2007

An integer programming clustering approach with application to recommendation systems

Mujing Ye
Iowa State University

An integer programming clustering approach with application to recommendation systems

by

Mujing Ye

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Olafsson, Major Professor
John Jackman
Dan Zhu

Iowa State University

Ames, Iowa

2007

UMI Number: 1447502

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Recommendation systems have become an important research area. Early recommendation systems were based on collaborative filtering, which uses the principle that if two people enjoy the same product they are likely to have common favorites. We present an alternative recommendation approach based on finding clusters of similar customers using integer programming model which is to find the minimal number of clusters subjected to several similarity measures. The proposed recommendation method is compared with collaborative filtering, and the experimental results show that it provides relatively high prediction accuracy as well as relatively small variance.

CHAPTER 1 INTRODUCTION

1.1 Motivation

Recommender systems are a specific type of information filtering (IF) technique that attempts to present to the user information items that the user is interested in. Examples of such application for recommendation systems include recommending movies, music, books, news, web pages and so on. In order to make good recommendation, the user's profile is compared to some reference characteristics, which may be from the information item (the item-based approach) or the other user's profiles (the collaborative filtering approach).

The most popular technique used in recommendation systems is collaborative filtering (CF), especially *k*-nearest neighbor collaborative filtering, such as those used by Netflix, Amazon, and other Web retailers. (Adomavicius and Tuzhilin, 2005) CF is based on the principle that if two people enjoy the same product, they're likely to have other favorites in common too. But behind this simple premise is a complex algorithm that incorporates millions of user ratings, tens of thousands of items, and ever-changing relationships between user preferences. For example, in Netflix published data set, there are over 17 thousand movie titles and over 100 million ratings from 480 thousand randomly-chosen, anonymous Netflix customers. (REF) To deal with this complexity, algorithms for recommendation systems are "trained" on huge datasets. One dataset used in Netflix's system contains the star ratings--one to five--that Netflix customers assign to movies. Using this initial information, good algorithms are able to predict future ratings, and therefore can suggest other films that an individual might like.

Clustering is the process of grouping the data into classes or clusters, so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters. (Han 2006) Several similarity measures are often used such as

Euclidean distance measures, cosine measures, correlation coefficient measure etc. However, current clustering algorithms need decision-makers to pre-estimate the number of clusters or the number of nearest neighbors, which is normally a vague parameter since we hardly know in advance the patterns of the dataset. The minimal number of clusters is a very important parameter in data mining clustering problems since it shows minimal clusters to represent the original large data set and it also can improve the accuracy of current clustering methods. Similarly, *k*-nearest neighbor CF is designed to recommend items for the user according to other *k* users that are most likely to the target user (whom the recommendation system attempts to recommend items for). Since we generally don't know *k* beforehand, the k^{th} user may be far away from the target user, hence the predicted value often has a large variance if *k* is not suitable chosen. Moreover, most CF algorithms only employ one similarity measure to determine similarity among users or items, causing that prediction may not be good in some cases due to the drawbacks of some similarity measures themselves. For example, distance-based similarity measure takes only the impact of the distance into account, cosine measure considers the direction of items which is often more used in document retrievals, while correlation coefficient measure has advantage with the scaled value between -1 and 1.

1.2 Objective

In this study, we design a new clustering method by using an integer programming model, aiming at minimizing the total number of clusters and subjected to several similarity measures. The proposed model is applied to predict the movie rates of Netflix recommendation system after dynamically generating the most correlated data set for each target customer according to his/her preference indicated by his/her historical records of movie rates. Prediction is based on other similar customers in the same cluster by employing the proposed integer programming clustering model. The results provide relatively high prediction accuracy as well as small variance compared to collaborative filtering algorithms.

1.3 Thesis Organization

Chapter 2 reviews relevant literatures about most commonly used clustering methods, typical similarity measures and recommendation systems. In chapter 3, we develop a new clustering method by using integer programming model. We report extensive experiments about the suitable conditions that the model can be applied as well as parametric analysis. In chapter 4, we further apply the proposed integer programming model to the Netflix published data set to predict the movie rates by finding clusters of similar customers. Conclusion and future work are drawn in chapter 5.

CHAPTER 2 LITERATURE REVIEW

2.1 Clustering Methods

Clustering is the unsupervised classification of instances into similar groups (clusters). The methods of clustering technique have two big categories: hierarchical clustering algorithms and partitional clustering algorithms. According to Jain, Murty and Flynn (1999) as showed in Figure 1, these also have several subcategories respectively.

A hierarchical clustering method is to group instances into a hierarchy of clusters that describes the degree of similarity between those instances. Hierarchical clustering methods can be further classified as either agglomerative or divisive, depending on if the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion. Distance among clusters for most hierarchical clustering algorithms are single-link (Sneath and Sokal 1973), complete-link (King 1967), or minimum-variance (Ward 1963; Murtagh 1984). Especially, the single-link and complete-link algorithms are most popular used. The single-link method uses the minimal distance between instances in the two clusters, while the complete-link applies the maximum of all pair-wise distances in the two clusters.

The partitional techniques usually produce clusters by optimizing a criterion function in order to organize objects into partitions, and each partition represents a cluster. The *k*-means algorithm is to employ a squared error criterion (McQueen 1967) and iteratively generate cluster centers. Several variants of *k*-mean are *k*-medoids, CLARANS and so on. The best-known graph-theoretic divisive clustering algorithm is provided by Zahn (1971), who constructs the minimal spanning tree (MST) and delete the longest MST edges in order to generate clusters. The Expectation Maximization (EM) algorithm is the representative for the mixture resolving approach with the underlying assumption that the patterns of clusters

are drawn from one of several distributions, and the goal is to identify the parameters of each distribution (Dempster et al. 1977). Other mixture resolving and model-seeking algorithms are conceptual clustering, neural network approach and so on. (Han 2006) A problem accompanying the use of a partitional algorithm is the choice of the number of desired output clusters, which is normally hard to know in advance especially for high dimensional data set.

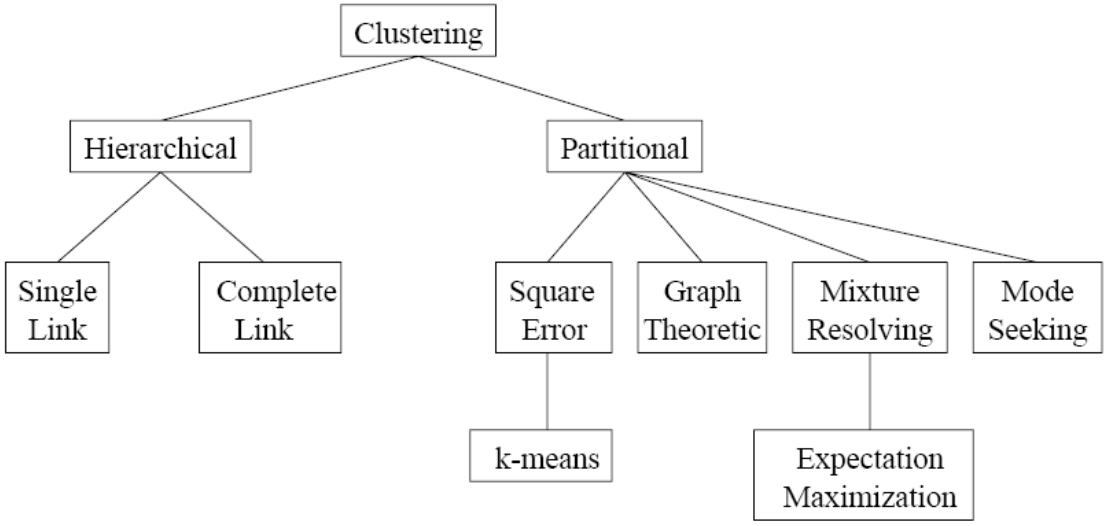

Figure 1 Clustering Methods

In addition to classical clustering methods mentioned above, there was extensive work done on optimization methods in clustering techniques as reviewed by Olafsson, Li and Wu (2006).

Vinod (1969) first provides integer programming formulations for the clustering problem. In one of his formulation, decision variables are binary to indicate if the instance is assigned to the cluster and the objective is to minimize the total assignment cost. Rao (1971) improves Vinod's work by minimizing two compactness measures, in which one measure is the within cluster sum of squares and the other, the maximum distance within clusters. Bradley et al. (1996) and Bradley and Mangasarian (2000) also provide mathematical programming formulations for the clustering problem, but they focus on identifying cluster

centers (decision variables). However, the solution from Bradley et al. (1996) turns out to be the cluster centers such that each instance in the cluster is as close as possible to the center, which means that the solution may be a cluster set where the clusters are not well separated. Bradley and Mangasarian (2000), on the other hand, employ mathematical programming to find the best cluster planes instead of the best centers. They propose an iterative algorithm, like *k*-means algorithm, to find the plane that minimize the sum of squares of distances of each instance to the cluster.

More recent work defines clustering problem as set covering (e.g. Shmoys, 1999). For example, the problem of locating the best clusters mirrors problems in facility location (Shmoys et al., 1997), and especially the *k*-center and *k*-median problems. Several other research work focus on the scalability of clustering algorithms. (Bradley et al. 1998; Ng and Han, 1994; Zhang et al., 1996)

Another clustering problem is sequential clustering, which is equivalent to the partitional clustering problem, except that the instances are ordered and this order must be maintained in the clustering. Hwang (1981) shows how to find optimal clusters for this problem, and Joseph and Bryson (1997) show how to find so called *w*-efficient solution to sequential clustering using linear programming.

2.2 Similarity Measures

Similarity is quite difficult to measure, since the quantity reflects the strength of relationship between two objects. This quantity is usually having range of either 0 to 1 or normalized into -1 to 1. If the similarity between object *i* and object *j* is denoted by s_{ij}, we can measure this quantity in several ways depending on the scale of measurement (or data type) that we have. Typical similarity measures include Euclidean distance measure, cosine measure and correlation coefficient measure. (Aldenderfer and Blashfield, 1984)

2.2.1 Euclidean distance measure

Euclidean Distance is the most common use of distance. In most cases when people talk about distance, they refer to Euclidean distance. Euclidean distance or simply 'distance' examines the *root of square differences* between coordinates of a pair of objects. This can be written mathematically as

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2},$$

where x_{ik} means the element of instance i for attribute k.

2.2.2 Cosine measure

Cosine measure, also called angle measure, actually is the cosine angle value between two objectives (vectors), which accounts for similarity rather than distance or dissimilarity. Thus, higher value of angular similarity indicates the two objects are similar. The value of angular measure is [-1, 1] similar to cosine. It is often called as *Coefficient of Correlation*. This can be written mathematically as

$$s_{ij} = \frac{\sum_{k=1}^{n} x_{ik} x_{jk}}{(\sum_{k=1}^{n} x_{ik}^2 \sum_{r=1}^{n} x_{jk}^2)^{\frac{1}{2}}},$$

where x_{ik} means the element of instance i for attribute k.

2.2.3 Correlation coefficient measure

Correlation coefficient is standardized angular separation by centering the coordinates to its mean value. The value is between -1 and +1. It measures similarity rather than distance or dissimilarity. This can be written mathematically as

$$s_{ij} = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{(\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)^2 \sum_{k=1}^{n}(x_{jk} - \bar{x}_j)^2)^{\frac{1}{2}}},$$

where

$$\bar{x}_i = \frac{1}{n}\sum_{k=1}^{n} x_{ik}\ ,\ \bar{x}_j = \frac{1}{n}\sum_{k=1}^{n} x_{jk}\ .$$

2.3 Recommendation System

Recommendation systems help customers to find the items that they would like to purchase by predicting the rate of the item or producing a list of top-N recommended items for a given customer. The work on recommendation systems can be traced back to cognitive science (Rich, 1979), information retrieval (Salton, 1989), forecasting theories (Armstrong, 2001), management science (Murthi and Sarkar, 2003) and so on. However, recommendation systems emerged as independent research area since the mid of 1990s when the focus turned to rating structure of recommendation systems. In general, many customers provide rates to different items. The space of items can be very large, ranging in hundreds of thousands or even millions of items in some applications, such as recommending books, CDs or movies. Some pairs of customer-item have rates, while others are missing (unrated), as showed in Table 1. The goal of the recommendation system is to predict the missing rates, so that we can recommend to customers the item(s) with the highest estimated rating(s).

Table 1 A Fragment of Rating Matrix for a Movie Recommendation System

Customers	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	?	4	5	5
Cindy	2	2	4	?
David	3	?	5	2

Collaborative filtering (CF) has been the most successful recommendation technique. CF is the method of making predictions or recommendation (filtering) about the interests of a user by collecting taste information from many other customers (collaborating). Suppose a list of m customers $U = \{u_1, u_2, ..., u_m\}$ and a list n items $I = \{i_1, i_2, ..., i_n\}$, where each customer u_i has a list of items I_{u_i}. Opinions are generally given by the customer as a rating sore within a certain range. Hence it forms a $m \times n$ customer-item data set as a rating matrix, **A**, where each a_{ij} in **A** represents the preference score (rating) of i^{th} customer on j^{th} item.

There have been many collaborative systems developed in the past decades. Grundy system was the first recommendation system that proposed to build models of users based on a limited amount of information on each individual user using stereotypes (Rich, 1979). Later, the Tapestry system relied on each user to identify like-minded users manually (Goldberg et al., 1992). GroupLens (Konstan et al., 1997; Resnick et al., 1994), Video Recommender (Hill et al., 1995) and Ringo (Shardanand and Maes, 1995) were the first systems to use CF algorithms to automate prediction. Other examples of collaborative recommender systems include the book recommendation system from Amazon.com and the Jester system that recommends jokes.

Two main categories were developed about CF algorithms in the past, Memory-based (user-based) algorithms and Model-based (item-based) algorithms (Breese, Heckerman and Kadie, 1998). Memory-based CF algorithms employ statistical techniques to find a set of neighbor users, which have similar historical preference with the target user. Once a neighborhood of the users is determined, then systems use different algorithms to combine the preferences of neighbors to make a prediction or top-N recommendation for the target user. Model-based CF algorithms provide item recommendation by finding similar items that co-rated by different users. The model building process can also be performed by different machine learning algorithms such as Bayesian network, clustering and rule-based approaches. The Bayesian network model provides a probabilistic model for CF problems (Breese,

Heckerman and Kadie, 1998). Clustering model considers CF problems as classification problems by clustering similar users in same class (Breese et al., 1998; Basu et al., 1998; Ungar and Foster, 1998), but those clustering methods are fallen in classical clustering techniques. The rule-based approach determines recommendation based on the strength of the association between co-purchased items applying association rule discovery algorithms (Sarwar et al., 2000).

The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering or recommendation system for music tastes could make predictions about which music a user should like given a partial list of that user's tastes (likes or dislikes).

For the user-based *k*-nearest neighbor CF, suppose rating vector for the target user to be I_{u_a}, the algorithm applies Pearson correlation coefficient or cosine-based approach to find out the most similar *k* other users. The formulas with Pearson correlation coefficient similarity measure is as follows:

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R}_i)(R_{u,j} - \overline{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R}_i)^2 \sum_{u \in U} (R_{u,j} - \overline{R}_j)^2}},$$

where $R_{u,i}$ is the rating vector of user *i* and $R_{u,j}$ is the rating vector of user *j*; $\overline{R}_i$ is the average rate for user *i* and $\overline{R}_j$ is the average rate for user *j*; *U* is a set of rates from co-rated items. The formula with cosine similarity measure is as follows:

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\left\|\vec{i}\right\|_2 \bullet \left\|\vec{j}\right\|_2}.$$

The predicted rate is defined as aggregating rating values of the most similar *k* other users to predict the rate of the target user as follows. Here, $r_{c,s}$ is the predicted rate for the customer *c* to item *s*. *C*' is the set of the most similar *k* other customers. Here, *q* is the

normalized value, normally defined as $q = 1/\sum_{c' \in \hat{C}} |sim(c,c')|$. We listed three ways of aggregation. In this thesis, we choose (b) to compute rates for the target user from the most similar *k* other users.

$$r_{c,s} = \underset{c' \in \hat{C}}{aggr}\ r_{c's}$$

$$\text{(a)}\ r_{c,s} = \sum_{c' \in \hat{C}} r_{c',s} / k$$

$$\text{(b)}\ r_{c,s} = q \sum_{c' \in \hat{C}} sim(c,c') \times r_{c',s}$$

$$\text{(c)}\ r_{c,s} = \bar{r}_c + q \sum_{c' \in \hat{C}} sim(c,c') \times (r_{c',s} - \bar{r}_{c'})$$

CHAPTER 3 MATHEMATICAL MODEL

3.1 Mathematical Formulation by QIP model

A cluster is defined to be a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. The clustering approach in this study is concerned with finding the minimal number of clusters given constraints of several similarity measures. We design to find the minimal number of clusters because decision makers often prefer to group a large data set into a smallest amount of clusters, each of which can be viewed as an individual, so as to efficiently deal with individual clusters instead of the large data set. Similarity measures including cosine measure and distance measure are two basic metrics in various clustering methods. The smaller the angle between two vectors is, the more similar they are. The less distance between two vectors, the more likely they belongs to the same cluster. As mentioned before, two traditional similarity measures, angle measure and distance measure, partially reflect the similarity of the compared objects, and they are complementary with respect to the distance and direction. The complementary feature of two measures suggests that it would be useful to consider both measures as our constraints in grouping vectors.

The number of clusters in classical clustering methods is often predetermined, like *k*-means, *k*-medoids, CLARANS and so on, or it is left users to specify the desired number of clusters as a termination condition, like agglomerative or divisive hierarchical clustering. It is seldom as a decision variable in the previous clustering approaches. So, we attempt to minimize the number of clusters as an objective function. After initially finding the optimal number of clusters, we may further apply other current clustering approaches to better cluster the data set. A quadratic integer programming clustering model is as follows:

Index sets:

NC={1,…K}, set of clusters,

NI={1,…, n}, set of instances,

NB={(i,j): angle(v_i, v_j)< b_{ij} or distance(v_i, v_j)> d_{ij} }

Problem parameters:

b_{ij}: minimal similarity requirement between vector i and vector j;

d_{ij}: maximal distance requirement between vector i and vector j;

angle(v_i, v_j): angle measure between vector i and vector j;

distance(v_i, v_j): distance measure between vector i and vector j;

M: large positive number;

K: the maximum number of clusters, normally $K \leq n$;

Decision variables:

$$Y_k = \begin{cases} 1 & \text{if cluster } k \text{ has elements} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ik} = \begin{cases} 1 \text{ if instance } i \text{ assigned to cluster } k \\ 0 \text{ otherwise} \end{cases}$$

Quadratic Integer Programming (QIP) Model:

$$\min \sum_{k \in NC} Y_k$$

s.t.

$$\text{angle}(v_i, v_j) \geq b_{ij} X_{ik} X_{jk} \quad \forall i, j \in NI \quad \forall k \in NC \qquad (1)$$

$$\text{distance}(v_i, v_j) X_{ik} X_{jk} \leq d_{ij} \quad \forall i, j \in NI \quad \forall k \in NC \qquad (2)$$

$$\sum_{k \in NC} X_{ik} = 1 \quad \forall i \in NI \qquad (3)$$

$$\sum_{i \in NI} X_{ik} \leq M Y_k \quad \forall k \in NC \qquad (4)$$

$$X_{ik}, Y_k = \{0,1\} \quad \forall i \in NI \quad \forall k \in NC \qquad (5)$$

In the above formulation, the objective function is to minimize the total number of clusters. Constraint (1) requires that angle measure should no less than b_{ij} if vector *i* and vector *j* belong to the same cluster *k*. If vector *i* and vector *j* belong to different clusters, then constraint (1) is redundant. Similarly, constraint (2) requires that vectors in the same cluster are no larger than maximal distance requirement. Constraint (3) assures that each vector can be only assigned in one cluster. The LHS of constraint (4) counts the number of element in cluster *k*. If cluster *k* has a positive number of elements, Y_k has to be 1 since *M* is a large positive number. If cluster *k* is empty, then Y_k can be either 1 or 0. But the objective is to minimize the sum of Y_k, so Y_k tends to be 0 in this case.

3.2 Example of QIP model

Suppose 3 instances, v_1, v_2 and v_3, to be assigned to 2 clusters (K=2). The distance between any two instances within the same cluster is no more than *d*, and the minimal cosine similarity is *b*. M=1000. Then the above quadratic model would be as follows:

$$\min Y_1 + Y_2$$

s.t.

$$\text{angle}(v_1, v_2) \geq bX_{11}X_{21}$$

$$\text{distance}(v_1, v_2)X_{11}X_{21} \leq d$$

$$\text{angle}(v_1, v_2) \geq bX_{12}X_{22}$$

$$\text{distance}(v_1, v_2)X_{12}X_{22} \leq d$$

$$\text{angle}(v_1, v_3) \geq bX_{11}X_{31}$$

$$\text{distance}(v_1, v_3)X_{11}X_{31} \leq d$$

$$\text{angle}(v_1, v_3) \geq bX_{12}X_{32}$$

$$\text{distance}(v_1, v_3)X_{12}X_{32} \leq d$$

$$\text{angle}(v_3, v_2) \geq bX_{31}X_{21}$$

$$\text{distance}(v_3, v_2)X_{31}X_{21} \leq d$$

$$\text{angle}(v_3, v_2) \geq bX_{32}X_{22}$$

$$\text{distance}(v_3, v_2)X_{32}X_{22} \leq d$$

$$X_{11} + X_{12} = 1$$

$$X_{21} + X_{22} = 1$$

$$X_{31} + X_{32} = 1$$

$$X_{11} + X_{21} + X_{31} \leq 1000Y_1$$

$$X_{12} + X_{22} + X_{32} \leq 1000Y_2$$

$$X_{ik}, Y_k = \{0,1\} \quad i = 1,2 \text{ or } 3 \quad k = 1 \text{ or } 2$$

If we let the decision vector be $X = (X_{11}, X_{21}, X_{31}, Y_1, X_{12}, X_{22}, X_{32}, Y_2)$, any quadratic form of $X_{ik}X_{jk}$ can be represented as $X'FX$, here F is the constant square matrix. For example,

$$X_{11}X_{21} = (X_{11}, X_{21}, X_{31}, Y_1, X_{12}, X_{22}, X_{32}, Y_2) \begin{pmatrix} 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} X_{11} \\ X_{21} \\ X_{31} \\ Y_1 \\ X_{12} \\ X_{22} \\ X_{32} \\ Y_2 \end{pmatrix}$$

Other constraints with quadratic form are similar with above example. The total amount of quadratic form is $q \times \binom{n}{2}$, where n is the total number of instances and q is types of similarity measures.

3.3 IP model

The QIP model includes a quadratic term in both constraints (1) and (2), which is very hard to handle in practice due to F often bring a many sparse matrix. Actually, if

angle(v_i, v_j) $\geq b_{ij}$ and distance(v_i, v_j) $\leq d_{ij}$, it means $X_{ik}X_{jk}$=1 or 0 for any k, otherwise $X_{ik}X_{jk}$=0. That is, if two instances, i and j, do not satisfy all similarity measures, then $X_{ik}X_{jk}$=0 for any k. Since X_{ik} or X_{jk} is binary variables, $X_{ik}X_{jk}$=0 is equivalent to $X_{ik}+X_{jk}\leq 1$. Hence, we modify the above QIP model into a linear IP model as follows. Constraint (6) separates two vectors into different clusters if the two vectors do not satisfy either the angle measure or the distance measure. The number of constraint (6) depends on the actual problem. The more compact of data, the less the number of (6). So, before solving the problem, we dynamically generate (6) by checking all similarity requirements for any two vectors. The model is modified as follows:

$$\min \sum_{k \in NC} Y_k$$

s.t.

$$X_{ik} + X_{jk} \leq 1 \quad \forall k \in NC \ \forall (i, j) \in NB \qquad (6)$$

$$\sum_{k \in NC} X_{ik} = 1 \quad \forall i \in NI \qquad (3)$$

$$\sum_{i \in NI} X_{ik} \leq MY_k \quad \forall k \in NC \qquad (4)$$

$$X_{ik}, Y_k = \{0,1\} \quad \forall i \in NI \quad \forall k \in NC \qquad (5)$$

Since the above modified model is linear (IP model), it is much easier to solve by most of mathematical software like Tomlab/CPLEX which includes an advanced Matlab interface to the state-of-the-art large-scale mixed-integer linear solver CPLEX. However, the coefficient matrix of (6) is normally large and sparse. In order to avoid "out of memory" error when running CPLEX, we need use storage of sparse coefficient matrix.

3.4 Iris data set Testing and Parametric Analysis

We try to use a classic numeric dataset, iris dataset, to test our model. The iris dataset is about seminal work collected by the eminent statistician R.A. Fisher in the mid-1930s and

is arguably the most famous dataset used in data mining. It contains 50 examples for each of three types of plant: *Iris setosa, Iris versicolor,* and *Iris virginica.* It is excerpted in Table 3 that there are four attributes: *sepal length, sepal width, petal length,* and *petal width* (all measured in centimeters). And all attributes have values that are numeric. The following set of rules might be learned from this dataset:

If petal length < 2.45 then Iris setosa

If sepal width < 2.10 then Iris versicolor

If sepal width < 2.45 and petal length < 4.55 then Iris versicolor

If sepal width < 2.95 and petal width < 1.35 then Iris versicolor

If petal length ≥ 2.45 and petal length < 4.45 then Iris versicolor

If sepal length ≥ 5.85 and petal length < 4.75 then Iris versicolor

Table 2 Iris Data

No.	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
…					
51	7	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
…					
101	6.3	3.3	6	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3	5.8	2.2	Iris-virginica
…					

We cluster the iris data set only considering distance measure for simplicity, which requires the minimal cosine similarity to be 0. From numerical experiments, we found that when

maximal distance d between 2.6 and 3.8, the IP model returned optimal 3 clusters, and when d between 3.9 and 7, the IP model returned 2 optimal clusters. If d is set too small, like 1.5, it appeared that things degenerated quite quickly to a very large number of clusters, so we couldn't find any feasible solution because the maximal number of clusters was smaller than the optimal number of clusters. It also indicates that even if we don't know the optimal solution, it should be greater than 60 when maximal distance is 1.5. When d is set too large it appeared that the two more similar clusters were simply combined (100 instances) and separated, or at least approximately separated, from the more distinct cluster (50 instances). When d was set large enough ($d \geq 7.1$), then all clusters became merged into one cluster. According to experiments, the optimization can find the correct number of clusters under a fairly wide range of distance parameters. Although the clusters are not exactly right possibly because we set all allowed maximal distance of points within one cluster the same, the primary purpose of this approach is to find the correct number of clusters. After initially finding the optimal number of clusters, we can further apply other heuristic approaches, like k-means algorithm to cluster the data set. If k=2, the k-means approach results in 2 clusters with 50×100 and k=3 with 50×50×50. The proposed model and parameter analysis also give us an intuitive image of the pattern of data set. For iris data set, we have hunch that there are 2 more similar clusters and another cluster is far away from these 2 clusters.

Table 3 Parametric Tests for Iris Data

Max No. of Clusters	Max Distance	Solution (No. of Clusters)	Optimality	No. of Cluster Elements
20	1.5	None	-	
40	1.5	None	-	
60	1.5	None	-	
25	1.8	C=8	feasible	
8	1.8	C=7	feasible	
7	1.8	C=6	feasible	
6	1.8	C=6	feasible	
6	2	C=5	optimal	
5	2.2	C=5	optimal	
5	2.4	C=4	optimal	
5	2.6	C=3	optimal	
3	2.61	C=3	optimal	51×55×44
3	2.62	C=3	optimal	50×61×39
3	2.63	C=3	optimal	51×45×54
3	2.64	C=3	optimal	50×61×39
3	2.65	C=3	optimal	51×48×51
3	2.66	C=3	optimal	37×62×51
3	2.67	C=3	optimal	50×48×52
3	2.68	C=3	optimal	35×64×51
3	2.69	C=3	optimal	51×35×64
3	2.7	C=3	optimal	51×49×50
3	2.8	C=3	optimal	46×51×53
3	2.9	C=3	optimal	53×51×46
3	3	C=3	optimal	52×24×74
3	3.1	C=3	optimal	54×67×69
3	3.2	C=3	optimal	27×52×71
3	3.3	C=3	optimal	52×25×73
3	3.4	C=3	optimal	53×22×75
3	3.5	C=3	optimal	32×41×77
3	3.6	C=3	optimal	60×49×41
3	3.7	C=3	optimal	31×58×61
3	3.8	C=3	optimal	44×19×87
3	3.9	C=2	optimal	75×75
2	4	C=2	optimal	59×91
2	4.1	C=2	optimal	56×94
2	4.2	C=2	optimal	54×96
2	4.3	C=2	optimal	54×96
2	4.4	C=2	optimal	54×96
2	4.5	C=2	optimal	54×96
2	4.6	C=2	optimal	54×96
2	4.7	C=2	optimal	54×96
2	4.8	C=2	optimal	51×99
2	4.9	C=2	optimal	50×100
2	5	C=2	optimal	50×100
2	5.1	C=2	optimal	50×100

Table 3 (Continued)

2	5.2	C=2	optimal	50×100
2	5.3	C=2	optimal	50×100
2	5.4	C=2	optimal	50×100
2	5.5	C=2	optimal	50×100
2	5.6	C=2	optimal	50×100
2	5.7	C=2	optimal	50×100
2	5.8	C=2	optimal	50×100
2	5.9	C=2	optimal	50×100
2	6	C=2	optimal	50×100
2	6.1	C=2	optimal	48×102
2	6.2	C=2	optimal	30×120
2	6.3	C=2	optimal	28×122
2	6.4	C=2	optimal	11×139
2	6.5	C=2	optimal	24×126
2	6.6	C=2	optimal	24×127
2	6.7	C=2	optimal	46×104
2	6.8	C=2	optimal	55×95
2	6.9	C=2	optimal	51×99
2	7	C=2	optimal	1×149
2	7.1	C=1	optimal	150

3.5 Numerical Experiments

We simulated 2 dimensional clusters in the square space of 5000 by 5000, within each cluster the maximal allowed distance is 20 and the minimal allowed cosine similarity is 0. We use PC computer with CPU 3GH and 3.5G RAM to test the different problem size as well as total amount of time needed. From Table 5, we found that the IP model could efficiently find the exact clusters up to about 100 instances when actual number of clusters is small (no more than 7). However, in No. 5 and No. 3, the amount of time spent in solving the problems by CPLEX exceeds its maximal set time, 300 second, so in this case those solutions are only feasible but not optimal.

Table 4 Scalability Test

No.	No. of Instances	Max No. of clusters	Simulated No. of clusters	Solution to No. of Clusters	Total Amount of Time (s)
1	44	10	2	2	0.51
2	59	10	4	4	25.47
3	100	10	6	7	300.26
4	150 (iris)	6	3	3	98.40
5	166	20	8	20	309.48

From the above experiments the IP model can only be suitable for solving a small amount of data set. In the proposed IP model, the number of constraints varies due to the characteristic of data set. When instances sparsely lie in the searching space, there would be lots of constraints (6). As the number of instances goes up, number of constraints (6) increases dramatically, so the coefficient matrix for this problem is correspondingly large and sparse. For example in 20 randomly generated 2 dimensional instances, if the maximal number of clusters is set 20, the coefficient matrix is 960 by 420 with 8380 non-zeros in the reduced MIP problem. However, for a problem that randomly generated 100 instances with maximal number of clusters 50, the coefficient matrix is 126479 by 5050 with 605620 non-zeros in the reduced MIP problem. (CPLEX first aggregates the coefficient matrix, then pre-solves and reduces the MIP problem.) Hence, when the problem size is too large, which depends on number of instances and number of maximal clusters, time spent solving the MIP problem by CPLEX would be not efficient enough and sometimes even cannot find the optimal solution.

For a problem with a large number of instances and a small number of clusters, which is generally a clustering problem in most cases, the model is still suitable as we did with iris data that has 150 instances and 3 optimal clusters. We may also iteratively try to find the optimal solution by reducing the maximal number of clusters. For example, we didn't find the optimal solution to No. 1 in Table 6 since the amount of time exceeds the limit (300) we set. However, if we can find a feasible solution (7), by setting maximal number of clusters to be 10, then we can use maximal number of clusters (7) to resolve the MIP problem, since the minimal number of clusters should be no larger than 7. Then as showed in No. 2 in Table 6, CPLEX solver needs 14.81 seconds (less than 300) to resolve the problem, so the solution found (6) is the optimal solution. Of course, if we use maximal number of clusters to be 6, CPLEX only needs 2.53 seconds, as showed in case 4. In this way, the coefficient matrix for

the problem can be largely reduced by reducing the maximal number of clusters, until the optimal cluster can be found.

Table 5 Reduce Max No. of Clusters

No.	No. of Instances	Max No. of clusters	Simulated No. of clusters	Solution to No. of Clusters	Total Amount of Time (s)
1	100	10	6	7	300.26
2	100	10	6	7	300.21
3	100	7	6	6	14.88
4	100	6	6	6	2.53

We repeat to test iris data set for 5 times and the amount of time spent on each repeat is close to 98.2 seconds with standard error only 0.14. All solutions found by CPLEX solver from 5 repeats are same. Hence, the CPELX solver is quite stable if the optimal solution can be found.

Table 6 Time Variability Test

No	No. of Instances	Max No. of clusters	Simulated No. of clusters	Solution to No. of Clusters	Total Amount of Time (s)
1	150 (iris)	6	3	3	98.12
2	150 (iris)	6	3	3	98.40
3	150 (iris)	6	3	3	98.27
4	150 (iris)	6	3	3	98.17
5	150 (iris)	6	3	3	98.05
Time ± S.E.					98.202±0.14

In order to further test the effectiveness of the IP model, we apply cluster separation to measure the compactness of data set, which provides us a quantity that how clusters are separated to each other. The separation measure we use is as follows:

$$F_{between} = \frac{1}{d}\frac{1}{K-1}\sum_{k=1}^{K}\sum_{i=1}^{n}(1-a_{ik})\left\|x_i-\gamma_k\right\|_2 .$$

Here d is the number of attributes, K is the total number of clusters, n is the total number of instances, a_{ik} is 1 if instance i belongs to cluster k, and $\left\|x_i-\gamma_k\right\|_2$ is square of distance between instance i and cluster center k. We simulated 5 different clusters with 100 instances in Table 8 in order to test variance of time needed by CPLEX solver. We can see that the

larger separation between clusters, the less amount of time spent in solving the IP model. With the increase in the density of instances in the given space, the time needed to obtain the optimal solution of the IP model will be largely increased. For the last 3 cases, problem No. 4 was solved in relatively small amount of time because each simulated cluster is well separated, while problem No. 3 and No. 5 both had two clusters out of 5 close to each other which lead to more amount of solving time.

Table 7 Separation and Total Time

No	No. of Instances	Max No. of clusters	Simulated No. of Clusters	Solution to No. of Clusters	Separation	Total Amount of Time (s)
1	100	10	5	5	4.4519e+008	14.266±0.126
2	100	10	5	5	1.6879e+008	14.286±0.104
3	100	10	5	5	6.6180e+007	139.8±0.249
4	100	10	5	5	5.1732e+007	92.66±0.167
5	100	10	5	5	4.2298e+007	276.23±3.204

3.6 Space Decomposition Technique

Decomposition technique is often used to break a complex problem down into easily-understood and achievable parts, e.g. decomposition linear programming (Bazaraa, et al., 2005), benders decomposition (Kall and Wallace, 2003), hyper-tree decomposition (Gottlob, et al., 2001) and so on. Space decomposition is to separate the search space into mutually exclusive subspaces. In each subspace plus the related boundary space associated with the previous subspace, we apply IP model to find out the minimal number of clusters. Since number of instances in each subspace is normally small and solvable by CPLEX solver, space decomposition technique becomes a good method to deal with clustering a large number of instances locating in a large space with relatively small actual number of clusters.

According to experiments in section 3.6, it is not efficient to apply IP model to a large number of instances and a large number of optimal clusters due to limited memory of the computer. However, we notice that if we decompose searching space into several subspaces, most clusters within the subspace would remain the same and all the connected subspaces

together form into the original space. For example, we may separate space of 2nd dimension for the example in section 3.4 into [0, 400], [400, 800] and [800, 1200], and only those instances near boundary of each interval would be affected the final optimal solution. If we have clustered instances within subspace [0, 400], then select those instances that have distance from boundary less near $d_{ij}/2$ (named sensitive instances), add sensitive instances as well as all instances that is fallen in the same clusters as sensitive instances into next subspace to reapply IP model to new data. After going through all subspaces, we get the same clustering results as we do for the original space. In this way the model can cluster a median size instances up to thousands of instances in reasonable time, showed in Table 9.

We also simulated 2 dimensional clusters within 5000 by 5000 space with large number of instances. The number of subspace should be carefully chosen, since we know CPLEX solver can efficiently solve about 100 instances with 6 possible clusters, in most of cases, we separate space that each subspace has about 50 instances in average so that even though the boundary line crosses one of clusters, the total number of instances in the next subspace would not exceed 100. If one of subspace has too many instances, we may only find feasible solution for this subspace which causes the solution found about the number of clusters is larger than the simulated number of clusters.

Table 8 Scalability Test by Space Decomposition Technique

No	No. of Instances	No. of Subspace	Simulated No. of cluster	Solution to No. of Clusters	Separation	Total Amount of Time (s)
1	166	3	8	8	3.55E+08	93.98±0.51
2	214	5	10	10	4.13E+08	22.58±0.39
3	302	6	10	10	1.1552e+009	5.59±0.06
4	399	8	14	14	1.8804e+009	134.192±1.13
5	509	14	14	14	2.2032e+009	199.13±1.46
6	600	15	12	12	1.8753e+009	488±0.77
7	1008	8	15	15	7.6040e+009	99.32±0.39

3.7 Heuristic Solving Procedure

There is an iterative heuristic technique that can be used to find the sub-optimal solution for the IP model. First we randomly select an instance, say a target instance, and scan data set to select all other instances that satisfy similarity measures with the target instance. Second, we apply the IP model to the selected data set to come out a cluster that contains the target instance. Third, we remove the cluster that contains the target instance and go back the first step. This is a heuristic method that works well if we only are interested in a cluster that contains a certain target instance. Otherwise, the method tends to be an approximate way to obtain the minimal number of clusters for the overall data set, since clustering a subset of data set in this case only can reach local optimal solution. For example, suppose four instances {A, B, C, D}, A-B, A-C and C-D are similar but B-C is not similar. Then the selected sub data set for A is {A, B, C}. If clustering result for the sub data set is two clusters with {A, C} and {B}, the heuristic procedure will turn out to be 3-mininal number of clusters, instead of the optimal solution, 2-mininal number of clusters with {A, B} and {C, D}. Hence, this heuristic procedure is efficient for a large data set in which we focus on only one instance at a time.

3.8 Discussion

From the above experiments, the IP model can find the minimal number of clusters in most cases up to thousands of instances. However, there are still some worst cases of data set that the IP model can not solve. For example, the data set has no obvious number of clusters at all and all data are randomly located in the space, then the optimal number of clusters would be very large which makes the problem too large for the CPLEX solver to solve. However, there is no meaning to find minimal number of clusters in this case since no obvious number of clusters. More, if there are a large number of instances in a small space,

then the space decomposition technique would fail, because instances are too density distributed in the boundary that each repeat has still many instances.

Although the IP model can find the minimal number of clusters, the optimal solution may not be unique, especially, instances near boundary of clusters may also belong to other neighbor clusters. Time spent to solve the problem can work as a controller to detect if an optimal solution is found. If the amount of maximal running time is exceeded, then the solution must be a feasible solution, not necessary optimal. If a feasible solution can be found, we can try by reducing the maximal number of clusters and rerun the IP model, in this way reducing the matrix size in the IP model, in order to get an optimal solution.

CHAPTER 4 NETFLIX MOVIE RATE PREDICTION

Netflix, an online movie rental service, published a data set that contains millions of movie rates from thousands of customers. We suppose each customer belongs to a cluster, either individually or in groups, that shares similar characteristics. Their carefully elicited preferences and a history of their ratings of the movies are maintained in a database. Customers are assigned to clusters (or groups) that are constructed such that significant differences between clusters in the distribution of preferences and significant similar preference within each cluster. Rating of the customer to a movie is predicted according to other customer preference in the same cluster. We design to find out joint customers such as a group of friends or a family. It will help us not only predict the kind of movies the customer already touched but also some other kinds of movies the customer may potentially be highly interested according to its "friends".

In collaborative filtering algorithm, *k*-nearest neighbor CF algorithm requires predefined number of nearest customers to determine movie rate for the target customer. We propose to apply IP model to find out the minimal number of clusters from the most related data set for the target customer. Since the number of elements in the cluster varies as data set changes, the number of most similar customers for target customers is therefore different. Then we predict the movie rate for the target customer by aggregating movie rates of other customers in the same cluster.

4.1 Dataset Description

One of the most challenge parts of the task is the scalability to deal with the huge datasets, because there are 17770 files, one file per movie, and more than 100 million

customer records. In this thesis, we only use about 10% of the total dataset for simplicity, but still data set is huge. In each movie file, the data set is formatted as follows:

CustomerID,Rating,Date

- MovieIDs range from 1 to 17770 sequentially.

- CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users.

- Ratings are on a five star (integral) scale from 1 to 5.

- Dates have the format YYYY-MM-DD.

In this study, we ignore influence of Date, since we suppose the preference of customers will not change with time, as the same assumption in the most collaborative filtering algorithm. Introducing Date for each customer for movie-customer pair will cause complicity which may need other model like time series to do with it.

4.2 Data Reduction

Each customer watches many movies and each movie can be seen by different customers. For a given customer-movie pair, we may have a rating that specifies the customer's preference to the movie. Hence, it forms a network. In Figure 3, m_i represents a movie i, C_j represents a customer j and weight of every arc is the rate for every movie-customer pair. Then our job is to find out patterns of movie-customer pairs with similar weights.

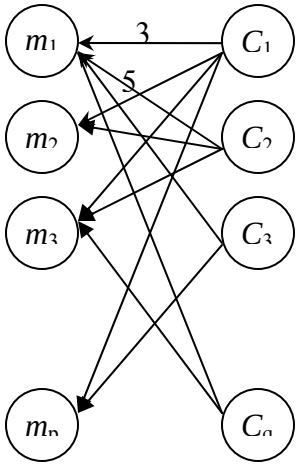

Figure 2 Movie-customer Pair Network

In order to deal with scalability, we need reduce the dataset to the *most related subset*. If we want to predict rate for MovieID1 watched by CustomerID1, which is a weight for m_1-C_1 pair, then a population that has similar preference as CustomerID1 to MovieID1 should be other customers in the MovieID1 file with m_1-C_j pair. Of course, there are some customers similar to CustomerID1 but outside MovieID1 file. However, those customers have no help for us to predict the rate between m_1-C_1 pair, because they have no such pair related to m_1 at all. So, the relevant customers are customers in the same movie file. More, each customer will not only be interested in one movie, but some other movies. So, we need find out all movie-customer pairs for customers in m_1 file by scanning database once.

For example, there are 547 customers in MovieID1 file and the total amount of movie-customer pair related to customers in MovieID1 file is 55780. This data subset is used to predict each customer in movieID1 file. For each customer, the data subset is still large. To further compact the dataset, we use each customer as an instance and represent each customer as a long vector about rates for different movies. (See Table 10)

Table 9 Movie-customer Matrix

Customer id	MovieID1	MovieID2	MovieID3	…
CustomerID1	2	3	n/a	…
CustomerID2	n/a	5	n/a	…
…	…	…	…	…

The total amount of movies seen by customers in movieID1 file, 1698, is still very large. So, in order to predict rate for MovieID1-CustomerID1, Table 10 is a large and sparse matrix with 1698 columns. Since we only attempt to predict rate about CustomerID1 who has only watched 197 movies. So, we can reduce the number of columns to 197 movies for customerID1 in movieID1 file. In this way, we find a most related matrix with 547 by197 to predict rate for MovieID1-CustomerID1 pair, which shows all rates of movies watched by CustomerID1 as well as at least one movie rate from other customers in the movie fileID1. These data subset is suitable for us to do clustering analysis. We call the reduced movie-customer matrix as most related matrix for m_i-C_j pair which can be generated as follows:

- choose movie file *i* that includes all customers that watched movie *i*
- scan data base once to collect every movies as well as corresponding rates that watched by customers in movie *i* file and form movie-customer matrix
- reduce collected movie-customer matrix by preserving only movie columns watched by customer *j*
- return reduced movie-customer matrix as most related matrix

Here, we need mask the column that includes m_i-C_j entry in most related matrix, because we want to predict rate of m_i-C_j pair.

4.3 Similarity Measures for Missing Data

Table 10 contains many "n/a" values that represent unrated movie rates. To calculate similarity with missing data, we need only think about the same movies that have been co-rated by both two customers, and ignoring movies one customer watched but the other

doesn't or both don't. This is consistent to the idea mentioned in collaborative filtering algorithms.

For example, Pearson similarity measure, correlation coefficient measure, with missing data can be calculated as follows. Suppose two vectors to be $v_1 = (3, n/a, 4)$ and $v_2 = (2,3,5)$. Then, actual vectors in calculation are $v_1^{'} = (3,4)$ and $v_2^{'} = (2,5)$

So, $sim(v_1, v_2) = \dfrac{(3-3.5)\cdot(2-3.5)+(4-3.5)(5-3.5)}{\sqrt{[(3-3.5)^2+(4-3.5)^2][(2-3.5)^2+(5-3.5)^2]}} = 1$

For cosine similarity measure, simply selecting columns that have both rates for any two customers may not good enough since when some customers only have watched a few movies, these customers are not good to participate in providing inference for other customers. So, we should better modify cosine similarity measure. The cosine measure is defined as $sim(v_1, v_2) = \dfrac{v_1 v_2}{|v_1||v_2|}$. For the missing data, we have two vectors for example, $v^1 = (3, n/a, 4)$ and $v^2 = (2,3,5)$, then $sim(v^1, v^2) = \dfrac{(3,4)\cdot(2,5)}{\sqrt{3^2+4^2}\sqrt{2^2+5^2}} = 0.965$

In order to account for dimension influence of two vectors, we add an adjusting weight for all similarity of two vectors as follows:

$$sim'(v_1, v_2) = \frac{n}{L}\frac{v_1 v_2}{|v_1||v_2|},$$

where n is the amount of available columns that both v_1 and v_2 have rates. L is the maximal amount of available columns in the two vectors. The more elements of two vectors, the more information they provide, hence, the more similar they may have. And the closer the similarity to 1, the more similar that two customers have. This simple measurement can help us find if two customers have similarity according to past performance. However, in this thesis, we choose Pearson similarity so as to be consistent to collaborative filtering algorithm.

4.4 Movie Rate Prediction

Although after data reduction, most related matrix for every customer still contains several hundred of instances. From numerical experiments that we did in section 3, applying clustering model directly into most related matrix will be hard to find the minimal number of clusters as well as elements in the clusters. For example, in order to predict the movie rate for the first customer id=1488844 and movie id=0000001, most related matrix is 547×197, where number of instances is 547. So, the IP model may not be easy to apply to most related matrix directly. However, notice that we only are attempt to predict rate for customer id=1488844 and movie id=0000001, so to speak, we only care about the cluster that contains customer id=1488844. So, we can apply heuristic procedure to find out similar customers for the target customer. Before clustering the data set, we first filter instances that are close to the target instance.

We applied only Pearson similarity measure in the constraints which was to keep consistent to collaborative filtering algorithms. And we predicted rates of first 20 customers in movie id=0000001 in Table 11. The procedure is as follows:

- find out most related matrix for each customer C_j, the target instance
- filter instances out that are close to the target instance
- run IP model with Pearson similarity measure constraint only on the filtered matrix
- take the cluster out that contains the target customer C_j
- predict movie rate by aggregating rates of neighbors of C_j in the same cluster

We set filtering criteria with Pearson similarity between 0.5 and 0.7, clustering criteria Pearson similarity greater than 0.75 when apply the IP model. For example, for customer id=1488844, Instance No.=327, most related matrix 547 instances. After filtering with Pearson similarity greater than 0.5, we have 36 instances including the target instance (No. 327), showed in Table 11. After clustering by the IP model, we found there is only one

neighbor of instance no.327, no. 489 in the same cluster. So, the predicted rate for customer no. 327 and movieID=0000001 is 3.

Table 10 Filtered Instances for Customer ID=1488844

MovieID / Instance No.	1	8	17	30	…	1905	1918	1925	1939
327	3	4	2	3		5	3	3	3
28	5	3	0	3		4	0	0	0
32	5	0	0	0		0	0	0	0
39	3	0	0	0		5	0	0	0
51	1	0	0	0		0	0	0	0
54	4	0	0	0		0	0	0	0
71	5	0	0	0		0	0	0	0
77	3	0	0	0		5	0	0	0
101	5	0	0	3		5	0	0	3
111	5	0	0	0		5	0	0	0
113	5	0	0	0		0	0	0	0
125	5	0	0	0		0	0	0	0
206	5	0	4	0		0	3	0	0
212	4	0	0	0		5	0	0	0
228	4	5	0	0		5	0	0	0
230	3	0	0	0		0	0	0	0
255	4	0	0	2		4	3	0	0
257	4	0	4	0		5	0	0	0
278	4	0	0	0		0	2	0	0
286	3	0	0	0		0	0	0	0
292	5	0	0	0		2	0	0	0
337	5	0	0	0		0	0	0	0
355	5	0	0	0		0	0	0	5
397	4	0	0	0		4	2	0	0
403	3	0	0	0		0	0	0	0
406	4	0	0	3		0	3	0	0
410	3	0	0	0		5	0	0	0
440	4	0	0	0		4	0	0	0
452	4	0	0	4		5	0	0	0
466	4	0	0	0		0	0	0	0
467	3	0	0	0		0	0	0	0
489	3	0	0	3		0	0	0	0
493	5	0	0	0		0	0	0	0
513	5	0	0	0		0	0	0	0
532	4	0	0	0		0	0	0	0
537	5	0	0	0		0	0	0	0

Similarly, using the above procedure, we calculated the first 20 customers in MovieID 0000001 file. We found that prediction accuracy was about 88%, and its standard variance was 0.132. We noticed that customer id 124105 had relatively larger prediction errors than others, mostly because the customer only watched a few movies making prediction insufficient. For example, customer id 124105, instance No. 37 in the Table 13, had only 6 movie records to predict first movie. Prediction for customer id 893988 is also bad but it may belong to outliers since this is true data set and movie rates from customers

Table 11 Movie Rate Prediction for the First 20 Customers
(Similarity≥0.8 if neighbor exits, otherwise Similarity≥0.75)

Customer id	Movie Rate	Predicted Rate	Similarity	No. of Neighbors	No. of Movies
1488844	3	3.00	0.80	1	197
822109	5	4.00	0.80	4	16
885013	4	4.00	0.80	1	32
30878	4	3.98	0.80	2	113
823519	3	3.00	0.75	1	59
893988	3	4.50	0.75	2	41
124105	4	3.00	0.80	6	7
1248029	3	3.68	0.80	4	118
1842128	4	3.98	0.80	3	13
2238063	3	3.53	0.80	57	3
1503895	4	3.34	0.80	4	7
2207774	5	4.00	0.80	1	53
2590061	3	3.61	0.75	2	61
2442	3	2.50	0.80	2	35
543865	4	3.55	0.75	2	74
1209119	4	3.55	0.80	2	29
804919	4	4.03	0.75	4	22
1086807	3	3.00	0.80	1	12
1711859	4	4.00	0.80	1	72
372233	5	-	0.75	-	81
Avg abs error rate			0.12		
Standard variance			0.13		

are relatively subjective. Customer id 2238063 should also be outliers since the customer has only two other movies for prediction which causes most of other customers are similar to the target customer. We cannot predict the 20th customer in that we do not find neighbor customers that have correlation coefficient similarity greater than 0.75.

We also found weak negative correlation between prediction error and similarity (-0.197), or prediction error and No. of movies (-0.199), indicating that more similarity leads to less prediction error and more amount of movies participated in prediction leads to less prediction error.

Table 12 Cluster for Customer ID=124105

Movie id / Instance No.	1	191	563	694	1110	1220	1905
76	3	0	0	3	5	5	4
369	4	0	0	3	5	0	4
502	2	0	0	3	5	0	0
2	3	4	0	3	4	0	4
37	4	5	5	4	5	5	5
335	3	5	4	3	5	5	0
527	3	5	0	3	0	0	4

4.5 Collaborative Filtering

In order to further make sure the effectiveness of the proposed integer programming method in recommendation system, we choose currently most popular used collaborative filtering algorithm in recommendation system, *k*-nearest neighbor CF, as a comparison. Since we don't know *k* in advance, we choose *k*=1, *k*=5 and *k*=10 respectively to apply CF to predict movie rates for the first 20 customers in MovieID 0000001, as Table 14, 15 and 16. From the experiments, we noticed that when *k*=1, prediction accuracy was not good, only about 72% with standard variance 0.22, when *k*=5, prediction accuracy was about 83% with standard variance 0.13, while when *k*=10, prediction accuracy was about 83% with standard

variance 0.11. For larger *k*, the prediction accuracy was still almost the same as *k*=10 since the further away from the target customer, the smaller weights in aggregation for prediction.

In Table 16, it shows smaller standard variance than that of the proposed clustering method in Table 12. However, the proposed method also provides us reliable indication, such as similarity parameter used in the IP model, number of movies participated in prediction and so on. If similarity parameter in the IP model is 0.8, then the prediction accuracy is mostly at least 0.8. The more number of movies in *most related matrix* means that more information is available. Since we relax similarity parameter to 0.75 if no neighbors available to the target instance, we actually tend to less prediction accuracy and increase variance of prediction results. Hence, we believe that if more data are used, better prediction accuracy and smaller variance will be reached in the proposed method.

Table 13 Collaborative Filtering C_j-M_1 Pairs, k=1

Customer id	Movie Rate	Predicted Rate
1488844	3	3
822109	5	3
885013	4	4
30878	4	5
823519	3	1
893988	3	4
124105	4	3
1248029	3	5
1842128	4	3
2238063	3	5
1503895	4	3
2207774	5	4
2590061	3	4
2442	3	3
543865	4	4
1209119	4	3
804919	4	2
1086807	3	3
1711859	4	3
372233	5	3
Avg abs error rate	0.28	
Standard variance	0.22	

Table 14 Collaborative Filtering C_j-M_1 Pairs, k=5

Customer id	Movie Rate	Predicted Rate
1488844	3	3.60
822109	5	4.00
885013	4	4.03
30878	4	3.81
823519	3	3.11
893988	3	4.00
124105	4	3.20
1248029	3	4.00
1842128	4	2.39
2238063	3	3.80
1503895	4	4.00
2207774	5	4.59
2590061	3	4.00
2442	3	3.60
543865	4	3.60
1209119	4	4.00
804919	4	3.59
1086807	3	3.99
1711859	4	3.59
372233	5	3.92
Avg abs error rate		0.17
Standard variance		0.13

Table 15 Collaborative Filtering C_j-M_1 Pairs, k=10

Customer id	Movie Rate	Predicted Rate
1488844	3	3.88
822109	5	4.09
885013	4	4.02
30878	4	4.20
823519	3	3.06
893988	3	4.00
124105	4	2.99
1248029	3	4.19
1842128	4	3.37
2238063	3	3.70
1503895	4	3.71
2207774	5	4.16
2590061	3	3.69
2442	3	3.50
543865	4	3.42
1209119	4	4.00
804919	4	3.69
1086807	3	3.50
1711859	4	3.41
372233	5	3.60
Avg abs error rate		0.17
Standard variance		0.11

4.6 Discussion

The proposed IP model to cluster customers has a function of variable *k* in CF since clustering customers in one group will result in different number of neighbors to predict the movie rates for the target customer. Hence, the prediction accuracy shows better than the standard CF algorithm which pre-defines number of neighbors, *k*.

Although *k*-nearest neighbor CF selects *k* most similar customers for the target customer and the more similarity to the target customer the neighbors have, the larger weights it will be used in predicting movie rates, it can not guarantee that those neighbors also have similar preference. For example, we want to predict a movie rate for customer A, who has two neighbors, B and C, according to CF algorithm, however, due to multi-dimensional data set, B and C may not similar to each other. In the IP model, we try to cluster data set into different groups, each of which has high similarity, so that those customers in one cluster are all similar to each other. This is one reason that the proposed IP model applied to recommendation system can work well in accuracy and variance.

CHAPTER 5 CONCLUSION AND FUTURE WORK

The thesis studied a new clustering method by proposing an integer programming model, aiming at minimizing the total number of clusters and subjected to several similarity measures. The model was applied to Netflix recommendation system in order to predict movie rates for customers according to their historical preference. The proposed method was compared with collaborative filtering algorithm and experimental results showed that it provided relatively high prediction accuracy as well as relatively small variance.

However, there are still some drawbacks in the proposed IP model.

The model is solvable for only middle size data set up to thousands of instances due to large size of matrix coefficients and limited computer memory. To solve the scalability of model, we should better provide a similar model with more compact matrix coefficients. Or we may use heuristics, like genetic algorithm, taboo search algorithm and so on to apply directly to the QIP/IP model.

The parameters in the IP model are not easy to be best specified. If we do parametric analysis for every data set in order to find out the best parameters, it will be very time consuming. So, how to set good parameters for the IP model is still a challenging problem.

Finding the minimum number of clusters works well by applying the IP model. However, cluster elements found by the IP model are often not unique if there is an overlap region between two clusters. One way to solve this problem is to use the IP model to identify the minimal number of clusters, and then we use other standard clustering methods to find out the exact elements for each cluster.

BIBLIOGRAPHY

Adomavicius G. and Tuzhilin A., 2005, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transaction on Knowledge and Data Engineering, Vol.17: 734-749.

Aldenderfer, M.S. and Blashfield, R.K., 1984. *Cluster Analysis*. Beverly Hills, CA: Sage Press.

Ahmoys, D.B., Tardos, E. and Aardal, K., 1997. Approximation algorithms for facility location problems. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pp. 265-274.

Armstrong, J.S., 2001. Principles of Forecasting—A Handbook for Researchers and Practitioners. Kluwer Academic.

Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D., 2005. Linear Programming and Network Flows, third edition, John Wiley & Sons, Inc., New York, NY.

Bradley, P.S., Fayyad, U.M., Mangasarian and Street, N., 1996. Clustering via concave minimization. *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, Vol.9: 368-374

Breese, J.S. Heckerman, D. and Kadie, C,1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proc. 14th Conf. Uncertainty in Artificial Intelligence*.

Breese, J. S., Heckerman, D., and Kadie, C., 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *In Proceedings of the 14th Conference on Uncertainty in Arti_cial Intelligence*, pp. 43-52.

Basu, C., Hirsh, H., and Cohen, W., 1998. Recommendation as Classification: Using Social and Content-based Information in Recommendation. *In Recommender System Workshop'98*. pp. 11-15

Bradley, P.S., Fayyad, U.M. and Reina, C., 1998a. Scaling clustering algorithms to large databases. *In: Proceedings of ACM Conference on Knowledge Discovery in Databases*, pp. 9-15.

Bradley, P.S. and Mangasarian, O.L, 2000, *k*-plane clustering. *Journal of Global Optimization.* Vol.16 (1), 23-32.

Bradley, P.S., Mangasarian, O.L. and Street, N., 1996. Clustering via concave minimization, *Advances in Neural Information Processing Systems*, Vol. 9. MIT Press, Cambridge, MA, pp. 368-374.

Dempster, A. P., Laird, N. M., and Rubin, D. B., 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B.*, Vol.39 (1): 1–38.

Gottlob G., Leone, N. and Scarcello, F., 2001. Hypertree Decompositions: A Survey. *MFCS* 37-57.

Goldberg, D., Nichols, D., Oki, B.M. and Terry, D., 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Comm.ACM*. Vol. 35 (12): 61-70.

Han, J. and Kamber, M., 2006. Data mining: Concepts and Techniques. *2nd edition,* Elsevier Inc.

Hill, W., Stead, L., Rosenstein, M., and Furnas, G., 1995. Recommending and Evaluating Choices in a Virtual Community of Use. *Proc. Conf. Human Factors in Computing Systems*

Hwang, F., 1981. Optimal partitions. *Journal of Optimization Theory and Applications*. Vol.34: 1-10.

Jain, A.K., Murty, M. N. and Flynn**,** P. J., 1999. Data clustering: A review. *ACM Computing Surveys (CSUR)*, Vol. 31 (3): 264-323.

Joseph, A. and Bryson, N., 1997. W-efficient partitions and the solution of the sequential clustering problem. *Annals of Operations Research: Nontraditional Approaches to Statistical Classification*. Vol.74: 305-319.

Kall, P. and Wallace, S.W., 1994. *Stochastic Programming*, John Wiley & Sons, Chichester.

King, B., 1967. Step-wise clustering procedures. *J. Am. Stat. Assoc,* Vol.69: 86–101.

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., et al. 1997. GroupLens: Applying Collaborative Filtering to Usenet News. *Comm. ACM*, Vol. 40 (3): 77-87.

Mark S. Aldenderfer and Roger K. Blashfield, 1984. Cluster Analysis, Sage Publications, Inc.

Mcqueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

Murtagh, F. 1984. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *Comput. J.*, Vol.26: 354–359.

Murthi B.P.S.,and Sarkar, S., 2003. The Role of the Management Sciences in Research on Personalization. *Management Science*. Vol. 49 (10):1344-1362.

Ng, R. and Han, J., 1994. Efficient and effective clustering method for spatial data mining. In: *Proceedings for the 1994 International Conference on Very Large Data Bases*, pp. 144-155.

Olafsson, S., Li, X. and Wu, S., 2006. Operations research and data mining. *European Journal of Operational Research*, doi:10.1016/j.ejor.2006.09.023

Rao, M.R., 1971. Cluster analysis and mathematical programming. *Journal of the American Statistical Association.* Vol.66: 622-626

REF: http://www.netflixprize.com

Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P. and Riedl, J., 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proc. 1994 Computer Supported Cooperative Work Conf.*

Rich, E., 1979. User Modeling via Stereotypes, *Cognitive Science*. Vol. 3 (4): 329-354.

Salton, G.,1989. Automatic Text Processing. Addison-Wesley.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J., 2000. Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the ACM EC'00. Conference*. Minneapolis, MN. pp. 158-167

Shmoys, D.B., 1999. Approximation algorithms for clustering problems. In: *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pp. 100-101.

Shmoys, D.B., Tardos, E., Aardal, K., 1997. Approximation algorithms for facility location problems. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pp. 265-274.

Shardanand U. and Maes, P., 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". *Proc. Conf. Human Factors in Computing Systems*

Sneath, P. H. A. and Sokal, R. R., 1973. *Numerical Taxonomy*. Freeman, London, UK.

Ungar, L. H., and Foster, D. P., 1998. Clustering Methods for Collaborative Filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*.

Vinod, H.D., 1969. Integer programming and the theory of grouping. *Journal of the American Statistical Association*. Vol. 64: 506-519

Ward, J. H. JR., 1963. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc*, Vol.58: 236–244.

Zahn, C. T., 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput. C-20* (Apr.): 68–86.

Zhang, T., Ramakrishnan, R. and Livny, M., 1996. BIRCH: An efficient data clustering method for very large database. In: *SIGMOD Conference*, pp. 103-114.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Sigurdur Olafsson for his guidance, patience and support throughout this research. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. John Jackman and Dr. Dan Zhu. I would additionally like to thank Dr. Dan Zhu for her critical thoughts of my thesis work and Dr. John Jackman for his potential purport in my graduate study.