Factorization-Based Large Scale Recommendation Algorithms

Thesis booklet

István Pilászy

Under the supervision of
Dr. Domonkos Tikk, Dr. Tadeusz P. Dobrowiecki

Budapest University of Technology and Economics, Hungary
Department of Measurement and Information Systems

## Contents

1 Introduction 2

2 Research outline 2
   2.1 Finding accurate matrix factorizations 4
   2.2 Finding fast matrix factorizations 4
   2.3 Content-based filtering 5
   2.4 Explaining recommendations 5

3 Organization of the dissertation 5

4 Summary of the new results 5

5 Applicability of the results 8

6 List of publications 8
1 Introduction

Recommender systems attempt to profile user preferences over items, and model the relation between users and items. The task of recommender systems is to recommend items that fit user’s tastes, in order to help the user in selecting/purchasing items from an overwhelming set of choices. Such systems have great importance in applications such as e-commerce, subscription based services, information filtering, etc. Recommender systems providing personalized suggestions greatly increase the likelihood of a customer making a purchase compared to unpersonalized ones. Personalized recommendations are especially important in markets where the variety of choices is large, the taste of the customer is important, and last but not least the price of the items is modest. Typical areas of such services are mostly related to art (especially books, movies, music), fashion, food and restaurants, gaming and humor.

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

There are two basic strategies that can be applied when generating recommendations. Content-based approaches profile users and items by identifying their characteristic features, such as demographic data for user profiling, and product information/descriptions for item profiling. The profiles are used by algorithms to connect user interests and item descriptions when generating recommendations. However, it is usually laborious to collect the necessary information about items, and similarly it is often difficult to motivate users to share their personal data to help create the database for the basis of profiling.

Therefore, the alternative approach, termed collaborative filtering (CF), which makes use of only past user activities (for example, transaction history or user satisfaction expressed in ratings), is usually more feasible. CF approaches can be applied to recommender systems independently of the domain. CF algorithms identify relationships between users and items, and make assumptions using this information to predict user preferences.

Netflix Prize Netflix initiated the Netflix Prize competition in October 2006, to improve their recommender system called Cinematch, which was developed in many years. The goal of the competition is to predict users’ ratings on given movies. Predictions are evaluated in terms of RMSE (Root Mean Squared Error). The competition involved a lot of research effort on CF algorithms. Due to the competition, the field of CF has completely changed, and former methods are now completely obsolete. Before the competition, many researches focused on Pearson-correlation-based k-nearest neighbors methods. However, during the competition it turned out that matrix factorization-based approaches are much better than the aforementioned methods, in training time, prediction time and accuracy.

I got involved into this competition in November 2006, as a member of team Gravity. We had been leading the competition between January 2007 and April 2007, and were always at the top.

2 Research outline

The problem of collaborative filtering (CF) can be defined as follows: users provide ratings on items (books, movies, products, etc), and then expect to be recommended with items they are likely to like. The recommendation can be based on the user’s former ratings and the similar users’ former ratings. Users can express their opinion in a numeric scale. A good CF system
is able to predict how the user will rate items, and based on the predictions it can recommend items with the highest predicted rating. In a probabilistic setting, the problem can be modeled by the random triplet \((U, I, R)\), where \(U\) is the user identifier, \(I\) is the item identifier, and \(R\) is the rating given by \(U\) on \(I\). A realization of this random triplet is called a rating. The goal is to model the relationship between \(R\) and \((U, I)\), such that for example the root mean squared error (RMSE) is minimized:

\[
\text{RMSE} = \sqrt{E\{(R - \hat{R})^2\}},
\]  

(1)

Here \(\hat{R}\) is the estimate of \(R\).

Let \(r_{ui}\) denote the rating of user \(u\) on item \(i\), and \(\hat{r}_{ui}\) the prediction of the model for \(r_{ui}\). Let \(N\) denote the number of users, and \(M\) the number of items. The ratings can be arranged into a matrix, denoted by \(R \in \mathbb{R}^{N \times M}\), where the \((u, i)\)-th element is \(r_{ui}\) for known ratings, and unknown for unknown ratings.

**Matrix factorization** Matrix factorization (MF) is one of the most often applied techniques for CF problems. Numerous MF variants have been already published and were validated on the NP dataset as well. The idea behind MF techniques is very simple. Suppose we want to approximate the matrix \(R\) as the product of two smaller matrices:

\[
\hat{R} = PQ^T,
\]  

(2)

where \(\hat{R}\) is the approximation of \(R\) at the known positions. \(P\) is an \(N \times K\) and \(Q\) is a \(M \times K\) matrix. We call \(P\) the user feature matrix and \(Q\) the item feature matrix, and \(K\) is the number of features (factors) in the given factorization.

The above equation can be rewritten in the following form for each element of \(R\):

\[
\hat{r}_{ui} = p^T_u q_i = \sum_{k=1}^{K} p_{uk} q_{ik},
\]  

(3)

where \(p_u\) is the transpose of the \(u\)-th row of \(P\), the user feature vector; \(q_i\) is the transpose of the \(i\)-th row of \(Q\), the item feature vector, \(p_{uk}\) is the \((u, k)\)-th element of \(P\), and \(q_{ik}\) is the \((i, k)\)-th element of \(Q\).

Sometimes a different formula is used:

\[
\hat{r}_{ui} = b_u + c_i + p^T_u q_i
\]  

(4)

Where \(b_u\) and \(c_i\) are the user and item biases resp.

Computing the above factorization is usually done by minimizing the following cost function:

\[
\sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2 + \lambda p_u^T p_u + \lambda q_i^T q_i
\]  

(5)

Where the first term ensures small training errors, and the second and third terms – called regularization terms – ensure small model weights, thus better generalization.

**Evaluation** There are some publicly available datasets for CF experiments. The largest is the Netflix Prize dataset, containing 100 480 507 ratings of 480 189 users on 17 770 movies, on a 1 to 5 scale. The MovieLens datasets are similar, but smaller: there are 3 variants, amongst which the MovieLens-1M is the most prevalent, containing 1 000 000 ratings of 6040 users on 3900 movies on a 1 to 5 scale. The Jester dataset is also commonly used, containing 4 136 360 ratings of 73 421 users on 100 jokes, on a -10 to 10 scale.
2.1 Finding accurate matrix factorizations

A MF variant, called BRISMF was found to be very efficient for these datasets [P6]. Another variant, called ALS (alternating least squares) was also found to be efficient for different reasons [1]. It has been pointed out that larger $K$ values yield better performance at the cost of increased training time [P1, 5]. However, for any dataset many learning parameters needs fine tuning to obtain better performance. My goal was to allow finding an optimized model with a large $K$ value efficiently.

My second goal was to explore whether it is possible to find two or more matrix factorization model such that the linear combination of the predictions of these models are more accurate than any of the two (or more) models.

For MF another cost function can be set up in the following form:

$$\sum_{(u,i) \in \mathcal{R}} (r_{ui} - \hat{r}_{ui})^2 + \sum_{u=1}^{N} \lambda p_u^T p_u + \sum_{i=1}^{M} \lambda q_i^T q_i, \quad (6)$$

which may be more intuitive, since the regularization terms are not proportional to the number of ratings of users or items. However, the prediction accuracy of this setting is worse. This naturally leads to the following question: what it the best way of regularization. I examined whether it is possible to find a better regularization than (5).

Compared to BRISMF, a very different factorization approach was proposed by Paterek [4], which is called NSVD1. It uses the following formula for prediction:

$$r_{ui} = b_u + c_i + p_u^T q_i \quad \text{where} \quad p_u = n_u^{-0.5} \sum_{j:(u,j) \in \mathcal{R}} w_j \quad (7)$$

where $n_u$ is the number of ratings of $u$. In this formula, $p_u$ is not a free variable: we have a second set of item feature vectors ($w_j$), and $p_u$ depends on what movies the user rated, but not on how they are rated. Thus, the free parameters of the model are: two sets of item feature vectors ($q_i$ and $w_j$) and user and item biases ($b_u$ and $c_i$). The rationale behind this formula is the following: if $u$ watches a movie, this gives information about the user’s taste, although this information is not as valuable as an explicit rating. The information is captured by the $w_j$ variables.

The linear blending of the prediction of NSVD1 and BRISMF provides a much better performance than a standalone BRISMF or NSVD1. I examined whether the sum of the two prediction formulas can result in a more accurate model. The prediction formula looks like:

$$\hat{r}_{ui} = b_u + c_i + p_u^T q_i' + n_u^{-0.5} \left( \sum_{j:(u,j) \in \mathcal{R}} w_j^T q_i'' \right) \quad (8)$$

2.2 Finding fast matrix factorizations

First, I dealt with the following problem: if user $u$ gives a new ratings, and wants to receive recommendations instantly, then how can we give recommendations to her without rerunning the whole optimization process of BRISMF.

The drawback of the algorithm I found is that after receiving many new ratings, rerunning the whole optimization process is unavoidable. The learning parameters of BRISMF have influence on its running time. I examined whether it is possible to optimize the parameters to find a good trade off between model accuracy and running time in the full optimization process.

The datasets mentioned above are called explicit feedback datasets, because users express explicitly their opinion about items. In many situations only a weaker information is available which is called implicit feedback. If items are movies, then the implicit feedback can be for
example that the user watched the movie, but we do not know whether she liked or not after watching. In these datasets, the whole $R$ is filled by zeros or ones (typically), which makes the optimization process very costly. Hu et al. [2] suggested an efficient method, but the running time is still cubic in $K$. I examined how the learning process can be speed up by applying the Sherman-Morrison formula. Moreover, I examined other possible applications of this formula in the field of CF.

2.3 Content-based filtering

Paterek’s NSVD1 can be seen as a neural network where the input is the binarized ratings of the user, and the output is the user’s actual ratings on items. The challenge is to train this neural network in a feasible time. An efficient training algorithm was published in [P3].

Based on this published method, my idea was to swap the role of users and items, and use movie metadata as the input, and the users’ rating on the movie as the output for this neural network (NN). After training this NN, we can predict users preferences on new movies (these movies are new in the sense that they have no rating at all). My goal was to investigate how many explicit ratings are equal to movie metadata, in terms of prediction accuracy.

2.4 Explaining recommendations

Hu et al. [2] provided an explanation algorithm for ALS-based MF that can explain any prediction by listing those ratings of the active user $u$ that have the most influence on the prediction. My goal was to carry over this method for BRISMF-based MFs.

3 Organization of the dissertation

The first chapter gives an introduction into the field. The second chapter surveys the literature. The next four chapters describe my theses:

- Methods that aim to improve on the accuracy of MF
- Methods that aim to improve on the speed of MF: how to efficiently handle new users, or new ratings of users; how to speed up the model building process;
- Methods for content-based filtering: The third part deals with how textual description of movies can be used to give recommendations to users, especially recommending new movies, i.e. movies that no one has rated (yet). Again, the focus is on getting the lowest possible error of predictions. I point out that the average of 10 ratings of a movie is more predictive than the textual description of the movie.
- Method for explaining recommendations of BRISMF.

The next two chapters summarize the results and direct further research.

4 Summary of the new results

Thesis group 1: I proposed many methods for improving RMSE of matrix factorizations. These methods are published in [P1, P2, P3, P4, P5, P6].

Thesis 1.1

I suggested using 8 parameters in BRISMF ($\eta^p, \eta^q, \lambda^p, \lambda^q, \eta^{pb}, \eta^{qb}, \lambda^{pb}, \lambda^{qb}$) instead of 4 ($\eta^p, \eta^q, \lambda^p, \lambda^q$) or 2 ($\eta, \lambda$). I applied automatic parameter optimization to set the parameters. I suggested user-subsampling and using small $K$ values to speed up the optimization process. This may result in a $30 \times$ speedup of the parameter optimization process. With the above considerations I achieved 6.04% improvement over Netflix’s Cinematch algorithm.
I also evaluated the proposed parameter optimization procedure on the MovieLens and Jester datasets. I concluded that the introduction of 4 or 8 parameters is useless for these datasets. I also evaluated whether user-subsampling and the usage of small $K$ values are useful to improve the speed of parameter optimization. I concluded that using small $K$ values is useful. Subsampling the users is useful on the MovieLens dataset, but not on the Jester dataset.

**Thesis 1.2**
I suggested finding automatic BRISMF parameterizations that improve on a combination of previously existing methods. I showed on the Netflix Prize dataset, that the proposed approach works: on BRISMF#250, two models (MLMF#200 and MLMF#80) can improve by $0.0011 + 0.0004 = 0.0015$ RMSE points. This result allows to achieve the same performance as of BRISMF#1000, but within less than half of that time and memory requirement (1000 factors versus 250+200+80 factors).

I also evaluated the method on the Jester and MovieLens datasets. Again, the method can find models which blends well with a given model.

**Thesis 1.3**
I suggested that the regularization and learning rate parameters of BRISMF should depend on the number of ratings of the active user and item. I proposed to replace each of these parameters with a function with 7 parameters. Again, parameters are optimized with an automatic parameter optimization algorithm. This method can achieve 6.53% improvement over Netflix’s Cinematch algorithm. I also tested the method on the Jester dataset with success, and on the MovieLens dataset with moderate success.

**Thesis 1.4**
I suggested a simple modification of the learning rule of BRISMF algorithm: in the best epoch, reset $P$ and keep the learning process going on until RMSE on a validation set improves. This modification can improve the accuracy of BRISMF-based matrix factorizations. On the Netflix Prize dataset: The RMSE performance of BRISMF#250 and BRISMF#1000 on Quiz can be boosted by 0.0027 and 0.0021 resp. I also evaluated the method on the Jester dataset with success, and on the MovieLens without success. However, I pointed out, that this may be due to the small number of users in that dataset.

**Thesis 1.5**
I suggested a hybrid method that efficiently alloys BRISMF and NSVD1. The idea is to sum the output the two methods, and train them simultaneously. This can be seen as training two methods on the residual of each other. I put BRISMF and NSVD1 in the framework of multi-layer perceptrons. I experimented with the Netflix Prize data, where the method (Hybrid#1000-80) achieved 6.61% improvement over Cinematch. I also pointed out that the approach blends well with the method of Thesis 1.3: the blending of this 6.61% and that 6.39% approaches can achieve 7.04% improvement over Cinematch.

**Thesis group 2**: I proposed many methods to speed up BRISMF and ALS based matrix factorizations, and also to efficiently handle new ratings of users, or new users. These methods are published in [P4, P6, P8]

**Thesis 2.1**
I proposed a modification for BRISMF that can efficiently handle new ratings of users. I pointed out by extensive experiments on the Netflix Prize dataset, that the method is efficient in terms of both speed and accuracy. As a result, only a small subset of the database is enough to get a reliable estimate of $Q$, and then fixing this matrix we can recompute $P$ for each user separately, using all of her ratings. I also evaluated the method on the MovieLens dataset with success, and on the Jester dataset with moderate success.

**Thesis 2.2**
I proposed to optimize the parameters of BRISMF such that the best RMSE is achieved
in the first few epochs. I pointed out by experiments on the Netflix Prize dataset, that only 200 seconds is enough to get a model in 2 epochs with Probe10 RMSE = 0.9071, on a Pentium M 2GHz (Dothan) CPU. I also tried the idea on the Jester and MovieLens datasets. The results are acceptable. On the MovieLens dataset, I concluded that if we had more users, the results would be more favorable.

**Thesis 2.3**
I examined the applicability of the Sherman-Morrison formula for alternating least squares based matrix factorizations. I showed that by using the Sherman-Morrison formula ALS based algorithms can be speed up both on explicit and implicit feedback datasets. For explicit feedback, I improved from \(O(K^2 \cdot \|R\| + K^3 \cdot N)\) to \(O(K^2 \cdot \|R\|)\) the time complexity of the recomputation of the user feature matrix \(\mathbf{P}\). For implicit feedback \(O(K^2 \cdot \|R^+\| + K^3 \cdot N)\) is reduced to \(O(K^2 \cdot \|R^+\|)\). For the item feature matrix \(\mathbf{Q}\) analog results hold. If a user provides a new rating, recomputing \(\mathbf{p}_u\) can be speed up from \(O(K^3)\) to \(O(K^2)\).

I also pointed out that SMF can also be applied to reduce the complexity of greedy feature selection algorithms on \(F\) predictors from \(O(F^5)\) to \(O(F^4)\).

**Thesis 2.4**
I examined the applicability of linear kernel ridge regression for ALS based matrix factorizations. I showed, how linear kernel ridge regression can speed up the recomputation of \(\mathbf{p}_u\) in ALS for users with \(n_u < K\) ratings: the original \(O(K^2 \cdot n_u + K^3)\) is reduced with SMF to \(O(K^2 \cdot n_u)\), and with KRR to \(O(K \cdot n_u^2)\). I performed experiments to show how much computational performance gain can be brought by KRR.

I pointed out that the addition or deletion of ratings can be handled in \(O(K^2)\) by SMF, and if \(n_u < K\), then in \(O(K \cdot n_u)\) with KRR + SMF, while the traditional way requires \(O(K^3)\) operations. These proposed methods allows ALS to be used in practical recommender systems, where the system must respond instantly, when a new user rating arrives.

**Thesis group 3:** Methods for content-based filtering on new items. These methods are published in [P9].

**Thesis 3.1**
Based on Paterek’s NSVD1, I suggested a method for content-based filtering with movie-metadata. I suggested evaluating content-based filtering methods on new movies. I pointed out that the proposed approach outperforms baseline methods: it can reach X10 RMSE = 0.9990, while with baseline approaches, I achieved only X10 RMSE = 1.0305.

**Thesis 3.2**
I pointed out that when we give 10 ratings on new movies (thus they are not new anymore), the performance of even a simple bias predictor is better, than the above proposed method’s performance on new movies (0.9990). From this, we can conjecture that from the viewpoint of recommender systems, even the average of 10 ratings of a new movie is more valuable (more predictive) than the textual description of a movie, containing the genre, actors, title and plot summary of the movie.

The importance of Thesis 3.2 is the following: many papers are published in the field of CF/CBF claiming that metadata can be useful in recommender systems. Many top contenders of the Netflix Prize competition claim that it is useless. In my opinion, metadata can improve only a badly parameterized method. This experiment tried to relate the usefulness of the two data sources, and concluded that movie metadata is practically useless (but only movie metadata).

**Thesis 4:** Explaining recommendations:
Based on the results of Hu et al. [2], which provides a method for explaining recommendations of ALS-based MFs, I proposed a method which is able to explain recommendations of BRISMF. I performed some experiments that demonstrate the viability of the algorithm. The results are
similar to those obtained by Hu et al.’s method. I showed how the gradient descent algorithm can be related to the weighted ridge regression: if we are given with the dual model of a linear model, we can compute an importance weight for each example, such that running weighted ridge regression with those importance weights, the result will be (almost) equal to that model. In this way, Hu et al.’s method – which can explain the predictions of weighted ridge regression – can be applied to explain the predictions of BRISMF algorithms. This method is published in [P7].

5 Applicability of the results

Each of my proposed methods deal with real aspects of recommender systems:

- The first thesis group focuses on getting the best accuracies. It has been pointed out [3] that small improvements in the prediction accuracy can have high effect on the ordering of recommendations. Thus, accurate matrix factorizations – unless they need too many computational resources – are of real interest.
- The second thesis group focuses on getting the good accuracies in short time. Moreover, if a user provide new ratings, she can expect to be recommended instantly. I gave a method for explicit feedback, and another methods for implicit feedback problems.
- The third thesis group gives a new algorithm which can recommend new items without any ratings, based solely on their metadata. However, I conclude that movie metadata is not useful to give predictions. Exploring other domains with the algorithm is in my research plan.
- The fourth thesis gives an algorithm than can explain to the user why it thinks that the item recommended by BRISMF will be suitable for the user. In many situations, a good explanation can make the user trust the recommendations.

6 List of publications


http://szovegbanyaszat.typotex.hu/page.php?id=27

References


