who your friends are, and I will tell you who you are" [3]. It has been suggested that, people prefer to rely on recommendations from their friends more than the recommendations comes from anonymous people [24]. Since social network popularity is increasing nowadays, making use of friends information makes sense. Still, the research in this type of recommender system is in its early phase.

Hybrid Recommender Systems: This type of recommendation system is based on the combination of early mentioned recommender systems. The idea behind using combination of techniques is basic. For instance; if a hybrid system uses techniques A and B, then this system tries to use advantages of A in order to diminish the disadvantages of technique B and vice versa.

It is possible to give a concrete example from the literature. For instance, collaborative filtering methods lack suggesting items to the newly presented users since

they have no ratings at first. This is known as cold start problem. Content based approaches, on the other hand, does not need rating information of the item. Because it makes use of features of an item for recommendation. Hence combining these two techniques could solve the new item problem that is present in collaborative filtering.

Since our focus will mainly be on Content Based Recommendation and Collaborative Filtering, they will be explained in detail below is Section 2.1 and 2.2.

2.1 Content Based Recommendation

Content based recommendation methods estimate the utility of an item i, by the help of the utilities assigned by the user to items which are similar to i. For instance while recommending movies to a user, the system tries to identify common features of items that the user gave a high rating score in the past (these common features could be subject matter, genres, directors, specific actors etc.). After that the movies that have a high degree of similarity to the user's preferences are recommended [2].

Profile based information retrieval methods have significant effect on most of the content based recommendation techniques [2]. In profile based approaches each user is represented by a profile containing information about the taste and preference of the user. This information is obtained from the user explicitly via questionnaires through signing up for the service or, learned implicity from transactional behaviour. Similarly each item is represented by a profile that is gathered from the item descriptions or item features. Usually the item profiles are defined by keywords, that represents a set of most important words associated with the item. In order to determine the importance of a keyword k_i for an item d_j a weighting measure w_{ij} is identified.

There are many definitions of the weighting measure w_{ij}. In information retrieval, the most used measure is the *tf * idf* measure, which stand for *term frequency * inverse document frequency*. The importance of a keyword in a set is represented by *tf * idf* measure. As it can be seen it is the product of two measures *term frequency* and *inverse document frequency*. The *term frequency* stands for the number of times a keyword appears for the corresponding item, that is to say frequency of the keyword in item description. The *inverse document frequency* stands for the relevance of the

keyword in the set of all items. The *idf* is identified as the ratio of the number of items containing the keyword to the number of all items. Shortly *tf * idf* of a keyword *k* in an item *d* could be calculated by the following equation [2, 24].

$$tf * idf(d,k) = tf(d,k) * log(\frac{|D|}{df(k)})$$

Here |*D*| refers to number of items present in the set and *df(k)* is the number of items in the set which the keyword *k* appears. Normalization is applied in order to balance different factors. It is possible to use other normalization techniques and motives.

As mentioned earlier, content based systems recommend items similar to the ones preferred by a user. In order to represent past preference of a user a vector of keyword weights is constructed via keyword analysis techniques stated above. The weight vector is matched with the item vectors of unrated items in order to compute the utility score *u(c,d)*. According to the utility score most similar items are chosen for recommendation. The similarity is measured based on correlation and association coefficients used in information retrieval and cluster analysis. Association coefficients are widely used. Cosine coefficient is the most commonly used measure. So as to calculate the cosine coefficient between a user profile vector *a* and an item profile vector *d* the calculation below is used.

$$Cos(t_a,t_d) = \frac{t_a.t_d}{|t_a||t_d|}$$

Here t_a and t_d are representing the *tf*idf* of the user and the item vector respectively. Cosine measure is one of the many variants of the inner product similarity measure such as inner product measure, dice measure and pseudo-cosine measure. It is a version where Euclidean length normalization technique is employed and it comes under the association coefficients.

Correlation coefficient is derived from Pearson product moment correlation coefficient which is used widely in statistics. Correlation coefficient is calculated by the standard formula stated below

$$P(a,d) = \frac{Cov(a,d)}{\sigma_a \sigma_d}$$

Here *Cov(a,d)* represents the covariance between a & d and σ represents the standard deviation between the data set a and d.

There are several limitations of content based models such as overspecialization, limited content analysis and new user problem, even though they are widely used [2, 21].

Overspecialization

Recommender systems are expected to recommend a range of items to the users. If not, a user is limited to be recommended items that are similar to those rated in the past. For instance, a user with no experience with the French cuisine would never receive a recommendation for trying even the greatest French cuisine in the neighborhood. This problem is overcame by incorporating some randomness to the user profiles.

Besides, the problem with overspecialization is not only recommending different items that user has never seen before, but also recommending too similar items that a user has already seen. For example, the user should not be recommended two different news article mentioning the same event. To sum up, the diversity of recommendations is desirable, since the users should not be presented a homogenous set of items, instead a range of options.

Limited Content Analysis

Content based techniques are limited by the explicit keywords or features associated with the objects that are going to be recommended. Hence effectiveness of these keywords or features is of vital importance for each item in the set. So as to acquire these explicit keywords or features either the content must be in a form that can be parsed automatically or it must be assigned through an interactive task by users, assigning the features or keywords manually from a list of most commonly used tags. While the first method is trivial by making use of information retrieval techniques, the other one is a challenging tasks since users are usually are not interested in providing other than minimum required data. Thus the text documents are easy to cope with since automatic extraction techniques applies. On the other hand, domains like multimedia data, graphical images, audio streams and video streams have an intrinsic problem

with automatic feature extraction.

The other problem with limited content analysis is that, if two distinct items are associated with the same set of features or keywords, it will be hard to distinguish these two items. For instance, since text based documents are usually associated with their most important keywords, content based systems will lack in terms of distinguishing between a well written article and a badly written one, if they have the same set of keywords.

New User Problem

This problem is also known as *cold start problem* in all types of recommendation systems. As mentioned before content based recommenders recommend items to users according to the past behaviour of the users. Therefore the user has to rate a sufficient number of items before the recommender system infers user preferences and interests and generate recommendations accordingly. Hence a new user who has rated a few items is not going to be presented non-trivial recommendations.

2.2 Collaborative Filtering

In contrast with content based recommendation methods, collaborative filtering methods try to predict the adequacy of items for a user by making use of past behaviour of *other users* that are similar to the user, as it is illustrated by Mortensen [17] in Figure 2.1. For instance, in a movie recommender system the recommendations presented to a user is based on the "peers" of the user that is found by the collaborative recommender system. Collaborative recommender systems try to find users who have similar taste in movies, i.e. rate the same movies similarly, in order to assign peers of a user. Thus, the user will only be suggested to the movies that are most liked by the "peers" of the user.

There are two primary approaches for collaborative filtering systems:

- Neighborhood Models
- Latent Factor Models

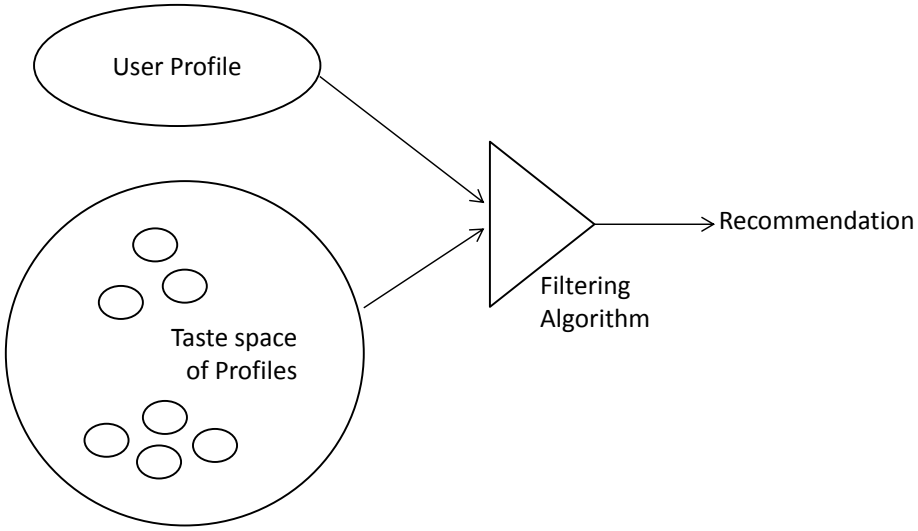


Figure 2.1: Collaborative Filtering.

Neighborhood models are the most common approach to collaborative filtering. The main idea of this approach is using the similarity of users or items. For example, two users are considered to be similar if they have rated the same set of items similarly.

Latent factor approaches, on the other hand, represents users and items as vectors in the same latent factor space through a reduced number of hidden factors. Therefore, users and items could be compared directly such that, the rating of a user to an item is predicted by the proximity (inner product for instance) between the related latent factor vectors [5].

Collaborative recommender systems do not face shortcomings that are encountered in content based recommender systems, as these systems use other users ratings and they are able to recommend items independent from their contents. Yet collaborative recommender systems have their own limitations such as sparsity, new user problem and new item problem [2].

Sparsity

In any recommender system the number of items that need to be predicted is usually much more than the number of items for which ratings have already been acquired. Therefore it is important to generate effective predictions from a small number of

known ratings. Likewise, a collaborative recommender systems's success depends on whether a critical mass of users is available or not. For instance, considering movie recommendation system, there may be many movies that are rated by only a few users. Even though they get high scores from only a few users, these movies have a smaller chance to be recommended. Further, the systems will not be able to find peers of the users whose tastes are unusual when compared to the rest of the users, leading to poor recommendations.

One way to deal with the problem of sparsity is to incorporate user profile information while calculating similarity of users [22]. Namely, two users could be considered similar if they belong to the same demographic segment, besides having rated the same movies similarly. The other way to address the problem of sparsity is to use associative retrieval frameworks in order to explore transitive associations among users through their past activities and feedback [10].

New User Problem

This problem resembles to the problem in content based recommender systems, since the systems must learn the user's preferences and interests to give adequate recommendations.

Several techniques proposed in order to solve this problem and most of them use *hybrid* recommendation through combining collaborative and content based recommender systems. An alternative approach is to make a new user to rate a set of items that is prepared before. Therefore a number of rating will be available for the system to generate successful recommendations [2].

New Item Problem

New items are always added to recommender systems. Since collaborative systems are based solely on the user's preferences, the system will not be able to recommend the newly added items unless they are rated by a substantial number of users. Hybrid approaches are addressed in order to solve this problem.

2.2.1 Neighborhood Models

The predictions done by neighborhood models are based on the relationships of similar users or items.

Algorithms based on the user-user similarity predict rating of an item for a particular user according to the ratings expressed to the same item by the similar users of this particular user. Algorithms based on the item-item similarity, on the other hand, compute the user preference for a particular item according to the ratings of the user given to similar items in the past. Item-item similarity is more preferable since number of items are typically smaller when compared to the number of users [5].

The crucial part of the neighborhood model is the similarity computations. Therefore mechanisms such as Pearson Correlation and Vector Cosine Similarity are used while detecting the similarity of the users or items [28, 26].

2.2.2 Latent Factor Models

Latent factor models predict the rating by characterizing both users and items on for example 5 to 100 factors that are inferred from rating patterns. For example, if movie domain is concerned; the meaning of factor changes according to the users and items (movies). For users, each particular factor measures how much the user likes movies which have a high score on the corresponding movie factor. For movies, the discovered factors may measure obvious dimensions such as horror versus drama, orientation to children or adults, amount of action; non-trivial dimensions such as strangeness or depth of a character development; or entirely un-interpretable dimensions [14].

Matrix factorization methods play a key role in some of the most successful realizations of latent factor models [14], as well as SVD methods. These techniques are discussed in detail in section 2.3.

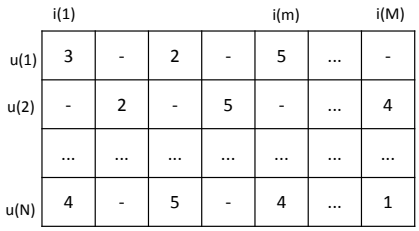


Figure 2.2: User-item rating matrix.

2.3 Matrix Factorization Techniques

Matrix factorization technique, being a hot research topic, has been studied by many researchers [12, 8, 6, 31, 9, 13].

Matrix factorization, in its basic form, characterizes users and items through vectors of factors inferred from item rating patterns. In fact recommendations are elicited by a high correspondence between user and item factors. As a result of combining both good scalability and predictive accuracy, these methods have become very popular in recent years. Furthermore, they are very flexible in terms of modeling various real life situations.

Recommender systems are based on different types of input data. The input data is often placed in a matrix with one dimension representing users and other dimension representing items of interest. High quality *explicit feedback*, including explicit input of users regarding their interests in items, is the most convenient data for the recommender system. One example of *explicit feedback* may be shown as star rating that Netflix collects for movies. The term explicit feedback is going to be referred as *ratings* from now on. Since any user probably rate a small percentage of the available items, the ratings will compose a sparse matrix. An illustration of user-item rating matrix can be seen in Figure 3.6.

One strength of matrix factorization is that additional information could easily be incorporated to matrix factorization. For example, when explicit feedback is not available, user preferences can be inferred by recommender systems from user's behaviour, such as browsing history, search pattern, purchase history etc. Since implicit feedback is based on presence or absence of an event, the representation is going

to be a densely filled matrix.

There are three well known techniques of matrix factorization. They are stochastic gradient descent, alternating gradient descent and singular value decomposition. Stochastic gradient descent and alternating gradient descent are based on the same model which is referred as "Basic Matrix Factorization Model". However, singular value decomposition uses a model which is a little bit different from these techniques. Therefore first the notion of "Basic Matrix Factorization Model" is explained, then stochastic gradient descent and alternating gradient descent models are presented, and finally singular value decomposition is discussed.

Basic Matrix Factorization Model

Matrix factorization models are based on mapping users and items to a joint latent factor space of dimensionality f to model user item interactions as inner products in that space [18].

Let r_{ui} denote the rating of user u to item i, for a given set of users $U = \{u_1, ..., u_N\}$, and set of items $I = \{i_1,..., i_M\}$. These ratings compose a user-item matrix, $\mathbf{R} = [r_{ui}]_{N \times M}$. The value of r_{ui} is usually a binary value indicating "like" and "dislike" or a certain range containing integer values, indicating different levels of preferences (e.g., $r_{ui} \in \{1, ..., 10\}$), even though it can take any real valued number.

Since the rating matrix $\mathbf{R}$ is highly sparse because of unknown entries, the set of known indices is denoted as follows:

$$\mathrm{T} = \{(u,i) : r_{ui} \text{ is known}\}$$

The goal of the recommender system is to predict the rating, $\hat{r}_{ui}$, that user u would give to item i, for an unknown user item pair. Moreover let $\mathrm{T}_{u.}$ be the set of items that have been rated by user u, and $\mathrm{T}_{.i}$ be the set of users who rated item i be denoted as follows:

$$\mathrm{T}_{u.} \equiv \{i : (u,i) \in T\}$$

$$\mathrm{T}_{.i} \equiv \{u : (u,i) \in T\}$$

As mentioned above, matrix factorization uses all known ratings to decompose the rating matrix $\mathbf{R}$ into the dot product of two low-rank latent feature matrices in order to predict unknown ratings in $\mathbf{R}$. User latent features are represented by $\mathbf{P}_{N \times K}$, and item features are represented by $\mathbf{Q}_{M \times K}$

$$\mathbf{R} \approx \hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T = \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ . \\ . \\ . \\ \mathbf{p}_N \end{bmatrix}}_{N \times K} \underbrace{\begin{bmatrix} \mathbf{q}_1{}^T & \mathbf{q}_2{}^T & \mathbf{q}_3{}^T & \cdots & \mathbf{q}_M{}^T \end{bmatrix}}_{K \times M} \tag{2.1}$$

Each user is associated with a vector $p_u \in \mathbb{R}^f$ where (u = 1,2,...,N), and each item is associated with a vector $q_i \in \mathbb{R}^f$ where (i = 1,2,...,M). These feature vectors are K dimensional vectors, where $K \ll \min\{M,N\}$. For a user u, p_u represents the amount of interest that the user has in items that are high on the corresponding factors either positive or negative. Likely, for an item i, the elements of q_i represents the amount that item possesses those factors, again either positive or negative. The result of the dot product $p_u {q_i}^T$ captures the user u's overall interest in the item i's characteristics. This result, is an approximation of user u's rating for the item i, denoted by r_{ui}, leads to following estimation.

$$\hat{r}_{ui} \;=\; p_u {q_i}^T \tag{2.2}$$

The main challenge is the mapping of each item and user to factor vectors $p_u,\ q_i \in \mathbb{R}^f$. After this mapping is completed by the recommender system the rating a user will give to any item can easily be estimated by using Equation 2.1.

The factorization of Equation (2.1) can be achieved by solving the following optimization problem, mathematically:

$$\min_{\mathbf{P},\mathbf{Q}} \|\mathbf{R} - \mathbf{P}\mathbf{Q}^T\|^2 \tag{2.3}$$

where $\|.\|$ is the Frobenius norm. Also it is common to include a regularization penalty on the sizes of **P** and **Q** in order to prevent over-fitting. After adding regularization penalty above equation turns into the following:

$$\min_{\mathbf{P},\mathbf{Q}} \|\mathbf{R} - \mathbf{P}\mathbf{Q}^T\|^2 + \lambda(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2) \tag{2.4}$$

Moreover it has been argued that [18] one should always penalize $\|\mathbf{p}_u\|^2$ and $\|\mathbf{q}_i\|^2$ by different amounts. Therefore the equation becomes:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i) \in T} (r_{ui} - \mathbf{p}_u {\mathbf{q}_i}^T)^2 + \lambda \left(\sum_u \|\mathbf{p}_u\|^2 + \gamma \sum_i \|\mathbf{q}_i\|^2 \right) \tag{2.5}$$

To sum up, recommender system tries to minimize the regularized squared error on the set of available ratings in order to learn the user and item factor vectors p_u and q_i*)*.[18]

Two approaches minimizing Equation 2.5 are stochastic gradient descent and alternating gradient descent (or alternating least squares - *ALS*).

2.3.1 Stochastic Gradient Descent

This algorithm loops through all known set of ratings in order to optimize the equation 2.4. For each known tuple, the systems predicts r_{ui} and computes the prediction error.

$$e_{ui} = r_{ui} - p_u {q_i}^T \tag{2.6}$$

After that, parameters are modified by a magnitude proportional to γ in the op-

posite direction of the gradient as follows [14]:

$$p_u \leftarrow p_u + \gamma.(e_{ui}.q_i - \lambda.p_u) \tag{2.7}$$

$$q_i \leftarrow q_i + \gamma.(e_{ui}.p_u - \lambda.q_i) \tag{2.8}$$

2.3.2 Alternating Gradient Descent

Equation 2.4 is not convex, since both p_u and q_i are unknowns. However, if one of the unknowns are fixed, the optimization problem becomes quadratic. Hence it can be solved optimally.

Therefore alternating gradient descent techniques rotate between fixing the p_u's and fixing the q_i's. When all q_i's are fixed, the system recomputes the p_u's by solving the optimization problem, and vice versa.

Let $\bigtriangledown_u^{BL}$ denote the derivative of L_{BL} with respect to p_u, and similarly $\bigtriangledown_i^{BL}$ denote the derivative of Eq. (2.5) with respect to q_i. Then [18],

$$\bigtriangledown_u^{BL} \propto \sum_{i \in T_{u.}} -(r_{ui} - \mathbf{p}_u \mathbf{q}_i{}^T)\mathbf{q}_i + \lambda \mathbf{p}_u \tag{2.9}$$

$$\bigtriangledown_i^{BL} \propto \sum_{i \in T_{i.}} -(r_{ui} - \mathbf{p}_u \mathbf{q}_i{}^T)\mathbf{p}_u + \lambda\gamma \mathbf{q}_i \tag{2.10}$$

for every u = 1,2,...,N and i = 1,2,...,M. At iteration (k+1), the updating equations for $\mathbf{p}_u$ and $\mathbf{q}_i$ is as follows:

$$\mathbf{p}_u{}^{(k+1)} = \mathbf{p}_u{}^{(k)} - \eta \bigtriangledown_u^{BL} \left(\mathbf{p}_u{}^{(k)}, \mathbf{q}_i{}^{(k)}\right) \tag{2.11}$$

$$\mathbf{q}_i{}^{(k+1)} = \mathbf{q}_i{}^{(k)} - \eta \bigtriangledown_i^{BL} \left(\mathbf{p}_u{}^{(k)}, \mathbf{q}_i{}^{(k)}\right) \tag{2.12}$$

where η is the step size or learning rate [18].

2.3.3 Singular Value Decomposition

Singular value decomposition is also a matrix factorization technique whose key idea is to factorize user-item rating matrix to a product of two lower rank matrices **P** and **Q**.

In addition, SVD employs another matrix *D*, which is a diagonal matrix of rank *K*, in order to design *P* and *Q* to be orthogonal, i.e. $P^T P = I$ and $Q^T Q = I$. Therefore, Eq. (2.3) becomes the following [18]:

$$\min_{\mathbf{P},\mathbf{Q}} \|\mathbf{R} - \mathbf{PDQ}^T\|^2 \tag{2.13}$$

Hence, SVD tries to minimize equation (2.13).

In this thesis Stochastic Gradient Descent technique is used.

CHAPTER 3

CONTENT-BOOSTED COLLABORATIVE FILTERING AND USER-ITEM SUBGROUPS

In order to improve the recommendation accuracy of recommender systems in movie domain we aim to merge two successful algorithms, i.e. MCoC and Content-boosted matrix factorization technique by Xu et al. [30] and by Chang et al. [11] respectively. We show that merging these two successful approaches result in better prediction results. In this chapter we present these two algorithms in detail and then present the proposed process of merging the two.

3.1 USER-ITEM SUBGROUPS

Recommender systems based on collaborative filtering, typically associate a user with a group of like-minded users according to their preferences over all of the items. Then the items enjoyed by others in the group, that is not seen by the user, is recommended. Mainly, it is presumed that users with similar behaviour (e.g. item preferences), will have similar tastes on all the items. However, it is stated by Xu et al. [30] that this assumption is not always credible since, two users that have similar tastes on one item subset may have completely distinct tastes if another item subset is in question. Furthermore, they say that one user's interest is usually not dispersive over all items, rather it is focused on some topics. Hence, it can be said that a group of users are having similar tastes, on a subset of items. Therefore, in the rest of the thesis a subset of items and a group of users interested in these items is referred as a **user item subgroup**. It is expected that subgroups will support capturing similar user tastes

when a subset of items considered.

A cluster is defined as a collection of data samples that have close relationships or whose features are similar. Clustering is widely used as an intermediate process since obtained clusters are used for further analysis in collaborative filtering algorithms. Lots of collaborative filtering models that use clustering, exploit item clusters, user clusters or co-clusters while designing algorithms [7, 30]. In all of these models, a user or item is presumed to be in only one cluster. However, in reality it is more likely for users or items to belong multiple clusters or subgroups.

There are several distinct categories of clustering methods as illustrated in Figure 3.1 [30]. Partitioning the users into several distinct groups may be the most straightforward way for clustering. There are several approaches for partitioning users. The user set may be clustered according to user-user similarity and clusters may be used as neighborhoods. On the contrary, user rating data could be used in order to cluster items or k-means or Gibbs sampling could be used in order to partition users and items separately. Then users can be re-clustered based upon the number of items in each item cluster they rated and similarly items can be clustered based upon the number of users in each user cluster that rated them. All of these algorithms can be referred as *one-sided* since they only make use of either users or clusters while partitioning. Such partitioning algorithms are illustrated in Figure 3.1.a and Fig 3.1.b.

There are some other algorithms that use *two-sided* clustering model. These algorithms can be referred as co-clustering based collaborative filtering models, since they use co-clustering techniques traditionally. For instance, user and item neighborhoods may be obtained by co-clustering and then predictions may be generated based on the average ratings of co-clusters. Distinct co-clusters with both users and items can be generated by some row and column changes as seen in Figure 3.1.c.

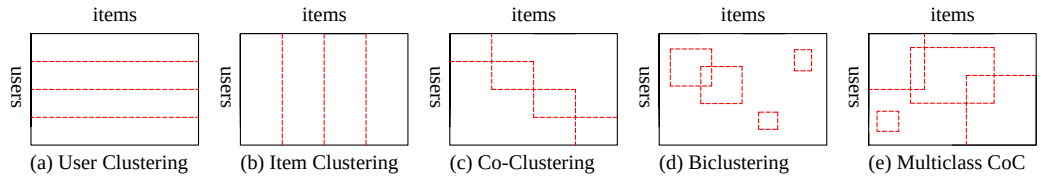


Figure 3.1: Clustering methods.

The limitation of both one-sided and two-sided (co-clustering) algorithms is that, a user or an item can belong to only one cluster. However it may be beneficial for recommender systems to provide the ability of clustering users and items in such a way that a user or an item can belong to several clusters. For instance, in movie domain, a movie can be popular in distinct groups from different aspects or, a user may like multiple movie topics. Hence the multiclass co-clustering (MCoC) model, which is shown in Figure 3.1.e, makes more sense by allowing users and items to exist in multiple subgroups at the same time [30].

Another clustering model which is named as biclustering model, which is seen in Figure 3.1.d. and which is mostly studied in gene expression data analysis [15, 19]. The MCoC and bicluster model may seem alike, however their main difference is that, in contrast to MCoC, in bicluster model all of the rows and columns may not be covered since this model tries to find some maximum biclusters with low residue scores. Therefore we will be focusing on the MCoC model.

3.1.1 Xu's Algorithm

The primary goal is to find possible user item subgroups and use these subgroups to enhance the performance of collaborative recommender systems. There are two main issues to consider in this algorithm. The first issue is how to cluster users and items into meaningful user item subgroups via the user item rating matrix we have. The second is how to utilize these subgroups with existing collaborative filtering algorithms.

The algorithm by Xu et al. [30] defines a way to deal with these two issues. They use MCoC to find meaningful subgroups and present a strategy to combine these subgroups with collaborative filtering systems.

3.1.2 MCoC Problem Formulation

Suppose T is a user item rating matrix and $T\ in\ ^{nxm}$ where n is the number of users and m is the number of items. Let each element of T_{ij} denote the preference of user i to item j and let $\mathbf{u}_i$ denote i-th user and $\mathbf{y}_j$ denote j-th item.

The main goal is to divide the users and items into c subgroups simultaneously, where a user or an item can belong to multiple subgroups. Therefore it is called as multi-clustering problem or shortly MCoC.

They represent MCoC result as a partition matrix $P \in [0,1]^{(n+m)\times c}$. Each element of P that is represented by P_{ij}, indicates whether a user or an item i belongs to subgroup j. If a user or an item i belongs to subgroup j then $P_{ij} > 0$, if not then $P_{ij} = 0$. The magnitude of each element indicates the relative weight of a user's or an item's i belonging to subgroup j, hence each row of partition matrix P sums to 1. If the number of groups that each user or item can belong to is fixed, which is a number that is smaller than the number of subgroups say k ($1 \leq k \leq c$), then there are exactly k non-zero values in each row of P. If k is fixed to be 1 ($k = 1$) then the clustering problem turns into traditional Co-Clustering problem. The matrix P is designed to be as follows:

$$P = \begin{bmatrix} Q \\ R \end{bmatrix} \tag{3.1}$$

where $Q \in [0,1]^{n\times c}$ represents the partition matrix of users and $R \in [0,1]^{m\times c}$ represents the partition matrix of items.

3.1.3 MCoC Solution

A very simple yet reasonable solution is proposed to solve MCoC problem. They follow the intuition that if a user and an item have a high rating score then they are expected to occur in one or more subgroups together. They employ the following loss function for modeling user-item relationships, so as to make the strongly associated users and items together.

$$\epsilon(Q,R) = \sum_{i=1}^{n}\sum_{j=1}^{m}(\| \frac{q_i}{\sqrt{D_{ii}^{row}}} - \frac{r_j}{\sqrt{D_{jj}^{col}}} \|^2 T_{ij}) \tag{3.2}$$

where $\mathbf{q}_i$ represents the i-th row of Q and $\mathbf{r}_j$ represents the j-th row of R. $D^{row} \in \mathbb{R}^{n\times n}$ stands for the diagonal matrix of users and $D^{col} \in \mathbb{R}^{m\times m}$ stands for the diagonal

matrix of items where $D_{ii}^{row} = \sum_{j=1}^{m} T_{ij}$ and $D_{jj}^{col} = \sum_{i=1}^{n} T_{ij}$. Because of the fact that the loss function is based only on the user-item interaction information, minimizing Eq. (3.1) means that the user vector q_i and item vector r_j should be very close if they have a high rating score. Furthermore, there is no restriction to missing user-item rating pairs. By some linear algebra derivations the loss function could be rewritten as follows:

$$\begin{aligned}
&\epsilon(Q, R) \\
&= \sum_{i=1}^{n} \| q_i \|^2 + \sum_{j=1}^{m} \| r_j \|^2 - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{2 q_i^T r_j T_{ij}}{\sqrt{D_{ii}^{row} D_{jj}^{col}}} \\
&= Tr(Q^T Q + R^T R - Q^T S R) \\
&= Tr\left(\begin{bmatrix} Q^T & R^T \end{bmatrix} \begin{bmatrix} I_n & -S \\ -S^T & I_m \end{bmatrix} \begin{bmatrix} Q \\ R \end{bmatrix} \right) \\
&= Tr(P^T M P),
\end{aligned} \tag{3.3}$$

where

$$S = (D^{row})^{-\frac{1}{2}} T (D^{col})^{-\frac{1}{2}}, \; M = \begin{bmatrix} I_n & -S \\ -S^T & I_m \end{bmatrix}. \tag{3.4}$$

$I_{n \times n}$ and $I_{m \times m}$ represent identity matrices of size $n \times n$ and $m \times m$ respectively. Because of the several strong constraints on partition matrix P, that are described in Section 3.1.2, solution of the loss function means solving the following optimization problem.

$$\begin{aligned}
\min_{P} \quad & Tr(P^T M P) \\
\text{s.t.} \quad & P \in \mathbb{R}^{(m+n) \times c}, \\
& P \geq 0, \\
& P \mathbf{1}_c = 1_{m+n} \\
& |P_i| = k, i = 1, ..., (m+n).
\end{aligned} \tag{3.5}$$

Each element of P is forced to stay in the range of [0,1] by the help of the combined constraints $P \geq 0$ and $P\,\mathbf{1}_c = \mathbf{1}_{m+n}$, where c is the total number of subgroups and k is the number of subgroups that a user or an item can belong to ($1 \leq k \leq c$). $|.|$ stands for the cardinality constraint that is the number of non-zero elements in a vector. P_i represents the i-th row of the matrix P.

Due to the fact that Eq. (3.5) is nonconvex and discontinuous, its solution is not straightforward. However, Xu et al. [30] proposed an efficient approximation method in order to solve this problem. Their method consists of two primary stages. The first stage is to map all users and items into a shared low dimensional space. The second stage is discovering subgroups.

First Stage: Map users and items to low dimensional space.

It is suggested that optimal r-dimensional embedding X^* can be obtained by solving the following problem, that preserves the user-item preference information meanwhile.

$$\begin{aligned} \min_{X} \quad & Tr(X^T M X) \\ \text{s.t.} \quad & X \in \mathbb{R}^{(m+n)\times c}, X^T X = I. \end{aligned} \tag{3.6}$$

where arbitrary scaling of X is preserved by the constraint $X^T X = I$. They state that it can be inferred from Eq. (3.2) and Eq. (3.3) that M is a semidefinite matrix. The optimal solution X^* that minimizes (3.6) is the same as the solution of eigenvalue problem $MX = \lambda X$. As a result $X^* = [\mathbf{x}_1, ..., \mathbf{x}_r]$. $\mathbf{x}_1, ..., \mathbf{x}_r$ represent eigenvectors of matrix M that is sorted by their corresponding eigenvalues.

Second Stage: Discover subgroups.

After mapping users and items to the shared low dimensional space, the partition matrix P is needed to be computed in order to find user-item subgroups.

A user or an item can belong to one or more subgroups. If it can belong to only one subgroup, it is suggested that clustering can be done via k-means clustering, resulting each row of the partition matrix P having only one non-zero value. However,

if it can belong to multiple subgroups, then it is suggested to use fuzzy c-means for clustering, which is the iterative optimization minimizing the following criterion function.

$$J_m(P,V) = \sum_{i=1}^{m+n}\sum_{j=1}^{c}(P_{ij})^l d(\mathrm{x}_i, \mathrm{v}_j)^2, \tag{3.7}$$

where P_{ij} indicates whether the user or item $\mathbf{x}_i$ is a member of cluster j and v_j indicates the center of the cluster j. The function d indicates the distance function and the parameter l is the weighting exponent that controls the fuzziness of the partition. They prefer to use the Euclidian distance for d and 2 for l. P and V is updated as follows at each iteration.

$$P_{ij} = (d(\mathrm{x}_i, \mathrm{v}_j))^{2/(1-l)} \Bigg/ \left[\sum_{k=1}^{c}(d(\mathrm{x}_i, \mathrm{v}_k))^{2/(1-l)}\right], \tag{3.8}$$

and

$$\mathrm{v}_j = \left[\sum_{i=1}^{m+n} P_{ij}^l x_i\right] \Bigg/ \left[\sum_{i=1}^{m+n} P_{ij}^l\right]. \tag{3.9}$$

where i = 1,..., (n + m) and j = 1,...,c. The algorithm is designed to stop when the objective function is improved less than a threshold value ε after two successive steps. After the iteration stops, only top-k biggest elements are kept in each row, and normalized in such a way that the sum would be one.

The process of obtaining the partition matrix P is illustrated in Figure 3.2

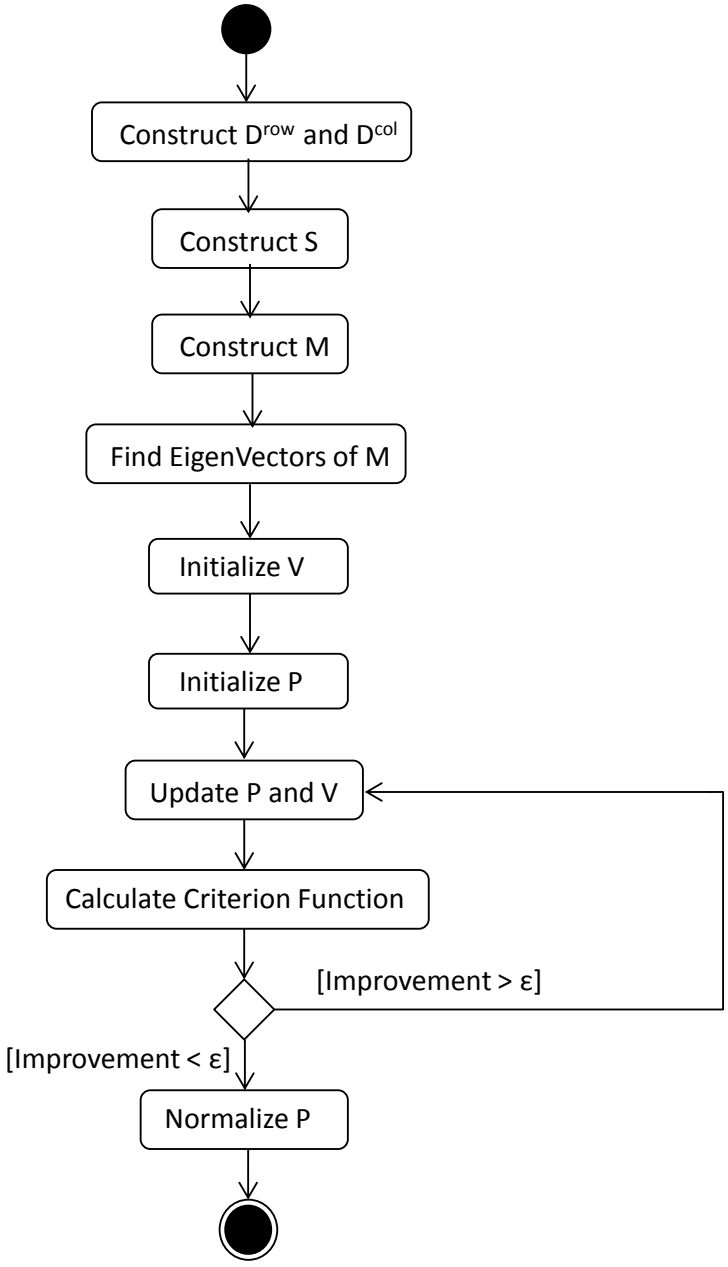


Figure 3.2: Obtaining the partition matrix of users and items.

3.1.4 Embedding Subgroups into Recommendation

Having user-item subgroups, the question is how to combine these subgroups into recommendation process. They propose to apply some CF algorithm on each subgroup and then merge the obtained prediction results together. Since the only input of CF algorithms is the user-item rating matrix, a user-item rating matrix could be

extracted for each subgroup from the big user-item rating matrix *T*. As a result the prediction scores of each subgroup will be obtained. When it comes to merge these results, a unified framework is proposed in [30] to handle the cases that a user or item belonging to several or zero subgroups. The framework is as follows:

$$Y_{ij} = \begin{cases} \sum_k Pre(\mathrm{u}_i, \mathrm{y}_j, k).\delta_{ik} & \text{if } \mathrm{u_i} \text{ and } \mathrm{y_j} \text{ belong to one or more subgroups} \\ 0 & \text{otherwise.} \end{cases}$$

where *Pre(u_i, y_j, k)* is the prediction score of user *i* to item *j* in subgroup *k*. δ_{ik} is an indicator value of user *i* that represents whether the subgroup *k* is the user's most interesting subgroup among all subgroups shared with item *j*. In other words, if the weight of the user *i* item *j* tuple of the subgroup *k* in the partition matrix is more than the other subgroups shared with item *j* then this indicator value will be greater than the other subgroups' indicator values. The indicator value could be the weight of the user-item tuple in the subgroup *k*, or it can be a hard weight (i.e. 0 or 1) according to the weights in the partition matrix, where the subgroup that has the maximum weight among the shared subgroups with a specific item is 1 and the others are 0. By the help of such a framework all cases of users and items could be handled.

The process of embedding subgroups into recommendation is illustrated in Figure 3.3

However notice that by this strategy the prediction scores for each user-item tuple are not obtained, since a user and an item tuple may not belong to any subgroups. In other words, prediction scores for some user-item tuples are unknown. Therefore, each user is suggested to unseen items that have top scores from the available prediction scores and the unknown prediction scores are not taken into consideration. To overcome this problem we will propose a solution while explaining merging process of the two algorithms.

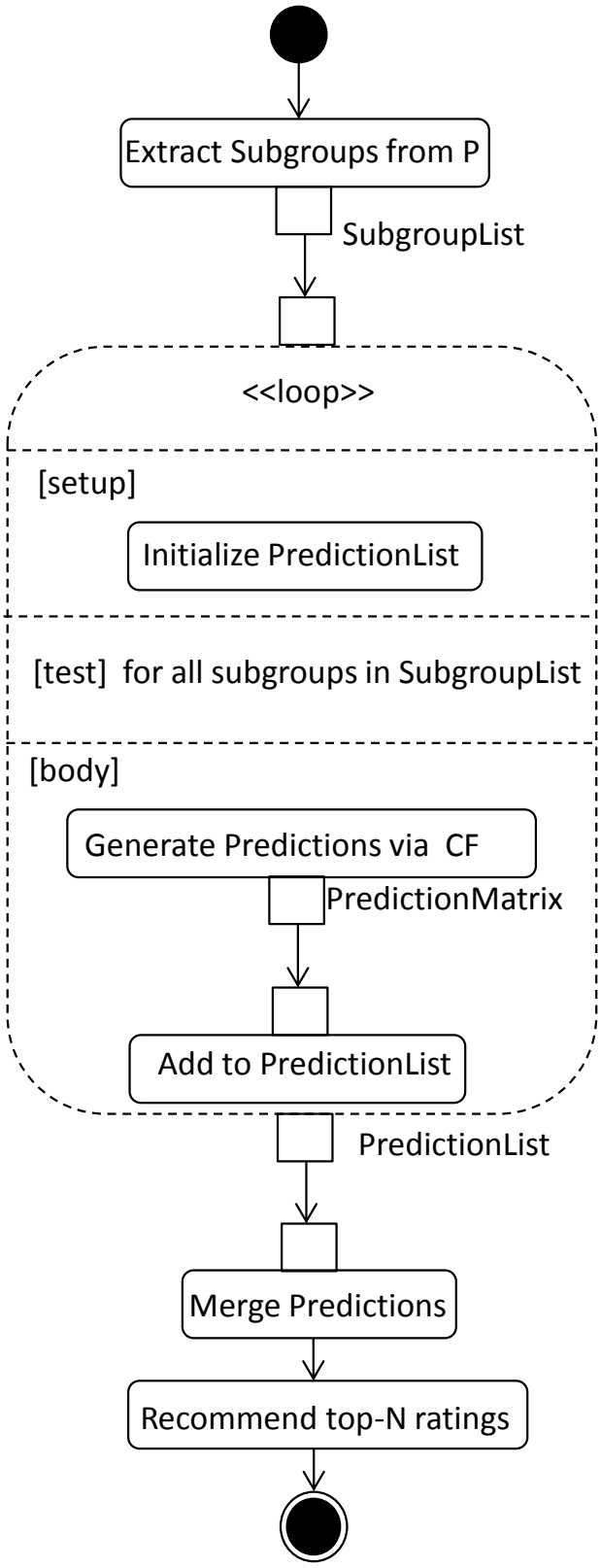


Figure 3.3: Embedding subgroups into recommendation.

3.2 CONTENT-BOOSTED COLLABORATIVE FILTERING

Despite the fact that collaborative filtering approaches has higher recommendation accuracy compared to content-based filtering, they ignore any information that is related to the content information.

Latent factor models attract attention because of their highly allusive ability of describing various aspects of data [27]. And matrix factorization technique is the basis of some of the most successful realizations of latent factor models [14]. The main idea of matrix factorization is to map users and items to a shared latent factor space, where high correspondence between user and item factors results in a recommendation. These techniques are successful techniques. Yet, they lack considering content information in the recommendation process, that could contribute significant and meaningful information in terms of improving recommendation accuracy. Considering such an improvement Chang et al. [11] proposed a model that incorporates usable content information directly into the matrix factorization approach. The details of the proposed algorithm, content boosting problem and the solution to the problem are covered in the next sections.

3.2.1 Chang's Algorithm

To enhance the accuracy of the collaborative filtering based recommendations, Chang et al. [11] offer an algorithm that incorporates content information into recommendation process. They suggest that item attributes could be used as content information. They advise to use movie genres, list of actors, keywords or producers as the content information in movie domain. In the proposed model, they extend the matrix factorization approach by placing these item attributes as a part of the latent factor vectors of items.

3.2.2 Problem Formulation

In order to make use of item content information, that could be useful in terms of improving significant and meaningful information about the user's interests, they embed

the content information directly into the matrix factorization technique. They employ a set called *G(i)* containing the attributes of item *i*. If item content information is considered to be genre of movies then, the set *G(i)* of a particular movie *i* will store the genres that the movie is related to. The set will be composed of values that indicate the genres, such as horror, thriller, drama, comedy etc. If the movie is related to a specific genre then the value corresponding to that genre in the set will be 1, it will be 0 otherwise.

3.2.3 Problem Solution

As a solution, they propose to embed the item attributes into item factor vector q_i directly. Before item attributes are embedded into item feature vector, the factor vector of item i is represented as $\mathbf{q}_i = (f_{i1}, f_{i2}, ..., f_{im})$. Similarly, the factor vector of user u is represented as $\mathbf{p}_u = (f_{u1}, f_{u2}, ..., f_{um})$, where m is the number of latent factors. Therefore the feature vector of each user and item is of size (1xm). The size of the item feature vector increases to (1x(k+m)) after placing item attributes to the first k columns of the vector. The size of the user feature vectors is also increased to be compatible with item feature vectors, making them also of size (1x(k+m)). However, note that the first k columns of the user feature vectors are initialized with random values, while the first k columns of item feature vectors are initialized with item attributes. After content information is embedded into $\mathbf{q}_i$, the size of the feature vector of item i, and the size of the user feature vector $\mathbf{p}_u$ are increased, they look like as follows:

$$\mathbf{q}_i = (g_{i1}, g_{i2}, ..., g_{ik}, f_{i(k+1)}, ..., f_{i(m+k)}) \tag{3.10}$$

$$\mathbf{p}_u = (f_{u1}, f_{u2}, \dots, f_{u(m+k)}) \tag{3.11}$$

where k is the number of attributes. Hence Eq. (2.6) turns into follows:

$$\hat{r}_{ui} = (f_{u1}, f_{u2}, \dots, f_{u(m+k)}) \times (g_{i1}, g_{i2}, ..., , g_{ik}, f_{i(k+1)}, ..., f_{i(m+k)})^T \tag{3.12}$$

Let c_1 be

$$c_1 = f_{u1} \times g_{i1} \tag{3.13}$$

Then the formula (3.12) is equivalent to following.

$$\hat{r}_{ui} = c_1 + c_2 + ... + c_k + \sum_{d=k+1}^{m+k} f_{ud} \times f_{id} \tag{3.14}$$

where c_l is the user preference on item attribute l. The meaning of above equation is adding constant biases to the matrix factorization model. In matrix factorization approach the data is viewed from a high level perspective. Hence, a global cost function is minimized while iterating. Therefore, incorporating bias to the matrix factorization model can express personalized preferences of individuals.

In order to obtain the user and item factor vectors p_u and q_i, the method that is presented in Eq. (2.5) is used. The major difference is that, since item attributes are placed in the first k rows of the item vector q_i, these columns are not altered in the iterations. In other words, while Eq. (2.8) is applied, the first k columns that represent item attributes (q_{i1} to q_{ik}) are not updated during the iteration.

Since the first k columns of q_i are assigned to corresponding item attributes instead of random values, it is needed to add k columns to user vector p_u. Then by iterating over the set of known ratings T and updating user and item factor vectors via moving them in the opposite direction of the gradient, the rating scores can be predicted.

The process of content-boosted matrix factorization is illustrated in Figure 3.4

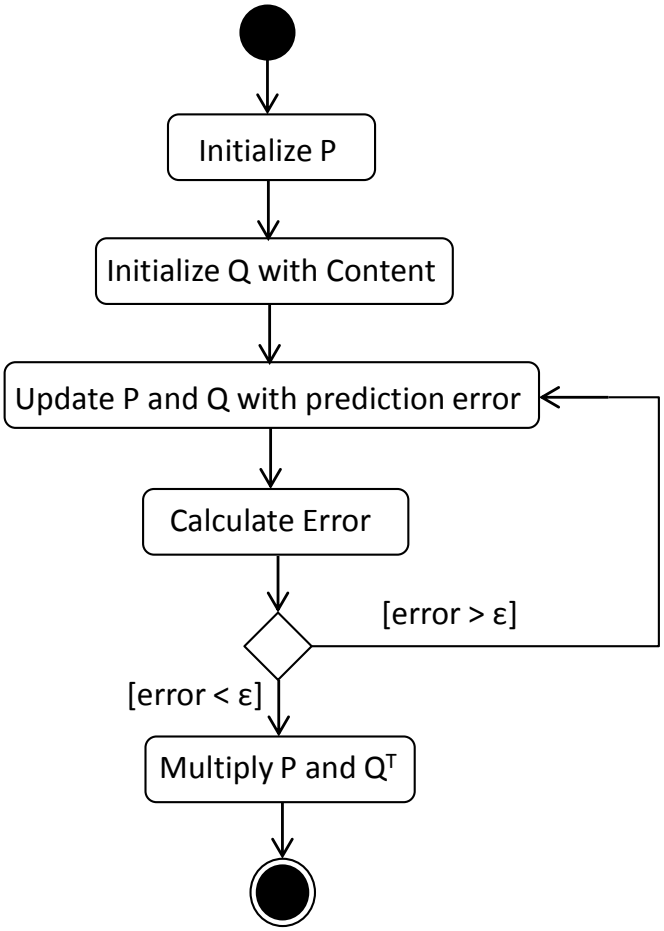


Figure 3.4: Content-boosted Matrix Factorization.

3.3 CONTENT-BOOSTED MATRIX FACTORIZATION VIA USER-ITEM SUBGROUPS

We propose an algorithm that uses content information directly in the recommendation process while providing better suggestions to users by considering their taste via user-item subgroups. In order to accomplish that, we merge Content-Boosted matrix factorization technique described in Section 3.2 and MCoC algorithm described in Section 3.1. First the stages of the algorithm are discussed and this process is illustrated on a small user-item rating matrix. Also the activity diagram of this process is illustrated in Figure 3.5.

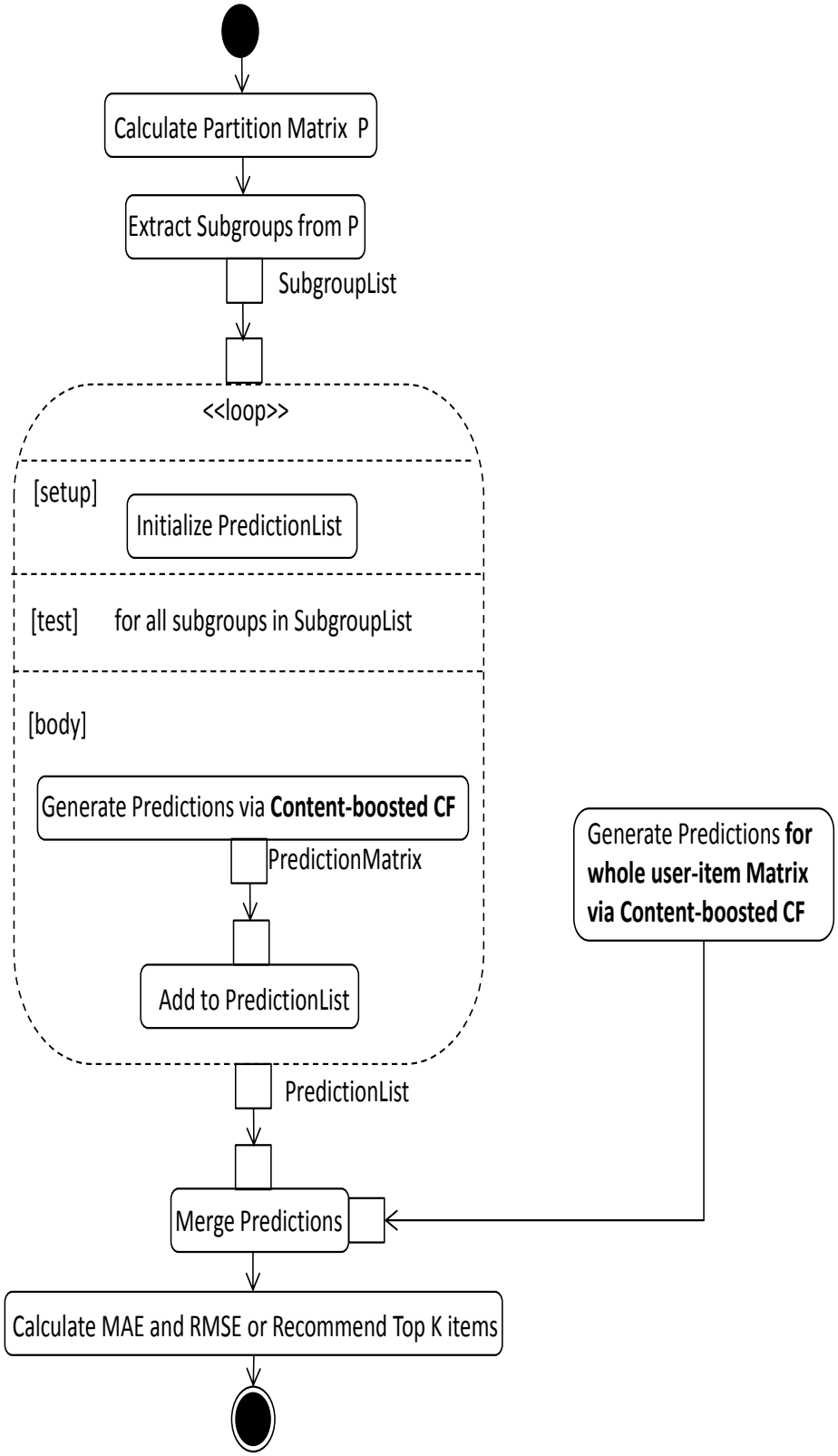


Figure 3.5: Grouped and content-boosted Matrix Factorization.

The proposed method consists of three stages:

1. Discover user-item subgroups
2. Run Content-Boosted matrix factorization on all subgroups
3. Run Content-Boosted matrix factorization on the whole set and merge with subgroup predictions

First of all the big user-item rating dataset is divided into two parts. The first part is composed of the randomly selected 80% of all rating scores in the dataset and is used for training purposes. Therefore it is called the training matrix. The remaining 20% of the rating scores is used for testing purposes while evaluating the results. Hence it is called the test matrix.

Stage 1: Discovering Subgroups

The aim is to find like-minded users on a subset of items. To discover user-item subgroups we use MCoC algorithm. Hence a partition matrix is created for users and items that indicates whether they belong to any subgroups or not.

Since the information we have is only rating scores, the partition matrix is derived from these scores by some derivations. The idea of MCoC algorithm is that, if rating score of user *i* to the item *j* is high, then the vectors of user *i* and item *j* should be close. Therefore the loss function presented in Equation 3.2 ($\epsilon(Q,R) = \sum_{i=1}^{n}\sum_{j=1}^{m}(\| \frac{q_i}{\sqrt{D_{ii}^{row}}} - \frac{r_j}{\sqrt{D_{jj}^{col}}} \|^2 \ T_{ij})$) is used. Minimizing the loss function means that the user and item vectors are close, and therefore the possibility of this user and this item belonging to the same subgroup is high. Note that, D^{row} is the diagonal degree matrix of users, and D^{col} is the diagonal degree matrix of items. Each diagonal entry of the D^{row} matrix is the sum of all rating scores that the user gave to the items in the dataset. Similarly, each diagonal entry of the D^{col} matrix is the sum of all rating scores that are given to the item by all users in the dataset. The visual representations of these matrices are provided in the section 3.3.1 in order to clarify the process.

Since minimizing the loss function 3.2 is the same as minimizing the transition function 3.5 ($Tr(P^TMP)$) the matrix M should be computed. Since $M =$

$\begin{bmatrix} I_n & -S \\ -S^T & I_m \end{bmatrix}$ where $S = (D^{row})^{-\frac{1}{2}} T (D^{col})^{-\frac{1}{2}}$, and T is the user-item training matrix, the matrix M is obtained.

Since matrix M is a big matrix and computations with big matrices are costly and time consuming, eigenvectors of M is computed and the smallest eigenvectors are used to find the partition matrix P.

The partition matrix is computed with the fuzzy c-means soft clustering method. The fuzzy c-means algorithm is an iterative optimization that minimizes the criterion function 3.7 ($J_m(P,V) = \sum_{i=1}^{m+n} \sum_{j=1}^{c} (P_{ij})^l d(\mathrm{x}_i, \mathrm{v}_j)^2$). During the iterations the partition matrix P and the center of cluster V matrices are updated. When the improvement of the function is less than a threshold, then the algorithm halts and the partition matrix P is obtained.

The rows of the partition matrix represent the users and items, the columns represent the subgroups. In fact, the partition matrix vertically consists of two parts; the first part is for users the second part is for items. In the first part each row indicates a user's belonging to the subgroups and, likewise in the second part of the matrix each row indicates an item's belonging to the subgroups. The columns of the partition matrix include values between 0 and 1. These values sum up to 1 and each column represents the weight of a user's or an item's belonging to the subgroup. However, the partition matrix should be normalized in a way that a user or an item can belong to a maximum number of subgroups. According to MCoC algorithm a user can belong to at most *k* subgroups where $k = \lceil log2(c) \rceil$, if there are *c* subgroups. Therefore the rows of the matrix are normalized in such a way that at most *k* nonzero values exist in each row.

Considering a subgroup, if the corresponding weights of a user and an item are greater than zero then they belong to this subgroup. In this manner, the user-item tuples that belong to the same subgroups are extracted and new matrices are created for these subgroups. Newly created matrices are composed of rating scores of the users to the items that belong to the same subgroups and the scores are obtained from the training matrix. Notice that all the newly created matrices are a portion of the training matrix. These portions may overlap, since a user-item tuple may belong

to more than one subgroup. Also some portions of the training matrix may not be covered in case of a user-item tuple not sharing any subgroups.

Stage 2: Running Content-Boosted matrix factorization on all subgroups

Newly created subgroup rating matrices can be treated as user-item rating matrix independent from each other. Therefore Content-Boosted matrix factorization can be applied to all of them. Then the results of all subgroup matrices can be merged.

While applying Stochastic Gradient Matrix Factorization technique, user and item feature vectors are initialized with small random values and they are updated in the opposite direction of the gradient until the prediction error is minimized. Consequently, the multiplication of the user and item feature matrices lead to a new prediction matrix. In order to be able to run Content-Boosted matrix factorization, on the other hand, the genre information of the items is embedded into the item feature vectors. If there are G number of genre categories available in the dataset, then the first G columns are initialized with the genre information of the items instead of random values. The genre information is represented by a 0 or 1. If the item is of the specified genre then the value is set to 1, 0 otherwise. For instance, if there are four genre categories in the dataset, then each item vector is preceded with four 0 or 1 values (e.g. 1 0 0 1). In this manner, the genre information of each item is placed at the beginning of each item feature vector. The important point is not to update the first G columns of the item vectors while iterating over and updating user, item feature vectors. Thus the genre information of the items are directly used in the recommendation process.

In order the user feature vectors to be compatible with the item feature vectors, they are also preceded with G more columns. In other words, G number of random values are placed in front of the user factor vectors. Note that, the first G columns of user feature vectors are initialized with random values, while item feature vectors' are initialized with the genre information.

After embedding genre information into item feature vectors and adjusting the user factor vectors accordingly, Stochastic Gradient Descent algorithm is applied to these vectors. After the updating of the user and item feature vectors is over, the multiplication of these matrices leads to the new user-item rating matrix.

The prediction score matrix of each subgroup are obtained, after running the algorithm on all of the subgroup matrices. These matrices are merged according to the user's interest in the items. The partition matrix is used in order to find the subgroup in which user is the most interested for each item, and the prediction score is taken from the corresponding subgroup's matrix. This process is applied for all user-item tuples. However, a user-item tuple may not share any subgroups, as a consequence of the MCoC algorithm. In that case the prediction scores of these tuples are unknown. Such user-item tuples are handled in the next stage.

Stage 3: Run Content-Boosted matrix factorization on the whole set and merge with subgroup predictions

In order to cover prediction scores of the tuples that are unknown after merging subgroup predictions, Content-Boosted Matrix Factorization is applied to the whole training dataset. The resulting prediction score matrix is merged with the matrix that is obtained by merging subgroup matrices. As a result, the prediction score of each user-item tuple becomes available. Thus prediction accuracy of the resulting matrix can be evaluated and unseen items that have high rating scores can be recommended to the users.

3.3.1 Illustrative Example

Recommendation algorithms give more accurate results provided that a big dataset is used. However, in order to explain the proposed algorithm, an illustration of the process on a small user-item rating matrix that is composed of 8 users and 16 items is presented (Figure 3.6). The user-item rating matrix is used only for illustration purposes and is not a real life matrix. In fact the rating scores are randomly selected.

The dataset contains 45 rating scores. First, the dataset is divided into two parts. The first part is used for training purposes and contains 36 rating scores (i.e. 80% of the whole dataset), and the remaining 9 rating scores are used for testing purposes (i.e. 20% is the dataset). The training and the test matrices can be seen in Figure 3.7 and in Figure 3.8 respectively.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	0	4	0	3	2	0	0	5	0	0	0	4	0	3	0	0
u2	2	0	0	5	0	0	3	0	5	0	0	0	0	0	0	0
u3	0	3	0	0	3	0	0	4	0	0	0	0	0	0	2	5
u4	3	0	0	2	0	0	5	0	0	2	0	0	1	0	0	5
u5	0	0	4	0	3	0	0	0	3	0	0	5	0	1	0	0
u6	4	0	3	0	0	2	0	5	0	0	0	0	3	0	0	4
u7	2	0	0	3	4	0	5	0	2	0	0	3	0	2	0	0
u8	0	3	0	0	2	0	4	0	0	0	5	0	0	2	0	4

Figure 3.6: Example user-item rating matrix.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	0	4	0	3	2	0	0	5	0	0	0	4	0	3	0	0
u2	0	0	0	0	0	0	3	0	5	0	0	0	0	0	0	0
u3	0	3	0	0	0	0	0	4	0	0	0	0	0	0	0	5
u4	3	0	0	0	0	0	5	0	0	2	0	0	1	0	0	5
u5	0	0	4	0	3	0	0	0	3	0	0	5	0	1	0	0
u6	4	0	3	0	0	0	0	5	0	0	0	0	3	0	0	4
u7	2	0	0	3	4	0	0	0	2	0	0	3	0	2	0	0
u8	0	3	0	0	2	0	4	0	0	0	0	0	0	2	0	0

Figure 3.7: Training matrix.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u2	2	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
u3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	2	0
u4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
u5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u6	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
u7	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
u8	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	4

Figure 3.8: Test matrix.

Stage 1: Discovering Subgroups

Subgroups are created from the partition matrix P. In order to compute the partition matrix P, matrix M should be computed first. Therefore S, D^{row} and D^{col} matrices are needed. Each diagonal value of D^{col} matrix is the summation of rating scores that are given to that item. For instance, the first item's corresponding value is 9, which is the summation of the rating scores 3 + 4 + 2. After each item's value is computed, the resulting matrix D^{col} is presented in Figure 3.9. Similarly, D^{row} matrix is computed. Each diagonal value of D^{row} matrix is the summation of that user's rating scores available in the dataset. For example, the first user's diagonal value is 21, which is the summation of the rating scores 4 + 3 + 2 + 5 + 4 + 3. Each user's value is computed in the same way. After each user's rating scores is computed the resulting D^{row} matrix is presented in the Figure 3.10.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u2	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u3	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
u4	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
u5	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0
u6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u7	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0
u8	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0
u9	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
u10	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
u11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u12	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0
u13	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
u14	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0
u15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14

Figure 3.9: D^{col} matrix.

	i1	i2	i3	i4	i5	i6	i7	i8
u1	21	0	0	0	0	0	0	0
u2	0	8	0	0	0	0	0	0
u3	0	0	12	0	0	0	0	0
u4	0	0	0	15	0	0	0	0
u5	0	0	0	0	16	0	0	0
u6	0	0	0	0	0	19	0	0
u7	0	0	0	0	0	0	16	0
u8	0	0	0	0	0	0	0	11

Figure 3.10: D^{row} matrix.

Having computed D^{row} and D^{col} matrices, the S matrix is computed by $S = (D^{row})^{-\frac{1}{2}} T (D^{col})^{-\frac{1}{2}}$ and the resulting matrix is presented in Figure 3.11.

$$\begin{bmatrix} 0 & 0{,}3 & 0 & 0{,}2 & 0{,}1 & 0 & 0 & 0{,}3 & 0 & 0 & 0 & 0{,}3 & 0 & 0{,}2 & 0 & 0 \\ 0{,}2 & 0 & 0 & 0{,}4 & 0 & 0 & 0{,}2 & 0 & 0{,}4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0 & 0{,}2 & 0 & 0 & 0{,}3 & 0 & 0 & 0 & 0 & 0 & 0 & 0{,}3 & 0{,}3 \\ 0{,}2 & 0 & 0 & 0{,}1 & 0 & 0 & 0{,}3 & 0 & 0 & 0{,}3 & 0 & 0 & 0{,}1 & 0 & 0 & 0 \\ 0 & 0 & 0{,}4 & 0 & 0{,}2 & 0 & 0 & 0 & 0{,}2 & 0 & 0 & 0{,}4 & 0 & 0{,}1 & 0 & 0 \\ 0{,}3 & 0 & 0{,}2 & 0 & 0 & 0{,}3 & 0 & 0{,}3 & 0 & 0 & 0 & 0 & 0{,}3 & 0 & 0 & 0{,}2 \\ 0{,}1 & 0 & 0 & 0{,}2 & 0{,}2 & 0 & 0{,}3 & 0 & 0{,}1 & 0 & 0 & 0{,}2 & 0 & 0{,}2 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0 & 0{,}1 & 0 & 0{,}2 & 0 & 0 & 0 & 0{,}5 & 0 & 0 & 0{,}2 & 0 & 0{,}2 \end{bmatrix}_{8x16}$$

Figure 3.11: S matrix.

Since matrix M is computed by $M = \begin{bmatrix} I_n & -S \\ -S^T & I_m \end{bmatrix}$, after the computations obtained matrix is presented in Figure 3.12.

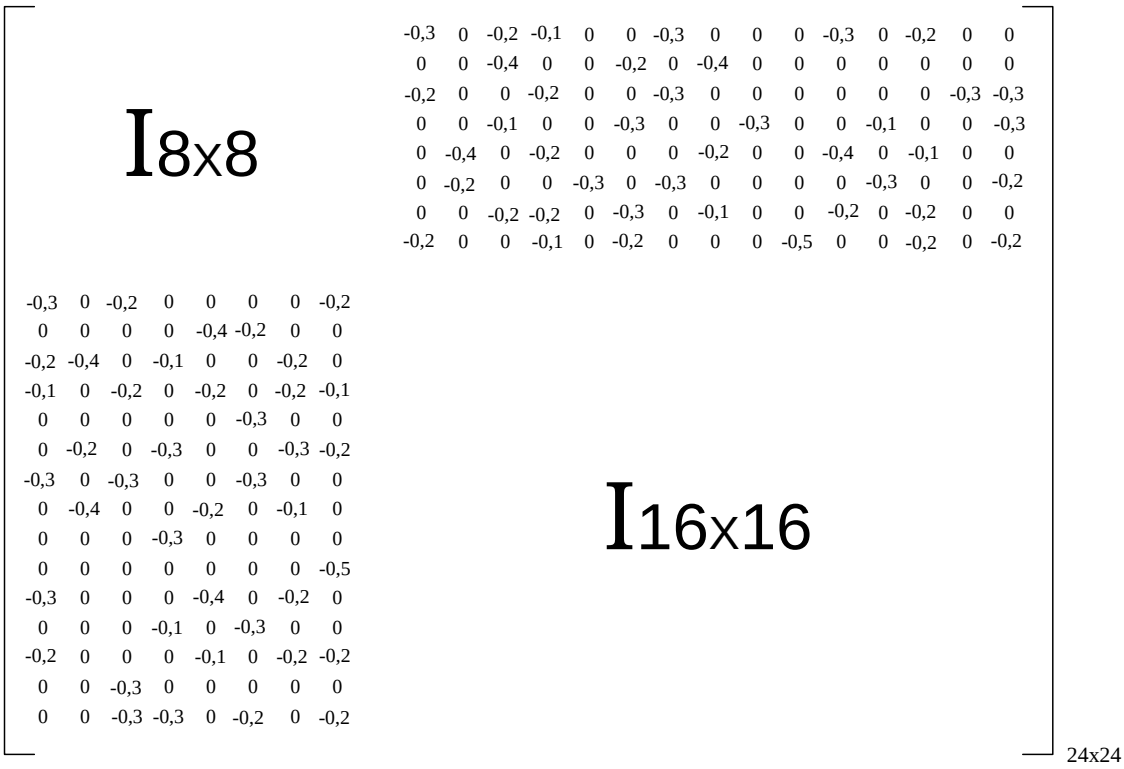


Figure 3.12: M matrix.

After eigenvectors of matrix M are extracted and the criterion function is optimized, a partition matrix is obtained. In this example the number of subgroups is 3. Therefore the maximum number of subgroups that a user or an item can belong to is $\lceil log2(3) \rceil$ = 2. The resulting partition matrix is presented in 3.13.a. Note that this matrix is not normalized, that is the number of nonzero values in a row is greater than 2. Since the user-item rating matrix is not a real life rating matrix and available rating scores are too many when compared to a real life rating matrix, all of the user-item tuples belong to some of the subgroups. In a real life dataset however, it is more likely that some user-item tuples do not belong to any subgroups. In order to cover such cases the normalized partition matrix that will be used in computations is presented in Figure 3.13.b. Notice that, 8th user, 12th and 15th items do not belong to any subgroups, that is the weights of the subgroups are 0.

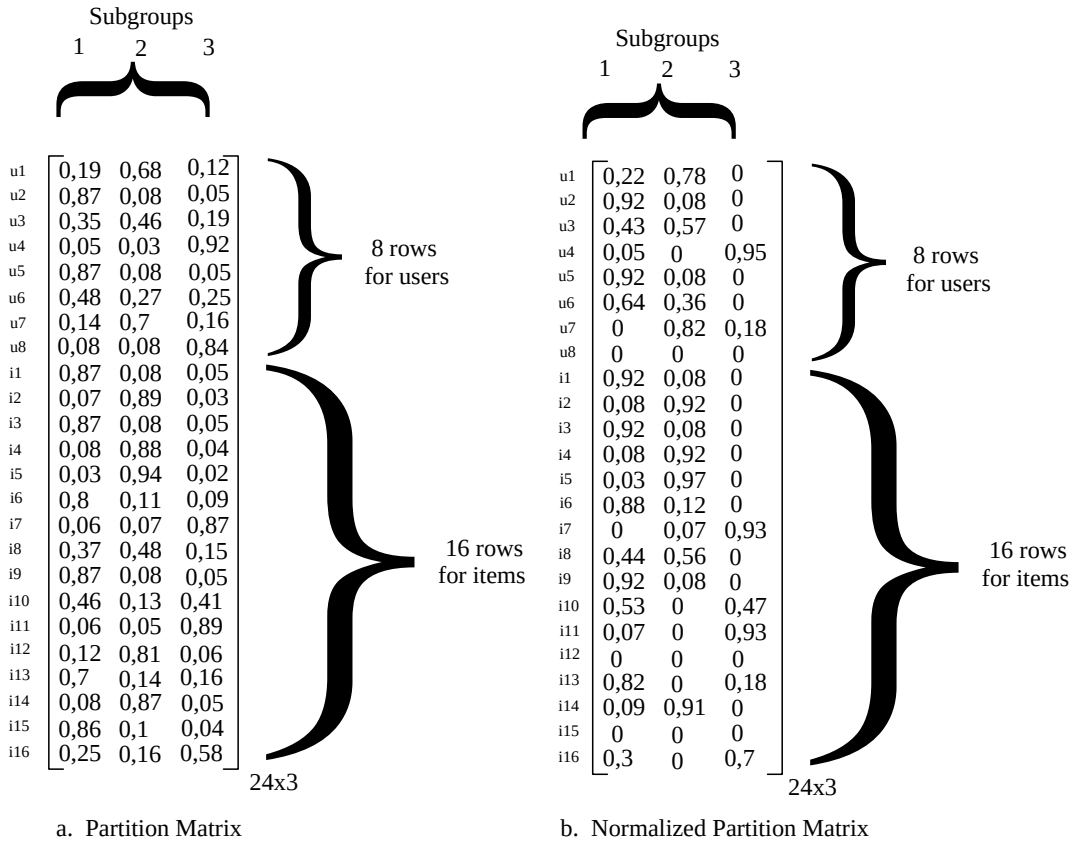


Figure 3.13: Partition & Normalized Partition matrix P.

According to the partition matrix 3.13.b, there are three subgroups. Therefore three new group matrices are created from the training matrix rating scores. The group matrices are created based on the weights of the users and items in the subgroups. If a user-item tuple belong to the first subgroup, i.e. the weight of the first subgroup is greater than zero, then the corresponding rating score in the training matrix is inserted into the first subgroup matrix. In this manner, all the subgroup matrices are created. The created subgroup matrices are presented in Figures 3.14 3.15 3.16.

Stage 2: Running Content-Boosted matrix factorization on all subgroups

The subgroup matrices can be thought independent from each other, and the Content-Boosted Matrix Factorization algorithm can be applied to all of them separately. In this algorithm, as in the Stochastic Gradient Descent algorithm, the prediction matrix is obtained from the multiplication of user item feature vectors. In the

traditional Stochastic Gradient Descent algorithm, if 5 latent factors are considered for instance, the prediction matrix is computed by $R = PQ^T$, where P is the user feature vector of size 8x5 and Q is the item feature vector of size 16x5 for this dataset. Hence the multiplication of these feature vectors results in a prediction matrix of size 8x16, which is of the same size with the original user-item rating matrix, and the recommendations are provided from the obtained prediction matrix. The representations of the user and item feature vectors can be seen in Figure 3.17.a.

	i1	i2	i3	i4	i5	i7	i8	i9	i10	i13	i14	i16
u1	0	4	0	3	2	0	5	0	0	0	3	0
u2	2	0	0	5	0	0	0	5	0	0	0	0
u3	0	3	0	0	3	0	4	0	0	0	0	5
u4	3	0	0	2	0	0	0	0	2	1	0	5
u5	0	0	4	0	3	0	0	3	0	0	1	0
u6	4	0	3	0	0	0	5	0	0	3	0	4

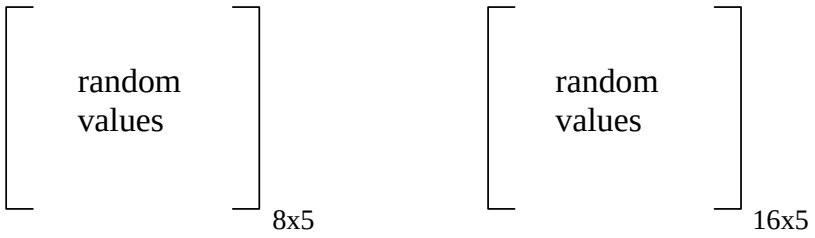
Figure 3.14: First subgroup matrix.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i14
u1	0	4	0	3	2	0	0	5	0	3
u2	2	0	0	5	0	0	3	0	5	0
u3	0	3	0	0	3	0	0	4	0	0
u5	0	0	4	0	3	0	0	0	3	1
u6	4	0	3	0	0	2	0	5	0	0
u7	2	0	0	3	4	0	5	0	2	2

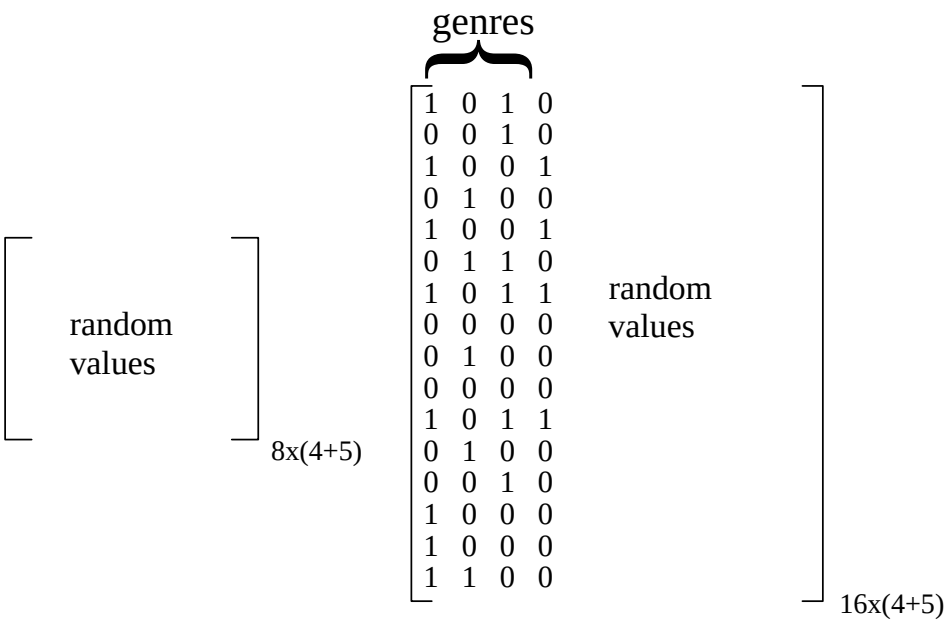
Figure 3.15: Second subgroup matrix.

	i7	i10	i13	i16
u4	5	2	1	5
u7	5	0	0	0

Figure 3.16: Third subgroup matrix.



a. Without content information



b. With content information

Figure 3.17: User and item feature vectors of Content-Boosted Matrix Factorization algorithm.

In the Content-Boosted Matrix Factorization algorithm, since the genre information of the items are embedded into the item feature vector the size of the item feature vector increases. In this dataset we assumed that the items have four categories of genre information, where the items can belong to multiple genres, and provided the genres of the 16 items. Hence the item feature vector becomes the size of 16x(4+5), and the columns of item features are preceded with the corresponding genre information. Since the multiplication of the user and item feature vectors lead to the prediction matrix, their size should match. Therefore the size of the user feature

vector is also increased and it becomes size of 8x(4+5). The visual representations of these user and item feature vectors can be seen from Figure 3.17.b.

After embedding content information into item feature vectors, the Content-Boosted Matrix Factorization algorithm is applied to the subgroup matrices and three different result matrices are obtained. Then the results of these matrices are merged by the framework proposed by Xu et al. [30], where the rating score of a user-item tuple is obtained from the subgroup vector, which is the subgroup that the user is most interested in (the subgroup that has the maximum weight among all the subgroups shared with that item in the partition matrix). The resulting matrix of merged subgroup matrices is presented in Figure 3.18. Note that there are not any prediction scores of user number 8, since this user does not belong to any subgroups in the partition matrix. Hence the prediction score of this user will be obtained in next stage.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i13	i14	i15	i16
u1	3,1	4,2	2,3	3,1	2,8	2,7	1,5	4,4	2,3	1	4,8	1,4	4,1	3,4
u2	2,1	3,7	1,4	0,1	1,5	2,8	3,2	1,4	4,8	1,2	4,2	3,4	0,1	4
u3	0,4	3,1	0,2	0,7	3,1	2,8	2	3,8	1,7	1,4	3	4,5	2,2	3,6
u4	2,8	1,9	0,4	2	0,2	1	4,9	4	2,2	3	0,3	1,3	1,2	4,2
u5	3,3	3,4	3,9	1	2,4	0,3	4,2	3	3,3	5	2	1,4	0,8	1,1
u6	4,2	4,1	2,9	1,6	2,8	2	4,5	4,9	3,3	2,6	2,9	3	0,8	4,3
u7	1,7	1,4	3,2	2,9	4,1	2,1	4,5	3,8	2,7	3,1	1,6	2	1,1	3

Figure 3.18: Merged subgroup matrix.

Stage 3: Run Content-Boosted Matrix Factorization on the whole set and merge with subgroup predictions

The aim of this stage is to cover the user-item tuples that have no recommendation scores because of not belonging to the same subgroup with that item. Therefore the Content-Boosted Matrix Factorization technique is applied to the whole training matrix 3.7. The obtained prediction matrix is merged with the prediction matrix of merged subgroups. For each user-item tuple, the predicted rating score is obtained from the merged subgroup matrix as long as it is available. Otherwise, the predicted rating score is obtained from the Content-Boosted Matrix Factorization predicted ma-

trix. After merging, the prediction matrix of this dataset is presented in Figure 3.19.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	3,1	4,2	2,3	3,1	2,8	2,7	1,5	4,4	2,3	1	0,1	4	4,8	1,4	4,1	3,4
u2	2,1	3,7	1,4	0,1	1,5	2,8	3,2	1,4	4,8	1,2	4,5	3,7	4,2	3,4	0,1	4
u3	0,4	3,1	0,2	0,7	3,1	2,8	2	3,8	1,7	1,4	2,4	4,3	3	4,5	2,2	3,6
u4	2,8	1,9	0,4	2	0,2	1	4,9	4	2,2	3	1,4	5	0,3	1,3	1,2	4,2
u5	3,3	3,4	3,9	1	2,4	0,3	4,2	3	3,3	5	1,2	2,6	2	1,4	0,8	1,1
u6	4,2	4,1	2,9	1,6	2,8	2	4,5	4,9	3,3	2,6	2	2	2,9	3	0,8	4,3
u7	1,7	1,4	3,2	2,9	4,1	2,1	4,5	3,8	2,7	3,1	3,7	1	1,6	2	1,1	3
u8	3,5	3	3,9	3,9	2	2,5	4	4,6	0,2	1,1	5	1,4	1,8	2,1	1,3	4

Figure 3.19: Merged matrix.

After the prediction matrix is obtained the prediction accuracy and top-N recommendations can be computed based on the test matrix. First, seen items, i.e. the items that have nonzero values in the training matrix, are removed from the resulting prediction matrix, in order not to recommend the items that exist in the training matrix. Figure 3.20 represents the unseen item matrix matrix, from which the seen items are extracted.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	3,1	0	2,3	0	0	2,7	1,5	0	2,3	1	0,1	0	4,8	0	4,1	3,4
u2	2,1	3,7	1,4	0,1	1,5	2,8	0	1,4	0	1,2	4,5	3,7	4,2	3,4	0,1	4
u3	0,4	0	0,2	0,7	3,1	2,8	2	0	1,7	1,4	2,4	4,3	3	4,5	2,2	0
u4	0	1,9	0,4	2	0,2	1	0	4	2,2	0	1,4	5	0	1,3	1,2	0
u5	3,3	3,4	0	1	0	0,3	4,2	3	0	5	1,2	0	2	0	0,8	1,1
u6	0	4,1	0	1,6	2,8	2	4,5	0	3,3	2,6	2	2	0	3	0,8	0
u7	0	1,4	3,2	0	0	2,1	4,5	3,8	0	3,1	3,7	0	1,6	0	1,1	3
u8	3,5	0	3,9	3,9	0	2,5	0	4,6	0,2	1,1	5	1,4	1,8	0	1,3	4

Figure 3.20: Prediction matrix after the seen items are removed.

In Figure 3.21, the framed scores represent the prediction scores of the unseen item matrix, that overlap with the test matrix. From the framed scores MAE and RMSE values are obtained. In order to compute MAE, the error between the prediction matrix and test matrix is calculated for each user, and the summation of error values averaged by the number of users (Equation 4.1). RMSE is computed by the

same fashion except that the summation of the squared error values that are averaged by the number of users is square rooted (Equation 4.2). For example the error value of the 3rd user is $|3.1 - 3| + |2.2 - 2| = 0.3$, and the squared error value is $(3.1 - 3)^2 + (2.2 - 2)^2 = 0.05$. The obtained MAE and RMSE values of the whole dataset are 0.51 and 0.45 respectively.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12	i13	i14	i15	i16
u1	3,1	0	2,3	0	0	2,7	1,5	0	2,3	1	0,1	0	4,8	0	4,1	3,4
u2	2,1	3,7	1,4	0,1	1,5	2,8	0	1,4	0	1,2	4,5	3,7	4,2	3,4	0,1	4
u3	0,4	0	0,2	0,7	3,1	2,8	2	0	1,7	1,4	2,4	4,3	3	4,5	2,2	0
u4	0	1,9	0,4	2	0,2	1	0	4	2,2	0	1,4	5	0	1,3	1,2	0
u5	3,3	3,4	0	1	0	0,3	4,2	3	0	5	1,2	0	2	0	0,8	1,1
u6	0	4,1	0	1,6	2,8	2	4,5	0	3,3	2,6	2	2	0	3	0,8	0
u7	0	1,4	3,2	0	0	2,1	4,5	3,8	0	3,1	3,7	0	1,6	0	1,1	3
u8	3,5	0	3,9	3,9	0	2,5	0	4,6	0,2	1,1	5	1,4	1,8	0	1,3	4

Figure 3.21: Rating scores overlapping the test matrix

MAP and Precision@10 values are computed according to Equation 4.4 and Equation 4.3 respectively. Considering the 3rd user, the ranked list of 10 items {14, 12, 5, 13, 6, 11, 15, 9, 10, 4} are recommended. Notice that the nonzero values in the test matrix (i.e. 5 and 15) are recommended to the user. After computation for each user the MAP and Precision@10 values are 0.23 and 0.1 respectively.

CHAPTER 4

EVALUATION AND EXPERIMENTS

In order to evaluate our method, we performed experiments based on MovieLens dataset. We compared the performance of the following four algorithms in terms of MAE, RMSE, Precision@10 and MAP.

1. Traditional Stochastic Gradient Based MF (TMF)
2. Traditional Stochastic Gradient Based MF via Subgroups (GMF)
3. Content-boosted Stochastic Gradient Based MF (CMF)
4. Content-boosted Stochastic Gradient Based MF via Subgroups (CGMF)

The first algorithm refers to traditional MF algorithm that is based on the Stochastic Gradient Descent technique that is explained in Section 2.3.1. This algorithm, from now on, will be referred as traditional MF or TMF in short.

The second algorithm GMF is the technique that is explained in Section 3.1. It discovers subgroups and run TMF algorithm on these subgroups. Since the prediction results for all the set of user-item tuples are not available we run TMF algorithm on the whole set and fill the missing prediction values from the generated results by TMF. This approach will be referred as GMF hereafter.

The third algorithm CMF is explained in Section 3.2. This method incorporates content information into TMF by employing item attributes of the movies. We choose to use the genre of the movies as the item attribute. The details of the dataset are presented in detail in section 4.1.

The last algorithm is our proposed approach. In this method, we first discover subgroups via creating subgroups method that is explained in Section 3.1 and run the CMF algorithm for all of these subgroups. We then run CMF algorithm on the whole user item rating matrix. Finally we merge the prediction results obtained from subgroups and fill the missing prediction values from the prediction results of running CMF algorithm on the whole set.

We use 5-fold cross-validation and apply same initializations for the shared values, in evaluating the performance of all algorithms.

4.1 Dataset

We performed experiments on the MovieLens 100k dataset [1]. This dataset is a widely used dataset that includes 100,000 rating scores provided by 943 users on 1682 items. The basic information about movies and users are also available. The scores range from 1 to 5. Each user has at least 20 scores available in the dataset. This version of the dataset provides 19 different genres for each movie. Each movie can belong to one or more genres.

In the experiments, we considered the genres information only; however, additional content information can be used with a few modifications of our method (i.e. adding some more columns to the user and item matrices). We choose the training set to be 80 percent of the dataset and choose testing set to be the remaining 20 percent of the dataset.

4.2 Evaluation Metrics

There are several metrics while evaluating recommender systems such as, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Precision@K, Mean Avarage Precision (MAP), Receiver Operating Characteristic curves (ROC curves), F1 metric [30]. MAE and RMSE are mostly used to measure the accuracy of the predicted user ratings. Precision@K and MAP, on the other hand, are used to evaluate the quality of the recommended top K items. While conducting the experiments we

aimed to measure the predictive accuracy and the quality of the top 10 recommendations of all methods in order to compare them properly. Therefore we used MAE, RMSE, Precision@10 and MAP evaluation metrics during the evaluation.

The definition of the MAE and RMSE is as follows [11]:

$$MAE = \frac{1}{|T|} \sum_{(u,i) \in T} |r_{ui} - \hat{r}_{ui}| \tag{4.1}$$

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (r_{ui} - \hat{r}_{ui})^2} \tag{4.2}$$

where r_{ui} represents the real rating and $\hat{r}_{ui}$ represents the predicted rating. T refers to the known set of user ratings since r_{ui} is not known for all user-item tuples in the real test dataset. Naturally, $|T|$ is the cardinal number of the test set. Therefore, getting lower values for MAE and RMSE means that the predictions of the recommendation systems are more accurate.

The definition of the Precision@K and MAP is as follows [29, 30]:

$$Precision@K = \frac{relevant\ items\ in\ top\ K\ items}{K} \tag{4.3}$$

$$MAP(U) = \frac{1}{|U|} \sum_{j=1}^{|U|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{4.4}$$

where precision is the number of items that are relevant to the user in the top K items, and Average Precision is the average of the precisions computed at the point of each correctly recommended item $(d_1, ..., d_{mj})$ in the ranked list, for a single user j. The value is then averaged over the user set U to find the Mean Average Precision. The bigger values of Precision and MAP means the recommendation is more successful.

4.3 Experiments

We implemented four recommendation methods that we described and we checked whether our approach improved MAE, RMSE, Precision@10 and MAP values or not.

The performance of recommendation algorithms highly depends on the parameter selection, i.e. an algorithms performance may drastically vary based upon the selected parameter values. In order to be confident with our results, we used the same initialization procedures and same parameter values in all of the methods when applicable.

First for TMF, the initialization of P and Q matrices and the choice of λ and γ are the key points. We initialized P and Q with small random values. We realized that increasing the number of factors K produced better RMSE values, however it increased memory consumption linearly and training time almost linearly. Similarly increasing λ and decreasing γ cause same effect of increasing memory consumption and training time. Therefore, we choose K= 5, λ = 0.02 and γ = 0.0002.

For GMF, the key parameters are r and k. Since the performance of using just a few eigenvectors is competitive, as experienced in by Xu et al. [30], we choose r to be 3. As for parameter k we set it k= $\lceil log_2(c) \rceil$ as Xu et al. [30] suggested. We choose c = 3, 10, 20 and respectively k = 2, 4, 5 in our experiments. The same initializations are applied while running TMF for generating prediction results for all subgroups and the whole set. When it comes to merging the prediction results, we used the hard weight for δ_{ik} , i.e. 0 or 1, as Xu et al. [30].

For CMF, the key parameters is almost the same as TMF, except content information i.e. k. In order to incorporate item attributes we initialized first k columns of Q to be the corresponding item's attributes. Moreover, we initialized the first k columns of P to be zero. We used the same values for K, λ and γ, i.e. K= 5, λ = 0.02 and γ = 0.0002. Since 19 genres are available in MovieLens dataset we used k = 19.

Lastly; for our method CGMF, we used the same parameter values of GMF while discovering subgroups to generate recommendations. However, notice that CGMF uses CMF in order to generate recommendations for all subgroups and the whole

training set, while GMF uses TMF. We chose c = 3, 10, 20 and respectively k = 2, 4, 5. While applying CMF to the subgroups and the whole set we used the same parameter values as in CMF. The same merging strategy is used as in GMF.

4.4 Results and Discussions

We ran the four recommendation methods under the same circumstances and compared their MAE, RMSE, Precision@10 and MAP values.

We considered TMF results as baseline, in order to compare the other three algorithms. MAE and RMSE values of the four algorithms are presented in Table 4.1. The table presents the MAE and RMSE values that are obtained respectively by subgroup numbers 3, 10 and 20. Since the TMF and CMF are independent from the subgroup number, their values are constant for all of the subgroups.

Table 4.1 shows that both GMF and CMF algorithms improve the prediction accuracy when compared to TMF. Also increasing the number of subgroups increases GMF algorithm's prediction accuracy. When it comes to our approach, it reveals a dramatic improvement in both MAE and RMSE values. A detailed comparison of MAE and RMSE values are presented in Table 4.1.

Table 4.1: MAE and RMSE values.

	c = 3		c = 10		c = 20	
Method	MAE	RMSE	MAE	RMSE	MAE	RMSE
TMF	0.792	1.165	0.792	1.165	0.792	1.165
GMF	0.714	0.941	0.663	0,768	0.689	0.845
CMF	0.78	1.03	0.78	1.03	0.78	1.03
CGMF	0.665	0.770	0.603	0.633	0.632	0.731

Also the four algorithms are compared in terms of MAE and RMSE in the same Chart, in order to understand the improvements more clearly. They are presented in Chart 4.1 and Chart 4.2 respectively. We see from charts that our approach show very good performance in terms of recommendation accuracy.

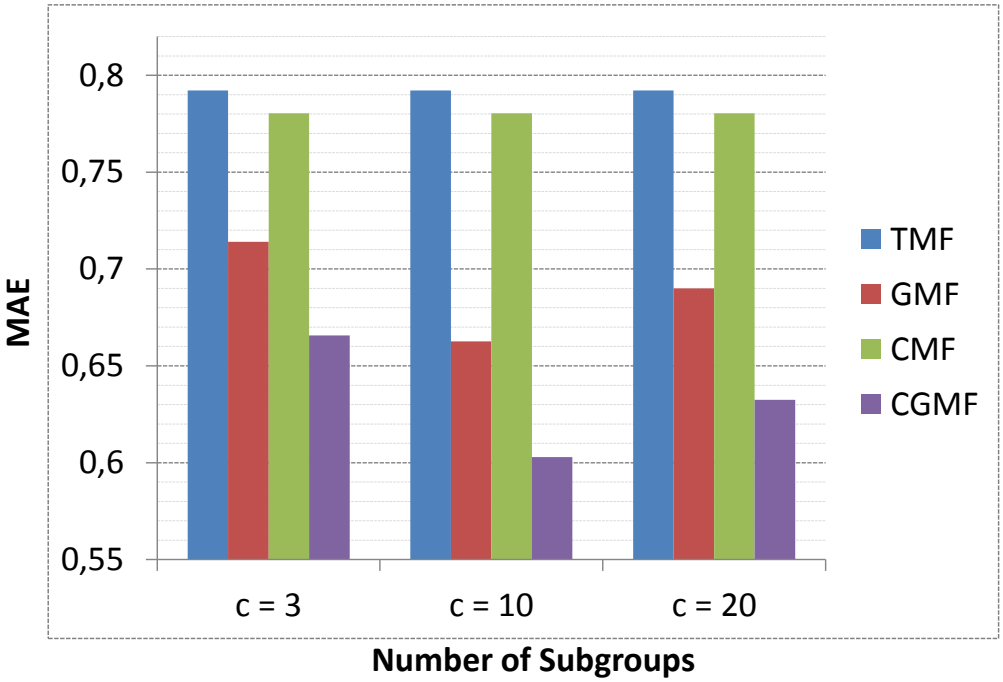


Figure 4.1: MAE values of four algorithms.

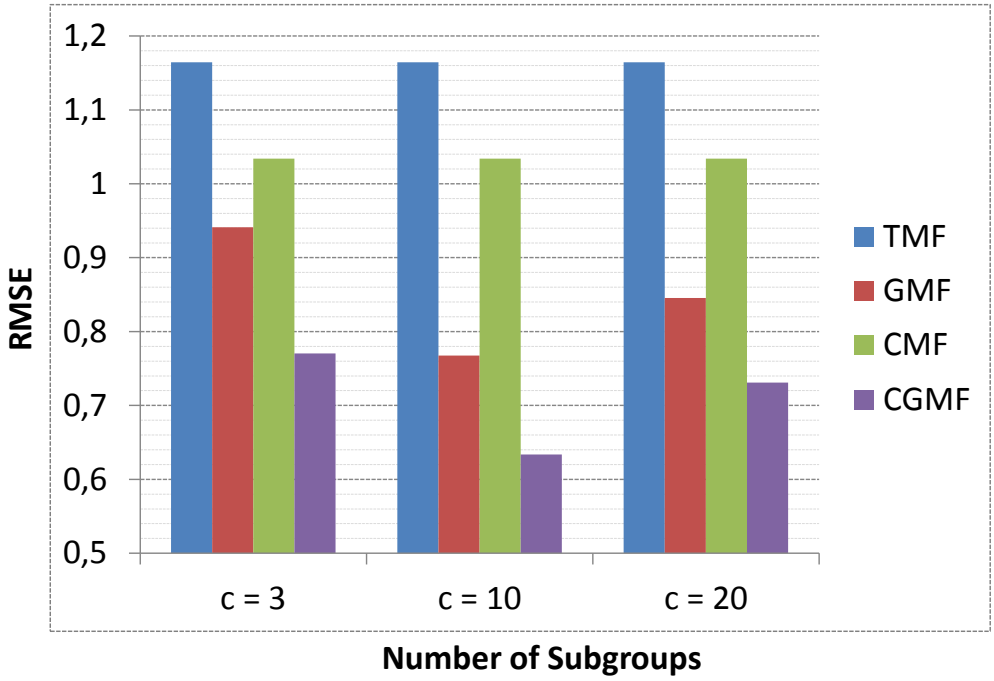


Figure 4.2: RMSE values of four algorithms.

Similarly, in order to compare the quality of top 10 items, MAP and Precision@10 values of all algorithms are presented in Table 4.2. The table presents the MAP and Precision@10 values that are obtained by subgroup numbers 3, 10 and 20, respectively. Since the TMF and CMF are independent from the subgroup number, their values are constant for all of the subgroups.

Table 4.2: MAP and Precision@10 values.

	c = 3		c = 10		c = 20	
Method	MAP	Prec@10	MAP	Prec@10	MAP	Prec@10
TMF	0.0144	0.0062	0.0144	0.0062	0.0144	0.0062
GMF	0.0167	0.0063	0.0194	0.0073	0.0213	0.0079
CMF	0.0214	0.0083	0.0214	0.0083	0.0214	0.0083
CGMF	0.0224	0.0089	0.0227	0.0084	0.0246	0.0096

Also the four algorithms are compared in terms of MAP and Precision@10 in the same Chart, in order to understand the improvements more clearly. They are presented in Chart 4.3 and Chart 4.4 respectively. TMF algorithm's MAP and Precision@10 values are weak when compared to the algorithms used by Xu et al. [30]. Therefore the performance of the other three algorithms are also weak since they are based on the TMF. However, it can be seen from the charts 4.3 and 4.4, that our approach increases MAP and Precision@10 values when compared to other three algorithms.

By using CGMF, ultimate recommendations are obtained from merging the recommendations that are obtained from subgroups and the recommendations obtained from applying Content-Boosted Matrix Factorization on the whole training matrix without subgroups. While conducting experiments we realized that the recommendations originated from subgroups are almost 65-70% of the total recommendations, while remaining 30-35% recommendations are obtained from the Content-Boosted Matrix Factorization applied on the whole training matrix. The percentages of the recommendation origins are presented in Table 4.3 for the subgroup numbers 3, 10 and 20.

Table 4.3: Percentage of subgroup vs. non-subgroup originated recommendation.

Method	c = 3		c = 10		c = 20	
	Sub	Non-Sub	Sub	Non-Sub	Sub	Non-Sub
GMF	61	39	84	16	72	28
CGMF	64	36	87	13	73	27

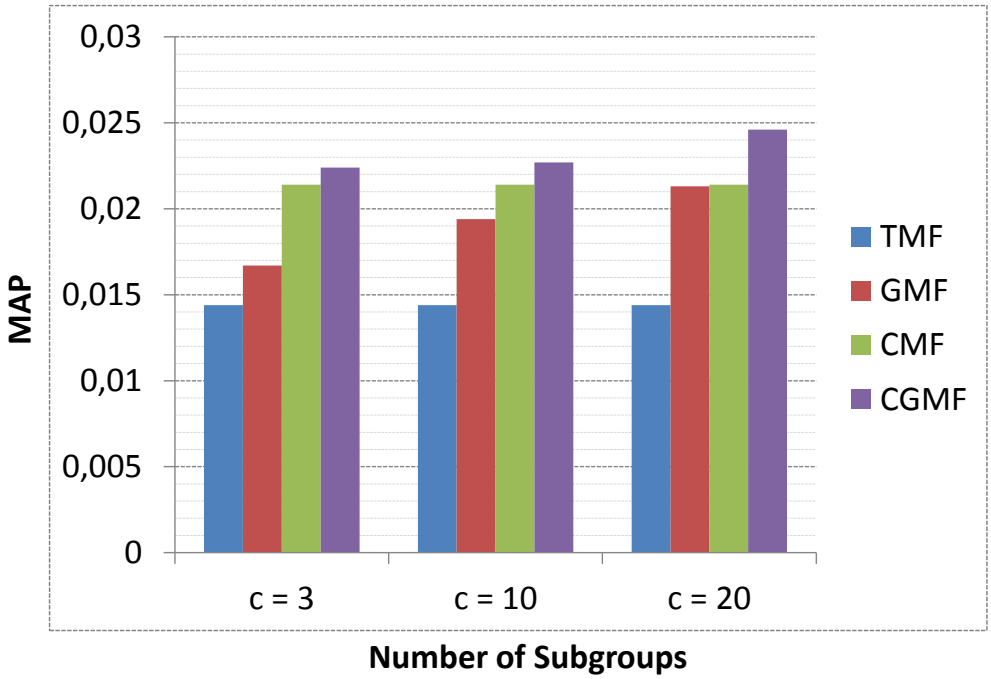


Figure 4.3: MAP values of four algorithms.

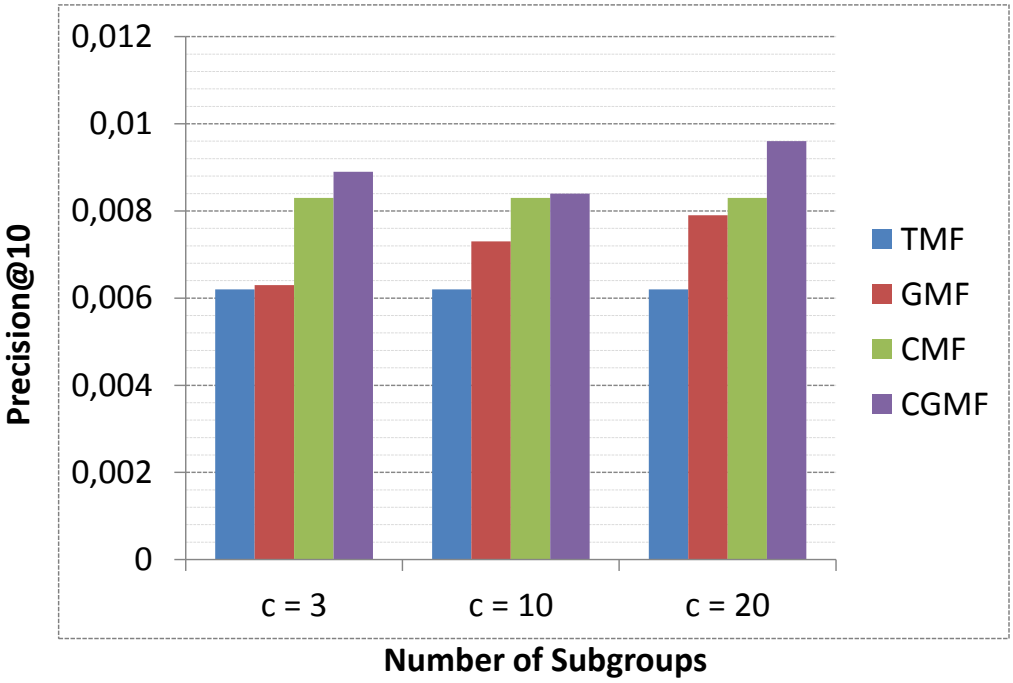


Figure 4.4: Precision@10 values of four algorithms.

CHAPTER 5

CONCLUSION

CF algorithms lack incorporating content information into recommendation process and group like-minded users based solely on same ratings to the same items, without thinking that users' taste may change according to the available item set. Hence we proposed an approach that offers a solution to both of these problems. We use an algorithm called MCoC in order to get user-item subgroups. The users in the same subgroup share similar tastes over the items that are present in that subgroup. For the CF technique to be used on these subgroups, we employ a content-boosted matrix factorization technique. The content-boosted matrix factorization technique makes use of the genre information of the movies in the set. In other words, the genre information of the movies is directly embedded into the item factor matrix that is used in MF technique.

We first discover subgroups via MCoC technique and run content-boosted MF technique over all of these subgroups. In this manner, the prediction results of all subgroups are obtained. However, since a user or an item may belong to zero, one or more subgroups, the prediction result for the whole set is not present after this process. In order to overcome this problem we also run the content-boosted CF algorithm on the whole user-item set and merge the results in order to have missing predictions caused by user-item subgroups.

In order to evaluate our approach we compared the following four methods.

1. Traditional Stochastic Gradient Based MF (TMF)
2. Traditional Stochastic Gradient Based MF via Subgroups (GMF)

3. Content-boosted Stochastic Gradient Based MF (CMF)

4. Content-boosted Stochastic Gradient Based MF via Subgroups (CGMF)

We performed our experiments on the Movielens 100k dataset. In the evaluation process we used 5-fold cross-validation. Namely, we set the %80 of the whole set to be the training set, and remaining %20 of the set as testing set. We partitioned the dataset into five mutually exclusive sets and assign one of them as testing set and others as training set, and we do it for each partitioned set. Hence, in total we covered the whole set. We ran all algorithms with the same initial values of parameters when applicable. Also we provided different number of subgroups in order to better test our approach. To sum up, we ran each algorithm five times with different training and test sets, we initialized shared parameters in the same way, and took the average of the obtained results.

We compared the accuracy of these algorithms in terms of MAE and RMSE values obtained from the experiments explained above. We observed that both CMF and GMF methods improve recommendation accuracy when compared to TMF. Recommendation accuracy of our approach, on the other hand, that combines CMF and GMF, is very satisfying. The results of the experiments showed that our approach improves the recommendation accuracy dramatically by both incorporating content information into recommendation process, and grouping like-minded users via user-item subgroups, when compared to other three methods.

As a future work, some other content information besides genre information of movies could be incorporated in the recommendation process of content-boosted MF. Moreover, better subgroups may be discovered to make all user item tuples belong to some subgroups. Also, the merging algorithm of the prediction results may be enhanced in order to recommend user a diverse set of items.

REFERENCES

[1] Movielens, 2014.

[2] G. Adomavicius and A. 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] O. Arazy, N. Kumar, and B. Shapira. Improving social recommender systems. *IT Professional*, 11(4):38–44, 2009.

[4] R. Burke. The adaptive web. chapter Hybrid Web Recommender Systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg, 2007.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[6] P. Forbes and M. Zhu. Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 261–264, New York, NY, USA, 2011. ACM.

[7] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[8] A. Gupta, A. Mohapatra, and T. Tenneti. Towards a hybrid approach to netflix challenge. 2009.

[9] B. Hidasi and D. Tikk. Enhancing matrix factorization through initialization for implicit feedback databases. In *Proceedings of the 2Nd Workshop on Context-awareness in Retrieval and Recommendation*, CaRR '12, pages 2–9, New York, NY, USA, 2012. ACM.

[10] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[11] Q. L. R. H. R. Z. Huiyuan Chang, Dingxia Li. Content-enhanced matrix factorization for recommender systems. In *Applied Mechanics and Materials* [11], pages 1084–1089.

[12] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 135–142, New York, NY, USA, 2010. ACM.

[13] Y.-D. Kim and S. Choi. A method of initialization for nonnegative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, April 15-20, 2007, Honolulu, Hawaii, USA*, pages 537–540. IEEE, 2007.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[15] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1):24–45, 2004.

[16] T. Mahmood and F. Ricci. Improving recommender systems with adaptive conversational strategies. In C. Cattuto, G. Ruffo, and F. Menczer, editors, *Hypertext*, pages 73–82. ACM, 2009.

[17] M. Mortensen. Design and evaluation of a recommender system. 2007.

[18] J. Nguyen and M. Zhu. Content-boosted matrix factorization techniques for recommender systems. *Statistical Analysis and Data Mining*, 6(4):286–301, 2013.

[19] A. Oghabian, S. Kilpinen, S. Hautaniemi, and E. Czeizler. Biclustering methods: Biological relevance and application in gene expression analysis. *PloS one*, 9(3):e90801, 2014.

[20] U. Panniello, A. Tuzhilin, and M. Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65, Feb. 2014.

[21] S. A. P. Parambath. Matrix factorization methods for recommender systems, 2013.

[22] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[23] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Mar. 1997.

[24] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[25] J. B. Schafer, J. A. Konstan, and J. Riedl. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, Jan. 2001.

[26] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[27] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, June 2009.

[28] Wikipedia. Collaborative filtering Wikipedia, the free encyclopedia, 2014. [Online; accessed 20-November-2004].

[29] Wikipedia. Mean average precision Wikipedia, the free encyclopedia, 2014. [Online; accessed 20-November-2004].

[30] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 21–30, New York, NY, USA, 2012. ACM.

[31] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. A fast parallel sgd for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 249–256, New York, NY, USA, 2013. ACM.