Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem

Gábor Takács*
Dept. of Mathematics and Computer Science
Széchenyi István University
Egyetem tér 1.
Győr, Hungary
gtakacs@sze.hu

István Pilászy
Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
pila@mit.bme.hu

Bottyán Németh
Dept. of Telecom. and Media Informatics
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
bottyan@tmit.bme.hu

Domonkos Tikk
Dept. of Telecom. and Media Informatics
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
tikk@tmit.bme.hu

ABSTRACT

Collaborative filtering (CF) approaches proved to be effective for recommender systems in predicting user preferences in item selection using known user ratings of items. This subfield of machine learning has gained a lot of popularity with the Netflix Prize competition started in October 2006. Two major approaches for this problem are matrix factorization (MF) and the neighbor based approach (NB). In this work, we propose various variants of MF and NB that can boost the performance of the usual ensemble based scheme. First, we investigate various regularization scenarios for MF. Second, we introduce two NB methods: one is based on correlation coefficients and the other on linear least squares. At the experimentation part, we show that the proposed approaches compare favorably with existing ones in terms of prediction accuracy and/or required training time. We present results of blending the proposed methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*parameter learning*

*All authors are also affiliated with Gravity Research & Development Ltd., H-1092 Budapest, Kinizsi u. 11., Hungary, info@gravitrd.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

General Terms

Algorithms, Experimentation

Keywords

Collaborative filtering, matrix factorization, neighbor based methods, Netflix Prize

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 the user's taste, 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 (esp. books, movies, music), fashion, food & restaurants, gaming & 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.com 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 (see details in Section 5.1).

There are two basic strategies that can be applied when generating recommendations. *Content-based approaches* profile users and items by identifying their characteristic features using, such us demographic data for user profiling, and product information/descriptions for item profiling. The profiles can be 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 user activities of the past (for example, transaction history or user satisfaction expressed in rating) is usually more feasible. CF approaches can be applied for recommender systems independently of the domain. CF algorithms identify relationships between users and items, and make associations using this information to predict user preferences.

In this paper, we focus on the case when users express their opinion of items by means of ratings. In this framework the user first provides ratings of some items usually on a discrete numerical scale, and the system then recommends other items based on ratings already provided by other users.

1.1 Related Work

The first works on the field of CF have been published in the early 1990s. In [7], the Tapestry system is presented that uses collaborative filtering to filter mails simultaneously from several mailing lists based on the opinion of other users on readings. In [12], the GroupLens system is described that was one of the pioneer applications of the field where users could rate articles on a 1–5 scale after having read them, and were then offered suggestions. In [5], the underlying techniques of predicting user preferences are divided into two main groups. *Memory-based* approaches operate on the entire database of ratings collected by the vendor or service supplier. On the other hand, *model-based* approaches use the database to estimate or learn a model, and then apply this model for prediction.

Over the last broad decade many CF algorithms have been proposed that approach the problem by different techniques, including similarity/neighborhood based approaches [12, 16], personality diagnosis [10], Bayesian networks [5], restricted Boltzman machines [14], and various matrix factorization techniques [6, 8, 15, 17]. In [19], an MF approach is presented that incorporates demographic information and ratings to enhance plain CF algorithms. In [5], the major CF approaches of the early years are surveyed, while in [11], short descriptions of most recent methods are given.

The NP competition boosted the interest in CF, and yielded the publication of a number of new methods. We should also mention here the NP related KDD Cup and Workshop [3], which indicated the possible directions of scalable and accurate CF methods. We feature among them the matrix factorization and neighbor based approaches.

Several matrix factorization techniques have been successfully applied to CF, including singular value decomposition [15], probabilistic latent semantic analysis [8], probabilistic matrix factorization [13], maximum margin matrix factorization [17] and alternating least squares [1].

Simon Funk (Brandyn Webb) published a detailed implementation of a regularized MF with separate feature update.[1] In [9], a bunch of novel techniques is introduced, including biased MF, kernel ridge regression on the residual of MF, linear model for each item, and asymmetric factor model (NSVD1, NSVD2). In [1], an improved neighborhood based approach is applied successfully, which removes the global effect from the data, and calculates optimal similarity values by solving regression problems.

2. PROBLEM DEFINITION

We define the problem of *collaborative filtering* (CF) in the following setting. The problem can be modeled by the random triplet (U, I, R), where

- U taking values from $\{1, \dots, N\}$ is the *user identifier*,
- I taking values from $\{1, \dots, M\}$ is the *item identifier*,
- R taking values from $\mathcal{X} \subset \mathbb{R}$ is the *rating value*.

A realization of (U, I, R) denoted by (u, i, r) means that user u rated item i with value r. The goal is to estimate R from (U, I) such that the root mean squared error of the estimate,

$$\mathrm{RMSE} = \sqrt{\mathbf{E}\{(\hat{R} - R)^2\}}, \tag{1}$$

is minimal, where $\hat{R}$ is the estimate[2] of R.

In practice, the distribution of (U, I, R) is not known, we are only given a finite sample, $\mathcal{T}' = \{(u_1, i_1, r_1), (u_2, i_2, r_2), \dots, (u_t, i_t, r_t)\}$, generated by it. The sample $\mathcal{T}'$ can be used for training predictors. We assume "sampling without replacement" in the sense that (user ID, item ID) pairs are unique in the sample, which means that users do not rate items more than once. Let us introduce the notation $\mathcal{T} = \{(u, i) : \exists r : (u, i, r) \in \mathcal{T}'\}$ for the set of (user ID, item ID) pairs. Note that $|\mathcal{T}'| = |\mathcal{T}|$, and typically $|\mathcal{T}| \ll N \cdot M$, because most of the users rate only a few items. Denote the set of items rated by the u-th user by $\mathcal{T}_u = \{i : (u, i) \in \mathcal{T}\}$. Denote the set of users who rated the i-th item by $\mathcal{T}^{(i)} = \{u : (u, i) \in \mathcal{T}\}$.

The sample can be represented as a partially specified matrix denoted by $\mathbf{R} \in \mathbb{R}^{N \times M}$, where the matrix elements are known in positions $(u, i) \in \mathcal{T}$, and unknown in positions $(u, i) \notin \mathcal{T}$. The value of the matrix $\mathbf{R}$ at position $(u, i) \in \mathcal{T}$, denoted by r_{ui}, stores the rating of user u for item i. For clarity, we use the term (u, i)-th rating in general for r_{ui}, and (u, i)-th training example if $r_{ui} : (u, i) \in \mathcal{T}$.

When we predict a given rating r_{ui} by $\hat{r}_{ui}$ we refer to the user u as *active user*, and to the item i as *active item*. The (u, i) pair of active user and active item is termed *query*.

3. MATRIX FACTORIZATION

Matrix factorization (MF) is one of the most often applied techniques for CF problems. The idea behind MF techniques is very simple. Suppose we want to approximate the matrix $\mathbf{R}$ as the product of two matrices:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q},$$

where $\mathbf{P}$ is an $N \times K$ and $\mathbf{Q}$ is a $K \times M$ matrix. This factorization gives a low dimensional numerical representation of

[1] `http://sifter.org/~simon/journal/20061211.html`

[2] In general, superscript "hat" denotes the prediction of the given quantity, that is, $\hat{x}$ is the prediction of x.

both users and items. Note, that $\mathbf{Q}$ and $\mathbf{P}$ typically contain real numbers, even when $\mathbf{R}$ contains only integers.

In the case of the given problem, the unknown ratings of $\mathbf{R}$ cannot be represented by zero. For this case, the approximation task can be defined as follows. Let p_{uk} denote the elements of $\mathbf{P} \in \mathbb{R}^{N \times K}$, and q_{ki} the elements of $\mathbf{Q} \in \mathbb{R}^{K \times M}$. Let $\mathbf{p}_u^T$ denote the transpose of the u-th row of $\mathbf{P}$, and $\mathbf{q}_i$ the i-th column of $\mathbf{Q}$. Then:

$$\hat{r}_{ui} = \sum_{k=1}^{K} p_{uk} q_{ki} = \mathbf{p}_u^T \mathbf{q}_i \tag{2}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui} \quad \text{for } (u,i) \in \mathcal{T} \tag{3}$$

$$\text{SSE} = \sum_{(u,i)\in\mathcal{T}} e_{ui}^2 = \sum_{(u,i)\in\mathcal{T}} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2$$

$$\text{RMSE} = \sqrt{\text{SSE}/|\mathcal{T}|}$$

$$(\mathbf{P}^*, \mathbf{Q}^*) = \underset{(\mathbf{P},\mathbf{Q})}{\arg\min}\, \text{SSE} = \underset{(\mathbf{P},\mathbf{Q})}{\arg\min}\, \text{RMSE} \tag{4}$$

Here $\hat{r}_{ui}$ denotes how the u-th user would rate the i-th item, according to the model, e_{ui} denotes the training error measured at the (u,i)-th rating, SSE denotes the sum of squared training errors. Eq. (4) states that the optimal $\mathbf{P}$ and $\mathbf{Q}$ minimizes the sum of squared errors only on the known elements of $\mathbf{R}$.

3.1 Regularized MF (RISMF)

To avoid overfitting, we apply regularization by penalizing the magnitude of vectors, which introduces the following variables:

$$e'_{ui} = \frac{1}{2}(e_{ui}^2 + \lambda \mathbf{p}_u^T \mathbf{p}_u + \lambda \mathbf{q}_i^T \mathbf{q}_i) \tag{5}$$

$$\text{SSE}' = \sum_{(u,i)\in\mathcal{T}} e'_{ui}$$

$$(\mathbf{P}^{*\prime}, \mathbf{Q}^{*\prime}) = \underset{(\mathbf{P},\mathbf{Q})}{\arg\min}\, \text{SSE}' \tag{6}$$

Eq. (6) states that the optimal $\mathbf{P}$ and $\mathbf{Q}$ try to give a good approximation of $\mathbf{R}$ while containing only small numbers.

In order to minimize RMSE$'$, which is equivalent to minimize SSE$'$, we apply a simple incremental gradient descent[3] method to find a local minimum of SSE$'$, where one gradient step intends to decrease e'_{ui}.

Suppose we are at the (u,i)-th training example, r_{ui}, and its approximation $\hat{r}_{ui}$ is given. We compute the gradient of e'_{ui} for $k \in \{1, \ldots, K\}$:

$$\frac{\partial}{\partial p_{uk}} e'_{ui} = -e_{ui} \cdot q_{ki} + \lambda \cdot p_{uk}, \frac{\partial}{\partial q_{ki}} e'_{ui} = -e_{ui} \cdot p_{uk} + \lambda \cdot q_{ki}. \tag{7}$$

We update the weights in the direction opposite to the gradient:

$$\begin{aligned} p'_{uk} &= p_{uk} + \eta \cdot (e_{ui} \cdot q_{ki} - \lambda \cdot p_{uk}), \\ q'_{ki} &= q_{ki} + \eta \cdot (e_{ui} \cdot p_{uk} - \lambda \cdot q_{ki}), \end{aligned} \tag{8}$$

that is, we change the weights in $\mathbf{P}$ and $\mathbf{Q}$ to decrease the square of actual error and keep the weights of $\mathbf{P}$ and $\mathbf{Q}$ small. Note that opposed to Simon Funk, we train all values of $\mathbf{P}$ and $\mathbf{Q}$ simultaneously. We remark when the training has finished, each value of $\mathbf{R}$ can be computed easily using eq. (2), even for unknown ratings. In other words, the model ($\mathbf{P}^{*\prime}$ and $\mathbf{Q}^{*\prime}$) provides a description of how an arbitrary user would rate any item, though for known ratings the value may be considerably different from the predicted value. We refer to this method as Regularized Incremental Simultaneous MF (RISMF).

[3] Also known as stochastic gradient descent.

3.2 Extension of RISMF (XRISMF)

The presented MF has 4 important meta-parameters: λ, η, and the initial value of $\mathbf{P}$ and $\mathbf{Q}$, which we denote by $\mathbf{P}^0$ and $\mathbf{Q}^0$ resp. We can get the proper setting of these parameters via trial by error or a parameter optimization algorithm to maximize the performance of a recommender system.

If we introduce more parameters, then we are able to get better performance, but the task of setting these parameters will become harder. In the followings, we introduce many-many parameters to get a very general method, then we present some special cases that will have advantageous properties.

Let p_{uk}^0 and q_{kj}^0 denote the elements of $\mathbf{P}^0$ and $\mathbf{Q}^0$ resp. We replace λ and η with $\lambda^p(u,i,k)$, $\lambda^q(u,i,k)$, $\eta^p(u,i,k)$ and $\eta^q(u,i,k)$ resp.

The formulas of eq. (8) for the (u,i)-th rating and the k-th feature become the following:

$$\begin{aligned} p'_{uk} &= p_{uk} + \eta^p(u,i,k) \cdot (e_{ui} \cdot q_{ki} - \lambda^p(u,i,k) \cdot p_{uk}), \\ q'_{ki} &= q_{ki} + \eta^q(u,i,k) \cdot (e_{ui} \cdot p_{uk} - \lambda^q(u,i,k) \cdot q_{ki}) \end{aligned} \tag{9}$$

For the training algorithm of this MF, see Algorithm 1. Note that we use different terminal conditions in Algorithm 1 and in eq. (6), because we optimize in the former case for the validation set, not for the training set.

Input: $\mathcal{T}'$: training set,
$\eta^p(u,i,k), \eta^q(u,i,k)$: learning rate,
$\lambda^p(u,i,k), \lambda^q(u,i,k)$: regularization factor,
K: number of features,
$\mathbf{P}^0 = \{p_{uk}^0\}$ and $\mathbf{Q}^0 = \{q_{kj}^0\}$:
initial value of user and item feature matrices
Output: $\mathbf{P}^*$,$\mathbf{Q}^*$: the user and item feature matrices
Partition $\mathcal{T}'$ into two sets: $\mathcal{T}'_{\text{I}}$, $\mathcal{T}'_{\text{II}}$ (validation set)
Initialize $\mathbf{P}$ and $\mathbf{Q}$ with $\mathbf{P}^0$ and $\mathbf{Q}^0$ resp.
loop until the terminal condition is met. One epoch:
  for each element (u,i,r_{ui}) of $\mathcal{T}'_{\text{I}}$:
    compute e_{ui} according to eq. (3);
    for each $k \in \{1, \ldots, K\}$:
      update $\mathbf{p}_u$, the u-th row of $\mathbf{P}$, and
      $\mathbf{q}_i$, the i-th column of $\mathbf{Q}$ according to eq. (9);
    end
  end
  calculate the RMSE on $\mathcal{T}'_{\text{II}}$;
  if RMSE on $\mathcal{T}'_{\text{II}}$ was better than in any previous
      epoch:
    Let $\mathbf{P}^* = \mathbf{P}$ and $\mathbf{Q}^* = \mathbf{Q}$.
  end
  terminal condition: RMSE on $\mathcal{T}'_{\text{II}}$ does not
  decrease during two epochs.
end

Algorithm 1: Training algorithm for XRISMF

3.3 RISMF with biases (BRISMF)

We get BRISMF by introducing a bias feature per user and per item for RISMF. BRISMF is defined by specializing XRISMF in the following way:

- Let $p^0_{\bullet 1} = 1$. Let the rest of $\mathbf{P}^0$ be small random numbers.
- Let $q^0_{2\bullet} = 1$. Let the rest of $\mathbf{Q}^0$ be small random numbers.
- Let $\eta^p(u,i,1) = 0$. Let $\eta^p(u,i,2) = \eta^{pb}$. Let $\eta^p(u,i,k) = \eta^p$ for $k > 2$.
- Let $\lambda^p(u,i,1) = 0$. Let $\lambda^p(u,i,2) = \lambda^{pb}$. Let $\lambda^p(u,i,k) = \lambda^p$ for $k > 2$.
- Let $\eta^q(u,i,2) = 0$. Let $\eta^q(u,i,1) = \eta^{qb}$. Let $\eta^q(u,i,k) = \eta^q$ for $k > 2$.
- Let $\lambda^q(u,i,2) = 0$. Let $\lambda^q(u,i,1) = \lambda^{qb}$. Let $\lambda^q(u,i,k) = \lambda^q$ for $k > 2$.

These settings mean that we set the first feature of each user and the second feature of each item to 1, and never change these values. We have different learning rate and regularization for users and items, and for bias and non-bias features. In short, BRISMF has the following meta-parameters: $\eta^p, \eta^{pb}, \lambda^p, \lambda^{pb}, \eta^q, \eta^{qb}, \lambda^q, \lambda^{qb}$, and two random variables to initialize P^0 and Q^0.

An interesting propery of this MF is that it converges much faster and is more accurate than RISMF.

3.4 BRISMF with special regularization (SBRISMF)

In RISMF, the regularization of user and item feature vectors is done per rating (eq. (5)). Thus the weight penalization of a user's or item's feature vector is proportional to the number of ratings of that user or item resp.

However, a CF problem may require that the features of users or items with many ratings should be penalized less. To achieve this goal, we introduce an extension of BRISMF by specializing XRISMF in the following way: We replace regularization and learning rate parameters with functions with 7 parameters such that the value of those functions depend only on the number of ratings of active user and item. Let

$$\begin{aligned} f(u,i,[p_1,p_2,p_3,p_4,p_5,p_6,p_7]) = p_1 + \\ + p_2/\log(|\mathcal{T}_u|) + p_3/\sqrt{|\mathcal{T}_u|} + p_4/|\mathcal{T}_u| + \\ + p_5/\log(|\mathcal{T}^{(i)}|) + p_6/\sqrt{|\mathcal{T}^{(i)}|} + p_7/|\mathcal{T}^{(i)}| \end{aligned} \tag{10}$$

The definition of SBRISMF has a similar structure to BRISMF:

- Let $p^0_{\bullet 1} = 1$. Let the rest of P^0 be small random numbers.
- Let $q^0_{2\bullet} = 1$. Let the rest of Q^0 be small random numbers.
- Let $\eta^p(u,i,1) = 0$. Let $\eta^p(u,i,2) = f(u,i,[\eta^{pb}_1,\ldots,\eta^{pb}_7])$. Let $\eta^p(u,i,k) = f(u,i,[\eta^p_1,\ldots,\eta^p_7])$ for $k > 2$.
- Let $\lambda^p(u,i,1) = 0$. Let $\lambda^p(u,i,2) = f(u,i,[\lambda^{pb}_1,\ldots,\lambda^{pb}_7])$. Let $\lambda^p(u,i,k) = f(u,i,[\lambda^p_1,\ldots,\lambda^p_7])$ for $k > 2$.
- Let $\eta^q(u,i,2) = 0$. Let $\eta^q(u,i,1) = f(u,i,[\eta^{qb}_1,\ldots,\eta^{qb}_7])$. Let $\eta^q(u,i,k) = f(u,i,[\eta^q_1,\ldots,\eta^q_7])$ for $k > 2$.
- Let $\lambda^q(u,i,2) = 0$. Let $\lambda^q(u,i,1) = f(u,i,[\lambda^{qb}_1,\ldots,\lambda^{qb}_7])$. Let $\lambda^q(u,i,k) = f(u,i,[\lambda^q_1,\ldots,\lambda^q_7])$ for $k > 2$.

Now we have 7 times as many parameters as BRISMF, not counting the two random variables for initialization.

Though at first sight this parametrization seems as an unnecessary overcomplication, it can give a very accurate model, which blends well with other models.

4. NEIGHBOR BASED METHODS

Neighbor based methods exploit the observation that similar users rate similar items similarly. In this scheme for each query a set of similar users is selected from those ones who rated the active item. Or analogously, a set of similar items is selected from those ones that have been rated by the active user. The prediction is then calculated from the ratings of similar users (items) for the active item (user). The first variant is termed user neighbor based, and the second is termed item neighbor based approach.

We present our algorithms for such a CF setting in which the number of items is much smaller than the number of users (ten thousands vs. millions). For recommendation systems with so many users it is impossible to keep all user–user similarities in the memory, however it may be possible to do this for items. In such cases the item neighbor based variant is much more efficient than the user neighbor based one. Therefore the algorithms will be presented using the item neighbor based scheme. The corresponding user neighbor based variants can be easily obtained by exchanging the terms "user" and "item".

Now let us define the item neighbor based scheme more formally. Recall that the set of items rated by user u is denoted by by $\mathcal{T}_u$. Denote the neighborhood set of the query (u,i) by $\mathcal{N}_{ui}$ ($\mathcal{N}_{ui} \subseteq \mathcal{T}_u$). The prediction of the (u,i)-th rating is the following:

$$\hat{r}_{ui} = \sum_{j \in \mathcal{N}_{ui}} w(u,i,j) \cdot p(u,i,j), \tag{11}$$

where $w(u,i,j)$ is the j-th interpolation weight and $p(u,i,j)$ is the j-th subprediction at making prediction for the (u,i)-th rating. Various item neighbor based algorithms differ in how they define $\mathcal{N}_{ui}$, $w(u,i,j)$ and $p(u,i,j)$.

4.1 A correlation based approach (NB-CORR)

The most common approach is to use a correlation based similarity and restrict the number of relevant neighbors to a fixed number K. A plenty of slightly different variants exist within this algorithm family [11, 16, 18]. Here we propose our variant that is quite accurate on the Netflix Prize problem [4].

The empirical correlation coefficient c_{ij} between items i

and j can be calculated as the following:

$$\begin{aligned}
\mathcal{R}_{ij} &= \{u : (u,i),(u,j) \in \mathcal{T}\},\\
\mu_{ij} &= \frac{1}{|\mathcal{R}_{ij}|}\sum_{u\in\mathcal{R}_{ij}} r_{ui},\\
\sigma_{ij} &= \sqrt{\frac{1}{|\mathcal{R}_{ij}|}\sum_{u\in\mathcal{R}_{ij}} (r_{ui}-\mu_{ij})^2},\\
c_{ij} &= \frac{1}{|\mathcal{R}_{ij}|}\sum_{u\in\mathcal{R}_{ij}} \frac{(r_{ui}-\mu_{ij})(x_{uj}-\mu_{ji})}{\sigma_{ij}\sigma_{ji}}. \qquad (12)
\end{aligned}$$

If there are only a few users who rated both the i-th and j-th item, then the estimate (12) is bad: $|c_{ij}|$ is typically larger than the absolute value of the true correlation. Therefore it is useful to shrink c_{ij} towards zero to get an improved estimate c'_{ij}:

$$c'_{ij} = \frac{|\mathcal{R}_{ij}|}{|\mathcal{R}_{ij}|+\beta}\cdot c_{ij},$$

where β is called the first regularization term. The similarity between items i and j denoted by s_{ij} is defined as the following:

$$s_{ij} = \begin{cases} \left(c'_{ij}\right)^{\alpha} & \text{if } c'_{ij} \geq 0,\\ 0 & \text{otherwise,}\end{cases}$$

where α is called the amplification factor. The value s_{ij} is high, if items i and j are similar, and 0 if they are negatively correlated. Now we are ready to define $\mathcal{N}_{ui}$, $w(u,i,j)$ and $p(u,i,j)$:

$$\begin{aligned}
\mathcal{N}_{ui} &= \{\text{the } \min\{K,|\mathcal{T}_u|\} \text{ nearest neighbors of}\\
&\qquad \text{item } i \text{ among } \mathcal{T}_u, \text{ according to } s_{ij}\},\\
w(u,i,j) &= \begin{cases} \dfrac{s_{ij}}{\gamma + \sum_{k\in\mathcal{N}_{ui}} s_{ik}} & \text{if } i \neq j,\\ 1 & \text{otherwise,}\end{cases}\\
p(u,i,j) &= \begin{cases} r_{uj}-\mu_{ji} & \text{if } i \neq j,\\ \mu_i & \text{otherwise,}\end{cases}
\end{aligned}$$

where γ is called the second regularization term and μ_i denotes the average rating of item i in the training set.

An advantageous property of this algorithm is that it can be implemented without training phase. Having said this, it is useful to precompute the correlation coefficients and keep them in the memory, because this decreases prediction time.

4.2 A least squares approach (NB-LS-BIN)

One may say that the definitions of $\mathcal{N}_{ui}$, $w(u,i,j)$, and $p(u,i,j)$ are too complicated in NB-CORR. It would be interesting to see what happens if simpler definitions are used. Let $\mathcal{N}_{ui} = \mathcal{T}_u$, $w(u,i,j) = |\mathcal{T}_u|^{-0.5}\cdot s_{ij}$, and $p(u,i,j) = 1$. The only thing that has to be specified is s_{ij}.

This model was first introduced by Paterek in [9] under the name *linear model for each item*. He suggested using gradient descent for training, here we propose a different approach.

The goal is to find s_{ij} values such that the sum squared error

$$\begin{aligned}
\text{SSE} &= \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui}-r_{ui})^2\\
&= \sum_{(u,i)\in\mathcal{T}} \left(\left(\frac{1}{\sqrt{|\mathcal{T}_u|}}\sum_{j\in\mathcal{T}_u} s_{ij}\right) - r_{ui}\right)^2
\end{aligned}$$

is minimal.

Now we show that minimizing the SSE is equivalent to solving M linear least squares problems. Recall that the set of users who rated the i-th item is denoted by $\mathcal{T}^{(i)}$. Order the elements of $\mathcal{T}^{(i)}$ and denote the k-th element by $t_k^{(i)}$ ($k = 1,\ldots,|\mathcal{T}^{(i)}|$). Define $\mathbf{B}_i$ as a $|\mathcal{T}^{(i)}|\times M$ matrix and $\mathbf{r}_i$ as a $|\mathcal{T}^{(i)}|\times 1$ column vector such that

$$\begin{aligned}
[\mathbf{B}_i]_{kj} &= \begin{cases} |\mathcal{T}_v|^{-0.5} & \text{if } v \in \mathcal{T}^{(j)},\\ 0 & \text{otherwise,}\end{cases}\\
[\mathbf{r}_i]_k &= r_{vi},
\end{aligned}$$

where $v = t_k^{(i)}$. Denote the column vector $(s_{i1},\ldots,s_{iM})^T$ by $\mathbf{s}_i$. Using the above definitions the SSE can be written as

$$\text{SSE} = \sum_{i=1}^{M} \|\mathbf{B}_i\mathbf{s}_i - \mathbf{r}_i\|^2,$$

therefore the minimization of the SSE is equivalent to solving M linear least squares problems.

If the i-th item has only a few ratings, then minimizing $|\mathbf{B}_i\mathbf{s}_i - \mathbf{r}_i\|^2$ can lead to poor results because of overfitting. To handle this we modify the objective function by adding a regularization term:

$$\text{SSE}' = \sum_{i=1}^{M} \left(\|\mathbf{B}_i\mathbf{s}_i - \mathbf{r}_i\|^2 + \lambda\|\mathbf{s}_i\|^2\right),$$

where λ is the regularization factor. Note that this modification does not affect the computational complexity of the problem.

It can be shown with differentiation that the minimum of the SSE$'$ is at:

$$\mathbf{s}_i = \left(\mathbf{B}_i^T\mathbf{B}_i + \lambda\mathbf{I}_M\right)^{-1}\left(\mathbf{B}_i^T\mathbf{r}_i\right), \qquad (13)$$

where $\mathbf{I}_M$ is the $M\times M$ identity matrix.

Unfortunately, calculating (13) for all i is computationally expensive: we have to invert an $M\times M$ matrix for each item which is $O(M^4)$ operations in total[4]. Therefore we also present a faster approximate solution.

Let us define the $N\times M$ matrix $\mathbf{B}$ as:

$$[\mathbf{B}]_{ui} = \begin{cases} |\mathcal{T}_u|^{-0.5} & \text{if } (u,i)\in\mathcal{T},\\ 0 & \text{otherwise.}\end{cases}$$

The approximation of (13) is

$$\mathbf{s}_i = \alpha\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{I}_M\right)^{-1}\left(\mathbf{B}_i^T\mathbf{r}_i\right), \qquad (14)$$

where α is called the amplification factor. Calculating (14) for all i needs only $O(M^3)$ operations, because it is enough to compute the inverse only once and then perform a matrix-vector multiplication for each item.

[4]We neglected the cost of computing $\left(\mathbf{B}_i^T\mathbf{B}_i + \lambda\mathbf{I}_M\right)$ and $\left(\mathbf{B}_i^T\mathbf{r}_i\right)$.

5. EXPERIMENTATIONS

5.1 The Netflix Prize Dataset

We evaluated our algorithm against the Netflix Prize (NP) dataset, since currently this is the most challenging problem for the collaborative filtering and recommender system community, and our work was greatly motivated by it. Netflix initiated the Netflix Prize contest in order to improve their recommender system—called Cinematch—that provides movie recommendations to customers. The dataset released for the competition is substantially larger than former benchmark datasets, and contains about 100 million ratings from over 480k users on nearly 18k movies. For comparison, the well-known EachMovie dataset[5] only consists of about 2.8m ratings of 73k users and 1628 movies.

In the Netflix Prize dataset, a rating record is a quadruple (u, i, r_{ui}, d_{ui}) representing that user u rated item i as r_{ui} on date $d_{ui} \in \mathcal{D}$, where $\mathcal{D}$ is the ordered set of possible dates. The ratings r_{ui} are integers from 1 to 5, where 1 is the worst, and 5 is the best. The data were collected between October, 1998 and December, 2005 and reflect the distribution of all ratings received by Netflix during this period [4]. The collected data was released in a train-test setting. A hold-out set was created from the most recent ratings of the users, consisted of about 4.2 million ratings. Earlier ratings form the Training set. The Hold-out set were split randomly with equal probability into three subsets of equal size: Quiz, Test and Probe. The Probe set was added to the Training set and was released with ratings. The ratings of the Quiz and Test sets were withheld as a Qualifying set to evaluate competitors. The Quiz/Test split of the Qualifying set is unknown for the public. We remark that the date based division of the entire NP dataset into train-test sets reflects the original aim of recommender systems, which is the prediction of future interest of users from their past ratings/activities.

As the aim of the competition is to improve the prediction accuracy of user ratings, Netflix adopted RMSE as evaluation measure. The goal of the competition is to reduce by at least 10 percent the RMSE on the Test set, relative to the RMSE achieved by Cinematch.[6] The contestants have to submit prediction for the Qualifying set. The organizers return the RMSE of the submission on the Quiz set, which is also reported on a public leaderboard.[7] Note that the RMSE on Test set is withheld by Netflix.

In this experimentation section we evaluate the presented methods on a randomly selected, but fixed 10% subset of the Probe set, which we term as Probe10. [8] We added the rest of the Probe set to the training set. Unless we explicitly mention, from now on the RMSE values refer to the Probe10 RMSE. We have decided to report on RMSE values measured on the Probe10 set, since in our experiments the Probe10 RMSE are significantly closer to the Quiz RMSE than Probe RMSE, and Quiz RMSE tell us more about the accuracy of the predictor, since it excludes the impact of overtraining. On the other hand the rules of NP competition allows only 1 submission daily, which limits the number of the Quiz RMSE calculation drastically. We remark that we measured a maximum 0.0003 difference between the Probe10 and the Quiz RMSE values, while this was of an order of magnitude larger for the Probe set. This advantageous property nominates the Probe10 set for being a standard and Netflix-independent evaluation set for predictors trained on the NP dataset.

[5]It used to be available upon request from Compaq, but in 2004 the proprietary retired the dataset, and since then it is no longer available for download.

[6]The first team achieving the 10 percent improvement is promised to be awarded by a Grand Prize of $1 million by Netflix. Not surprisingly, this prospective award drawn much interest towards the competition. So far, more than 3 000 teams submitted entries for the competition.

[7]`http://www.netflixprize.com/leaderboard`

[8]A Perl script has been made available at our homepage, which selects the Probe10 from the original Netflix Probe set to ensure repeatability and comparability.

5.2 Results of NB-CORR and NB-LS-BIN

In the experiments similarities were calculated only between the 6000 items with most ratings. Most of the queries involve these items. For other items the prediction was made by a baseline method (50% user average, 50% item average). NB-CORR was run on the original while NB-LS-BIN on the double centered ratings matrix.

Model	Parameters	RMSE
NB-CORR	$K = 15, \alpha = 2,$ $\beta = 500, \gamma = 0.15$	0.9280
NB-LS-BIN	$\alpha = 3, \lambda = 4000$	0.9605

Table 1: Probe10 RMSE of neighbor based methods.

5.3 Results of Matrix Factorization

We recall that we applied linear combination of methods for blending. For the matrix factorization methods, we ordered the training examples user-wise and then by date.

To initialize the $\mathbf{P}$ and $\mathbf{Q}$ matrices, we use continuous uniform distribution with parameters $w^{\underline{p}}$ and $w^{\overline{p}}$ for $\mathbf{P}$, and $w^{\underline{q}}$ and $w^{\overline{q}}$ for $\mathbf{Q}$.

We introduce a new meta-parameter, G, which we term as global offset. If not mentioned, we assume $G = 0$. Otherwise we subtract this number from each rating before learning, and add back after prediction.

5.3.1 Comparing RISMF and BRISMF

We compare:

- a RISMF, which we name briefly as RISMF#0, with the following parameter settings: $K = 40, \eta = 0.01, \lambda = 0.01, w^{\underline{p}} = -w^{\overline{p}} = w^{\underline{q}} = -w^{\overline{q}} = -0.01$
- a BRISMF, which we name briefly as BRISMF#0, with the following parameter settings: $K = 40, \eta = 0.01, \lambda = 0.01, w^{\underline{p}} = -w^{\overline{p}} = w^{\underline{q}} = -w^{\overline{q}} = -0.01$

RISMF#0 reaches its optimal RMSE in the 13th epoch: 0.9214, while these numbers for BRISMF#0 are: 10th and 0.9113, which is a 0.0101 improvement.

We can obtain even better results if we set η to 0.007 and λ to 0.005: Probe10 RMSE is 0.9056 (after 10 epochs), which is a 0.0057 improvement. We refer to this MF in the following as BRISMF#1. The running time for this MF is only 14 minutes! Note that running time depends only on K, and on the optimal number of epochs, it is linear in both variables.

5.3.2 Accurate MFs

We applied parameter optimization to get accurate MFs. Also, we applied the "trial and error" method to get manually parameterized accurate MFs. Unless not mentioned otherwise, the parameter settings of the following MFs are found by parameter optimization. Here we describe some results of both:

- BRISMF#800: manually parameterized MF, with 800 features. Parameter are set to: $K = 800$, $w^{\underline{p}} = -w^{\overline{p}} = w^{\underline{q}} = -w^{\overline{q}} = -0.005, \eta^p = \eta^{pb} = 0.016, \eta^q = \eta^{qb} = 0.005, \lambda^p = \lambda^q = 0.010, \lambda^{pb} = \lambda^{qb} = 0, G = 3.6043$. After 9 epochs, learning rates are multiplied by 0.01, and the model is trained for another 2 epochs.
- MLMF#200: a BRISMF with 200 features.
- MLMF#80: a BRISMF with 80 features.
- SBRISMF#1000: an SBRISMF with 1000 features.
- SBRISMF#2000: an SBRISMF with 2000 features.
- SBRISMF#4000: the same as SBRISMF#2000, but $K = 4000$.

See Table 2 for the RMSE values of the each method and their blended versions.

#	Model	RMSE
1	BRISMF#800	0.8940
2	MLMF#200	0.9112
3	MLMF#80	0.9251
4	SBRISMF#1000	0.8905
5	SBRISMF#2000	0.8901
6	SBRISMF#4000	0.8895
7	NB-CORR	0.9280
8	NB-LS-BIN	0.9605
10	1+2	0.8928
11	1+2+3	0.8921
12	1+2+3+4	0.8887
13	1+2+...+5	0.8880
14	1+2+...+6	0.8877
15	1+2+...+7	0.8876
16	1+2+...+8	0.8873

Table 2: Probe10 RMSE of MFs, NB-CORR and NB-LS-BIN

5.4 RMSE values reported by other authors

We compare the presented Probe10 RMSE values (which differ from Quiz RMSE values at most by 0.0003) with other RMSE values reported for the Netflix Prize dataset.

Authors often report Probe RMSE or Quiz RMSE values.

In [1], a detailed description of their alternating least squares approach proposed to matrix factorization is provided. They report on Probe RMSE=0.9167 for their positive MF, and their methods need to run the specialized quadratic optimizer for $(17770+480189)\cdot n^*$ times, where n^* is "few tens". The computational complexity for the specialized quadratic optimizer is approx. $\mathcal{O}(K^3)$ for K features. They report on Probe RMSE=0.9071 and Quiz RMSE=0.8982 for their neighbor method executed on the residuals of their MF.

The best results are reported by [2], where they briefly describe their approach for the Netflix Progress Prize 2007. The MF methods described in that paper have the same computational complexity as mentioned above ([1]). They report only on Quiz RMSE values. Here we summarize the best results of their paper:

- Best stand-alone positive MF: Quiz RMSE=0.8998
- Best stand-alone positive MF with "adaptive user factors": Quiz RMSE=0.8955
- Best neighbor method on positive MF: Quiz RMSE=0.8953

Table 3 compares our best methods with the results reported by [2].

Method	Name of our method	Our Probe10	Our Quiz	Bell et al's Quiz
Simple MF	BRISMF #1000	0.8938	0.8939	0.8998
Tweaked MF	SBRISMF #4000	0.8895	0.8893	N/A
Neighbor on residuals of MF	N/A	N/A	N/A	0.8953

Table 3: Comparison of our best Probe10 and Quiz RMSE values against the Quiz RMSE values reported by [2]

Clearly, our matrix factorization methods outperform Bell et al.'s MF methods. Additionally, our methods are much faster.

6. CONCLUSIONS

This paper surveyed some of our approaches for collaborative filtering. We presented several MF and NB methods. We evaluated our algorithms against the Netflix Prize dataset. We showed that linear combination of various methods can significantly improve the accuracy of the blended solution. We showed that they compare favorably with existing methods in terms of prediction accuracy measured by RMSE, and time complexity. The experiments prove that the proposed methods are scalable to large recommender systems having with hundreds of millions of ratings.

7. REFERENCES

[1] R. M. Bell and Y. Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *Proc of. ICDM, IEEE International Conference on Data Mining*, 2007.

[2] R. M. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize. Technical report, AT&T Labs Research, 2007. `http://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf`.

[3] J. Bennett, C. Eklan, B. Liu, P. Smyth, and D. Tikk. KDD Cup and Workshop 2007. *ACM SIGKDD Explorations Newsletter*, 9(2):51–52, 2007.

[4] J. Bennett and S. Lanning. The Netflix Prize. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 3–6, San Jose, CA, USA, 2007.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI'98, 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan-Kaufmann, 1998.

[6] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of SIGIR'02: 25th ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 238–245, Tampere, Finland, 2002. ACM Press.

[7] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[8] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[9] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, San Jose, CA, USA, 2007.

[10] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI'00: 16th Conf. on Uncertainty in Artificial Intelligence*, pages 473–480, Stanford, CA, USA, 2000. Morgan Kaufmann.

[11] A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl. ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm. In *Proc. of WebKDD'06: KDD Workshop on Web Mining and Web Usage Analysis, at 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, 2006.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proc. of CSCW'94, ACM Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, United States, 1994. ACM Press.

[13] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

[14] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc. of ICML'07, the 24th Int. Conf. on Machine Learning*, pages 791–798, Corvallis, OR, USA, 2007.

[15] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system—a case study. In *Proc. of WebKDD'00: Web Mining for E-Commerce Workshop, at 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, MA, USA, 2000.

[16] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of WWW'01: 10th Int. Conf. on World Wide Web*, pages 285–295, Hong Kong, 2001. ACM Press.

[17] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. *Advances in Neural Information Processing Systems*, 17, 2005.

[18] G. Takács, I. Pilászy, B. Németh, and D. Tikk. On the Gravity recommendation system. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 22–30, San Jose, CA, USA, 2007.

[19] M. G. Vozalis and K. G. Margaritis. Using SVD and demographic data for the enhancement of generalized Collaborative Filtering. *Information Sciences*, 177(15):3017–3037, 2007.