The Pennsylvania State University
The Graduate School
Harold and Inge Marcus Department of Industrial and Manufacturing Engineering

MATRIX FACTORIZATION METHOD FOR LAGRE
RECOMMENDATION SYSTEM

A Thesis in
Industrial Engineering
by
Yunhang Li

© 2015 Yunhang Li

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2015
The thesis of Yunhang Li will be reviewed and approved* by the following:

Tao Yao  
Associate Professor of Industrial Engineering  
Thesis Adviser

Robert A. Novack  
Associate Professor of Business Logistics

Lingzhou Xue  
Associate Professor of Statistics

Harriet B. Nembhard  
Department Head, Industrial and Manufacturing Engineering

*Signatures are on file in the Graduate School
Abstract

Recommendation system is a very popular topic in recent years. It’s very efficient to utilize machine learning algorithms to improve the performance of recommendation systems. In this thesis, the Matrix Factorization (MF) is discussed including its basic model and some extensions: regularized MF and neighbor based MF. The process of processing data and effort to improve the process are also presented.
# TABLE OF CONTENTS

## CHAPTER 1 INTRODUCTION

1.1 Content Based Recommendation System ................................................................. 1
1.2 Dynamic Assortment Planning ....................................................................................... 2
1.3 Matrix Factorization Methods ....................................................................................... 5
1.4 Netflix Prize ................................................................................................................. 7

## CHAPTER 2 PROBLEM DESCRIPTION

2.1 Matrix Factorization Methods ....................................................................................... 10
   2.1.1 Basic MF ............................................................................................................. 11
   2.1.2 Regularized MF .................................................................................................. 13
   2.1.3 BRISMF ............................................................................................................ 14
   2.1.4 Neighbor based correction of Matrix Factorization .............................................. 14

## CHAPTER 3 CASE STUDY

3.1 Data Reading .............................................................................................................. 17
3.2 Model Realization ...................................................................................................... 21
3.3 Parameter Optimization .............................................................................................. 23

## CHAPTER 4 CONCLUSION

........................................................................................................................................ 25

## APPENDIX CODE

........................................................................................................................................ 26

## REFERENCES

........................................................................................................................................ 29
Chapter 1

INTRODUCTION

Recommender systems attempt to profile user preferences over items and models and the relation between users and items. The task of recommender systems is to recommend items that fit the user’s taste, in order to help the user in selecting/purchasing items from an overwhelming set of choices. Such systems have great importance in applications such as e-commerce, subscription-based services, information filtering, etc. Recommender systems providing personalized suggestions greatly increase the likelihood of a customer making a purchase compared to unpersonalized ones. Personalized recommendations are especially important in markets where the variety of choices is large, the taste of the customer is important, and typically the price of the items is modest. Typical areas of such services are mostly related to art (esp. books, movies, music), fashion, food & restaurants, gaming & humor.

With the growing significance of e-commerce, an increasing number of web-based merchant and rental services use recommender systems. Some of the major participants of e-commerce web, like Amazon.com and Netflix, successfully apply recommender systems to deliver automatically generated personalized recommendation to their customers. The importance of a good recommender system was recognized by Netflix, which led to the announcement of the Netflix Prize (NP) competition to motivate researchers to improve the accuracy of their recommender system called Cinematch.
1.1 Content Based Recommendation System

Content Recommendation is a new kind of online services that complement search by allowing media sites to direct users from articles they are currently reading to other web-based content they may be interested in consuming.

Besbes, Gur and Zeevi (2014) recently come up with a dynamic recommendation model in which users interact with the provider along their browsing path. In a myopic approach, which is often used in practice, click through rate (CTR) is the key indicator of the performance. However, they demonstrate CTR has an important limitation. There is significant value to be captured by accounting for the future path of users and one may do so in a relatively simple and practical fashion. They model and demonstrate the value of a new dimension of content through predictive analytics and develop heuristics to optimize dynamic recommendations based on this new dimension. Then they validate the value of these heuristics theoretically.

Besides this novel approach, there are two classes of methods solving recommendation system problem raised by previous researches. One of them takes the perspective of consumer, focusing on the main objective of maximizing the probability to click on a recommendation.

Decision tree learners build a decision tree by recursively partitioning text documents into subgroups until those subgroups contain only instances of a single class. A partition is formed by a test on some feature, such as the presence or absence of an individual word or phrase. Expected information gain is a commonly used criterion to select the most informative features for the partition tests. However, the decision tree bias is not ideal for unstructured text classification tasks. RIPPER is a rule induction algorithm closely related to decision trees that
supports multi-valued attributes, which leads to more powerful classifiers for semi-structured text documents.

The Nearest Neighbor Algorithm classifies a new item by comparing it to all stored items using a similarity function and determines the "nearest neighbor". The similarity function used by the nearest neighbor algorithm depends on the type of data. The vector space approach and the cosine similarity function have been applied to several text classification applications. It’s quite simple compared to performances with more complex algorithms.

Relevance Feedback allows users to rate documents returned by the retrieval system with respect to their information need, which can be used to incrementally refine the initial query. Rocchio’s algorithm is a widely used relevance feedback algorithm that operates in the vector space model. The algorithm is based on the modification of an initial query through differently weighted prototypes of relevant and non-relevant documents. Besides, empirical experiments have demonstrated that the approach leads to significant improvements in retrieval performance.

Linear Classifiers are algorithms that learn linear decision boundaries. There are a large number of algorithms and many of them have been successfully applied to text classification tasks. One of the algorithms that perform well with many features is exponential gradient (EG) algorithm. Kivinen7 prove a bound for EG’s error, which depends only logarithmically on the number of features. The important advantage of linear classifier is they can be performed on-line, while their hyper-plane’s generalization performance might not be optimal. Support vector machine is an approach to improve generalization performance.

Naïve Bayes is another well-performed algorithm used in recent research. There are two frequently used formulations of Naïve Bayes, the multivariate Bernoulli and the multinomial
model. Empirically, the multinomial naïve Bayes formulation was shown to outperform the multivariate Bernoulli model and this effect is noticeable in large vocabularies.89

Another common class of algorithms focuses on collaborative filtering. The fundamental assumption of CF is that if users X and Y rate n items similarly, or have similar behaviors, and hence will rate or act on other items similarly. There are some challenges for collaborative filtering: CF algorithms are required to have the ability to deal with highly sparse data, to scale with the increasing numbers of users and items, to make satisfactory recommendations in a short time period, and to deal with other problems like synonymy, shilling attacks, data noise, and privacy protection problems.

In Memory-based CF algorithms, every user is part of a group of people with similar interests. By identifying the so-called neighbors of a new user, a prediction of preferences on new items for him or her can be produced. Similarity computation between items or users is a critical step in memory-based collaborative filtering algorithms. There are many measures including Correlation-Based Similarity, Vector Cosine-Based Similarity and so on. Extensions of these algorithms include Default Voting, Inverse User Frequency, Case Amplification, Imputation-Boosted CF Algorithms and Weighted Majority Prediction10.

Model-based CF algorithms include Bayesian Belief Net CF Algorithms, where the simple Bayesian CF has worse predictive accuracy but better scalability than the Pearson correlation-based CF, and is less time-consuming11. TAN-ELR and NB-ELR have been proven to have high classification accuracy for both complete and incomplete data.12 Clustering models have better scalability than typical collaborative filtering methods because they make predictions within much smaller clusters rather than the entire customer base but its recommendation quality
is generally low. Markov decision processes (MDPs) Based CF Algorithms is conducted by considering the recommendation system as sequential optimization problem. Latent semantic CF technique relies on a statistical modeling technique that introduces latent class variables in a mixture model setting to discover user communities and prototypical interest profiles.

To take advantage of properties of different models, Hybrid Collaborative Filtering Techniques are often considered. The recommendation performances of algorithms combining memory-based and model-based CF are generally better than some pure memory-based CF algorithms and model-based CF algorithms. Probabilistic memory-based collaborative filtering (PMCF) and Personality diagnosis (PD) are two major techniques in this category.

1.2 Dynamic Assortment Planning

Dynamic Assortment Planning is such a system which can allows revisiting assortment decisions at each point in time as more information is collected about initially unknown demand/consumer preferences. The central to this is topic is to balance the trade-off between information collection (exploration), which leads to a clearer picture of demand, and revenue maximization (exploitation), that strives to make optimal assortment decisions at each point in time.

Caro and Gallien (2007) are wildly considered the first to study this problem. They study a finite horizon multi-armed bandit model with several plays per stage and Bayesian learning and then present a closed-form solution for this model with learning that only requires knowledge of the two first moments of demand. In their formulation, customer demand for a product is independent of demand and availability of other products, the rate of demand is constant throughout the selling season, and perfect inventory replenishment is assumed. Besides,
the model considers only the assortment problem faced by a single store. They derive bounds on
the value function and propose an index-based policy that is shown to be near optimal when
there is some prior information on demand.

Rusmevichientong et al. (2010)\textsuperscript{19} use the multinomial logit choice model to represent
demand, and learn the demand distribution by offering different product assortments, observing
resulting selections, and inferring the demand distribution from past selections and assortment
decisions. In their study, a solution to the Capacitated MNL problem (where a capacity constrain
exists) is found first by assuming mean utilities are known in advance. Then they adapt the
algorithm to the setting when the mean utilities are unknown and must be estimated from
historical data. They introduce a parameter estimation technique based on maximum likelihood
estimation (MLE) and develop new error bounds that relate the quality of the parameter
estimates (based on MLE) as a function of the number customers.

Honhon et al. (2012)\textsuperscript{20} raise a discrete-time dynamic program to solve this problem. Each
period, the firm chooses an assortment and sets prices to maximize the total expected profit over
a finite horizon. The consumers then choose a product from the assortment that maximizes their
own utility. The firm observes sales and updates beliefs in a Bayesian fashion based on the
consumer tastes. It’s presumed that the firm knows all consumer locations. The study shows that
the optimal assortment under censored information cannot be less informative than the optimal
assortment under uncensored information. One of their novel contributions is that they
demonstrate how to sensor consumer tastes information do to substitution behavior and how a
firm can control the quality of information obtained from the dynamic assortment.
Saure and Zeevi (2013)\textsuperscript{21} also use the theory of multi-armed bandit model. Their study assumes perfect inventory and replenishment while considering limited display capacity and ignores a variety of operational considerations. They establish fundamental bounds on the performance of good policies, with specially identifying the magnitude of loss relative to the oracle performance that any policy must incur and characterizing its dependence on the length of the selling horizon, the number of products, and the capacity constraint. Besides, it’s proposed a family of adaptive policies that achieve the fundamental bound. To summarize, they also come up with salient features of dynamic assortment problem distinguishing it from sequential decision making under model uncertainty.

Other related researches include: Bernstein et al. (2010)\textsuperscript{22} study the assortment problem with limited inventory by using a threshold policy.

Multi-armed Bandit Model (MAB), as the main model used in their study, is discussed in Saure and Zeevi’s another paper (2013)\textsuperscript{23}. They focus on a MAB formulation which allows for a broad range of temporal uncertainties in the rewards, while still maintaining mathematical tractability.

As a similar topic, Golrezaei et al. (2014)\textsuperscript{24} solve the problem of personalizing the assortment of products for each arriving customer by proposing a family of index-based policies coordinating the real-time assortment decisions with the back-end supply chain constraints.

1.3 Matrix Factorization Methods

The first works on the field of Collaborative Filtering (CF) have been published in the early 1990s. The Tapestry system\textsuperscript{25} used collaborative filtering to filter mails simultaneously
from several mailing lists based on the opinion of the community on readings. Over the last broad decade many CF algorithms have been proposed that approach the problem by different techniques, including similarity/neighborhood based approaches \(^{26,27}\), Bayesian networks \(^{28}\), restricted Boltzman machines (RBM) \(^{29}\), and various matrix factorization techniques \(^{30,31}\).

The NP competition boosted the interest in CF, and yielded a number of related publications. We should here mention the NP related 1st Netflix-KDDWorkshop in 2007 \(^{32}\), which brought together top contenders of the contest. The members of BellKor/KorBell team \(^2\) presented an improved neighborhood based approach in \(^{33}\), which removes the global effect from the data - can be considered as normalization - to improve the accuracy of similarity-based interpolative predictions. Paterek applied successfully various matrix factorization techniques \(^{34}\) by adding biases to the regularized MF, post-processing the residual of MF with kernel ridge regression, using a separate linear model for each movie, and by decreasing the parameters in regularized MFs. Kurucz et al showed the application of expectation maximization based MF methods for NP.

### 1.4 Netflix Prize

In this thesis, all algorithms are conducted on Netflix Prize database. Netflix Prize is a very famous competition in the area of machine learning, which attracted a lot of excellent teams to join. The reason of choosing this database is it would be easy to compare the result to other teams and evaluate the performance of the method.
In fall 2006, the movie rental company Netflix started a competition, the Netflix Prize. The goal of the competition is to design a recommender system which improves on the Netflix recommender system Cinematch by 10% with regard to the root mean squared error (RMSE) on a published database. This database contains training data in the form of about 100 million ratings from about 480,000 users on 17,770 movies. Each rating in this database is an integer between 1 and 5. A probe set is provided which can be used to test algorithms. Furthermore, Netflix published a qualifying set which consists of user-item pairs but no ratings (the items correspond to movies in this database). The ranking of a submitted solution is based on this data set. The Netflix dataset captures the difficulties of large recommender systems. First, the dataset is huge and therefore the runtime and memory usage of potential algorithms become important factors. Second, the ranking matrix is very sparse with about 99 percent of its entries being missing such that many users have voted for just a few movies. The algorithms presented in this article were tested on the Netflix dataset. However, their design is not specific to this dataset, thus the algorithms can be applied to other CF problems as well.
Chapter 2

PROBLEM DESCRIPTION

The Collaborative Filtering problem is defined as following: a set of \( I \) users and a set of \( J \) items are given. A rating record is a vector \((i, j, x_{ij})\) representing that user \( i \) rated item \( j \) as \( x_{ij} \), where \( i \in \{1, \ldots, I\} \), \( j \in \{1, \ldots, J\} \), and \( x_{ij} \in \mathcal{X} \subseteq \mathbb{R} \). Typically rating values can be binary \((\mathcal{X} = \{0, 1\})\), integers from a given range \((\text{e.g. } \mathcal{X} = \{1, 2, 3, 4, 5\})\), or real number of a closed interval \((\text{e.g. } \mathcal{X} = [-1, 10])\). We assume that a given user can rate a given item at most once. This justifies the use of subscripts \( ij \) for rating values. We are given a finite set of rating records, \( T \), which are used for the training. We refer to the set of all known \((i, j)\) pairs in \( T \) as \( \mathcal{R} \). Note that typically \( |\mathcal{R}| \ll I \cdot J \), because each user rates only a few items.

The rating values can be organized in a rating matrix \( X \) where elements indexed by \((i, j) \notin \mathcal{R}\) are unknown. In this thesis we adopt the evaluation measure of NP contest, the root mean squared error (RMSE), which is defined as:

\[
RMSE(T) = \frac{1}{|T|} \sum_{(i,j) \in T} (\hat{x}_{ij} - x_{ij})^2,
\]

where \( T(i, j) \) contains user-item pairs on which the ratings are predicted. The accuracy of predictors are evaluated on a validation set \( \nu \); naturally the ratings of \( \nu \) is not used at creation of the predictor. Since in recommendation systems the goal is to predict the user preferences from past ratings, the ratings of \( T \) precedes the ratings of \( \nu \) in time.
At the prediction of a given rating we refer to the user as active user, and to the movie as active movie. Superscript hat denotes the prediction of the given quantity, that is, \( \hat{x} \) is the prediction of \( x \).

### 2.1 Matrix Factorization Methods

MF techniques approximate \( X \) as a product of two much smaller matrices:

\[
X \approx UM
\]

where \( U \) is an \( I \times K \) and \( M \) is a \( K \times J \) matrix.

#### 2.1.1 Basic MF

In the case of the given problem, \( X \) has many unknown elements which cannot be treated as zero. For this case, the approximation task can be defined as follows. Let \( U \in \mathbb{R}^{I \times K} \) and \( M \in \mathbb{R}^{K \times J} \). Let \( u_{ik} \) denote the elements of \( U \), and \( m_{kj} \) the elements of \( M \). Let \( u_i^T \) denote a row of \( U \), and \( m_j \) a column of \( M \). Then:

\[
\hat{x}_{ij} = \sum_{k=1}^{K} u_{ik} m_{kj} = u_i^T m_j
\]

\[
e_{ij} = x_{ij} - \hat{x}_{ij} \text{ for } (i, j) \in \mathbb{R}
\]

\[
e'_{ij} = \frac{1}{2} e_{ij}^2,
\]

\[
SSE' = \frac{1}{2} SSE = \sum_{(i, j) \in \mathbb{R}} e'_{ij}
\]
\[
RMSE = \sqrt{\frac{SSE}{|\mathbb{R}|}}
\]

\[
(U^*, M^*) = \arg \min_{(U, M)} SSE = \arg \min_{(U, M)} SSE = \arg \min \text{RMSE}
\]

Here \(\hat{x}_{ij}\) denotes how the \(i\)-th user would rate the \(j\)-th movie, according to the model, \(e_{ij}\) denotes the training error on the \((i,j)\)-th rating, and SSE denotes the sum of squared training errors. The equation states that the optimal \(U\) and \(M\) minimizes the sum of squared errors only on the known element of \(X\).

In order to minimize RMSE, which is equivalent to minimize SSE’, we have applied a simple incremental gradient descent method to find a local minimum of SSE’, where one gradient step intends to decrease the square of prediction error of only one rating, or equivalently, either \(e^'_{ij}\) or \(e^2_{ij}\). Suppose we are at the \((i,j)\)-th training example, \(x_{ij}\) and its approximation \(\hat{x}_{ij}\) is given.

We compute the gradient of \(e^'_{ij}\):

\[
\frac{\partial}{\partial u_{ik}} e^'_{ij} = -e_{ij} \cdot m_{kj}
\]

\[
\frac{\partial}{\partial m_{kj}} e^'_{ij} = -e_{ij} \cdot u_{ik}
\]

We update the wrights in the direction opposite of the gradient:

\[
u^'_{ik} = u_{ik} + \eta \cdot e_{ij} \cdot m_{kj}
\]

\[
m^'_{kj} = m_{kj} + \eta \cdot e_{ij} \cdot u_{ik}
\]
that is, we change the weight in $U$ and $M$ to decrease the square of actual error, thus better approximating $x_{ij}$. Here $\eta$ is the learning rate. We refer to the method as Basic MF.

### 2.1.2 Regularized MF

In some cases of basic MF, some extreme predicted values would appear. For example in a case with range of (1,5), it’s possible to get a value of 10 from the model. Apparently such large feature values should be avoided. The common way of overcoming this consists in applying regularization by penalizing the square of Euclidean norm of weights, which results in a new optimization problem:

$$e'_{ij} = (e^2_{ij} + \lambda \cdot u_i^T \cdot m_k + \pi \cdot m_j^T \cdot m_k) / 2$$

$$\text{SSE}' = \sum_{(i,j) \in \mathbb{R}} e_{ij}'$$

$$(U^*, M^*) = \arg \min_{(U,M)} \text{SSE}'$$

Note that minimizing $\text{SSE}'$ is no longer equivalent to minimizing $\text{SSE}$. Similar to the Basic MF approach, we compute the gradient of $e'_{ij}$, and update the weights in the direction opposite of the gradient:

$$\frac{\partial}{\partial u_{ik}} e'_{ij} = -e_{ij} \cdot m_{kj} + \lambda \cdot u_{ik}$$

$$\frac{\partial}{\partial m_{kj}} e'_{ij} = -e_{ij} \cdot u_{ik} + \lambda \cdot m_{kj}$$
\[ u'_{ik} = u_{ik} + \eta \cdot (e_{ij} \cdot m_{kj} - \lambda \cdot u_{ik}) \]

\[ m'_{ij} = u_{ik} + \eta \cdot (e_{ij} \cdot u_{ik} - \lambda \cdot m_{ij}) \]

2.1.3 BRISMF

There is a simple way to boost the performance of Regularized MF, by fixing the first column of \( U \) and the second row of \( M \) to the constant value of 1. Under the expression “fixing to a constant value” we mean not to apply equations when updating \( u_{i1} \) and \( m_{2j} \), and in the initialization, to assign them the constant value instead of random values. The pair of these features (\( m_{ij} \) and \( u_{i2} \)) can serve as a bias feature. This simple extension speeds up the training phase and yields a more accurate model with better generalization performance. We refer to this method as BRISMF that stands for Biased Regularized Incremental Simultaneous MF. BRISMF is a special way of inserting constant values: all the inserted values are 1-s. We remark that the bias feature idea was mentioned also by Paterek.

2.1.4 Neighbor based correction of Matrix Factorization

The matrix factorization and the neighbor based (NB) approaches complement each other well:

The MF approach views the data from a high level perspective. MF can identify the major structural patterns in the ratings matrix. An appealing property of MF is that it is able to detect the similarity between two items, even if no user rated both of them.
The NB approach is more localized. It is typically good at modeling pairs of users/items and not so good at modeling interdependency within larger sets of users/items. NB methods are memory based, therefore they do not require any training.

It is known that the combination of MF and NB can lead to very accurate predictions. However, conventional NB methods scale up poorly for large problems. The price of additional accuracy is the loss of scalability.

Here we propose a scalable scheme for using the MF and NB approaches together. The idea is that we improve and existing MF model \((U, M)\) by adding an item neighbor based correction term to its answer in the prediction phase. The corrected answer for query \((i, j)\) is the following:

\[
\hat{r}_{ij} = u_i^T m_j + \gamma \frac{\sum_{k \in T_i \setminus \{j\}} s_{jk} (u_i^T m_k - r_{ik})}{\sum_{k \in T_i \setminus \{j\}} s_{jk}}
\]

where \(s_{jk}\) is the similarity between items \(j\) and \(k\), and \(T_i\) is the set items rated by user \(i\). The weight of the correction term \(\gamma\) can be optimized via cross-validation.

The similarity \(s_{jk}\) can be defined in many different ways. Here are two variants that proved to be useful for Netflix Prize problem.

(S1): Normalized scalar product based similarity.

\[
s_{jk} = \left( \frac{\sum_{l=1}^{K} m_{ij} m_{lk}}{\sqrt{\sum_{l=1}^{K} m_{ij}^2} \cdot \sqrt{\sum_{l=1}^{K} m_{lk}^2}} \right)^\alpha
\]
(S2): Normalized Euclidean distance based similarity.

\[ s_{jk} = \left( \frac{\sum_{l=1}^{K} (m_{lj} - m_{lk})^2}{\sqrt{\sum_{l=1}^{K} m_{lj}^2} \cdot \sqrt{\sum_{l=1}^{K} m_{lk}^2}} \right)^\alpha \]

In both case, the value \( s_{jk} \) can be calculated in \( O(K) \) time, thus \( \hat{r}_{ij} \) can be calculated in \( O(K \cdot |T_i|) \). We remark that one can restrict to use only the top \( S \) neighbors of the queried item, however it does not affect the time requirement, if we use the same function for \( s_{ij} \) and neighbor-selection.
Chapter 3

CASE STUDY

3.1 Data Reading

The training data is composed of 17,770 txt files, representing 17,770 movies. In each file, customer reviews of this movie are recorded. The first line of each file is the number of movie, which should be eliminated in data reading process. Each record is consisted of three columns: user, rate and date of review. An overview of the data file is showed below.

1:
822109, 5, 2005-05-13
885013, 4, 2005-10-19
30878, 4, 2005-12-26
823519, 3, 2004-05-03
893988, 3, 2005-11-17
124105, 4, 2004-08-05
1248029, 3, 2004-04-22
1842128, 4, 2004-05-09
2238063, 3, 2005-05-11
1503895, 4, 2005-05-19
2207774, 5, 2005-06-06
2590061, 3, 2004-08-12
2442, 3, 2004-04-14
543865, 4, 2004-05-28
1209119, 4, 2004-03-23
804919, 4, 2004-06-10
1086807, 3, 2004-12-28
1711859, 4, 2005-05-08

Figure 3.1 Raw txt data file

One way to read these data is SQL. The advantage of SQL is that it’s designed for multidimensional data. If all data are read well, it would be very convenient for users to conduct conditional inquires. Besides, the interface and operations of SQL are relatively easy. Reading data from external txt files was the first problem and it turned out to be very easy. The next problem was how to bulk read data and store them appropriately. Here the disadvantage of SQL showed up. It was complicated to make LOOP in SQL, compared to other languages. Since that
was the basis of bulk operation, there would be more issues in the future. SQL was abandoned finally due to its inconvenience.

```
mysql> select * from m00001
    -> limit 50;
+-------+---+-----------------+
| user  | rate| date            |
+-------+---+-----------------+
| 822109|  5 | 2005-05-13      |
| 885013|  4 | 2005-10-19      |
| 30878 |  4 | 2005-12-26      |
| 823519|  3 | 2004-05-03      |
| 893900 | 3  | 2005-11-17      |
| 124105 | 4  | 2004-08-05      |
| 1248029| 3  | 2004-04-22      |
| 1842128| 4  | 2004-05-09      |
| 2238063| 3  | 2005-05-11      |
| 1503895| 4  | 2005-05-19      |
| 2208774| 5  | 2005-06-06      |
| 2570061| 3  | 2004-08-12      |
| 2442  | 3  | 2004-04-14      |
| 543865 | 4  | 2004-05-28      |
| 1209119| 4  | 2004-03-23      |
| 804919 | 4  | 2004-06-10      |
| 1086807| 3  | 2004-12-28      |
| 1711859| 4  | 2005-05-08      |
| 372233 | 5  | 2005-11-23      |
| 1080361| 3  | 2005-03-28      |
| 1245640| 3  | 2005-12-19      |
| 558634 | 4  | 2004-12-14      |
| 2165002 | 4  | 2004-04-06      |
| 1181550| 3  | 2004-02-01      |
| 1227322 | 4  | 2004-02-06      |
+-------+---+-----------------+
```

**Figure 3.2** Importing data with SQL

The next available tool was SAS. This is a very mature application in the area of data analysis. It provides plenty of expressions to let user process data. SAS supports macro variables, which makes bulk-reading data much easier. In this process, the macro variable was used for identifying the file name, table name and then inputting the movie number in the table.
The final objective of manipulating data was to create a rate matrix. Rows represented users and columns were for movies in the matrix. All rates were recorded in it. These created tables recorded the numbers of row and column of each record, followed by its value. By concatenating data from different tables, a combined table was created. In this database, the numbers of movies are consecutive from 1 to 17,770. But the numbers of user are randomly assigned. Therefore, the number of movie can be easily used as column number, while user numbers had to be transferred to row numbers. The process of sorting was needed to make renaming effective.

**Figure 3.3** Importing data with SAS
The next step was to transfer it into a rate matrix used in the training model. Matlab was used in this process. Basically the sequence of user was the order with which records are in the matrix. And the movie number was directly used as the column number. When determining the row number, it was noticed if the customer of current record is as the same as the last one, rate would be put in the same row as before. Otherwise, a new row would be created. Then the matrix was obtained.
The final matrix is showed above. Since the customer generally rates only several movies, most values in the matrix are 0 (missing value). The row numbers here are not corresponding to the actual user numbers in the database. They just represent the relative positions in the chosen dataset. An index is need for looking up the relations between these two groups of numbers.

3.2 Model Realization

After importing the matrix into the model, it was noticed that the result was quite confusing. Almost all predictions were around 0. Following is the partial prediction result of 1000 customers’ expected rates on 50 films.
The reason was the gap between the ideal condition and the reality. In the first version model, it was assumed that there is no missing value in the initial matrix, which means, in the training set, every customer rates all of the movies. However, in reality, each customer could only rate several films so that other positions would be missing. When importing the Netflix dataset, all missing values were recorded as 0. The learning process tried to make all predicted values close to accurate values as much as possible, no matter whether it’s 0 or not. So the final results of these missing positions were quite close to 0. Apparently, they were not what we expect, since the main mission of recommendation system is to predict these missing values.

Actually, the learning process just focuses on those non-missing values. There is a finite set of user-movie pairs representing all training data. All learning activities happen in this set. These activities include calculating the total error, updating items in user and product matrix.

It was necessary to create an index of these pairs before iterations. The index was composed of two columns recording the numbers of row and columns of training data. This array then was used as the reference for conducting the iteration. In later loops, only items indicated by

---

**Figure 3.6** Results of the first try

<table>
<thead>
<tr>
<th>Column 1 through 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
<tr>
<td>0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000</td>
</tr>
</tbody>
</table>
the index were used for learning, while other items were ignored. In this way, a valid prediction was created after 1225 iterations and the partial results are shown below. Notice that values in column 7 are close 0. This is because none of these 1000 customers have rated Film 7.

![Figure 3.6 Results of the final try](image)

### 3.3 Parameter Optimization

There are several parameters to be optimized. The combination of these parameters significantly influences the performance of the model, including accuracy, speed. Practically, learning rate $\eta$ and penalizing coefficient $\lambda$ are the most important among these parameters. In most of papers, accuracy is measured by Root Mean Squared Error (RMSE). A series of cycles were conducted to see the best parameter combination.
<table>
<thead>
<tr>
<th>η</th>
<th>RMSE</th>
<th>λ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.015</td>
</tr>
<tr>
<td>0.001</td>
<td>0.001</td>
<td>0.3300</td>
</tr>
<tr>
<td>0.002</td>
<td>N/A</td>
<td>0.3246</td>
</tr>
<tr>
<td>0.003</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>0.004</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 3.1 Parameter cross-validation**

N/A means it doesn’t converge.

It is noticed that the model costs less time when η decreases, which is not shown in the table. But we can’t increase η as much as possible since the divergence might happen when η is growing. For this case, the best combination is η = 0.003 and λ = 0.001.
Chapter 4

CONCLUSION

This thesis presented matrix factorization based approaches for rating based collaborative filtering problems. It was proposed a few novel MF variants including regularized MF, neighbor based MF. I reported on the RMSE scores of the algorithms evaluated on the Netflix Prize dataset. The proposed method is shown to be very effective for the customer recommendation problem and it just takes minutes to solve the problem.
function [rmse]=regulmf
% Regularized Matrix Factorization

load Data50a;
% Test matrix %

load Ind50;
ind=ind;
% Load index %

[a b]=size(x);
c=15;
% # of features %

u=rand(a,c)*0.1;
m=rand(c,b)*0.1;

u(:,1)=0.1;
m(2,:)=0.1;
% For BRISMF %

xh=u*m;
e=x-xh;

torl=0.1;
ita=0.003;
lambda=0.001;
% Parameter Initialization %

r=length(ind)
sse(1)=0;
for i=1:a
    for j=1:b
        e2(i,j)=(e(i,j)^2+lambda*(u(i,:)*u(i,:)'+m(:,j)'*m(:,j)))/2;
    end
end
sse(2)=0;
for k=1:r
    sse(2)=sse(2)+e2((ind(k,1)),(ind(k,2)));
end
p=2;

while abs(sse(p)-sse(p-1))>torl % Iteration begins %
    for i=1:a
        for j=1:c
            e2(i,j)=e2(i,j)+ita*(e(i,j)-lambda*u(i,:)*u(i,:)'-m(:,j)'*m(:,j))/2;
        end
    end
end

rmse=sqrt(sse(2)/r);
end
if k==1
    m(k,(ind(i,2)))=m(k,(ind(i,2)))+ita*(e((ind(i,1)),(ind(i,2)))*u((ind(i,1)),k) -lambda*m(k,(ind(i,2))));
elseif k==2
    u((ind(i,1)),k)=u((ind(i,1)),k)+ita*(e((ind(i,1)),(ind(i,2)))*m(k,(ind(i,2))) -lambda*u((ind(i,1)),k));
else
    un=u((ind(i,1)),k)+ita*(e((ind(i,1)),(ind(i,2)))*m(k,(ind(i,2))) -lambda*u((ind(i,1)),k));
    m(k,(ind(i,2)))=m(k,(ind(i,2)))+ita*(e((ind(i,1)),(ind(i,2)))*u((ind(i,1)),k) -lambda*m(k,(ind(i,2))));
    u((ind(i,1)),k)=un;
end
end
end

xh=u*m;
e=x-xh;
for i=1:a
    for j=1:b
        e2(i,j)=(e(i,j)^2+lambda*(u(i,:)*u(i,:)'+m(:,j)'*m(:,j)))/2;
    end
end
p=p+1
sse(p)=0;
r=length(ind);
for k=1:r
    sse(p)=sse(p)+e2((ind(k,1)),(ind(k,2)));
end
abs(sse(p)-sse(p-1))

end

% Neighbor Based Correction %
gamma=0.5;
for i=1:a
    user=find(ind(:,1)==i);
l=size(user);
    prod=ind((user),2);
    rate=x(i,prod);
    for j=1:b
        s=zeros(l(1),1);
        s1=0;
        for h=1:l(1)
            s(h)=sqrt(sum(m(:,j).*m(:,(prod(h))))/sum(m(:,j).^2)/sum(m(:,(prod(h))).^2));
            s1=s1+s(h)*(xh(i,(prod(h)))-rate(h));
        end
        xh(i,j)=xh(i,j)+gamma*s1/sum(s(h));
    end
end
end

t = size(ind);
rmse = sqrt(sse(p)/t(1));
save result xh
end
REFERENCES

1 Omar Besbes, Yonatan Gur, Assaf Zeevi, “Optimization in Online Content Recommendation Services: Beyond Click-Through-Rates.”


34 A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD ’07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, San Jose, CA, USA, 2007.