Content Boosted Matrix Completion

by

Yan Zhao

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Mu Zhu

Waterloo, Ontario, Canada, 2014

Abstract

Recommender systems are widely used in modern business. Most recommendation algorithms are based on collaborative filtering. In this paper, we study different ways to incorporate content information directly into the matrix completion approach of collaborative filtering. These content-boosted matrix completion algorithms can achieve better recommendation accuracy.

Acknowledgements

I would like to thank all the people who made this work possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recommender systems are widely used to promote sales of products and services today. For example, Netflix (www.netflix.com) will recommend movies that one may be interested in based on his previously rated movies. In terms of underlying mathematical model, *collaborative filtering* is a popular algorithm used in recommender systems. The main idea of collaborative filtering is that users who have rated the same items closely are considered to be "similar". Then to a new user, the system recommends items that "similar" users have rated favorably before.

For a recent review and discussion of different collaborative filtering algorithms, we refer readers to [7]. In practice, most collaborative filtering algorithms are either based on matrix factorization or nearest neighbors [10]. However, the collaborative filtering problem can also be formulated as a *matrix completion* problem. This approach used to be considered as unattractive since typical formulation of matrix completion problem is NP-hard. However, recent introduction of *nuclear norm* [3] relaxes this problem to a convex one and solvable by semi-definite programming. Therefore, more attention has been paid to matrix completion problem recently. Various algorithms have been proposed to solve matrix completion problem and a recent review of those different algorithms can be found at [14].

To improve the performance of collaborative filtering algorithms, one key idea is to leverage supplemental information [18]. In this paper, inspired by successful usage of item content information in matrix factorization based collaborative-filtering [15], we focus on a particular type of supplemental information - content information about the individual items. Instead of directly solving matrix completion problem on original user-item matrix, we try to use content information to "pre-complete" part of user-item matrix. Then we solve matrix completion problem on this pre-processed user-item matrix. And as shown in experiment part, the incorporated content information indeed achieves better recommendation accuracy.

The rest of the paper is organized as follows. In Chapter 2, we give some background knowledge about matrix completion. In Chapter 3, we describe our approach to incorporate content information into matrix completion. In Chapter 4, we describe the experiments we conducted to evaluate our approach. Finally, we summarize this paper with some further discussion in Chapter 5.

Chapter 2

Matrix Completion

Firstly, we will briefly review the matrix completion problem.

2.1 Notation

Given a set of users $U = \{u_1, ..., u_N\}$, and a set of items $I = \{i_1, ..., i_M\}$, let r_{ui} denote the rating given by user u to item i. Then these ratings form a user-item rating matrix, $R = [r_{ui}]_{N \times M}$.

In practice, since users can only rate a small fractions of all items, the user-item rating matrix R is highly sparse with lots of unknown entries. Therefore we denote

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

as the set of indices for known ratings and

$$\bar{\Omega} = \{(u, i) : r_{ui} \text{ is unknown}\}$$

as the set of indices for unknown ratings.

Then given any user-item pair (u, i), the purpose of recommender systems is to predict the corresponding rating, which we denote by $\hat{r}_{ui}$. Similarly, these predictions form a user-item prediction matrix $\hat{R} = [\hat{r}_{ui}]_{N \times M}$.

2.2 Problem Formulation

The matrix completion problem [3] is typically formulated as

$$\begin{array}{ll} \underset{\hat{R}}{\text{minimize}} & \text{rank}(\hat{R}) \\ \text{subject to} & \hat{r}_{ui} = r_{ui}, \ (u,i) \in \Omega. \end{array} \tag{2.1}$$

That is, we want to fill in all unknown entries of R while keeping the rank of completed matrix $\hat{R}$ as low as possible. This formulation is quite intuitive since usually user ratings is mainly explained by several key features. And the value of rank corresponds to the number of features. To be distinguished from other formulations, we will use *rank minimization matrix completion* problem to denote Problem (2.1).

Now Suppose $\sigma_1(\hat{R}) \geq \sigma_2(\hat{R}) \geq \ldots \geq \sigma_{min(M,N)}(\hat{R}) \geq 0$ are singular values of $\hat{R}$, and let $\sigma = (\sigma_1(\hat{R}), \sigma_2(\hat{R}), \ldots, \sigma_{min(M,N)}(\hat{R}))^T$. Then we will have

$$\begin{aligned} \text{rank}(\hat{R}) &= \sum_{i=1}^{min(M,N)} I(\sigma_i(\hat{R}) \neq 0) \\ &= \|\sigma\|_0. \end{aligned}$$

Therefore the rank minimization matrix completion problem can be considered as a l_0-norm minimization problem of vector σ. Then like other l_0-norm minimization problems, it is NP-hard [15]. In addition, all known algorithms which provide exact solutions to rank minimization matrix completion problem require time doubly exponential in the dimension n of the matrix in both theory and practice [3] [5]. These facts drive people to seek convex relaxations of this problem.

Generally speaking, it has been well established [4] [6] that l_1-norm minimization is a good convex relaxation of l_0-norm minimization. Therefore we want to apply l_1-norm relaxation to rank minimization matrix completion problem. In particular, we can define

$$\|\hat{R}\|_* = \sum_{i=1}^{min(M,N)} \sigma_i(\hat{R})$$

as the nuclear-norm of $\hat{R}$. Then according to our assumption, all entries of vector σ is non-negative, so we have

$$\begin{aligned} \|\hat{R}\|_* &= \sum_{i=1}^{min(M,N)} |\sigma_i(\hat{R})| \\ &= \|\sigma\|_1. \end{aligned}$$

Now instead of rank minimization matrix completion problem, we formulate the *nuclear-norm minimization matrix completion* problem

$$\begin{aligned} \underset{\hat{R}}{\text{minimize}} \quad & \|\hat{R}\|_* \\ \text{subject to} \quad & \hat{r}_{ui} = r_{ui},\ (u,i) \in \Omega. \end{aligned} \tag{2.2}$$

Nuclear-norm minimization matrix completion problem is a l_1-norm minimization problem of vector σ and therefore can be seen as a convex relaxation of rank minimization matrix completion problem.

Like other l_1-norm minimization problems, the nuclear-norm matrix completion problem is convex and can be solved by semi-definite programming [3]. In general, nuclear-norm matrix completion may not produce the same solution to rank minimization matrix completion problem. However, it has been proven that, under certain conditions (e.g. restricted isometry property), the nuclear norm minimization can be guaranteed to produce the rank minimization solution [16]. That is, the solutions of matrix completion problem and nuclear-norm matrix completion problem can be exactly the same. This may be the key point that the research of nuclear-norm minimization is gaining intensive attention recently.

In our project we will use nuclear-norm minimization formulation as our optimization problem.

2.3 Algorithms

Although there is no efficient algorithm to solve rank minimization matrix completion problem, various algorithms have been proposed to solve nuclear-norm minimization matrix completion problem. Those algorithms can be classified as two categories: direct ones and approximation ones. Direct algorithms solve exactly the nuclear-norm minimization matrix completion problem while approximation algorithms usually make reasonable modification to the formulation of nuclear-norm minimization matrix completion problem. A review of different algorithms can be found at [14].

2.3.1 Direct Algorithms

The nuclear-norm minimization matrix completion problem can be solved directly by some convex programming packages such as SDPT3 [20], cvx [8][9] and TFOCS[1].

SDPT3 is capable to solve most convex optimization and very reliable. However, the implementations of optimization problems are not easy in practice, especially for uncommon problems. People need to follow a long user guide before they can use SDPT3 with proficiency.

CVX is more like a programming language than a solver. It supports different solvers as its computational engines including SPDT3. It works more like a wrapper and makes implementations of optimization problems easy to use and understand. For example, using cvx, nuclear-norm minimization matrix completion problem can be implemented as

```
ind = find(R);
cvx_begin
    variable R_hat(size(R));
    minimize( norm_nuc(R_hat) );
    subject to
        R_hat(ind) == R(ind);
cvx_end
```

which almost remains the same with its mathematical formulation.

TFOCS provides a set of Matlab templates that can be used to construct solvers for many convex problems. A demo of matrix completion problem implementation can be found at [17]. The advantage of TFOCS is that in practice it is very efficient in both time and space.

2.3.2 Approximation Algorithms

It is useful to have some definitions before we proceed. Let us define a projector $\mathcal{P}_\Omega(\cdot) : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ as

$$(\mathcal{P}_\Omega(A))_{ij} = \begin{cases} A_{ij} & \text{if } (i,j) \in \Omega. \\ 0 & \text{otherwise.} \end{cases}$$

Now nuclear-norm minimization matrix completion problem can be rewrite as

$$\begin{array}{ll} \underset{\hat{R}}{\text{minimize}} & \|\hat{R}\|_* \\ \text{subject to} & \mathcal{P}_\Omega(\hat{R} - R) = 0. \end{array} \tag{2.3}$$

Singular Value Thresholding Algorithm

For $X \in \mathbb{R}^{n_1 \times n_2}$ of rank r, consider its singular value decomposition

$$X = U\Sigma V^*, \Sigma = \text{diag}(\{\sigma_i\}_{1\le i\le r}) \in \mathbb{R}^{r\times r}, U \in \mathbb{R}^{n_1\times r}, V \in \mathbb{R}^{n_2\times r}.$$

For each $\tau \ge 0$, define *soft-thresholding operator* $\mathcal{D}_\Omega(\cdot)$ as

$$\mathcal{D}_\tau(X) := U\mathcal{D}_\tau(\Sigma)V^*, \quad \mathcal{D}_\tau(\Sigma) = \text{diag}(\{(\sigma_i - \tau)_+\}_{1\le i\le r}),$$

where $t_+ = \max(0, t)$. The authors of [2] proved that for all $\tau \ge 0$

$$\mathcal{D}_\tau(X) = \underset{Y}{\text{argmin}}\{\frac{1}{2}\|Y - X\|_F^2 + \tau\|Y\|_*\}.$$

Now the singular value thresholding algorithm is the following iterations starting with X^0

$$\begin{cases} \hat{R}^k = \mathcal{D}_\tau(X^{k-1}) \\ X^k = X^{k-1} + \delta_k \mathcal{P}_\Omega(R - \hat{R}^k) \end{cases}$$

until the stopping criterion is reached.

It can be shown that under certain conditions the sequence $\hat{R}_k$ above converges to the solution of the following problem

$$\begin{aligned} \underset{\hat{R}}{\text{minimize}} \quad & \tau\|\hat{R}\|_* + \frac{1}{2}\|\hat{R}\|_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\hat{R} - R) = 0. \end{aligned}$$

Intuitively, if τ is large enough, the term $\frac{1}{2}\|X\|_F^2$ is negligible. Therefore the above problem is a good approximation of the nuclear-norm minimization matrix completion problem. [2]

Accelerated Proximal Gradient Algorithm

Accelerated proximal gradient algorithm introduced in [21] solves the problem

$$\min_{\hat{R}\in\mathbb{R}^{n_1\times n_2}} \frac{1}{2}\|\mathcal{A}(\hat{R}) - b\|_2^2 + \mu\|\hat{R}\|_*.$$

Let $\mathcal{A}$ be $\mathcal{P}_\Omega$ and b be $\mathcal{P}_\Omega(R)$, then the above problem becomes

$$\min_{\hat{R}\in\mathbb{R}^{n_1\times n_2}} \mu\|\hat{R}\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(R - \hat{R})\|_2^2.$$

The rational of this approximation is as follows. The data for (2.3) may be contaminated with noise, and thus there may not be any low-rank matrices that satisfy the constraints in (2.3) [21]. Therefore we drop the constraint in (2.3) and add it as a penalty term into objective function. When μ is large, the objective function is still dominated by nuclear norm term. Therefore this formulation is a good approximation of the nuclear-norm minimization matrix completion problem.

Other Algorithms

Other algorithms includes:

- Fixed point continuation with approximate SVD [13].
- The alternating splitting augmented Lagrangian method [19].

2.3.3 Choice of Algorithm

Although approximation algorithms claim to be more efficient (at least in scenarios described in the corresponding papers), it is not guaranteed that they are better in all cases (in particular, for our experiment settings). In addition, nuclear-norm minimization matrix completion problem itself is already an modification of the original rank minimization matrix completion problem. Using approximation algorithms may be a bit far away from our original question. Therefore, we choose to use direct algorithms in our experiments. And among those direct algorithms, the TFOCS package turns out to be most efficient and hereby becomes our choice.

Chapter 3

Content Boosted Matrix Completion

In this chapter, we will describe our approach to incorporate content information into matrix completion problem.

Suppose that, for each item i, we have a content information vector $a_i = (a_{i1}, a_{i2} \ldots, a_{iD})^T$ of D attributes. These vectors form a content information matrix $A = [a_{ij}]_{M\times D}$. For simplicity, we further assume that A is a binary matrix. That is, $a_{id} \in \{0, 1\}$, indicating that whether item i has attribute d or not. Now our goal is to incorporate this content information matrix A into our nuclear-norm minimization matrix completion problem.

From the formulation of nuclear-norm minimization matrix completion problem, adding content information related penalty term to objective function seems to be an obvious choice. However, existing algorithms do not admit modification to problem formulations. If we want to add penalty term, current algorithms will fail and we need to propose a new algorithm.

Another way to incorporate content information is to consider the set Ω (i.e. the set of indices for known ratings). It is reasonable to believe that when Ω is larger, the predictions $\hat{R}$ should be more accurate (A confirmation experiment is described in Section 4.5). That is, the more ratings we know, the more accurate the predictions will be. Using incomplete rating matrix R and content information matrix A, it is not difficult to predict part of the unknown ratings. Then we will have a new set of indices for known ratings Ω' with $|\Omega'| > |\Omega|$.

Now consider problem

$$\begin{array}{ll} \underset{\hat{R}}{\text{minimize}} & \|\hat{R}\|_* \\ \text{subject to} & \hat{r}_{ui} = r_{ui},\ (u, i) \in \Omega'. \end{array} \tag{3.1}$$

Note that the only change is Ω' while the formulation remains the same. Then we can use existing algorithms to solve this problem. Therefore, our goal now is to increase size of Ω based on R and A.

In the following part, we present two approaches. One is based on sampling, and the other is based on regression model.

3.1 Sampling Method

Firstly from matrix A we calculate an item similarity matrix S. For $i, j \in I$,

$$s_{ij} = \begin{cases} 1 & \text{if } \dfrac{A_i \cdot A_j}{\|A_i\|\|A_j\|} \geq c \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where A_i is the i-th row of A. Then $S = [s_{ij}]_{M \times M}$. That is, if the cosine similarity between content information vectors of item i and j exceeds a threshold, then item i and j are considered as similar. Note that $S \in \mathbb{R}^{M \times M}$ is a binary, symmetric matrix.

Now for each $(u, i) \in \bar{\Omega}$, we define the similar rating vector $x \in \bar{\mathbb{R}}^M$. For $k \in I$

$$x_k = \begin{cases} r_{uk} & \text{if } (u, k) \in \Omega \text{ and } s_{ik} = 1 \\ -\infty & \text{otherwise.} \end{cases} \tag{3.3}$$

Then we define a truncated related rating vector $\bar{x}$ as

$$\bar{x} = \{x_k : x_k \neq -\infty\}. \tag{3.4}$$

This vector is the set of ratings that user u gives to items that are similar to i. Note that dimension of $\bar{x}$ varies with different (u, i). For a set of items that are considered as similar, it is reasonable to assume one certain user's ratings to those items are highly related. In particular, we can use Gaussian distribution to capture this relativity. Let m be mean of entries of vector $\bar{x}$ and v be variance of entries of vector $\bar{x}$. Now we can estimate mean and variance of this Gaussian distribution based on vector $\bar{x}$ as follows

$$r_{ui} \sim N(m, v).$$

We can sample from this distribution and thus predict r_{ui}.

In this way, we can predict almost all unknown ratings. However, quality is more important than quantity. We should fill in unknown rating with accurate predictions. It is natural to assume that predictions from a larger size of related vector is more accurate. Therefore, we will only predict r_{ui} when the dimension of corresponding $\bar{x}$ exceeds some threshold t. After predicting we will have some ratings besides Ω, they together forms Ω'. Then using TFOCS to solve problem (3.1), we are done with our problem.

We conclude the above processes with Algorithm 1.

Algorithm 1: Sampling based method

```
Input: $R$, $A$, $c$, $t$
Output: $\hat{R}$
begin
    $s_{ij} \leftarrow \begin{cases} 1 & \text{if } \frac{A_i \cdot A_j}{\|A_i\|\|A_j\|} \geq c \\ 0 & \text{otherwise.} \end{cases}$ for $i, j \in I$;
    for $(u, i) \in \bar{\Omega}$ do
        $x_k \leftarrow \begin{cases} r_{uk} & \text{if } (u,k) \in \Omega \text{ and } s_{ik} = 1 \\ -\infty & \text{otherwise.} \end{cases}$ for $k \in I$;
        $\bar{x} \leftarrow TruncateInfinity(x)$;
        if $length(\bar{x}) \geq t$ then
            sample $r_{ui}$ from $N(mean(\bar{x}), var(\bar{x}))$;
        end
    end
    $\Omega' \leftarrow$ all known entries in $R$;
    $\hat{R} \leftarrow$ solve problem (3.1) using TFOCS;
end
```

3.2 Regression Method

In the above section, let us consider one pair of $(u, i) \in \bar{\Omega}$. The item that corresponds to each entry of $\bar{x}$ always has similarity 1 with regard to item i. This is because similarity is truncated to be binary in (3.2). However, truncation may cause lost of information. In fact, using original similarities as predictor and ratings as response variable, we can build

measure	$s(a,b)$	range
cosine similarity [1]	$\frac{a \cdot b}{\|\|a\|\|\|\|b\|\|}$	$[0,1]$
Jaccard index [2]	$\frac{\|a \cup b\|}{\|a \cap b\|}$	$[0,1]$
Sørensen-Dice coefficient [2]	$\frac{2\|a \cup b\|}{\|a\| + \|b\|}$	$[0,1]$
Overlap coefficient [2]	$\frac{2\|a \cup b\|}{\min(\|a\|, \|b\|)}$	$[0,1]$

Table 3.1: Different similarity measures

[1]Apply to vectors

[2]Apply to sets and binary vectors. For binary vectors, $|\cdot|$ is number of non-zero entries.

a regression model and predict unknown ratings. In what follows, we will describe the details.

3.2.1 Naive Regression Method

Firstly we compute item similarity matrix S' without truncation. For $i, j \in I$,

$$s'_{ij} = \text{similarity between } A_i \text{ and } A_j, \tag{3.5}$$

where A_i is the i-th column of A. Then $S' = [s'_{ij}]_{M \times M}$. Note that the similarity is not limited to cosine similarity, various similarity measure of vectors can be used in this situation. Some of them are listed in Table 3.1.

Now for each $(u,i) \in \bar{\Omega}$, we define the similar rating vector $x' \in \bar{\mathbb{R}}^M$. For $k \in I$

$$x'_k = \begin{cases} r_{uk} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases} \tag{3.6}$$

Meanwhile we define a corresponding similarity vector $y' \in \bar{\mathbb{R}}^M$. For $k \in I$

$$y'_k = \begin{cases} s'_{ik} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases} \tag{3.7}$$

Then we truncate x' and y' like before

$$\bar{x'} = \{x'_k : x'_k \neq -\infty\},$$
$$\bar{y'} = \{y'_k : y'_k \neq -\infty\}.$$

Note that x' and y' take on $-\infty$ value at the same set of indices, therefore $\bar{x'}$ and $\bar{y'}$ are of the same length. For some index j, $\bar{x'}_j$ is the rating that user u gives to this item while $\bar{y'}_j$ is the similarity between this item and item i.

Now we can consider the following regression model

$$rating = \alpha \times similarity + \beta + \epsilon. \tag{3.8}$$

We train this model with data $rating = \bar{x'}$ and $similarity = \bar{y'}$. Remember that now we want to predict the rating that user u gives to item i (i.e. r_{ui}). The similarity between item i and itself, is exact s'_{ii}. Then we can predict r_{ui} using trained regression model and new data s'_{ii}.

Note that the above linear regression model may not be sufficient. The reason that we use linear regression model is for simplicity. It is actually a start point. Based on simple linear regression, we can use different optimizations and get better models. The improvement introduced in the next part is an example. Other non-linear transformation is also applicable.

Using the same strategy as above section, we only predict entries with dimension of $\bar{x'}$ exceeding some threshold t. That is, we exclude those with small data size. Because we believe that usually data of small size will incur uncertainty and inaccurate predictions. Now we will have some ratings besides Ω, they together forms Ω'. Then using TFOCS to solve problem (3.1), we are done with our problem.

We conclude the above processes with Algorithm 2.

The content attributes are binary, and the similarity measures we used (including cosine) are basically based on counting how many content features two items have in common. As a result, we will have many "zero" similarities. However the ratings with "zero" similarities are more like random numbers. Including those data will significantly decrease the performance of regression model. Figure 3.1 is an example of illustration. Therefore, we exclude those "zero" similarities as defined in (3.5) and (3.6).

Algorithm 2: Regression based method

```
Input: $R$, $A$, $SimilarityMeasure$, $t$
Output: $\hat{R}$
begin
    $s'_{ij} \leftarrow$ similarity between $A_i$ and $A_j$ using $SimilarityMeasure$ for $i, j \in I$;
    for $(u, i) \in \bar{\Omega}$ do
        $x'_k \leftarrow \begin{cases} r_{uk} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases}$ for $k \in I$;
        $y'_k \leftarrow \begin{cases} s'_{ik} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases}$ for $k \in I$;
        $\bar{x'} \leftarrow TruncateInfinity(x')$;
        $\bar{y'} \leftarrow TruncateInfinity(y')$;
        if $length(\bar{x'}) \geq t$ then
            $RegressionModel \leftarrow TrainRegressionModel(\bar{x'}, \bar{y'})$ ;
            $r_{ui} \leftarrow predict(RegressionModel, s'_{ii})$;
        end
    end
    $\Omega' \leftarrow$ all known entries in $R$;
    $\hat{R} \leftarrow$ solve problem (3.1) using TFOCS;
end
```

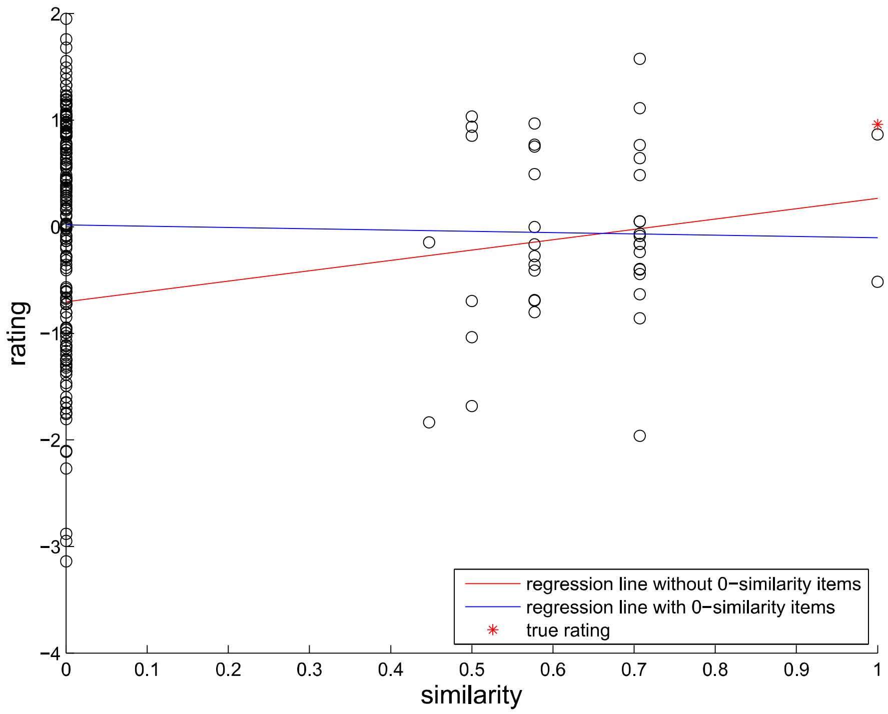

Figure 3.1: The effect of removing 0-similarity items.

In Figure 3.1, we use cosine similarity and self-similarity is 1 in this case. Therefore, the predicted rating is the value regression line at similarity = 1. Clearly, the regression line without "zero" similarities is a better prediction.

3.2.2 Improved Regression Method

Some improvements can be applied to Algorithm 2. An obvious one is as follows. It is natural that for one certain item, most items are not similar to it while only a small fraction have high similarity. This means most points will be in left part of the scatter plot (you can observe this if you remove all 0-similarity points in Figure 3.1). That is, the data are unbalanced. We can resolve this by taking *log* value of similarity. Since we exclude items 0 similarity, this modification is well defined. The improved algorithm is described in Algorithm 3. Figure 3.2 illustrates the improvement.

Algorithm 3: Improved regression based method

```
Input: $R$, $A$, $SimilarityMeasure$, $t$
Output: $\hat{R}$

begin
    $s'_{ij} \leftarrow$ similarity between $A_i$ and $A_j$ using $SimilarityMeasure$ for $i, j \in I$;
    for $(u, i) \in \bar{\Omega}$ do
        $x'_k \leftarrow \begin{cases} r_{uk} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases}$ for $k \in I$;
        $y'_k \leftarrow \begin{cases} s'_{ik} & \text{if } (u,k) \in \Omega \text{ and } s'_{ik} \neq 0 \\ -\infty & \text{otherwise.} \end{cases}$ for $k \in I$;
        $\bar{x'} \leftarrow TruncateInfinity(x')$;
        $\bar{y'} \leftarrow TruncateInfinity(y')$;
        if $length(\bar{x'}) \geq t$ then
            $RegressionModel \leftarrow TrainRegressionModel(\bar{x'}, log(\bar{y'}))$ ;
            $r_{ui} \leftarrow predict(RegressionModel, s'_{ii})$;
        end
    end
    $\Omega' \leftarrow$ all known entries in $R$;
    $\hat{R} \leftarrow$ solve problem (3.1) using TFOCS;
end
```

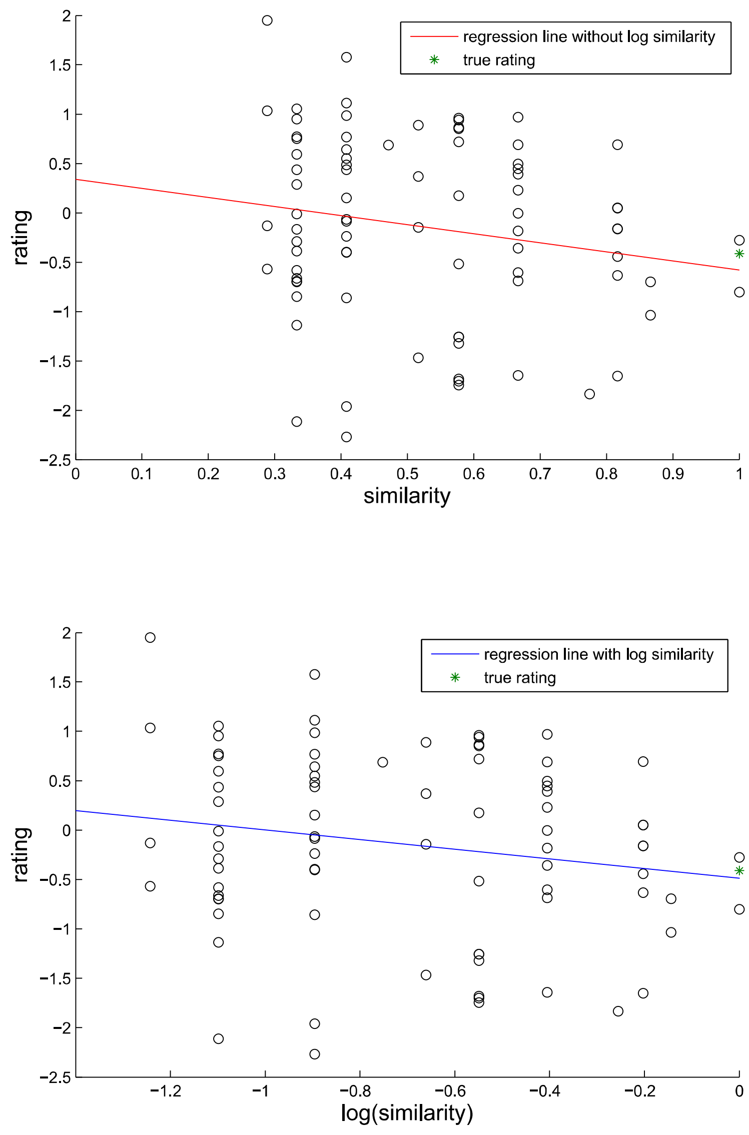

Figure 3.2: The effect of taking log of similarity.

In Figure 3.2, we use cosine similarity and self-similarity is 1 in this case. Therefore, in top part the predicted rating is the value regression line at similarity $= 1$ while in bottom part the predicted rating is the value regression line at similarity $= 0$ $(= log(1))$. Clearly, the regression line using log similarity is a better prediction.

Chapter 4

Experiments

In this chapter, we will show the performance of our algorithms.

4.1 Data

We used data set 'Movies', which is the MovieLens100K data set from GroupLens. The statistics of Movies data set is listed in Table 4.1. The item content information is an binary vector of whether movie i belongs to genre d. Notice that we have D genres in total and one movie can belong to multiple genres.

Statistics	Value
Number of users N	943
Number of items M	1682
Value of ratings r_{ui}	$\{1, 2, 3, 4, 5\}$
Number of attributes D	19
Number of known ratings $\lvert\Omega\rvert$	100000
Density ratio $\lvert\Omega\rvert/(MN)$	6.3%

Table 4.1: Statistics of 'Movies' data set

4.2 Evaluation

To evaluate our algorithms, we repeated each same experiment 10 times. Each time, we sampled 30% of the user-item pairs $(u, i) \in \Omega$ as test set, denoted by Ω_T. Using the remaining 70% as the known ratings (Ω), we recovered the user-item rating matrix $\hat{R}$. Ratings for all $(u, i) \in \Omega_T$ were predicted by $\hat{r}'_{ui}$ which is a truncation of $\hat{r}_{ui}$ when $\hat{r}_{ui}$ fell outside $[1, 5]$. That is

$$\hat{r}'_{ui} = \begin{cases} 1 & \text{if } \hat{r}_{ui} < 1 \\ \hat{r}_{ui} & \text{if } 1 \leq \hat{r}_{ui} \leq 5 \\ 5 & \text{if } \hat{r}_{ui} > 5 \end{cases}$$

Finally, we use root mean square error (RMSE) to evaluate our prediction:

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

4.3 Normalization

Before we proceed, we briefly talk about normalization. In collaborative-filtering algorithm, the ANOVA-type of model is often a simple but significantly useful method [7] [11]. The simplest ANOVA-type model is

$$r_{ui} = \mu + \alpha_u + \beta_i + \epsilon_{ui},$$

where ϵ_{ui} is white noise, μ is the overall mean, α_u represents a user-effect, and β_i represents an item-effect. It is common to normalize the user-item rating matrix R by removing such an ANOVA-type model before applying any algorithm. In all of our experiments, all algorithms were applied to $r_{ui} - \hat{\mu} - \hat{\alpha_u} - \hat{\beta_i}$, and the prediction was $\hat{r}_{ui} + \hat{\mu} + \hat{\alpha_u} + \hat{\beta_i}$, where $\hat{r_{ui}}$ is the prediction of our algorithms, and $\hat{\mu}, \hat{\alpha_u}, \hat{\beta_i}$ were the MLEs of μ, α_u, β_i. In order not to further complicate our notation, however, this detail will be suppressed and the notations, r_{ui} and R, is unchanged[15].

4.4 Parameter Setting

We used cross validation technique to set parameters. Our choice of parameter is listed in Table 4.2. For similarity measure, we used cosine similarity, Sørensen-Dice coefficient and

	Similarity Measure	t	c
Sampling Method	cosine similarity	70	0.5
Naive Regression Method	Sørensen-Dice coefficient	100	
Improved Regression Method	Sørensen-Dice coefficient	80	

Table 4.2: Choice of parameters

Jaccard index. Note that Algorithm 1 always use cosine similarity and c is not needed for Algorithm 2 and 3.

4.5 Size of Ω

In the beginning of Chapter 3, we have a key assumption that larger size of Ω implies more accurate prediction. We will illustrate this assumption in this section.

Firstly we randomly sampled 10000 known ratings as test set. Then each time, we random sampled different number of known rating from the rest 90000 entries as training set. And we ran matrix completion algorithm on training set and evaluated RMSE on test set. The results are in Figure 4.1. As we can see, if the number of known rating is large, then we will have more accurate prediction. This confirms our idea proposed in Chapter 3.

Also, the $|\Omega'|$ should not be significantly larger than $|\Omega|$. Because our goal is in whether filling in "some" entries first by sampling/regression based on content similarity can boost matrix completion. But we still want overwhelming majority of the predictions still come from matrix completion. Therefore in our experiment, we restricted that $|\Omega'| < 2|\Omega|$ by choosing proper parameters.

4.6 Results

Figure 4.2 summarize the results of our algorithms. "Baseline" applied matrix completion without any content information. For each algorithms, we used the optimal parameters in Table 4.2. Although Algorithm is the most simple one, the performance is best in our experiment. The Naive Regression Method (Algorithm 2) is worst, its accuracy is almost the same as original matrix completion algorithm. The effect of improvement in regression method is significant, result a 5% decrease in RMSE.

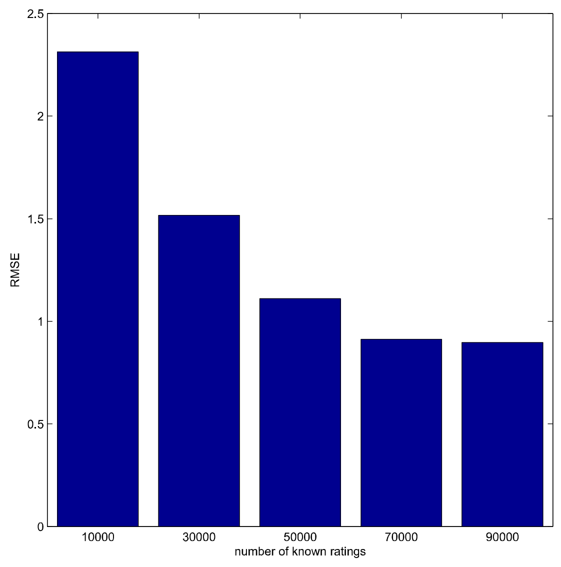

Figure 4.1: The effect of $|\Omega|$

In Figure 4.3, we are interested in the comparison of Algorithm 2 and 3. As for parameter, we set $t = 80$. The result is that Algorithm 3 has better prediction accuracy than Algorithm 2.

From Figure 4.2 and 4.3, we notice that the variance of RMSE is rather large in all our experiments. The reason is that the matrix completion algorithm we used (TFOCS) is not stable. This is a potential drawback of our method.

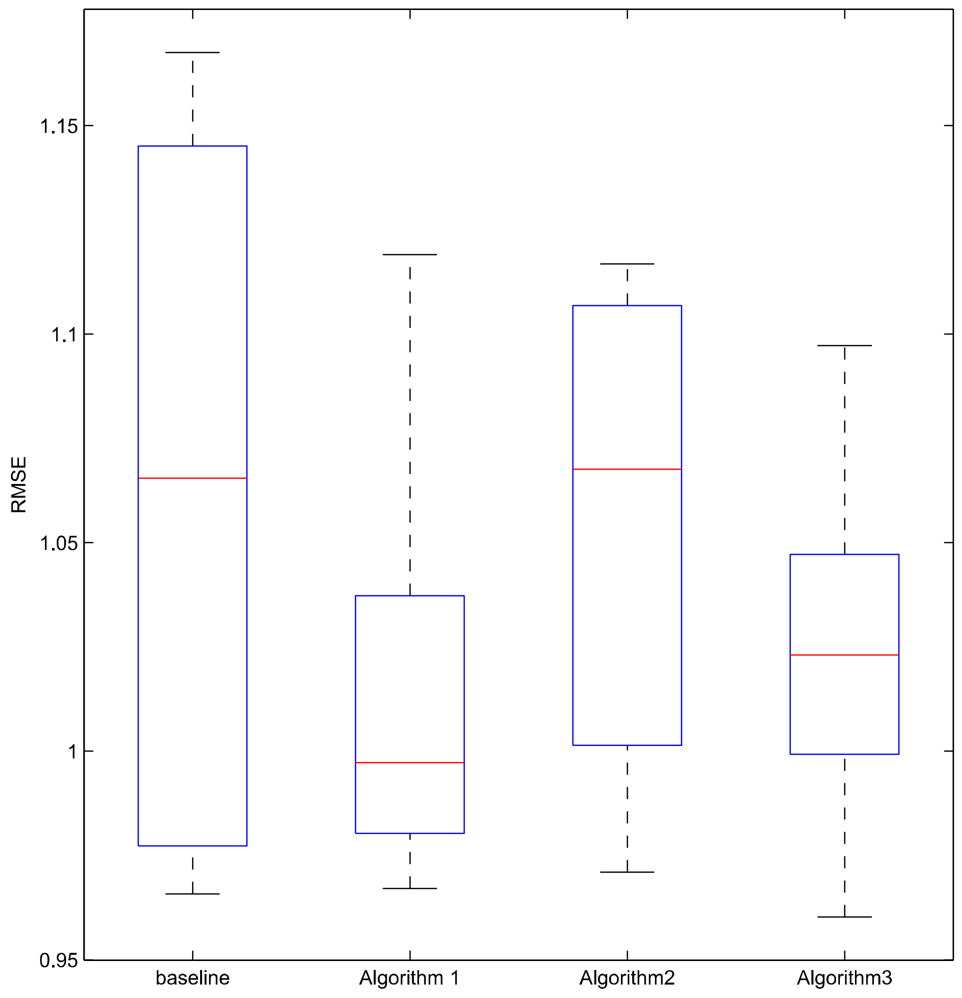

Figure 4.2: Performance of content boosted algorithms. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers (maximum and minimum), and outliers are plotted individually (red cross mark).

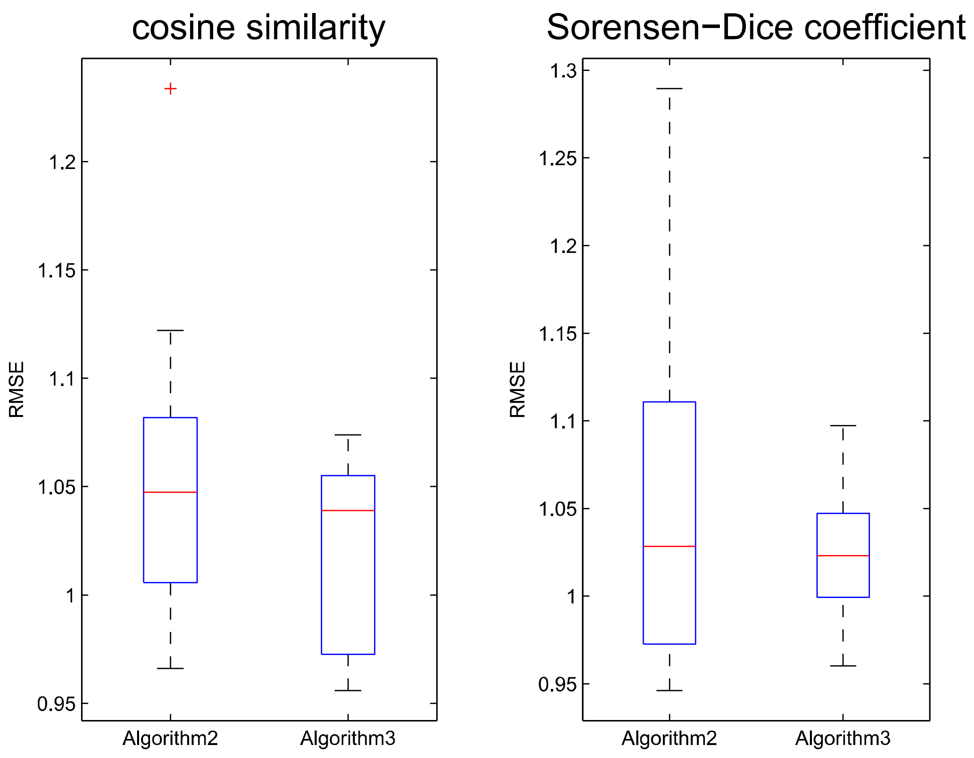

Figure 4.3: Comparison of Algorithm 2 and 3. Explanation for the box plot can be found at caption of Figure 4.2

Chapter 5

Summary

To sum up, we have proposed a novel approach to incorporate content information into the matrix completion approach for collaborative filtering. Our methodology focuses on increasing the size of know ratings before applying matrix completion. Experiments have shown that these content-boosted algorithms can achieve better recommendation accuracy.

Our approach include sampling method and regression method. However, other method that can increase the size of know ratings may also be helpful. Here we list some potential ideas:

- Improve the estimation of mean and variance of Normal distribution in sampling method.
- Use other distributions in sampling method.
- Optimize regression model in regression method
- Use other prediction algorithms instead of regression model in regression method.

The matrix completion approach for collaborative filtering is a research problem with various opportunities for further research. We hope that our project can inspire researchers to think about this problem in a more systematic way and contribute more to this problem.

References

[1] Stephen R Becker, Emmanuel J Candès, and Michael C Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.

[2] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[3] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[4] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[5] Alexander L Chistov and D Yu Grigor'ev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Mathematical Foundations of Computer Science 1984*, pages 17–31. Springer, 1984.

[6] David L Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[7] Andrey Feuerverger, Yu He, Shashi Khatri, et al. Statistical significance of the netflix challenge. *Statistical Science*, 27(2):202–231, 2012.

[8] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[9] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[10] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

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

[12] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[13] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[14] Marie Michenková. Numerical algorithms for low-rank matrix completion problems, 2011.

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

[16] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[17] CVX Research Inc RSS. Demo: Matrix completion., August 2014.

[18] Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. Knowledge infusion into content-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 301–304. ACM, 2009.

[19] Min Tao and Xiaoming Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[20] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[21] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 6(615-640):15, 2010.