2014

A COMPARATIVE STUDY AND EVALUATION OF COLLABORATIVE RECOMMENDATION SYSTEMS

Joshua C. Stomberg
Michigan Technological University

A COMPARATIVE STUDY AND EVALUATION OF COLLABORATIVE RECOMMENDATION SYSTEMS

By

Joshua C. Stomberg

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Computer Science

Report Advisor: *Dr. Laura Brown*

Committee Member: *Dr. Timothy Havens*

Committee Member: *Dr. Allan Struthers*

Department Chair: *Dr. Charles Wallace*

Contents

List of Figures

List of Tables

Abstract

Consumers currently enjoy a surplus of goods (books, videos, music, or other items) available to purchase. While this surplus often allows a consumer to find a product tailored to their preferences or needs, the volume of items available may require considerable time or effort on the part of the user to find the most relevant item. Recommendation systems have become a common part of many online business that supply users books, videos, music, or other items to consumers. These systems attempt to provide assistance to consumers in finding the items that fit their preferences. This report presents an overview of recommendation systems. We will also briefly explore the history of recommendation systems and the large boost that was given to research in this field due to the Netflix Challenge. The classical methods for collaborative recommendation systems are reviewed and implemented, and an examination is performed contrasting the complexity and performance among the various models. Finally, current challenges and approaches are discussed.

Chapter 1

Introduction

Currently, we are facing an ever growing deluge of information, music, videos, books, and other commercial merchandise. While this provides new opportunities to find goods tailored to our needs or preferences, the overall abundance serves as an increasing barrier to find items of interest or relevance. Additionally, high visibility is given to only the small fraction of items that are popular across diverse groups. This small fraction of items account for a large percentage of sales and has the effect of creating a long tail in the distribution, lots of items that individually have small sales volume, but together still constitute a significant percentage of total sales. The long tail distribution in both sales and visibility provides a hindrance to the discovery of items most relevant to a consumer's needs.

Recommendation systems have become an increasingly prevalent presence in our lives as our choices in music, videos, books, and even household goods expand. Companies like Netflix, Pandora, and Amazon look to harness the power of recommendation systems to increase their bottom line. However, they also represent an opportunity for users/customers to find relevant items without wasting precious time wading through the wide array of choices available to them.

In this report, we will be looking at recommendation systems that utilize a range of predictive models of gradually increasing complexity, each building upon previous models, and comparing their accuracy as measured by their root mean squared error (RMSE) on the MovieLens 100k dataset. In Chapter 2, we briefly discuss the background of recommendation systems, the three basic types of recommendation systems, and their various types of input and output. In Chapter 3, we explore some of the methods available within the collaborative type of recommender systems, specifically those implemented and evaluated in this project. In Chapter 4, we will evaluate the performance of several recommender systems on the MovieLens dataset. Finally in Chapter 5, we summarize the work in this project and discuss future directions.

Chapter 2

Background

The goal of a recommender system is to predict or recommend for every user, the items (video, audio, text, articles, ...) that would hold the greatest interest or satisfaction to that particular user. The standard recommendation systems in concerned with a set of users, U, and a set of items, I. Let there be a utility function, s, generally approximated by the set of ratings, R, that measures the usefulness, interest, or relevance of item i to user u. Initially, s is only defined with the elements in R. The purpose of a recommendation system is to extrapolate the value of s for unknown (u,i) pairs.

2.1 Recommendation Systems

Recommendations systems can be categorized in a wide range of types. There are three basic data models, multiple types of input data, and two basic output formats used in all recommendation systems. The most successful systems seek to leverage all model types and harness all available data. The three types of computation models are: (i) collaborative/social-based filters, which use ratings from thousands of users and statistical analyses to find correlations among users and/or items to create predictions; (ii) content-based filters, which use data from description and/or analysis of their item set and the user's ratings to create predictions; and (iii) knowledge-based methods, which use ratings and the action history of the userbase to learn rules that describe desired items, then uses those rules to create predictions based on a user's explicitly stated preferences.

Among the several types of system input are (i) text/video/audio, the actual content of the item or a description/analysis of the content; (ii) views/actions, can be binary in nature or numeric to capture repeated visits to, selection of, or sales of items; and (iii) ratings, generally an Likert scale with a numeric representation for expressing preferences (e.g., 1 = Hated It, 2 = Didn't Like It, 3 = Liked It, 4 = Really Liked It, 5 = Loved It).

The two types of system output are ratings and top-n lists. The first is a prediction of the rating given a specific user and item. The second looks at the predicted ratings for all

unrated items to create a list of the n items that a user is most likely to enjoy.

In their 2011 survey paper, Ekstrand et al. gives a comprehensive overview of basic collaborative filtering recommendation systems [1]. We follow their general progression of models in this project and extend upon them, analyzing additional models with various stages of bias removal and building upon each of them. We begin with basic statistical models that can be generated with just a few passes through the dataset. Next, we explore the models created by methods utilizing matrix factorization of the user-item rating matrix. This approach attempts to determine the latent factors within each item and each users preference for those factors in an SVD-like decomposition of the ratings matrix and then leverage that simpler representation to estimate future ratings. Then we examine more complex collaborative filters that attempt to identify similarity among users or items through the latent factors determined. This similarity identifies like items or users whose past ratings are used to predict our future expected rating.

2.1.1 Types

Collaborative- or social-based were the first and are likely the most widely used type of recommendation system. Systems of this type seek to leverage ratings data from large numbers of users to find items of interest to recommend. The first methods utilized the *k*-nearest neighbor algorithm. These methods sought to find users and/or items

with greatest similarity and use that additional information to make its predictions or recommendations.

The content-based systems use similar methods as the collaborative filtering systems, but utilize a different set of data. The data used by content-based systems relies on features inherent to the item and not on ratings given to items by the users. Pandora serves as an excellent example of a content-based recommender. Pandora analyzes songs to calculate what they term a song's "musical genome," and use that to find and play songs similar to the songs that a user has rated highly. Use of a content-based has several advantages and disadvantages. It allows for significantly greater accuracy over a social-based model during a "cold start," the initial period of a recommender system that starts with no ratings. It requires developing a more compact representation of the content in addition to choosing features from analysis of the content to provide reduced computation expenses. Some common features include: director, actors, genre (for movies), length, topic, author, word choice (for text), musician, lyrics, genre, key, and bpm (for music).

Knowledge-based systems attempt to learn rules and then use logic to make their recommendations. These systems work best in situations where ratings are sparse, due to the low frequency of their occurrence like house or car purchases, or where requirements need to be more precisely specified. Burke describes these systems as more of a conversational system as opposed to information filtering [2]. There are two basic types of knowledge-based systems: constraint-based, that work by satisfying rules, and case-based,

similarity metrics, system. The first might apply to home purchases. A prospective buyer specifies a price range and the systems works to provide them with available houses within that range. This type has a greater similarity to query-type systems than any other type of recommender systems.The second could be used in a local food finder that attempts to find nearby restaurants with food similar to other restaurants that the user has rated highly.

2.1.2 Uses

The most visible use of recommendation systems today comes in the form of commercial utilization. Amazon, Netflix, and Pandora are just a few of the companies that use recommendation systems in an effort to help customers/users find the items most relevant to them. Netflix uses your past ratings on movies to predict how much you will enjoy other movies. Pandora uses their thumb up/down ratings and content-based system to adjust the selection of songs played on each radio station. Amazon recommends related items based on past purchases, item views, and suggests related items that other customers purchased when viewing an item. These companies hope that by reducing the barrier to finding what their customers want they can increase sales, satisfaction, and stability in the consumer base.

2.1.3 History

The idea of a recommendation system was first proposed in the early 1990s [3]. Its goal was to help Usenet users find interesting and useful content on the infant internet. The earliest approaches would now be classified as collaborative filtering, they sought to find similar users to the questor and utilize the data available to build personal filters [3]. At the same time, the internet business bubble was just beginning to expand and recommendations systems were caught up with it. This increased commercialization of recommender systems drove improvement in a number of areas. First, recommender systems now needed to provide value in addition to the accuracy they were already demonstrating. Second, the size of the datasets being used had increased exponentially in the transition from a research to a commercial environment. Additionally, long delays in computation were no longer acceptable when being used in a rapidly changing online marketplace. Finally, marketing professionals were more interested in lists of items that would be most relevant to a user driving a shift in how these systems provide results. The Netflix prize would provide the next large boost to recommendation system research [3].

2.1.3.1 Netflix Prize

In 2006, Netflix announced a competition with a grand prize of $1 million to the team that could produce a recommendation algorithm that would beat their current system, Cinematch, by 10% [4]. The ratings data they provided consisted of over 100 million ratings over 17,700 movies, made by almost 500,000 users. Less than a week after the beginning of the competition, there were already teams that were scoring better on the quiz set than the Cinematch algorithm. However, it would take almost 3 years for the 10% improvement to be reached. Initially there was a large explosion in the number of teams that submitted results, over 20,000 teams from over 150 countries registered in the first eight months. However that initial interest tapered off quite sharply after the first year. The final year there was actually a fairly significant drop in the number of teams competing due to teams combining and joining their models in an effort to increase performance [5].

Chapter 3

Methods

Throughout this chapter I will be employing a subset of ratings data (Table. 3.1) arbitrarily extracted from the MovieLens dataset for use in illustrating the methods being employed.

	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

Figure 3.1: Example Set of Ratings

3.1 Bias Removal

We will start with three types of bias removal attempting to transform the set of ratings, R, into a more normalized distribution. The first type of bias removal calculates the mean rating, μ, over all users and items (see Fig. 3.2(a)).

$$\mu = \frac{\sum_{r_{ui} \in R} r_{ui}}{|R|}. \tag{3.1}$$

This has effect of centering the ratings. Thus for our example set, the general bias, or μ, is equal to 3.33.

The second and third types of bias removal repeats the same process for the subset of ratings for each user, R_u (in Fig. 3.2(b)), and item, R_i (in Fig. 3.2(c)), respectively. Due the the potential small sample sizes for user and/or item ratings, Laplace smoothing is employed to prevent outliers from having as great an impact. For our evaluation, we chose a value of $L = 25$, as suggested by Ekstrand and Funk [1, 6]. The user bias, B_u, is calculated for each user of the data set as

$$B_u = \frac{\sum_{r_{ui} \in R_u} (r_{ui} - \mu)}{|R_u| + L}. \tag{3.2}$$

	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

(a) Entire Set

	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

(b) Subsetted by User

	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

(c) Subsetted by User

Figure 3.2: Bias Calculations

Similarly, the item bias, B_i, is calculated as

$$B_i = \frac{\sum_{r_{ui} \in R_i} (r_{ui} - \mu)}{|R_i| + L}. \tag{3.3}$$

For our example data, we do not use Laplace smoothing and find the following vectors: user bias, $B_u = [0, 0.67, 0.33, 0, 0.16, -1.33]$, and item bias, $B_i = [0.67, -1, -0.83, 1, 0.67]$.

Each bias, B_u or B_i, is also calculated as a residual bias, b_u or b_i, after the removal of the

other as

$$b_u = \frac{\sum_{r_{ui} \in R_u} (r_{ui} - \mu - B_i)}{|R_u| + L}, \text{ and} \tag{3.4}$$

$$b_i = \frac{\sum_{r_{ui} \in R_i} (r_{ui} - \mu - B_u)}{|R_i| + L}. \tag{3.5}$$

For our example data, we do not use Laplace smoothing and find the following vectors, residual user bias, $b_u = [0.28, -0.16, -0.44, 0.39, 0.25, -0.42]$, and residual item bias, $b_i = [0.33, -0.56, -0.54, 0.67, 0.42]$.

The first model, BASEBIAS, is a basic system that uses the current mean rating, μ, calculated by Eq. 3.1, as the predicted rating, $\hat{r}_{ui} = \mu$. This is an extremely naive model, but it serves as a excellent starting point for construction of more complex models and as a baseline point from which future improvements may be measured.

The next model, USERBIAS, will attempt to start minimizing the prediction error of our baseline model by recognizing that users might have their own inherent biases that would be reflected in the ratings they give. Starting from the baseline mean, μ, we utilize the individual user's bias, B_u, calculated by Eq. 3.2 to improve our model, $\hat{r}_{ui} = \mu + B_u$.

Alternatively, the third model, ITEMBIAS, recognizes that just as there may be biases inherent to each user, there might also be a general bias associated with each item. As before, starting from the baseline mean, μ, we utilize the individual item's bias, B_i,

calculated by Eq. 3.3 to improve our model $\hat{r}_{ui} = \mu + B_i$.

The fourth model, U2I_BIAS, and fifth model, I2U_BIAS, are similar in nature. The models begin with using either user bias, B_u, or item bias, B_i, then computes the residual bias from the other as calculated in Eq. 3.4 or Eq. 3.5. The predicted ratings for the models, U2I_BIAS and I2U_BIAS, are then calculated as $\hat{r}_{ui} = \mu + B_u + b_i$ and $\hat{r}_{ui} = \mu + B_i + b_u$.

These five models serve as baseline predictors which our next models will use as a stepping stone to hopefully greater performance and are summarized in Table. 3.1.

Model Predictor	Model Name
$\hat{r}_{ui} = \mu$	BASEBIAS
$\hat{r}_{ui} = \mu + B_u$	USERBIAS
$\hat{r}_{ui} = \mu + B_i$	ITEMBIAS
$\hat{r}_{ui} = \mu + B_u + b_i$	U2I_BIAS
$\hat{r}_{ui} = \mu + B_i + b_u$	I2U_BIAS

Table 3.1
Bias Model Predictors

3.2 Singular Value Decomposition (SVD)

Any matrix, $\mathbf{M}$, can be decomposed into the following parts, $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{M}$ is a $m \times n$ matrix of real values, $\mathbf{U}$ is a $m \times k$ matrix of the right singular vectors of $\mathbf{M}$, $\mathbf{V}$ is a $n \times k$ matrix of the left singular vectors of $\mathbf{M}$, $\mathbf{\Sigma}$ is a $k \times k$ matrix of the singular values of $\mathbf{M}$, and k is an integer value between 1 and $\max(m,n)$ depending on the rank of the decomposition. The vectors in $\mathbf{U}$ and $\mathbf{V}$ form an orthonormal basis, and the values in $\mathbf{\Sigma}$ are

their importance. The use of matrix factorization not only provides the benefit of allowing the missing values within the matrix to be computed as predictions, but a reduced rank factorization allows for smaller model sizes, faster computation times, and dimensionality reduction among the data.

Due to the incomplete nature of $\mathbf{R}$, an alternative form of the decomposition is used as suggested by Paterek [7]. Instead of the standard $\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ formulation, we will use $\mathbf{R} = \mathbf{P}\mathbf{Q}^T$ where $\mathbf{P}$ and $\mathbf{Q}$ are $m \times k$ and $n \times k$ matrices. This formulation readily lends itself to interpretation of the vectors of $\mathbf{P}$ and $\mathbf{Q}$ as user interest and item membership in the *k features* respectively. Also, $\mathbf{P}$ and $\mathbf{Q}$ are found through an iterative method over the set of training data rather than being directly calculated. This method can be thought of as solving the following optimization problem

$$\arg\min_{\mathbf{P},\mathbf{Q}} \|I(\mathbf{R} - \mathbf{P}\mathbf{Q}^T)\|_2,$$

where I is an indicator function that returns 0 where $\mathbf{R}$ is undefined. An additional advantage of the PQ factorization is the use of $\mathbf{P}$ and $\mathbf{Q}$ as description vectors of users and items respectively in user-user and item-item collaborative filtering.

The main algorithm for performing the PQ factorization is show in Fig. 3.3. Each matrix is initialized with some non-zero value. The general consensus is to initialize both matrices with the same value. Then each column updated until squared error over known values shows less than a minimum improvement (MIN_IMPROV). For performance reasons, an

```
function TRAINPQ(k, R)
    P = 0.1
    Q = 0.1
    for i := 1 to k do
        PREV_ERROR = MAX_VALUE
        repeat
            for r_ui ∈ R do
                ERROR += UPDATEFACTOR(r_ui, u, i, k)
            end for
            ERROR = ERROR/||R||
        until ERROR+MIN_IMPROV < PREV_ERROR
    end for
end function
```

Figure 3.3: Pseudocode version of the PQ factorization

```
function UPDATEFACTOR(r, u, i, k)
    PRED = PREDICT(user, item)
    ERR = r − PRED
    P_uk = P_uk + γ ∗ (Q_ik ∗ ERR − λ ∗ P_uk)
    Q_ik = Q_ik + γ ∗ (P_uk ∗ ERR − λ ∗ Q_ik)
    return ERR^2
end function
```

Figure 3.4: Pseudocode used to update **P** and **Q**

additional criteria is often added to terminate the inner loop once a large number of passes have been completed without incrementing the outer loop.

The column update function uses a gradient descent method as described in the UPDATEFACTOR method of Fig. 3.4 that seeks to minimize the squared error over known values. The learning rate, γ, controls how quickly the factors converge. The most commonly used value for γ is 0.001, but can be up to 0.1. The normalization factor, λ, controls the penalty on the magnitude of the factors and prevents overfitting to the training data. Commonly used values for λ are in the iterval [0,0.2] [6, 8, 9, 7].

Both matrices are trained simultaneously, one column or *feature* at a time, using a ridge-based gradient descent method. A sum of squared error with each pass is used as termination criteria. Once that sum has failed to decrease by more than a certain amount from the prior pass we move onto to the next feature. As previously mentioned, there are a number of variables that control the training algorithm and influence the final model generated: the number of features (k), the learning rate (γ), and the normalization factor (λ). For this report these variables were selected using 3-fold cross-validation on the training set.

(a) Matrix Initialization

(b) After first pass on first factor

(c) After First Factor Completed

(d) After Second Factor Completed

Figure 3.5: PQ Decomposition of Example Data

In Fig. 3.5, we show how this works on the example data. We use $k = 2$, $\gamma = 0.01$, and $\lambda = 0$ for this example. We also limit the number of passes for each feature to 100 and initalize $\mathbf{P}$ and $\mathbf{Q}$ to $\mathbf{1}$ as seen in Fig. 3.5(a). In Fig. 3.5(b) we can see the changes to the first factor after a single pass though the known ratings. In Fig. 3.5(c) we can see the results after prediction error failed to decrease enough after the twenty-second pass. Finally, in Fig. 3.5(d) we can see the completed decomposition.

The next model, ZERO_SVD, uses a PQ factorization of the unadjusted ratings matrix, $\mathbf{R}$, to calculate predictions $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T$. The BASESVD model is similar, but builds upon the BASEBIAS model by adjusting the ratings matrix by the predictions of that model ($\mathbf{R} -$ BASEBIAS) $\approx \mathbf{P}\mathbf{Q}^T$. The prediction of BASE_SVD is then the combination of the matrix approximation with the baseline predictor, $\hat{\mathbf{R}} =$ BASEBIAS $+ \mathbf{P}\mathbf{Q}^T$. The USERSVD and the ITEMSVD models similarily builds upon the USERBIAS and the ITEMBIAS models. Likewise, the U2I_SVD and the I2U_SVD models builds upon the U2I_BIAS and the I2U_BIAS models respectively. A summary of the models can be found in Table. 3.2.

Model Predictor	Ratings Decomposition	Model Name
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T$	$\mathbf{P}\mathbf{Q}^T \approx \mathbf{R}$	ZERO_SVD
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T +$ BASEBIAS	$\mathbf{P}\mathbf{Q}^T \approx (\mathbf{R} -$ BASEBIAS$)$	BASESVD
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T +$ USERBIAS	$\mathbf{P}\mathbf{Q}^T \approx (\mathbf{R} -$ USERBIAS$)$	USER_SVD
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T +$ ITEMBIAS	$\mathbf{P}\mathbf{Q}^T \approx (\mathbf{R} -$ ITEMBIAS$)$	ITEM_SVD
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T +$ U2I_BIAS	$\mathbf{P}\mathbf{Q}^T \approx (\mathbf{R} -$ U2I_BIAS$)$	U2I_SVD
$\hat{r}_{ui} = \mathbf{P}_u\mathbf{Q}_i^T +$ I2U_BIAS	$\mathbf{P}\mathbf{Q}^T \approx (\mathbf{R} -$ I2U_BIAS$)$	I2U_SVD

Table 3.2
SVD Model Predictors

3.3 Collaborative Filtering

Collaborative filtering is an idea that is related to crowd sourcing. The basic idea is the use of large numbers of users and ratings to find similar items and users that can assist in the creation of predictions. Similarity measures are functions for determining how much one items is like another given a vector of features that describes them. Common similarity measures include cosine similarity and distance functions (Manhattan, Euclidean, Minkowski). For increased performance, item-item similarity is often used in conjunction with caching due to the lower volatility of their similarity measures. Requires a fairly significant number of ratings before any level of accuracy is guaranteed, however the accuracy of the systems will increase over time as more ratings, users, and items enter the system. New users and items need a certain number of ratings (items more so than users) before accurate predictions can be made even if the rest of the systems has achieved a higher level of accuracy.

Now that most of the bias has been eliminated from the ratings data, another approach to predicting ratings must be found if greater improvement is sought. One approach is to find similar users that have rated the item and use those ratings in our prediction. However, that raises the question of determining similarity between users.

3.3.1 Similarity Measures

In their paper, Herlocker et al. [10] provides an excellent discussion of the framework needed for collaborative filtering . For this project, we use cosine similarity, defined as follows:

$$\text{sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|},$$

where v_i is defined as a vector representation of the user or item. The representation of a user can be R_u, their entire set of ratings, or P_u, the vector of features from the SVD model can be used.The use of R_u has the advantage of being the most accurate representation of the user, but requires a specialized definition of the dot product operator to deal with ratings that are not available. On the other hand, P_u does not have that disadvantage, but could be a less accurate representation of the user. The representation of items is similar with identical tradeoffs.

Alternatively, similarity could be measured through distance, either Manhattan distance $d(v_1, v_2) = \sum_{n=|v_i|}^{j=0} |v_{1j} - v_{2j}|$, Euclidean distance $d(v_1, v_2) = \sqrt{\sum_{n=|v_i|}^{j=0} |v_{1j} - v_{2j}|^2}$, or even a multi-dimensional Minkowski distance $d(v_1, v_2) = \sqrt[k]{\sum_{n=|v_i|}^{j=0} |v_{1j} - v_{2j}|^k}$ measurement. However, the use of a distance measurement requires the various features to be normalized

to prevent any one feature from having an excessive impact on the measurement.

User\Item	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

(a) User-User Filtering

User\Item	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

(b) Item-Item Filtering

Figure 3.6: Neighborhood Creation for (U3,I2)

3.3.2 K-nearest neighbor

3.3.2.1 User-User Filtering

The algorithm we use for user based collaborative filtering begins with the set of training data after filtering for users that have a rating for the item we are attempting to predict (Fig. 3.6(a)). We then compute similarity, using the description vectors learned from the SVD models, between the remaining users and the user for whom we are attempting to predict. We select up to the 7 nearest neighbors to create a neighborhood K_u. Instead of using a weighted average of their ratings to directly compute a prediction, we seek to improve upon on SVD based models by computing the prediction error for our neighbors of the BASE model and then calculating an adjustment to our prediction using the weighted

average of the prediction error

$$\hat{r}_{ui} = \text{SVDMODEL}(u,i) + \frac{\sum_{a \in K_u} (r_{ai} - \text{BASE}(a,i) * sim(u,a))}{2\|K_u\|}. \qquad \text{(BASE\_UNN)}$$

Six models were generated using user-user collaborative filtering, each based on one of the six SVD models created; see Table. 3.3 for the complete list of collaborative filtering models.

3.3.2.2 Item-Item Filtering

An alternative approach to user based collaborative filtering instead uses items as the basis for comparison (Fig. 3.6(b)). The algorithm is similar to user based but instead looks for similar items already rated by that user to compute an adjustment to the prediction

$$\hat{r}_{ui} = \text{SVDMODEL}(u,i) + \frac{\sum_{a \in K_i} (r_{ua} - \text{BASE}(u,a) * sim(i,a))}{2\|K_i\|}. \qquad \text{(BASE\_INN)}$$

Item based algorithms have some advantage over user based as the description vector of items is often more stable, allowing the similarity measures to be previously computed and stored. Sarwar et al. discuss the various algorithms that can be used to perform item based collaborative filtering in their 2001 paper [11]. An additional six models were generated using item-item collaborative filtering, each based on one of the six SVD models created;

see Table. 3.3 for the complete list of collaborative filtering models.

Base Model	User based variant	Item based variant
ZERO_SVD	ZERO_UNN	ZERO_INN
BASE_SVD	BASE_UNN	BASE_INN
USER_SVD	USER_UNN	USER_INN
ITEM_SVD	ITEM_UNN	ITEM_INN
U2I_SVD	U2I_UNN	U2I_INN
I2U_SVD	I2U_UNN	I2U_INN

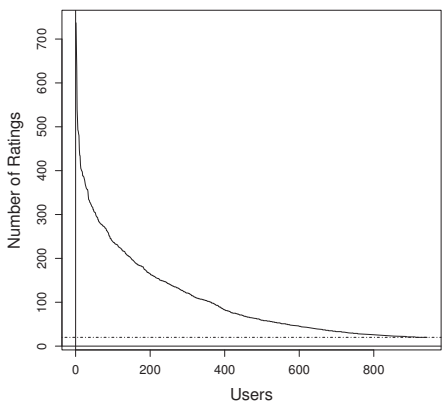
Table 3.3
Collaborative Filtering Models

Chapter 4

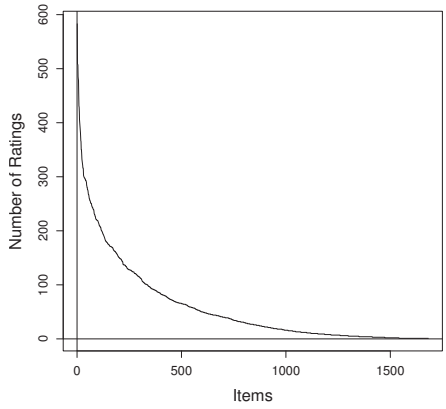
Experimental Evaluation

4.1 Source Data

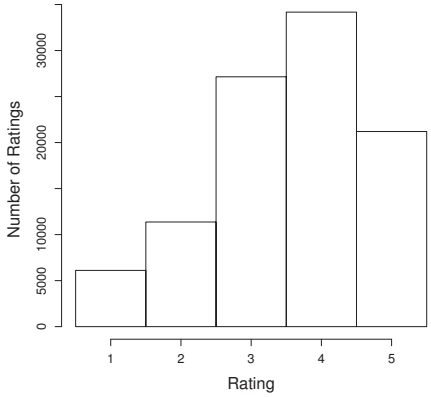
For this project, we will be using the 100k MovieLens dataset [12]. The dataset as provided by MovieLens contains 100,000 reviews from 943 users over 1682 items with every user having at minimum 20 ratings. Figure 4.1(a) and Figure 4.1(b) show the distribution of ratings per user and ratings per item. As might be expected, both follow a similar distribution with every item/user having at least a few ratings and a few items/users have a very large number of ratings. Figure 4.1(c) shows that there is a distinct skew in the ratings distribution toward higher ratings rather than a normal distribution that might be naively expected. The bias models will leverage this to predict future ratings.



(a) Distribution of Ratings Per User



(b) Distribution of Ratings Per Item



(c) Distribution of Ratings

Figure 4.1: MovieLens 100k data set

4.2 Results

4.2.1 Overview

After implementing each of the models mentioned in the previous chapter using the training data set, we now evaluate these models. A 4-way cross validation was performed for the purpose of performance analysis. Each fold was generated using a pseudorandom method that distributed user ratings randomly, but equally over the folds guaranteeing at least 5 ratings from each user would appear in each fold. The folds had slight (less than 0.1%) difference in size due to the pseudorandom nature of the fold generation. We compute the error for each element of the test set as follows,

$$error_{ui} = r_{ui} - \hat{r}_{ui}.$$

and compute the root mean squared error (RMSE) for each model. We plot the error over each element of the test sets, sorted in ascending order, creating a visual representation of the error distribution. The x-axis is then just an index. We also plot a sample-based pdf of the error. Here the x-axis is error and the y-axis is probability. Two of the error density plots have a larger y-axis scale due to the presence of the BASEBIAS model which has only discrete error values. We plot the models grouping them by model type in Fig. 4.2 though

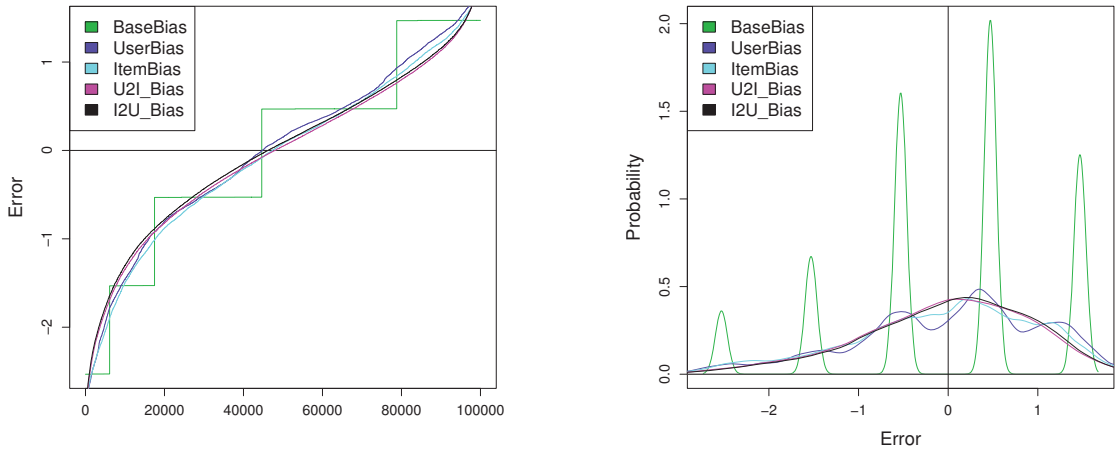
Model	RMSE1	RMSE2	RMSE3	RMSE4	RMSE	Std. Dev.
BaseBias	1.1234	1.1276	1.1252	1.1264	1.1257	0.00179
UserBias	1.0426	1.0465	1.0411	1.0462	1.0441	0.00241
ItemBias	1.0314	1.0362	1.0369	1.0362	1.0352	0.00269
U2I_Bias	0.9613	0.9662	0.9640	0.9672	0.9647	0.00312
I2U_Bias	0.9533	0.9574	0.9550	0.9587	**0.9561**	0.00242
Zero_SVD	0.9489	0.9528	0.9516	0.9557	0.9523	0.00232
Base_SVD	0.9569	0.9612	0.9612	0.9633	0.9607	0.00243
User_SVD	0.9461	0.9506	0.9494	0.9544	**0.9501**	0.00282
Item_SVD	0.9624	0.9661	0.9649	0.9680	0.9653	0.00254
I2U_SVD	0.9529	0.9570	0.9545	0.9583	0.9557	0.00220
U2I_SVD	0.9607	0.9656	0.9633	0.9666	0.9641	0.00234
Zero_UNN	0.9603	0.9650	0.9653	0.9687	0.9648	0.00266
Base_UNN	0.9601	0.9638	0.9660	0.9672	0.9643	0.00262
User_UNN	0.9536	0.9576	0.9588	0.9618	**0.9579**	0.00189
Item_UNN	0.9668	0.9691	0.9713	0.9729	0.9700	0.00263
U2I_UNN	0.9659	0.9697	0.9710	0.9728	0.9698	0.00292
I2U_UNN	0.9566	0.9599	0.9610	0.9634	0.9602	0.00267
Zero_INN	0.9443	0.9474	0.9458	0.9523	**0.9475**	0.00254
Base_INN	0.9454	0.9485	0.9483	0.9513	0.9484	0.00343
User_INN	0.9478	0.9468	0.9495	0.9526	0.9492	0.00340
Item_INN	0.9522	0.9521	0.9533	0.9568	0.9536	0.00347
U2I_INN	0.9560	0.9596	0.9589	0.9603	0.9587	0.00282
I2U_INN	0.9557	0.9527	0.9580	0.9538	0.9550	0.00345

Table 4.1
Accuracy of Implemented Models measured by RMSE

Fig. 4.5. We also plot the models grouping them by base model in Fig. 4.7 and Fig. 4.8.

In Table. 4.1, a summary of the various model's performance within each fold (RMSE*X*) and the mean RMSE and standard deviation (Std. Dev.) over all folds is given. The results will be discussed in greater detail by model type.

4.2.2 Bias Models



(a) Distribution **(b)** Density

Figure 4.2: Bias Model Error

We see a large range of performance among the bias models, from BASEBIAS to the composite bias models U2I_BIAS and I2U_BIAS. The improved performance between the USERBIAS and ITEMBIAS models may suggest item bias has a greater impact on the rating and therefore accuracy of the prediction than user bias. We can also see from the error distributions in Fig. 4.2 the immediate increase in accuracy once we move beyond the BASEBIAS model to include either user or item biases, and the additional increase when using both.

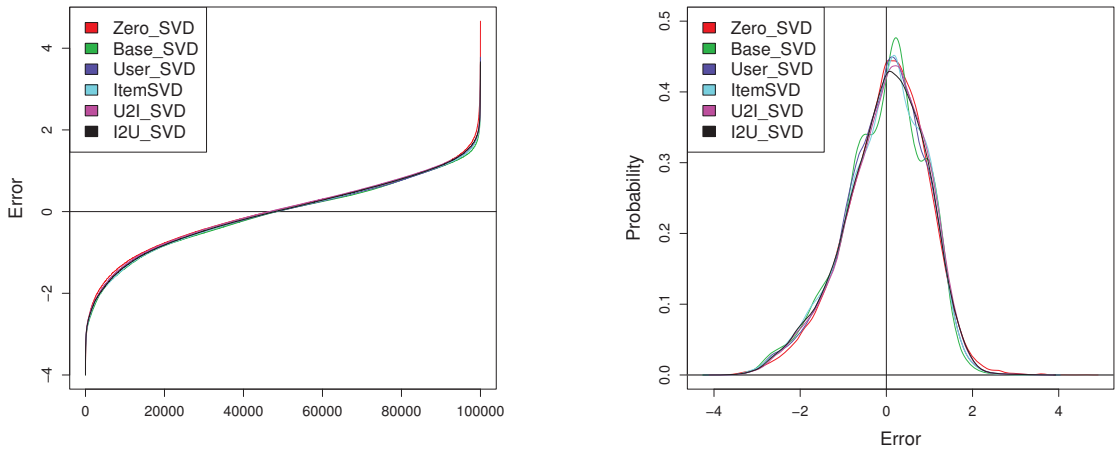
4.2.3 SVD Models

Model	K	γ	λ
ZERO_SVD	15	0.0025	0.2
BASE_SVD	40	0.01	0.2
USER_SVD	5	0.001	0.1
ITEM_SVD	40	0.02	0.2
I2U_SVD	5	0.01	0.2
U2I_SVD	5	0.02	0.2

Table 4.2
SVD Selected Parameters

As mentioned in 3.2, we began by choosing number of factors (K), learning rate (γ), and normalization factor (λ) using a three-fold cross validation over the set of training data. We evaluate twenty potential choices for K (5, 10, 15, ..., 100), eight different learning rates (0.001, 0.0025, 0.005, 0.0075, 0.01, 0.02, 0.03, 0.05), and six normalization factors (0, 0.015, 0.02, 0.05, 0.1, 0.2). We chose separately for each model (see Table. 4.2 for the values used).

We see a much smaller range of performance among the SVD models. The only model that stands out is the USER_SVD model, surprisingly outperforming the other models, even the I2U_SVD and the U2I_SVD models which incorporate an additional type of bias compensation. Also, its selected variables are different from any other model in both number of factors (5 vs. 40,75-80) and learning rate (0.005 vs 0.02-0.05). Its normalization factor is even different at 0.1 from all the others at 0.2. One point of interest is that ZERO_SVD model, even though its not significantly different in terms of performance,



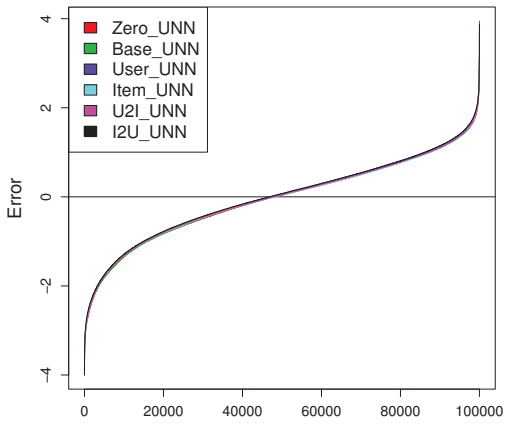
(a) Distribution **(b)** Density

Figure 4.3: SVD Model Error

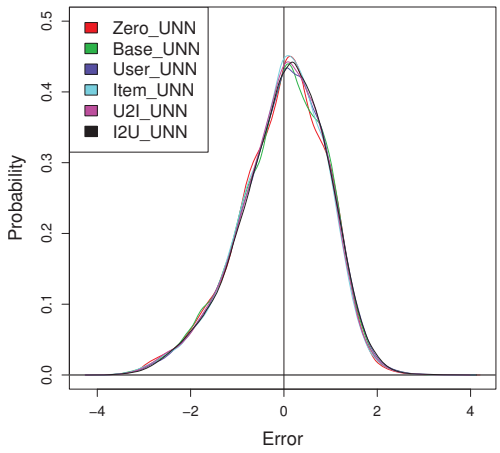
appears to have a slight positive offset relative to the other models in Fig. 4.3.

4.2.4 User Based Nearest Neighbor Models

Upon examining the user based nearest neighbors we find that contrary to our expectations, the performance of these models has decreased across the board with one exception. The I2U_UNN model registers a slight increase in performance measured by RMSE. Even the standard deviation, across the cross validation folds, of these models has increased over their SVD base models except for one. Although in this case that model is USER_UNN with the best performance in this model type.

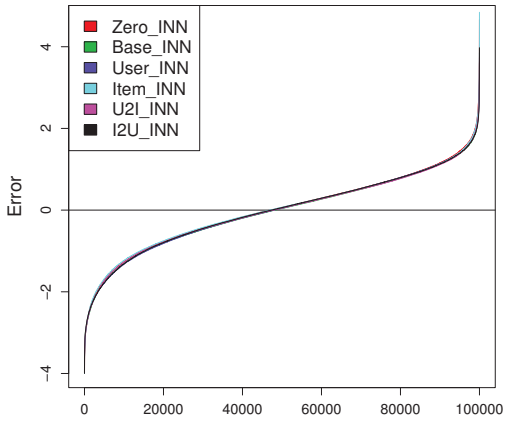


(a) Distribution

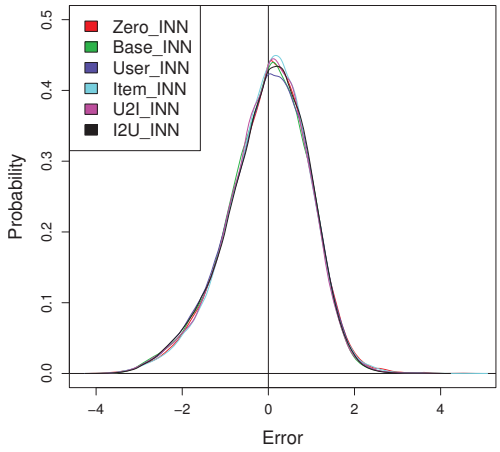


(b) Density

Figure 4.4: User Based Nearest Neighbor Error



(a) Distribution



(b) Density

Figure 4.5: Item Based Nearest Neighbor Error

4.2.5 Item Based Nearest Neighbor Models

When we examine the user based nearest neighbor models, we generally see slight improvement over their base SVD model. The sole exception is that U2I_INN has slightly worse performance than U2I_INN. Somewhat surprisingly, ZERO_INN has the best performance measured, though it is indistinguishable from both BASE_INN and USER_INN. Additionaly, we notice a small decrease in performance of the U2I_INN over its base model U2I_SVD. Overall, we see an increase in performance over the SVD type models with a continuing decrease in variance as measured by the standard deviation.

4.2.6 Summary

In summary, as can be seen in Fig. 4.7 and Fig. 4.8, models with increasing complexity generally produced more accurate predictions with the notable exception of the user based nearest neighbor filter. ZERO_INN produced the best results among item base models, though insignificantly better than BASE_INN and USER_INN. A more visual representation of the data in Table 4.1 can be seen in Fig. 4.6, where the average RMSE of the models are plotted with whiskers at plus or minus the standard deviation.

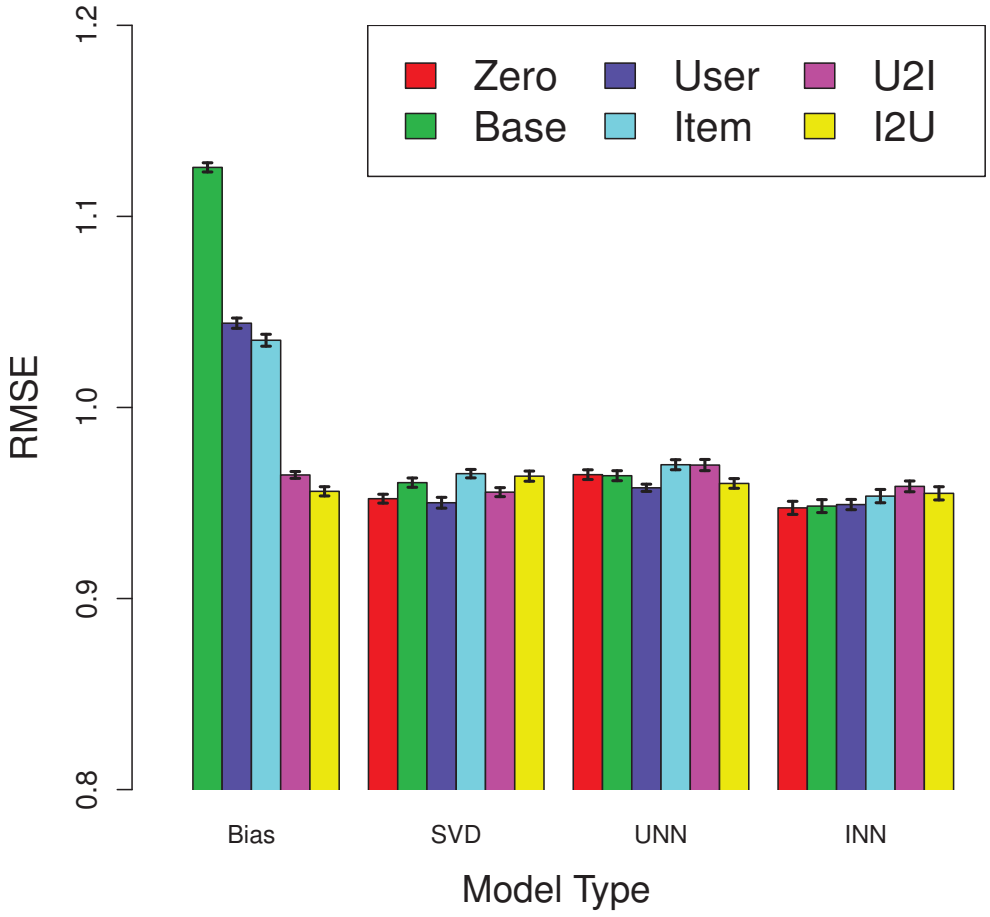
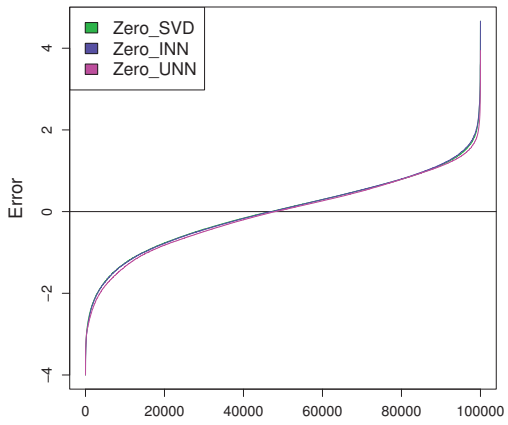
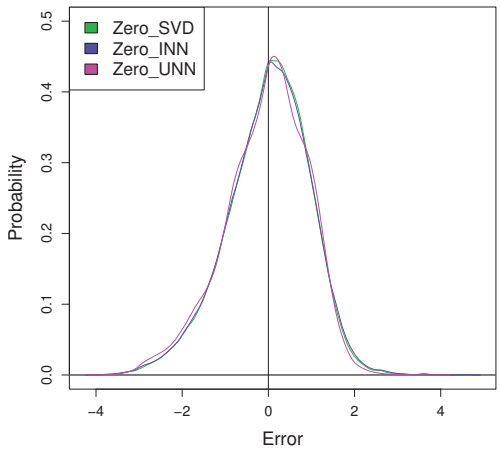


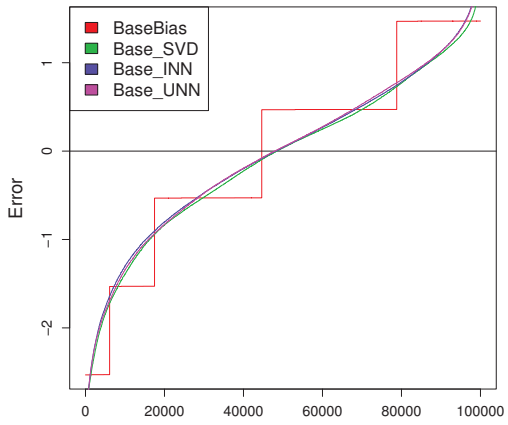
Figure 4.6: Overall Comparison



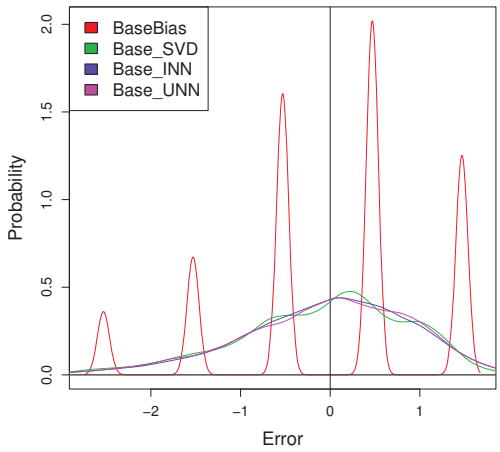
(a) Unaltered Error Distribution



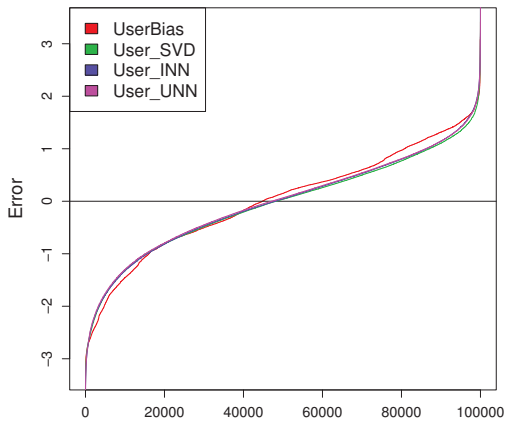
(b) Unaltered Error Density



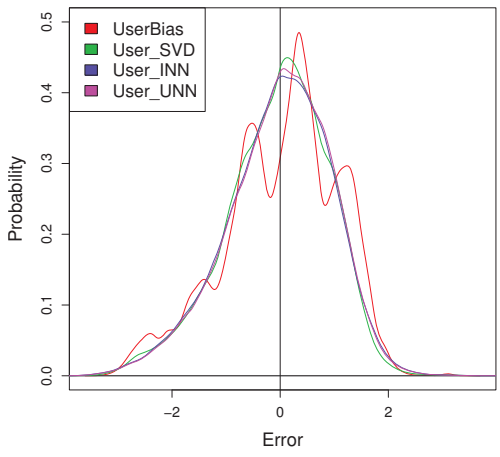
(c) Base Error Distribution



(d) Base Error Density

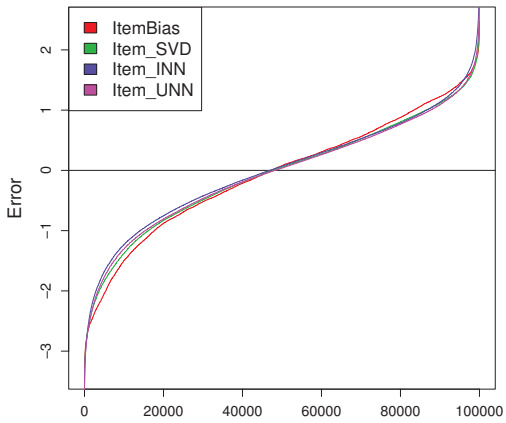


(e) User Error Distribution

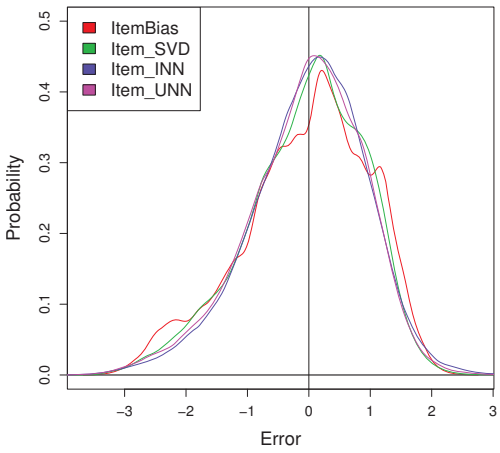


(f) User Error Density

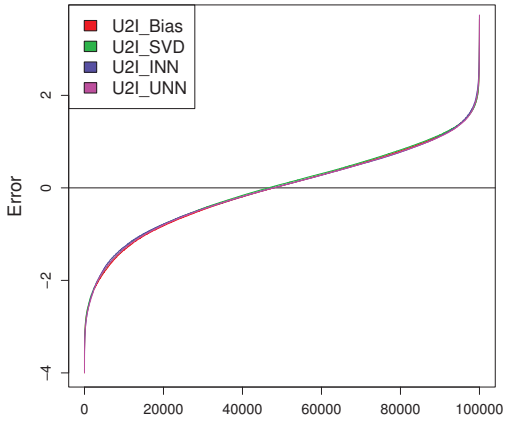
Figure 4.7: Error by Base Model



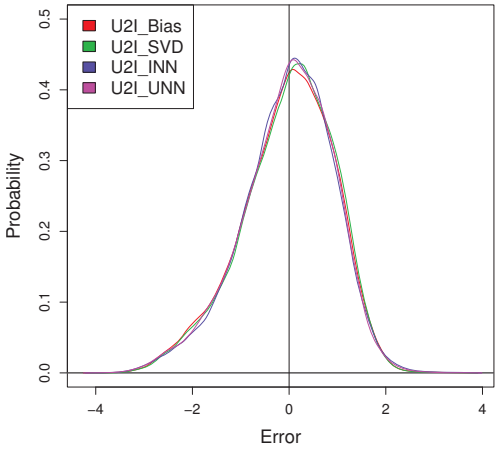
(a) Item Error Distribution



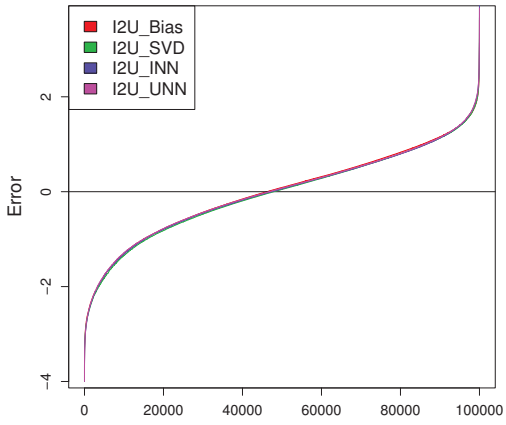
(b) Item Error Density



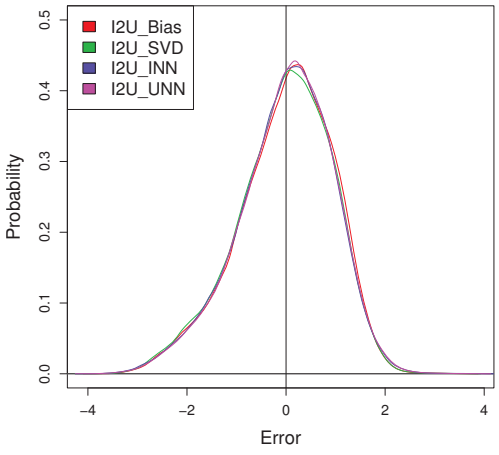
(c) U2I Error Distribution



(d) U2I Error Density



(e) I2U Error Distribution



(f) I2U Error Density

Figure 4.8: Error by Base Model (cont.)

Chapter 5

Future Work and State of the Art

5.1 Potential Improvements

In our examination of the various models' performances, we noticed two interesting things. The first is that ITEMBIAS performed slightly better than USERBIAS, conversely USER_SVD performed better than ITEM_SVD. This suggests that user biases might be more complex in nature than item biases, because of the increased performance upon transitioning to a more complex model. Therefore it would be interesting to examine the performance of a model that represents user bias in alternate formulations or taking into account other information. For example, the user bias can be estimated with respect to a particular genre, rather than a general user bias. Also, Paterek used a different approach in

his paper; instead of directly calculated bias, he learned values for the bias vector using a gradient descent method in parallel with the PQ decomposition [7].

Other areas for potential improvements in performance would be to start leveraging some of the additional data included within the MovieLens 100k dataset. This report examined models that only used the ratings data, but the dataset includes demographic data on the users, which could serve as another type of similarity measurement, along with genre and other miscellaneous data on the items (movies). We already mentioned the potential of user-genre biases. A further refinement of that approach would be to transform the boolean membership in genre, provided in the dataset, with a potentially more accurate partial membership using a fuzzy classifier or other similar methods. Alternatively, we could construct fuzzy user models to relate their interest in each genre [13].

5.2 State of the Art

5.2.1 Hybrid Models

Today many recommendation systems make use of hybrid models. Models that are neither purely content-based, nor purely collaborative-based. Instead these models seek to harness the advantages associated with each type while also compensating for their weaknesses. An example is that use of content-based model information can be used to compute item

similarity thus mitigating the Cold Start issue for new items, though not for new users [14]. The hybrid approach also allows the use of more sources of data enabling deeper analyses and hopefully more accurate predictions.

5.2.2 Ensemble Methods

Toward the end of the Netflix competition [4], performance had ceased to be improved through new types of models. Instead, teams were drawing from new techniques, collectively known as ensemble methods, to enhance performance by leveraging multiple models constructed using different techniques or parameter selection. The nature of using these diverse models allowed these systems to leverage that diversity to make better predictions. Each model could focus on a particular facet without loss of accuracy due to the large number of models in the system. The final winning team, *BellKor's Pragmatic Chaos* mention the use of tens of predictors comprised of hundreds of sub-models each [15, 16, 17].

5.2.3 Context-Aware Systems

Some areas of application, especially music, have begun exploring use of context-aware recommendations systems that use contextual information like weather, location, or time

as additional input to assist in finding music relevant to the listener [18, 19, 20]. Three different approaches are being used to integrate this information into the system: (i) pre-filtering, the set of items is filtered using the contextual information before reaching a more traditional model; (ii) post-filtering, output from a more traditional model is filtered based on the context before reaching the user; and (iii) contextual modeling, the model itself uses the contextual information together with user and item data to generate recommendations [21].

5.3 Additional Uses

With the growing maturity of recommendation systems, many new uses are being found outside of the traditional or commonly-known applications. Recommendation systems have been suggested for use in requirements engineering to assist in identifying stakeholders and eliciting desired features [22], finding a dentist [23], and improving product line development and configuration [24].

5.4 Conclusions

This report presented an overview of recommendation systems. Specifically, collaborative recommendation systems were discussed and implemented. An empirical evaluation was

performed to examine the performance of the models on the MovieLens dataset. Overall, basic statistical methods that remove the user and item bias do quite well. More complex models perform better (lower rating errors), with the best models using a collaborative filtering nearest neighbor approach.

References

[1] Ekstrand, M. D.; Riedl, J. T.; Konstan, J. A. *Foundations and Trends® in Human–Computer Interaction* **2011**, *4*(2), 81–173.

[2] Burke, R. *Encyclopedia of library and information systems* **2000**, *69*(Supplement 32), 175–186.

[3] Jannach, D.; Zanker, M.; Felfernig, A.; Friedrich, G. *Recommender systems: an introduction;* Cambridge University Press, 2010.

[4] Netflix Prize. `http://www.netflixprize.com`.

[5] Bennett, J.; Lanning, S. In *Proceedings of KDD cup and workshop*, Vol. 2007, page 35, 2007.

[6] Funk, S. *URL http://sifter. org/˜ simon/journal/20061211. html* **2006**.

[7] Paterek, A. *Proceedings of KDD cup and workshop* **2007**, *2007*, 5–8.

[8] Koren, Y.; Bell, R.; Volinsky, C. *Computer* **2009**, *42*(8), 30–37.

[9] Bell, R. M.; Koren, Y.; Volinsky, C. **2007**.

[10] Herlocker, J. L.; Konstan, J. A.; Borchers, A.; Riedl, J. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[11] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[12] Movielens 100k. Project, G. R. http://files.grouplens.org/datasets/movielens/ml-100k.zip.

[13] Al-Shamri, M. Y. H.; Bharadwaj, K. K. *Expert systems with applications* **2008**, *35*(3), 1386–1399.

[14] Grivolla, J.; Campo, D.; Sonsona, M.; Pulido, J.-M.; Badia, T. In *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*, Vol. 1, pages 297–301. IEEE, 2014.

[15] Töscher, A.; Jahrer, M.; Bell, R. M. *Netflix prize documentation* **2009**.

[16] Koren, Y. *Netflix prize documentation* **2009**, *81*.

[17] Piotte, M.; Chabbert, M. *Netflix prize documentation* **2009**.

[18] Wang, M.; Kawamura, T.; Sei, Y.; Nakagawa, H.; Tahara, Y.; Ohsuga, A. In *Semantic Technology;* Springer, 2014; pages 17–32.

[19] Ricci, F. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 865–866. ACM, 2012.

[20] Beach, A.; Gartrell, M.; Xing, X.; Han, R.; Lv, Q.; Mishra, S.; Seada, K. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 60–65. ACM, 2010.

[21] Panniello, U.; Tuzhilin, A.; Gorgoglione, M. *User Modeling and User-Adapted Interaction* **2014**, *24*(1-2), 35–65.

[22] Hariri, N.; Castro-Herrera, C.; Cleland-Huang, J.; Mobasher, B. In *Recommendation Systems in Software Engineering;* Springer, 2014; pages 455–476.

[23] Pradhan, S.; Gay, V. In *Trust Management VIII;* Springer, 2014; pages 221–228.

[24] Mazo, R.; Dumitrescu, C.; Salinesi, C.; Diaz, D. In *Recommendation Systems in Software Engineering;* Springer, 2014; pages 511–537.

[25] Konstan, J. A. *ACM Transactions on Information Systems (TOIS)* **2004**, *22*(1), 1–4.

[26] Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; Riedl, J. T. *ACM Transactions on Information Systems (TOIS)* **2004**, *22*(1), 5–53.

[27] Gower, S. **2014**.

[28] Kurucz, M.; Benczúr, A. A.; Csalogány, K. In *Proceedings of KDD Cup and Workshop*, Vol. 12, pages 31–38. Citeseer, 2007.

Appendix A

Code

A tarfile containing the code used to generate and evaluate the models in this report is available at http://www.cs.mtu.edu/ jcstombe/masters/mastersReportCode.tgz.

Appendix B

SVD Parameter Selection

A tarfile containing the results for the 3-fold validation of SVD parameter selection for each of the 4 folds is available at http://www.cs.mtu.edu/ jcstombe/masters/svdParameterSelection.tgz.