DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2016

Evaluating Prediction Accuracy for Collaborative Filtering Algorithms in Recommender Systems

SAFIR NAJAFI

ZIAD SALAM

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

KTH Computer Science
and Communication

Evaluating Prediction Accuracy for Collaborative Filtering Algorithms in Recommender Systems

SAFIR NAJAFI
ZIAD SALAM

Degree Project in Computer Science, DD143X
Supervisor: Atsuto Maki
Examiner: Örjan Ekeberg

CSC, KTH, 2016-05-11

Abstract

Recommender systems are a relatively new technology that is commonly used by e-commerce websites and streaming services among others, to predict user opinion about products. This report studies two specific recommender algorithms, namely FunkSVD, a matrix factorization algorithm and Item-based collaborative filtering, which utilizes item similarity. This study aims to compare the prediction accuracy of the algorithms when ran on a small and a large dataset. By performing cross-validation on the algorithms, this paper seeks to obtain data that supposedly may clarify ambiguities regarding the accuracy of the algorithms. The tests yielded results which indicated that the FunkSVD algorithm may be more accurate than the Item-based collaborative filtering algorithm, but further research is required to come to a concrete conclusion.

Referat

Bedömning av Förutsägelsers Precision för Kollaborativa Filtreringsalgoritmer i Rekommendationssystem

Rekommendationssystem är en relativt ny teknologi som ofta förekommer på bland annat e-handelsidor och streamingtjänster för att förutse användares åsikter om produkter. Denna rapport studerar två specifika rekommendationsalgoritmer, nämligen FunkSVD, en algoritm som bygger på matrisfaktorisering, samt Item-based Collaborative Filtering, som använder sig av likheten mellan olika element. Denna studie har som mål att undersöka hur träffsäkra algoritmernas förutsägelser blir mellan en mindre och en större datakollektion. Genom att använda korsvalidering på algoritmerna för de olika datakollektionerna försöker rapporten ta fram data som kan klargöra tvetydigheter kring algoritmernas prestanda. Testerna visade på resultat som indikerar att FunkSVD-algoritmen verkar prestera bättre än Item-based Collaborative Filtering-algoritmen, men mer forskning behöver bedrivas för att dra en konkret slutsats.

Contents

Chapter 1
Introduction

Recommender systems are an evolving technology often based on a learning system providing predictions to users of a web service or application, based on user behavior. The research on recommender systems is a consequence of the business possibilities, allowing for increased revenue, consumer-satisfaction, as well as user-friendliness [13]. Different approaches have been presented and studied in order to improve the performance and accuracy of recommender systems [13, 16, 22].

Improvement is still a matter of importance for developers of recommender systems. Lacking performance, accuracy and other problems are surfacing as services are frequently expanding in terms of supply range and customer base [22]. A recommender system might have trouble predicting user-preferred items for other reasons as well, often depending on the implementation strategy [15]. Many different approaches have been presented concerning several issues, but the prevention of one problem often increases the impact of others [22].

One important aspect of recommender systems is the requirement for precise recommendations. For example, one group of recommender systems may not require accurate results to present relevant data of some sort, while others might be in need of an algorithm which can produce results in between a narrow interval. Certain algorithms may be better suited for heavy constraints on precision and validity while others do better in contexts where low-cost computations are a more important feature.

Recommender systems operate on sets of data in order to provide recommendations. Commonly, datasets start out small and grow over time as more data related to products are obtained or new products added. An algorithm working efficiently and accurately for a particular dataset may perform differently on other datasets [3]. In order to determine the optimal algorithm for a specific dataset, tests can be conducted with different metrics to yield an approximation of an algorithm's performance. This in turn would provide guidelines for optimal algorithms depending on dataset properties and sizes.

1.1 Related Work

Different approaches and techniques have been proposed for predicting user behavior. A really interesting paper concerning the Item-based collaborative filtering algorithm has been published by Sarwar et al. [20]. The paper describes the item-based approach thoroughly and discusses optimal parameters and compares Item-based collaborative filtering to User-based collaborative filtering using the Movielens 100k dataset. Another interesting algorithm has been proposed by Simon Funk [10], namely FunkSVD, a matrix factorization algorithm. An experiment will be conducted in order to measure item-based collaborative filtering and FunkSVD thoroughly by comparing the algorithms' performances with each other on two different datasets. The objective is to measure which algorithm is more scalable.

1.2 Problem Statement

This report will address the issue of offline computation accuracy and how certain algorithm predictions scale based on the size of a dataset. The aim of this report is to acquire an understanding of when a given algorithm is the most effective for a growing dataset. Naturally, the result will not yield a definite answer as to when to use which algorithm, since there are external factors such as the structure of datasets and what kind of predictions are preferred. Hopefully this report will provide some guidelines to which algorithm is the most suitable for a dataset with specific properties and size.

Question: Which of the algorithms Item-based collaborative filtering and FunkSVD is optimal for predicting ratings for the Movielens 100k dataset? Does the result differ for the Movielens 1M dataset?

Chapter 2
Background

This section will outline the concept of recommender systems, describe several different approaches and the advantages and disadvantages of various implementations. The scope of this report lies in collaborative filtering and matrix factorization, but this section will brief the reader about related subjects necessary to make use of the results and analysis later in this report.

2.1 Recommender Systems

A recommender system's main goal is to estimate ratings for items and provide a personalized list of the recommended items to a given user [1, 27]. There are many different approaches to this task, such as the comparison of ratings, item or user attributes and demographic relations.[1, 15].

Every basic recommender system contain some type of background information, consisting of a set of users, a set of items and the relation between those which in many cases is the rating a user gives to an item [1, 15]. The rating of an item is often represented as an integer, for example a scale from one to five. [15].

These relations can for instance be described with bipartite graphs [15] and matrices [14]. A common tool of recommender algorithms are user profiles [17] and item attributes [15]. User profiles may contain certain characteristics, for example gender, age and occupation [28]. Attributes of an item depends on their domain. For instance, movies might have a genre, an actor list, a specific producer or other relevant attributes that link items together [17].
Recommender systems filtering can be categorized into three main approaches.

1. **Content-based filtering:** Recommends items by matching attributes with other items that a given user have rated [14, 15]. Further explained in section 2.2.

2. **Collaborative filtering:** Recommends items by comparing a given user with a set of users that have rated other items similarly [15]. Collaborative filtering systems have two main approaches for filtering, namely memory-based and model-based [19]. Further explained in section 2.3.

3. **Hybrid filtering:** Recommends items by combining different type of approaches together [1]. Further explained in section 2.4.

2.2 The Content-based Approach

In content-based recommender systems, a recommendation is based on the relation between the attributes of items that a given user has previously rated and items which the user have not yet rated [14]. The content-based approach utilizes the concept of monotonic personal interest, meaning a person's interest will most likely not change in the near future [19].

The process of how a basic content-based approach recommends an item involves having a set of items and a set of users [19]. Assuming every user have rated a subset of the items, a user profile can be constructed for each user containing their preferences [14, 19]. User preferences are computed by analyzing the similarities between the items that the user has previously rated [14, 19]. Recommending an item when a user profile is available is done by matching the profile attributes with items not yet rated by the user [14]. However lacking information in user profiles and items leads to less accurate results. The case of insufficient data to build a reliable model is called the start up problem [6]. Thus, the content-based approach is heavily reliant on the available information. On the contrary, collaborative filtering requires user-item rating relations.

2.3 Collaborative Filtering Approach

Collaborative filtering system recommends items to users by comparing ratings [19, 23, 27]. Opinions of real-life related users play a role in personal decision making [19]. A collaborative system is based on the same principle, but instead of considering geographically adjacent user opinions, the system searches globally for similar users [23]. However, basing recommendation on all users may not result in accurate predictions. In collaborative filtering, the issue is addressed by matching users that share rating similarities [19]. As opposed to content-based filtering, collaborative filtering provides recommendations by examining user-item rating relations [25].

Table 2.1. User-item rating relation. Rating is represented in a scale 1-5 where 5 is best and 1 is worst.

Rating Matrix	**The Avengers**	**The Revenant**	**The Martian**	**Deadpool**
Fred	2	4	5	1
Sara	?	5	?	2
John	5	2	2	4
Jessica	?	1	?	5

Table 2.1 is an example of how a collaborative filtering system may recommend an item using rating information from different users [19]. The user-item rating relation can be viewed as a matrix [23], as shown in the table. Each row in table 2.1 represents a user and each column a movie, the ratings are in scalar form and ranges from 1 to 5.

Given a user to provide a recommendation for, the system identifies other users based on the rating information by examining patterns in their rating behaviors [19]. The potential items to be recommended to the user will therefore be dependent on previous user ratings. For instance analyzing Sara in table 2.1 will most likely result in the system predicting a high rating for The Martian and a low rating for The Avengers to Sara. This is because Sara's rating pattern is more similar to Fred's than it is to John's and Jessica's. However, a larger quantity of users would be taken into consideration in a real system, possibly yielding a more accurate prediction [25].

One disadvantage present to collaborative algorithms is the requirement that systems have multiple users rating several items in order to produce accurate results [23].

2.4 Hybrid Approaches

In the attempt to provide more accurate recommendations, several hybrid approaches has been proposed [9]. A hybrid algorithm combines several approaches in order to provide more accurate results. The structure of such an implementation differ depending on the hybrid approach strategy. More specifically, a hybrid approach incorporates evaluation features from the content-based approach and the collaborative filtering approach [1]. The structural idea behind hybrid approaches may vary, descriptions of common ideas are outlined below:

- A hybrid system can predict recommendations from a dataset using both a content-based approach and a collaborative approach separately [1]. When the predictions are presented, a specific metric may be used to determine the quality of the recommendations in order to choose result set. Alternatively, the results can be combined in order to produce the highest quality results from both the approaches.

- A hybrid system may predict recommendations from a dataset using a collaborative approach incorporating some of the characteristics of a content based approach [1].
- A hybrid system may predict recommendations from a dataset using a content-based approach incorporating some of the characteristics of a collaborative approach [1].
- A unified approach, combining features from both the content-based and collaborative approaches, resulting in an algorithm with a unique implementation [1].

2.5 Memory-based Algorithms

Memory-based algorithms use statistical techniques to find sets of users, often called neighborhoods, that are users similar to the target user [20]. The entire dataset or a subset is used and every user is in a neighborhood of users that are similar to each other. A neighborhood-based collaborative filtering algorithm performs prediction by first calculating the similarity or weight. The similarity is computed by observing the distance correlation between users and items. A prediction to a user is produced by taking the weighted sum of all the ratings for that item on another item, for example [25]. Algorithms based on memory have been widely used in commercial purposes such as on the Amazon.com website [25].

However there are several limitations. An algorithm tries to find a set of users by evaluating similarities between items rated by users. Therefore the dataset must not be sparse or consist of a low number of items [25]. These type of limitations causes unreliable predictions. To get better and more reliable recommendations, model-based collaborative filtering can be utilized.

2.6 Model-based Algorithms

Model-based algorithms differ from memory-based by utilizing machine learning, data mining algorithms or other algorithms to find patterns and predict ratings by learning. Some approaches for model-based algorithms are clustering models, Bayesian models and dependency models [25].

2.7 Problem Domains

Today, recommender systems face a variety of problems, including implementation [22], user behavior [26] and performance issues[22]. In addition, the properties of datasets may also introduce difficulties [22].

2.7.1 Implementation Problems

Recommender systems are facing certain challenges, algorithms often have their independent disadvantages depending on their individual implementation. While some approaches are time-efficient and produce accurate predictions, they may have corner-case problems which produce exceptionally bad predictions [1]. Another example could be recommender systems requiring to recompute the offline training model often in order to provide accurate predictions, which may be a straining procedure, taking a lot of time and computational power. Implementation problems can generally be solved by incorporating other methods into an existing algorithm, creating new algorithms based on various principles, or gathering more data in order for predictions to become more precise [1]. Many of these solutions often affect another problem domain. For example the cold-start problem [4], where recently introduced items in a system are seldom recommended. The reason is lacking data associated with recently added items. There might be several solutions to such a problem. However, trying to solve the problem may result in unintended side-effects.

2.7.2 Performance Problems

Growing databases containing item and user information may present performance issues. Too vast systems contain a lot of data, which in turn demands a lot of computation to be made in order to present predictions to users. Services consisting of millions of items and users benefit from efficiently scaling recommender algorithms, otherwise the computation of recommendations may be too slow [22].

2.7.3 User Behavior Problems

How a user behaves when using a service is crucial to making accurate predictions [26]. The problem of user behavioral issues are possibly one of the hardest to address since a large part of the problem is due to human behavior in general. Users who do not contribute to the database by rating items are one of the problems [12], another one may be users that tend to exclusively rate items which they have a strong opinion about [2]. In the latter case, computation of user rating might prove inaccurate, because the value input of the computation does not truly reflect the general user opinion, but merely items which some users had a particular interest in submitting a rating for [2].

2.7.4 Dataset Problems

Datasets used to evaluate algorithms have a major impact on the results. When evaluating an algorithm, the best practice is to do it on datasets with similar domain features [12]. Dataset properties can have substantial effect on algorithms. These properties are commonly [12]:

- Density of the dataset. How many cells of the item-user matrix that is populated with data.
- Ratio between users and items.

Density is important since users only rate a small subset of all the items [2]. Datasets using implicit rating are more likely to be less sparse because no effort is required from the users [12]. Evaluating an algorithm on a sparse dataset can cause less accuracy, because it lacks rating data to find common items between users [2]. The relation between users and items may also affect algorithm performance. Algorithms that find similarities between users or items may achieve an increase in performance[11]. For example evaluating an item-based algorithm in a dataset with more users than items may lead to better results [8].

2.8 Evaluation

Recommender algorithm performance can be measured by testing against different datasets [20]. In this paper, cross-validation is used in order to test algorithm prediction performance which is further explained in subsection 2.8.4. Algorithm prediction performance is measured with different metrics. Which metric to use depends on the purpose of the algorithm and the goal of the measurement [12, 25]. The metrics relevant to this paper will be outlined below.

2.8.1 Mean Absolute Error (MAE)

Mean absolute error is a metric used to compute the average of all the the absolute value differences between the true and the predicted rating [5]. The lower the MAE the better the accuracy. In general MAE can range from 0 to infinity, where infinity is the maximum error depending on the rating scale of the measured application. Computing the Mean Absolute Error is accomplished with the following formula [12]:

$$\text{MAE} = \frac{1}{n} \Sigma_{i=1}^{n} |d_i| - |\hat{d}_i|$$

d_i is the actual rating
$\hat{d}_i$ is the predicted rating
n is the amount of ratings

2.8.2 Root Mean Square Error (RMSE)

Root mean square error computes the mean value of all the differences squared between the true and the predicted ratings and then proceeds to calculate the square root out of the result [5]. As a consequence, large errors may dramatically affect the RMSE rating, rendering the RMSE metric most valuable when significantly large errors are unwanted. The root mean square error between the true ratings and predicted ratings is given by [12]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(d_i - \hat{d_i})^2}$$

d_i is the actual rating
$\hat{d_i}$ is the predicted rating
n is the amount of ratings

2.8.3 Offline Experiments

In prediction analysis, offline experiments may be conducted on a dataset in order to decide optimal algorithms to predict certain information. The offline experiment assumes the user behavior from the time the data was collected still is accurate. Otherwise, such an experiment is more or less meaningless for a live application. In our case, the analysis rests solely on determining the general behavior of certain algorithms. Therefore we do not have to take into account the dataset's validity. An offline experiment may include predicting similarities between items, calculating neighborhood clusters, among other things [12, 21, 24].

2.8.4 Cross-validation

Cross-validation, sometimes referred to as rotation estimation, is a validation methodology for analyzing statistical data. One goal of cross-validation is to pair an algorithm and a dataset in order to build a model and determine the generalizability of the algorithm. The analysis of an algorithm can be compared to the analysis of another algorithm, thus cross-validation can be used to measure algorithm performance [18].

Cross-validation splits a dataset into z equally large partitions, one of the partitions being the test partition while the z - 1 rest partitions are the training partitions. The algorithms then trains a model with the training partitions and when the training is complete, the model is tested with the test partition, producing test data. This procedure continues until every partition has been the test partition [18].

Chapter 3
Algorithms

3.1 Item-based Collaborative Filtering

The Item-based collaborative filtering approach compares users and their ratings to a certain set of items. The algorithm calculates the nearest neighbors to a given user. Every neighbor's previously rated items are compared to the item in focus with an item similarity computation. When the most similar items have been determined, the prediction is computed by taking the weighted average of every neighbor's ratings of those items. The calculation of item similarities can be done with several methods including cosine-based similarity, correlation-based similarity and adjusted cosine similarity [20, 25].

3.1.1 Cosine-based Similarity

Cosine-based similarity is a method for computing similarities between items in Item-based based collaborative filtering. In cosine-based similarity an item is represented as a vector in the user-space. The angle between two of these vectors are computed and the cosine of the angle is the similarity of the items. For example two items m and n are computed as follows [20]:

$$similarities(m,n) = cosine(\bar{m},\bar{n}) = \frac{\bar{m}*\bar{n}}{\|\bar{m}\|_2 * \|\bar{n}\|_2}$$

Where the "*" represents the dot product of the two vectors.

The cosine-based similarity computation has one drawback. User rating scale is not taken into consideration during the computation. The calculations are performed over information from several columns, and each column represents a different user [20, 25]. The issue may be solved with adjusted cosine similarity which subtracts the corresponding user average rating from each co-rated (a user that has rated both item m and n) pair [20, 25]. The similarity for the two items m and n, using adjusted cosine similarity, are computed as follows [20]:

$$similarity(m,n) = \frac{\sum_u \in U\left(R_{u,m} - \bar{R_u}\right)\left(R_{u,n} - \bar{R_u}\right)}{\sqrt{\left(R_{u,m} - \bar{R_u}\right)^2}\sqrt{\left(R_{u,n} - \bar{R_u}\right)^2}}$$

$R_{u,m}$ is the rating of user u on item m.
$\bar{R_u}$ is the average rating of item m.

3.1.2 Correlation-based Similarity

Correlation-based similarity between two items m and n are calculated by computing the Pearson correlation [20]. The Pearson correlation for the item m and n rated by a set of users U is computed as follows [20, 25]:

$$similarity(m,n) = \frac{\sum_u \in U\left(R_{u,m} - \bar{R_m}\right)\left(R_{u,n} - \bar{R_n}\right)}{\sqrt{\left(R_{u,m} - \bar{R_m}\right)^2}\sqrt{\left(R_{u,n} - \bar{R_n}\right)^2}}$$

Where $\bar{R_u}$ is the average rating of user u.

3.1.3 Computing Prediction

Computing predictions can be achieved by several techniques. A simple method is computing the weighted sum of an item m for user u by taking the sum of all the rated items by user u that are similar to m. The purpose of this method is to apprehend how a user rates items that are similar to the item that the method is trying to predict rating for. The weighted sum for predicting an item m for user u is computed as follows [20]:

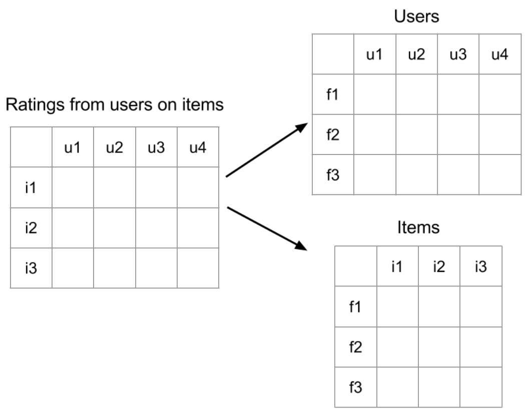
$$P_{u,m} = \frac{\sum_N (s_{m,k} * R_{u,k})}{\sum_N (|s_{m,k}|)}$$

Where $P_{u,m}$ is the prediction of item m for user u.
$R_{u,k}$ is the rating from user u on item k.
N is a set of similar items.
k is an element of N.
$s_{m,k}$ is the similarity value between two items.

3.2 FunkSVD

FunkSVD is an algorithm utilizing a matrix factorization method, namely Singular value decomposition. Singular value decomposition is used to reduce one matrix to two matrices with lower dimensions[1]. The two resulting matrices represent generalized items and users respectively. Predictions are made by computing the dot product of one or more item vectors and a user vector, yielding values which indicate the most suitable recommendations [10]. An example of how a Singular value decomposition reduces a matrix into two other matrices is displayed in figure 3.1. The amount of features are decided by an input variable to the factorization [10]. In the figure, u1 to u4 denotes the users, i1 to i3 denotes the items and f1 to f3 denotes the features. Features are usually called latent factors where Singular value decomposition is concerned.

Figure 3.1. Singular value decomposition in FunkSVD on a dataset



[1]In reality, singular value decomposition results in three matrices, but in the FunkSVD algorithm the third matrix is incorporated into the other matrices [10]

FunkSVD trains the model in order to achieve as accurate feature values as possible for items and users. When training is performed, the algorithm has a parameter which often is called iteration count. The iteration count decides how many times the training for a cell in the matrices is supposed to run. The iteration count may be a fixed amount of times to loop, or some other kind of condition which states when the looping is supposed to terminate [10].

A useful advantage to Singular value decomposition is that the lower ranked matrices often yield better estimates of the data than the original matrix [21]. The purpose of the matrix factorization is to generalize the data by extracting common user preferences, categorizing them to get an overview of the users. Due to the generalization, vast amounts of data can be represented as more intuitive information [10, 21].

Chapter 4
Experiment

We used the LensKit Recommender Toolkit 2.2.1, which is a recommender system evaluation framework [7], for our research. More detailed information about the purpose and usage of LensKit can be found at http://lenskit.org/. Our reasons for choosing LensKit Recommender Toolkit is the following:

- Relevant and interesting algorithms are already implemented and refers to valid scientific sources.
- Algorithm parameters are easily modifiable in order to alter algorithm performance.
- Intuitive and thorough documentation.

In order to represent our result data in graphs and tables we used MATLAB 8.4.0 and the R environment software because of the available types of graph plotting they contained.

We have conducted testing on the algorithms Item-based collaborative filtering and FunkSVD. Item-based collaborative filtering may be implemented in different ways, and the implementation we chose to use is the algorithm built into Lenskit and proposed by Sarwar et al. [20]. The reason being it appears to be a widely recognized implementation. FunkSVD is an algorithm implemented by Simon Funk, who placed third in the Netflix Prize. [10] Several algorithms have been implemented using matrix factorization, FunkSVD is one of them, but it has been studied and acknowledged.

4.1 Datasets

Testing has been conducted on two of the Movielens datasets provided by GroupLens[1]. The Movielens datasets contain user ratings for movies. The Movielens datasets are differently sized. Each dataset is independent from the other, the small dataset is not a subset of the bigger one [11]. The datasets used in this thesis are the 100k dataset released in 1988 and the 1M dataset released in 2003 [11]. Both datasets contain users that have rated at least 20 movies [11]. Parameter testing were conducted on the Movielens 100k dataset. It consists of 100.000 ratings from 943 users on 1682 movies with a density of 6.37% displayed in table 4.1. The parameters that gave best result on the Movielens 100k dataset were then used to evaluate the algorithms on the 1M dataset. The 1M dataset consists of one million ratings from 6040 users on 3706 movies with a density of 4.47% displayed in table 4.1. Both datasets are represented with a rating scale from 1 to 5 using a precision of one in the LensKit Tool Recommender.

The reason we chose these two specific datasets from Movielens is because they are widely used in multiple reports. They also use conventional ratings which is preferred when predicting ratings. Both datasets use the same type of rating scale and precision. Since the RMSE and MAE values are depended on the rating scale, the results will be more comparable. GroupLens provides multiple dataset sizes for research purposes but due to hardware performance issues this report is limited to evaluation on the 100k and the 1m dataset.

[1]http://grouplens.org

Table 4.1. Movielens dataset properties [11]

Name	Users	Movies	Rating-Scale	Ratings	Density
ML 100k	943	1682	1-5	100.000	6.30%
ML 1m	6040	3706	1-5	1.000.209	4.47%

4.2 Parameters

Algorithms in LensKit has certain default values when unmodified. In order to optimize the prediction accuracy, we labored with the parameters and methods. In order to do this, we gathered information from papers [20] and methodically investigated the prediction performance of them by running iterative tests. The improvements achieved by the adjustment are displayed in the results section, but the process of optimizing the algorithms are outlined below.

4.2.1 Item-based Collaborative Filtering Adjustment

From the paper published by Sarwar et al. [20] we discovered that neighborhood size can be an altering factor in the prediction accuracy of the Item-based collaborative filtering algorithm. Therefore, we wanted to investigate which neighborhood size would optimize the prediction results on our dataset and the LensKit implementation. We used the 100k dataset and cross-validated the algorithm with 10, 20, 30, .., 80 as neighborhood size input. The test results are displayed in the 5.1 results section.

The similarity computation of Item-based collaborative filtering may also affect prediction results, which was outlined in the Item-based collaborative filtering section in the background chapter of this paper. Therefore we conducted tests with the cosine-based similarity computation, adjusted cosine similarity computation and correlation-based similarity computation [20, 25]. The test results are displayed in the 5.1 results section.

4.2.2 FunkSVD Adjustment

The iteration count in FunkSVD is an important parameter which drastically may affect prediction accuracy. We performed some testing on the iteration count 60, 80, 100, .., 300. In parallel to the iteration count testing, we also conducted tests on the optimal amount of latent factors (feature vectors) in order to determine when the two parameters in combination performed the best. For every iteration count magnitude, we tested the latent factor count 10, 15, 20, .., 75. The results are displayed in the 5.2 results section.

4.3 The Scalability Experiment

When the algorithms have been optimized, the scalability may be measured. Both item-based collaborative filtering and FunkSVD will be cross-validated over the two datasets. The results will be analyzed, interpreted and compared with mean absolute error and root mean square error. Furthermore the change factor in the results from the 100k dataset to the 1M dataset will be computed in order to determine the scalability. The results are displayed in the 5.2 results section.

Chapter 5
Results

This chapter will outline the optimization and scalability experiment results by presenting the average values of the cross-validations.

5.1 Item-based Collaborative Filtering Optimization

Evaluation values for each of the three item similarity methods were tested with different neighborhood sizes. Each method was evaluated with both MAE and RMSE.

5.1.1 Cosine-based Similarity

Results from the running cosine-based similarity method with iterative neighborhood sizes are presented in figure 5.1. Both evaluation metrics indicated similar growth in rating accuracy when the neighborhood size is incremented. The lowest values for RMSE and MAE were obtained with different neighborhood sizes. RMSE indicates that neighborhood size 20 yields the best accuracy while MAE indicates that neighborhood size 10 yields the best accuracy.

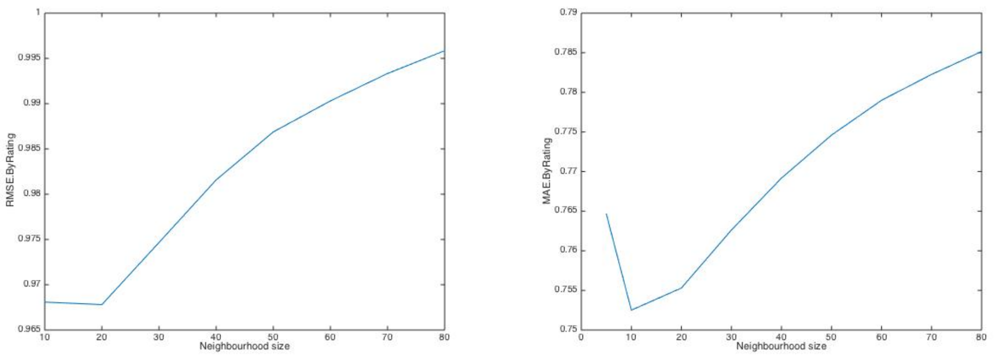


Figure 5.1. Cosine-based similarity. Iterating through different neighborhood size to find best possible accuracy.

5.1.2 Adjusted Cosine Similarity

Results from running adjusted cosine with different neighborhood sizes are presented in figure 5.2. The lowest value for adjusted cosine was achieved when neighborhood size were twenty with both RMSE and MAE. Both graphs grows similarly when reaching their minimum point.

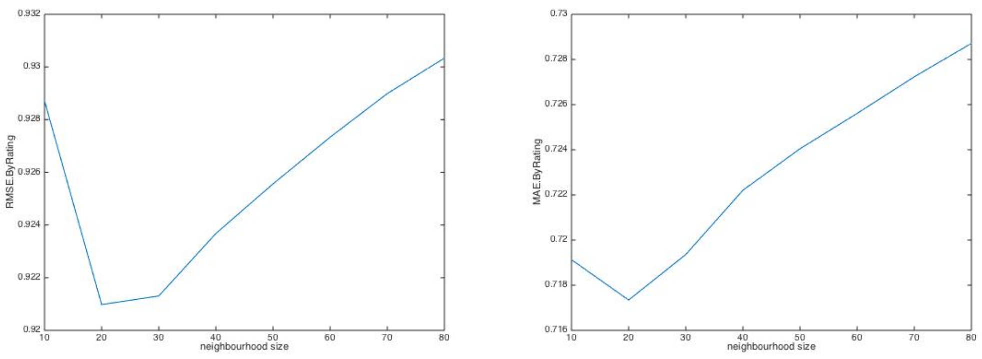


Figure 5.2. Adjusted-cosine item similarity. Iterating through different neighborhood size to find best possible accuracy.

5.1.3 Correlation-based Similarity

Results from running Pearson correlation with different neighborhood sizes are presented in figure 5.3. Both graphs resembles similar accuracy throughout the neighborhood sizes. The minimum value is computed for both RMSE and MAE when neighborhood size is 20.

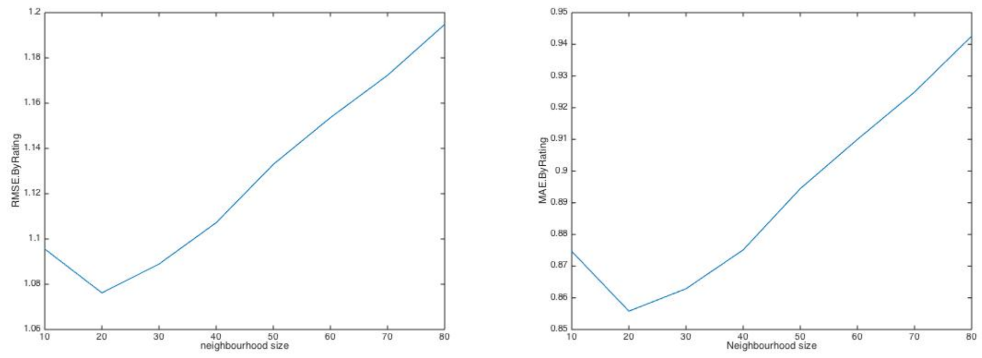


Figure 5.3. Pearson correlation similarity. Iterating through different neighborhood size to find best possible accuracy.

Table 5.1. The lowest values for each of the item similarity computations

	Cosine		Adjusted Cosine		Pearson Correlation	
Neighborhood size	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
10	-	0.752503	-	-	-	-
20	1.076190	-	0.920979	0.717344	1.076190	0.855794

5.1.4 Optimized Parameters

The lowest RMSE and MAE values obtained from each of the different parameter combinations are presented in table 5.1. The table shows that the MAE and RMSE values are at a minimum when adjusted cosine similarity with neighborhood size twenty.

5.2 FunkSVD Optimization

The lowest score for each iteration count and the corresponding latent factor is shown in figure 5.4. Latent factor 50 and iteration count 200 yielded the lowest RMSE and MAE scores. A summary of the data is presented in appendix B.

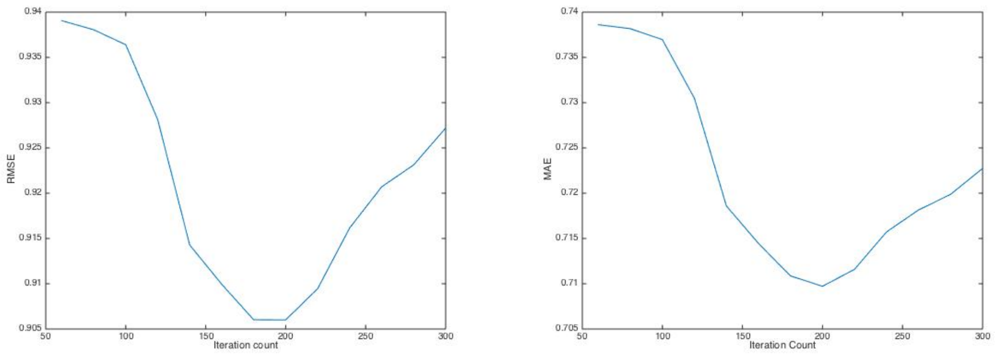


Figure 5.4. The lowest score for each iteration and corresponding latent factor

5.3 Evaluation Results

The results from section 5.1 revealed which parameter combination that yields the MAE and RMSE values. Both algorithms were evaluated on the 100k dataset as well as the 1M dataset. The distribution for each of the RMSE and MAE values after conducting cross-fold evaluation are presented as boxplots in appendix A. The average evaluation values for each metric are presented in table 5.2. The lowest values were obtained while cross-validating FunkSVD over both datasets. The change factor from the 100k dataset to the 1M dataset are presented in table 5.3.

Table 5.2. Evaluations on optimized algorithms with 100k and 1m dataset.

	100k		**1m**	
Algorithm	**RMSE**	**MAE**	**RMSE**	**MAE**
FunkSVD	0.905520	0.709531	0.863767	0.672917
Item-based	0.920979	0.717344	0.885014	0.689355

Table 5.3. Improvement from 100k to 1m.

	Improvement	
Algorithm	**RMSE**	**MAE**
FunkSVD	4.83%	5.44%
Item-based	4.06%	4.06%

Chapter 6
Discussion

The results indicate that the FunkSVD algorithm scales slightly better than the Item-based collaborative filtering algorithm when comparing the performance with regard to the datasets. However, the actual reason for the difference is unclear, it could mean FunkSVD is superior to Item-based collaborative filtering when datasets grow larger. On the contrary, the difference in performance could be related to the properties of the datasets. The varying densities of the datasets could be a major factor, the relation between the amount of users and items another, for example.

Assuming the algorithms work equally well no matter the dataset properties, we will now outline what we can deduce from the test results. The assumption would indicate that the FunkSVD algorithm always predicts the ratings more accurately for larger datasets than the Item-based collaborative filtering algorithm. Further evaluation on larger datasets is required in order verify the assumption. We may deduce that the improvement in performance for FunkSVD is higher both with respect to root mean square error and mean absolute error when comparing the test results.

One matter to consider is what impact the prediction accuracy has on the actual recommendation. The predicted ratings of the FunkSVD algorithm are indeed closer to the real ratings than the Item-based collaborative filtering algorithm, but it does not necessarily propose the outcome of a recommendation to a user will be more accurate. The tests does not give any clue as to which ratings have been most severely miscalculated, which makes the results hard to interpret. For example in the 100k dataset cross-validation, the FunkSVD algorithm has a RMSE of 0.90 and the Item-based collaborative filtering algorithm a RMSE of 0.92. The difference is significant, but it is hard to tell which of the predicted ratings were evaluated differently between the algorithms. Suppose a recommendation is needed for a user and both algorithms yield a set of suggestions. The suggestions could turn out to be equally relevant, because it is unknown which ratings are causing the most errors. If the low ratings distinguishes the algorithm errors, it is likely the two algorithms will perform equally well in providing the user with a relevant recommendation. On the other hand, if the high ratings distinguishes the algorithm errors, the algorithm with the lowest error will likely provide more relevant recommendations. Thus, the results should be considered as rough estimations, rather than definitive answers to what is truly optimal for an actual recommender system.

Central to understanding the results is to take into consideration what role the density of a dataset has. The results show that the FunkSVD algorithm scales better than the Item-based collaborative filtering algorithm when moving from a smaller dataset with higher density to a larger dataset with lower density. In order to determine whether the algorithms scale equally during circumstances where the dataset properties does not change, further testing is required.

Similarly the relation between users and items may well affect the result. As earlier mentioned, the Item-based approach tends to perform differently depending on the user-item ratio in a dataset. In effect, the scaling of the Item-based collaborative filtering algorithm may be faulty, because the smaller dataset contains more items than users, and the larger dataset contains more users than items. One might suspect the Item-based collaborative filtering algorithm would have done even worse if datasets with similar user-item ratio would have been used for the testing.

Another important matter to discuss is optimization. Our study does not suggest which of the algorithms is actually the most effective on the larger dataset when optimized for it specifically. However, for the purpose of this report, we do not believe such an experiment to be relevant. The goal is to provide a better perspective of how the algorithms in general perform when a dataset grows. To find the optimized algorithm for any given dataset, it is probably most accurate to test algorithms directly against the dataset. Hence, we do not bother to optimize the algorithms for the 1M dataset, since we strictly wanted to compare the same specifications on two datasets with different sizes. However, further testing can be conducted to analyze how the algorithms should be altered in order to perform optimally on datasets with varying properties.

Chapter 7
Conclusion

The prediction accuracy of a recommender system is heavily dependent on various factors. The properties of a dataset plays a major role as well as the parameters of the algorithms. What can be seen is that when both algorithms are optimized for the 100k dataset, the FunkSVD algorithm performs better on the larger, less dense dataset than the Item-based collaborative filtering algorithm. To elicit more rigorous results, testing algorithms on datasets with more similar properties to each other need to be performed. The fact that the Item-based collaborative filtering algorithm was given a convenient opportunity to scale well due to the dataset properties and still got outperformed by FunkSVD is a strong indicator that the FunkSVD algorithm is more versatile. We can conclude that the FunkSVD algorithm is seemingly more accurate than the Item-based collaborative filtering algorithm when it comes to scaling from small to large datasets.

Acknowledgements

We would sincerely like to thank Atsuto Maki for supervising this project, greatly improving the style and structure of the report as well as assisting with ideas and thoughts.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[2] G. Adomavicius and J. Zhang. Impact of data characteristics on recommender systems performance. *ACM Trans. Manage. Inf. Syst.*, 3(1):3:1–3:17, Apr. 2012.

[3] I. C. G. Alejandro BellogÃn Kouki, Pablo Castells Azpilicueta. Recommender system performance evaluation and prediction: An information retrieval perspective. 2012.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[5] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? - arguments against avoiding rmse in the literature. 2014.

[6] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[7] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM.

[8] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[9] A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, S. Reiterer, and M. Stettinger. *Recommendation Systems in Software Engineering*, chapter Basic Approaches in Recommendation Systems, pages 15–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[10] S. Funk. Netflix update: Try this at home, 2006.

[11] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.

[12] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[13] T. Iwata, K. Saito, and T. Yamada. Modeling user behavior in recommender systems based on maximum entropy. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1281–1282, New York, NY, USA, 2007. ACM.

[14] P. Lops, M. Gemmis, and G. Semeraro. *Recommender Systems Handbook*, chapter Content-based Recommender Systems: State of the Art and Trends, pages 73–105. Springer US, Boston, MA, 2011.

[15] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 519(1):1 – 49, 2012. Recommender Systems.

[16] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke. A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science*, 17(4):583–604, feb 2011.

[17] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[18] P. Refaeilzadeh, L. Tang, and H. Liu. *Encyclopedia of Database Systems*, chapter Cross-Validation, pages 532–538. Springer US, Boston, MA, 2009.

[19] M. Robillard, R. Walker, and T. Zimmermann. Recommendation systems for software engineering. *IEEE Software*, 27(4):80–86, July 2010.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.

[22] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering, 2002.

[23] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. *The Adaptive Web: Methods and Strategies of Web Personalization*, chapter Collaborative Filtering Recommender Systems, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[24] G. Shani and A. Gunawardana. Evaluating recommender systems. Technical Report MSR-TR-2009-159, November 2009.

[25] X. Su and M. T. Khoshgoftaar. A survey of collaborative filtering techniques, advances in artificial intelligence. 2009.

[26] A. Tuzhilin and G. Adomavicius. Integrating user behavior and collaborative methods in recommender systems. In *CHI 99 Workshop. Interacting with Recommender Systems*, 1999.

[27] X. Yang, Y. Guo, Y. Liu, and H. Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41:1 – 10, 2014.

[28] T. Yuan, J. Cheng, X. Zhang, S. Qiu, and H. Lu. Recommendation by mining multiple user behaviors with group sparsity. 2014.

Appendix A
Evaluation Data

Results from evaluating the optimized algorithms on the 100k and 1m dataset. The plots show how the values are distributed before calculating the average performance.

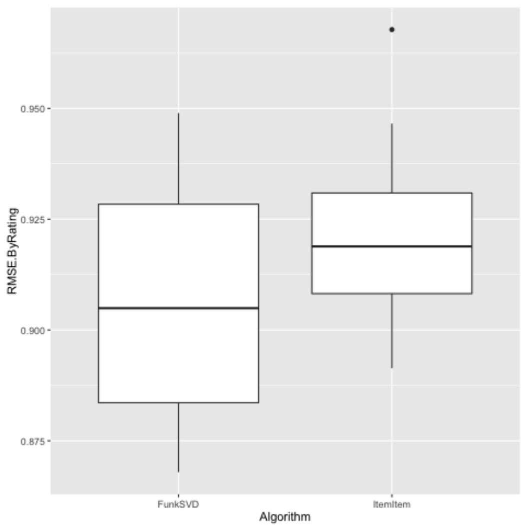


Figure A.1. The RMSE values from each cross-fold evaluation on the 100k dataset

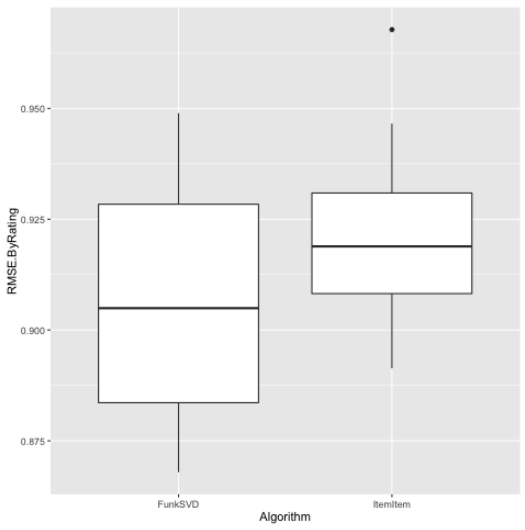


Figure A.2. The RMSE values from each cross-fold evaluation on the 1m dataset

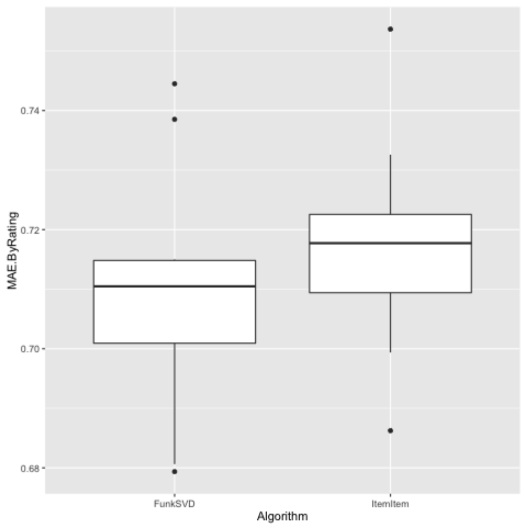


Figure A.3. The MAE values from each cross-fold evaluation on the 100k dataset

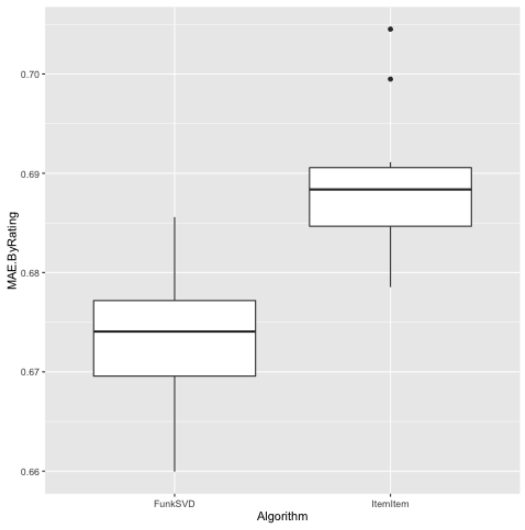


Figure A.4. The MAE values from each cross-fold evaluation on the 1m dataset

Appendix B
FunkSVD Parameter Testing

Average results from running evaluations on different parameter combinations containing.

Parameters	RMSE.ByRating	MAE.ByRating
FunkSVDIC60FC10	0.940418	0.739879
FunkSVDIC60FC15	0.939640	0.739107
FunkSVDIC60FC20	0.939169	0.738715
FunkSVDIC60FC25	0.939057	0.738610
FunkSVDIC60FC30	0.939308	0.738812
FunkSVDIC60FC35	0.939863	0.739277
FunkSVDIC60FC40	0.940555	0.739912
FunkSVDIC60FC45	0.940608	0.740332
FunkSVDIC60FC50	0.940182	0.740086
FunkSVDIC60FC55	0.940521	0.740303
FunkSVDIC60FC60	0.941430	0.741129
FunkSVDIC60FC65	0.942264	0.741907
FunkSVDIC60FC70	0.942765	0.742535
FunkSVDIC60FC75	0.942976	0.742924
FunkSVDIC80FC10	0.938801	0.738877
FunkSVDIC80FC15	0.938164	0.738282
FunkSVDIC80FC20	0.938050	0.738172
FunkSVDIC80FC25	0.938342	0.738410
FunkSVDIC80FC30	0.939041	0.738935
FunkSVDIC80FC35	0.939648	0.739597
FunkSVDIC80FC40	0.938539	0.738971
FunkSVDIC80FC45	0.939070	0.739525
FunkSVDIC80FC50	0.939801	0.740264
FunkSVDIC80FC55	0.940005	0.740602
FunkSVDIC80FC60	0.939964	0.740867
FunkSVDIC80FC65	0.939803	0.740993
FunkSVDIC80FC70	0.939720	0.741139
FunkSVDIC80FC75	0.939677	0.741213
FunkSVDIC100FC10	0.936488	0.737058
FunkSVDIC100FC15	0.936398	0.736961
FunkSVDIC100FC20	0.936525	0.737098
FunkSVDIC100FC25	0.936752	0.737341
FunkSVDIC100FC30	0.937540	0.738027
FunkSVDIC100FC35	0.937773	0.738257
FunkSVDIC100FC40	0.938139	0.738856
FunkSVDIC100FC45	0.938307	0.739135
FunkSVDIC100FC50	0.937082	0.738408
FunkSVDIC100FC55	0.935003	0.737037
FunkSVDIC100FC60	0.934816	0.737150
FunkSVDIC100FC65	0.934623	0.737092
FunkSVDIC100FC70	0.934553	0.737047

FunkSVDIC100FC75	0.934463	0.736962
FunkSVDIC120FC10	0.932832	0.733811
FunkSVDIC120FC15	0.932764	0.733685
FunkSVDIC120FC20	0.933246	0.733879
FunkSVDIC120FC25	0.934868	0.735021
FunkSVDIC120FC30	0.929368	0.730651
FunkSVDIC120FC35	0.929528	0.731540
FunkSVDIC120FC40	0.931849	0.733573
FunkSVDIC120FC45	0.930931	0.733114
FunkSVDIC120FC50	0.929970	0.732659
FunkSVDIC120FC55	0.929187	0.732169
FunkSVDIC120FC60	0.928992	0.732104
FunkSVDIC120FC65	0.928732	0.731841
FunkSVDIC120FC70	0.928115	0.731268
FunkSVDIC120FC75	0.927314	0.730482
FunkSVDIC140FC10	0.926911	0.728213
FunkSVDIC140FC15	0.927124	0.728125
FunkSVDIC140FC20	0.927541	0.728154
FunkSVDIC140FC25	0.926747	0.727809
FunkSVDIC140FC30	0.926916	0.728447
FunkSVDIC140FC35	0.925386	0.728086
FunkSVDIC140FC40	0.923706	0.726779
FunkSVDIC140FC45	0.921922	0.725516
FunkSVDIC140FC50	0.920540	0.724418
FunkSVDIC140FC55	0.919502	0.723533
FunkSVDIC140FC60	0.918742	0.722906
FunkSVDIC140FC65	0.917889	0.722174
FunkSVDIC140FC70	0.917211	0.721567
FunkSVDIC140FC75	0.916635	0.721049
FunkSVDIC140FC80	0.916168	0.720547
FunkSVDIC140FC85	0.915526	0.719885
FunkSVDIC140FC90	0.914794	0.719141
FunkSVDIC140FC95	0.914316	0.718579
FunkSVDIC140FC100	0.914277	0.718390
FunkSVDIC140FC105	0.914643	0.718613
FunkSVDIC160FC10	0.921572	0.723172
FunkSVDIC160FC15	0.921101	0.722168
FunkSVDIC160FC20	0.920842	0.722063
FunkSVDIC160FC25	0.920375	0.722120
FunkSVDIC160FC30	0.919651	0.722210
FunkSVDIC160FC35	0.918539	0.721725
FunkSVDIC160FC40	0.915368	0.719686
FunkSVDIC160FC45	0.912963	0.717654
FunkSVDIC160FC50	0.911924	0.716676
FunkSVDIC160FC55	0.911050	0.716013
FunkSVDIC160FC60	0.910429	0.715560

FunkSVDIC160FC65	0.909965	0.715071
FunkSVDIC160FC70	0.909973	0.714998
FunkSVDIC160FC75	0.909941	0.714780
FunkSVDIC160FC80	0.909944	0.714642
FunkSVDIC160FC85	0.909951	0.714457
FunkSVDIC180FC10	0.917230	0.718389
FunkSVDIC180FC15	0.916202	0.717837
FunkSVDIC180FC20	0.915508	0.717082
FunkSVDIC180FC25	0.914291	0.717014
FunkSVDIC180FC30	0.914303	0.717035
FunkSVDIC180FC35	0.914855	0.717855
FunkSVDIC180FC40	0.909088	0.713752
FunkSVDIC180FC45	0.907207	0.712041
FunkSVDIC180FC50	0.906595	0.711483
FunkSVDIC180FC55	0.906156	0.711012
FunkSVDIC180FC60	0.906019	0.710866
FunkSVDIC180FC65	0.906369	0.711048
FunkSVDIC180FC70	0.906943	0.711295
FunkSVDIC180FC75	0.907395	0.711515
FunkSVDIC200FC10	0.914756	0.715446
FunkSVDIC200FC15	0.914588	0.715801
FunkSVDIC200FC20	0.914979	0.716494
FunkSVDIC200FC25	0.913508	0.715651
FunkSVDIC200FC30	0.915033	0.715877
FunkSVDIC200FC35	0.913766	0.715977
FunkSVDIC200FC40	0.907974	0.711262
FunkSVDIC200FC45	0.906452	0.710348
FunkSVDIC200FC50	0.906007	0.709719
FunkSVDIC200FC55	0.906234	0.710272
FunkSVDIC200FC60	0.906914	0.710190
FunkSVDIC200FC65	0.907428	0.710314
FunkSVDIC200FC70	0.908265	0.710844
FunkSVDIC200FC75	0.909952	0.712496
FunkSVDIC220FC10	0.915706	0.715240
FunkSVDIC220FC15	0.915777	0.716509
FunkSVDIC220FC20	0.917743	0.717199
FunkSVDIC220FC25	0.917972	0.717170
FunkSVDIC220FC30	0.916502	0.716445
FunkSVDIC220FC35	0.916303	0.716596
FunkSVDIC220FC40	0.910001	0.712117
FunkSVDIC220FC45	0.909464	0.711593
FunkSVDIC220FC50	0.910101	0.712173
FunkSVDIC220FC55	0.911103	0.712335
FunkSVDIC220FC60	0.912436	0.712893
FunkSVDIC220FC65	0.914435	0.714581
FunkSVDIC220FC70	0.914435	0.714581

FunkSVDIC220FC75	0.915190	0.715378
FunkSVDIC240FC10	0.917900	0.716444
FunkSVDIC240FC15	0.919829	0.718777
FunkSVDIC240FC20	0.921878	0.720072
FunkSVDIC240FC25	0.923710	0.720565
FunkSVDIC240FC30	0.923432	0.721141
FunkSVDIC240FC35	0.920933	0.719020
FunkSVDIC240FC40	0.916166	0.715725
FunkSVDIC240FC45	0.916639	0.716323
FunkSVDIC240FC50	0.917563	0.716946
FunkSVDIC240FC55	0.917550	0.716946
FunkSVDIC240FC60	0.919897	0.717692
FunkSVDIC240FC65	0.921205	0.718785
FunkSVDIC240FC70	0.922164	0.719574
FunkSVDIC240FC75	0.922386	0.720387
FunkSVDIC260FC10	0.920705	0.718148
FunkSVDIC260FC15	0.924804	0.721802
FunkSVDIC260FC20	0.926705	0.723262
FunkSVDIC260FC25	0.930907	0.726117
FunkSVDIC260FC30	0.929048	0.725190
FunkSVDIC260FC35	0.925693	0.721710
FunkSVDIC260FC45	0.925076	0.721687
FunkSVDIC260FC50	0.926005	0.722621
FunkSVDIC260FC55	0.928450	0.723911
FunkSVDIC260FC60	0.929889	0.725199
FunkSVDIC260FC65	0.929595	0.724304
FunkSVDIC260FC70	0.929348	0.725491
FunkSVDIC260FC75	0.930918	0.726419
FunkSVDIC280FC10	0.923127	0.719874
FunkSVDIC280FC15	0.929608	0.725034
FunkSVDIC280FC20	0.931249	0.725883
FunkSVDIC280FC25	0.934591	0.728519
FunkSVDIC280FC30	0.935064	0.728664
FunkSVDIC280FC35	0.934229	0.727306
FunkSVDIC280FC40	0.928223	0.723545
FunkSVDIC280FC45	0.935036	0.729260
FunkSVDIC280FC50	0.934976	0.728949
FunkSVDIC280FC55	0.937151	0.730283
FunkSVDIC280FC60	0.941109	0.732450
FunkSVDIC280FC65	0.938891	0.731752
FunkSVDIC280FC70	0.940638	0.734679
FunkSVDIC280FC75	0.940310	0.732174
FunkSVDIC300FC10	0.927199	0.722739
FunkSVDIC300FC15	0.933433	0.727787
FunkSVDIC300FC20	0.936392	0.729819

FunkSVDIC300FC25	0.940578	0.732279
FunkSVDIC300FC30	0.942833	0.733320
FunkSVDIC300FC35	0.941756	0.732432
FunkSVDIC300FC40	0.941286	0.732240
FunkSVDIC300FC45	0.940538	0.732974
FunkSVDIC300FC50	0.943976	0.736105
FunkSVDIC300FC55	0.948792	0.737838
FunkSVDIC300FC60	0.946298	0.737685
FunkSVDIC300FC65	0.948191	0.738852
FunkSVDIC300FC70	0.950728	0.740835
FunkSVDIC300FC75	0.952842	0.740651

www.kth.se