A Unified Approach of Factor Models and Neighbor Based Methods for Large Recommender Systems

Gábor Takács*
Széchenyi István University
Egyetem tér 1.
Győr, Hungary
gtakacs@sze.hu

István Pilászy, Bottyán Németh, Domonkos Tikk†
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
{pila,bottyan,tikk}@tmit.bme.hu

Abstract

Matrix factorization (MF) based approaches have proven to be efficient for rating-based recommendation systems. In this paper, we propose a hybrid approach that alloys an improved MF and the so-called NSVD1 approach, resulting in a very accurate factor model. After that, we propose a unification of factor models and neighbor based approaches, which further improves the performance. The approaches are evaluated on the Netflix Prize dataset, and they provide very low RMSE, and favorable running time. Our best solution presented here with Quiz RMSE 0.8851 outperforms all published single methods in the literature.

1. Introduction

Recommender systems give personalized recommendations on items to users. Such systems help the user in selecting/purchasing items from an overwhelming set of choices, typical in e-commerce, information filtering, etc. 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.

With the growing significance of e-commerce, an increasing number of web-based merchant and rental services use recommender systems. The importance of a good recommender system was recognized by Netflix, which led to the announcement of the Netflix Prize (NP) competition in 2006. The participants of contest aim to create such a recommender system that outperforms Netflix's Cinematch by a given threshold on the fixed NP dataset. This competition motivated our present work as well.

The collaborative filtering (CF) approach makes use of user activities of the past (transaction history or user ratings) to model user preferences. Here, we focus on rating-based recommendation, when users express their opinion over items by means of ratings from a discrete or continuous numerical scale.

Matrix factorization (MF) based models are among the most efficient and accurate approaches in recommender systems (see Section 1.1). The main contribution of this work is a hybrid MF-NSVD1 based approach. We also describe a neighbor based correction of the hybrid model, which is a unification of factor models and neighbor based approaches. The proposed methods are evaluated on the Netflix Prize dataset, but this does not limit their applicability to this specific data. We point out that our hybrid solutions are very accurate and they produce lower RMSE error than other single models in the literature.

1.1. Related work

The history of the field of CF dates back to the early 1990s. Since then many CF algorithms have been proposed that approach the problem by different techniques, including neighborhood based approaches [9], and various matrix factorization techniques [6, 7, 8, 10], etc.

The NP competition boosted the interest in CF, and yielded a number of related publications, many of them proposed various MF approaches and neighbor based methods. Bell and Koren presented an improved neighborhood based approach in [2], which removes the global effect from the data—can be considered as normalization—to improve the accuracy of similarity based interpolative predictions. They also reported on various MF techniques (see e.g. [1, 3]) using alternate least squares at weight updates. Paterek applied successfully various matrix factorization techniques [7] by adding biases to the regularized MF, postprocessing

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

[†] Domonkos Tikk was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Science.

978-1-4244-2624-9/08/$25.00 ©2008 IEEE

the residual of MF with kernel ridge regression, using a separate linear model for each movie, and by decreasing the parameters in regularized MFs. The latter method was named as NSVD1 by Paterek.

We present in this paper an MF method which has many important characteristics: simultaneous feature training, regularization, bias features, early stopping criteria, and incremental gradient descent training algorithm. Though numerous already published MF methods manifest some of these characteristics, we present here an MF that has all of them, and as we will show, this greatly improves accuracy.

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* (N is the number of users),
- I taking values from $\{1, \dots, M\}$ is the *item identifier* (M is the number of items), and
- R taking values from $\mathcal{X} \subset \mathbb{R}$ is the rating value. Typical rating values can be binary or integers from a given range.

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 (RMSE) of the estimate,

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

is minimal (superscript "hat" denotes the estimate of the given quantity, that is, $\hat{R}$ is the estimate 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. 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 $r_{ui} : (u, i) \in \mathcal{T}$.

The goal of this CF setup is to create such predictors that aims at minimizing the error (1). In practice, we cannot measure the error because the distribution of (U, I, R) is unknown, but we can estimate the error on a validation set, $\mathcal{V}'$, that is disjoint from the training set $\mathcal{T}'$. Assuming that $\mathcal{T}'$ and $\mathcal{V}'$ are generated from the same distribution the estimate of RMSE can be calculated as

$$\widehat{\text{RMSE}} = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{(u,i) \in \mathcal{V}} (\hat{r}_{ui} - r_{ui})^2}. \tag{2}$$

For better readability, from now on we omit the "hat" from the RMSE.

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 methods

This section exploits our previous work [11] to introduce matrix factorization. We there discussed the basis of matrix factorization models, the Regularized MF, and the incorporation of constant values in the matrices. The goal of MF techniques is to approximate $\mathbf{R}$ as a product of two much smaller matrices: $\mathbf{R} \approx \mathbf{PQ}$, where $\mathbf{P}$ is an $N \times K$ and $\mathbf{Q}$ is a $K \times M$ matrix.

In the case of the given problem, $\mathbf{R}$ has many unknown elements which cannot be treated as zero. For this case, the approximation task can be defined as follows. Let $\mathbf{P} \in \mathbb{R}^{N \times K}$ and $\mathbf{Q} \in \mathbb{R}^{K \times M}$. Let p_{uk} denote the elements of $\mathbf{P}$, q_{ki} the elements of $\mathbf{Q}$, $\mathbf{p}_u^T$ the u-th row of $\mathbf{P}$, and $\mathbf{q}_i$ the i-th column of $\mathbf{Q}$. We also refer to p_{uk} and q_{ki} ($k \in \{1, \dots K\}$) as the features of the u-th user and the i-th item resp. With this denotation, $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$. The goal is to find $\mathbf{P}$ and $\mathbf{Q}$ such that training RMSE (2) is minimized; which is equivalent to minimizing SSE (sum squared error).

$$\text{SSE} = \sum_{(u,i) \in \mathcal{T}} e_{ui}^2, \quad \text{where } e_{ui} = \hat{r}_{ui} - r_{ui}$$

A good approximation (when SSE is low) may overfit, thus we apply regularization by penalizing the magnitude of weights:

$$e'_{ui} = \frac{1}{2} \left(e_{ui}^2 + \lambda^{(p)} \mathbf{p}_u^T \mathbf{p}_u + \lambda^{(q)} \mathbf{q}_i^T \mathbf{q}_i \right) \tag{3}$$

$$\text{SSE}' = \sum_{(u,i) \in \mathcal{T}} e'_{ui}, \tag{4}$$

where $\lambda^{(p)}, \lambda^{(q)} \geq 0$ are small constants. In order to minimize SSE$'$, we have applied a simple incremental gradient descent method to find a local minimum. Suppose we are at the (u, i)-th training example, r_{ui} and $\hat{r}_{ui}$ are given. We compute the gradient of e'_{ui}:

$$\frac{\partial e'_{ui}}{\partial \mathbf{p}_u} = e_{ui} \cdot \mathbf{q}_i + \lambda^{(p)} \cdot \mathbf{p}_u, \;\; \frac{\partial e'_{ui}}{\partial \mathbf{q}_i} = e_{ui} \cdot \mathbf{p}_u + \lambda^{(q)} \cdot \mathbf{q}_i. \tag{5}$$

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

$$\mathbf{p}'_u = \mathbf{p}_u - \eta^{(p)} \cdot \frac{\partial e'_{ui}}{\partial \mathbf{p}_u}, \ \mathbf{q}'_i = \mathbf{q}_i - \eta^{(q)} \cdot \frac{\partial e'_{ui}}{\partial \mathbf{q}_i}, \qquad (6)$$

where $\eta^{(p)}$ and $\eta^{(q)}$ are the learning rates. Note that all features of the u-th user and the i-th item are updated simultaneously.

One epoch iterates over each $(u, i) \in \mathcal{T}$. After each epoch, we calculate RMSE over $\mathcal{V}'$, and stop the learning when this value starts to increase. Thus, we use both regularization (weight decay) and early stopping to avoid overfitting. We refer to this method as Regularized Incremental Simultaneous MF, or shortly, as RISMF.

3.1. BRISMF

We found a simple way to boost the performance of RISMF, by fixing p_{u1} and q_{2i} to the constant value of 1. Under the expression "fixing to a constant value" we mean not to apply (6) when updating p_{u1} and q_{2i}, and initialize them with the constant value of 1. The pair for these features (q_{1i} and p_{u2}) can serve as a bias feature. This simple extension speeds up the training phase and yields a more accurate model with better generalization performance. We refer to this method as BRISMF that stands for Biased Regularized Incremental Simultaneous MF. In [11] the insertion of constant values into $\mathbf{P}$ and $\mathbf{Q}$ was proposed. In BRISMF, all the inserted values are 1-s. We remark that the bias feature idea was mentioned also by Paterek in [7],

4. Efficient training algorithm for NSVD1

In [7] Paterek introduced an interesting asymmetric factor model, called NSVD1. The free parameters of this model are two sets of item feature vectors ($\mathbf{q}_1, \ldots, \mathbf{q}_M$, $\mathbf{w}_1, \ldots, \mathbf{w}_M \in \mathbb{R}^K$), user biases ($b_1, \ldots, b_N \in \mathbb{R}$), and item biases ($c_1, \ldots, c_M \in \mathbb{R}$). The prediction of the (u, i)-th rating is calculated as the following[1]:

$$\hat{r}_{ui} = b_u + c_i + \left(|\mathcal{T}_u|^{-0.5} \sum_{j \in \mathcal{T}_u} \mathbf{w}_j \right)^T \mathbf{q}_i, \qquad (7)$$

where $\mathcal{T}_u$ is the set of movies rated by user u.

Paterek does not give any details about the training algorithm. He just states that the parameters can be trained by gradient descent. However, the naive implementation of gradient descent is very ineffective for this model. If all $\mathbf{w}_j$-s are updated at each training example, then the training will be very slow.

Here we propose an effective algorithm for training NSVD1. The trick is that for each user we introduce the variable $\mathbf{p}_u = |\mathcal{T}_u|^{-0.5} \sum_{j \in \mathcal{T}_u} \mathbf{w}_j$. We iterate over the training examples user-wise. At the (u, i)-th example we update $\mathbf{p}_u$, $\mathbf{q}_i$, b_u, and c_i. After we have finished processing the ratings of the u-th user, we distribute the change of $\mathbf{p}_u$ among the $\mathbf{w}_j$-s. The deatails can be seen in Algorithm 1. Note that the proposed algorithm produces the exactly same model as the naive implementation of gradient descent[2]. The only difference is running time.

```
Create random model.
for $u \leftarrow 1$ to $N$ do
    $\mathbf{p}_u \leftarrow |\mathcal{T}_u|^{-0.5} \sum_{j \in \mathcal{T}_u} \mathbf{w}_j$
    $\mathbf{p}'_u \leftarrow \mathbf{p}_u$
    for $i \in \mathcal{T}_u$ do
        $e_{ui} \leftarrow b_u + c_i + \mathbf{p}_u^T \mathbf{q}_i - r_{ij}$
        $\mathbf{p}''_u \leftarrow \mathbf{p}_u$
        $\mathbf{p}_u \leftarrow \mathbf{p}_u - \eta e_{ui}(\mathbf{q}_i + \lambda \mathbf{p}_u)$
        $\mathbf{q}_i \leftarrow \mathbf{q}_i - \eta e_{ui}(\mathbf{p}''_u + \lambda \mathbf{q}_i)$
        $b_u \leftarrow b_u - \eta e_{ui}$
        $c_i \leftarrow c_i - \eta e_{ui}$
    end
    for $j \in \mathcal{T}_u$ do
        $\mathbf{w}_j = \mathbf{w}_j + |\mathcal{T}_u|^{-0.5}(\mathbf{p}_u - \mathbf{p}'_u)$
    end
end
```

Algorithm 1: Training algorithm for NSVD1

5. A hybrid MF-NSVD1 approach

It is known that blending the outputs of different CF models can lead to very accurate results. It is an interesting question whether it is possible to integrate the different approaches together to get a hybrid model. Here we propose a model that effectively hybridizes MF and NSVD1 approaches. The prediction of the (u, i)-th rating is the following:

$$\begin{aligned} \hat{r}_{ui} &= \hat{r}'_{ui} + \hat{r}''_{ui}, \qquad (8) \\ \hat{r}'_{ui} &= \beta \mathbf{p}'^T_u \mathbf{q}'_i, \\ \hat{r}''_{ui} &= b_u + c_i + (1 - \beta) \mathbf{p}''^T_u \mathbf{q}''_i, \\ \mathbf{p}''_u &= |\mathcal{T}_u|^{-0.5} \sum_{j \in \mathcal{T}_u} \mathbf{w}_j, \end{aligned}$$

where $\mathbf{p}'_u$, $\mathbf{q}'_i \in \mathbb{R}^{K_1}$, $\mathbf{w}_j$, $\mathbf{q}''_i \in \mathbb{R}^{K_2}$, $b_u, c_i \in \mathbb{R}$ are the free parameters of the model (where u ranges from 1 to N,

[1] In Paterek's original model the normalization factor is $(|\mathcal{T}_u|+1)^{-0.5}$. We use $|\mathcal{T}_u|^{-0.5}$ for simplicity.

[2] This would not be true, if the normalization factor was $(|\mathcal{T}_u|+1)^{-0.5}$.

i and j range from 1 to M). and $\beta \in \mathbb{R}$, K_1 and K_2 are predefined constants. If $K_1 = 0$, we get the MF model, if $K_2 = 0$, we get the NSVD1 model.

The model can be converted into a matrix-factorization model:

$$\begin{aligned} \hat{r}_{ui} &= \mathbf{p}_u^T \mathbf{q}_i, \text{ where} \qquad (9)\\ \mathbf{p}_u &= (\beta \mathbf{p}'_u;\ b_u;\ 1;\ (1-\beta)\mathbf{p}''_u),\\ \mathbf{q}_i &= (\mathbf{q}'_i;\ 1;\ c_i;\ \mathbf{q}''_i), \end{aligned}$$

where ";" is the vector concatenation operator. Note that (8) and (9) are equivalent.

To learn this model, an efficient training algorithm can be derived from Algorithm 1 and the training algorithm for MF: We iterate over the ratings database per user (like in NSVD1). For each rating, we compute e_{ui}, and update NSVD1-features like in Algorithm 1, assuming error is $(1-\beta)e_{ui}$, and update MF features like in (6), assuming error is βe_{ui}, and also update bias features (b_u and c_i).

5.1. Neighbor based correction of factorization models

Here we propose an integrated scalable scheme for using factorization models (MF, NSVD1 or the hybrid model) and neighbor based (NB) approaches together.

Suppose we are given an existing MF model ($\mathbf{P}$,$\mathbf{Q}$). We showed in (9) that NSVD1 and the hybrid model can be converted into an MF model. If we assume $\mathbf{Q}$ is constant, the model of MF is linear in $\mathbf{P}$. One may argue that linear models are too simple to describe users' preferences. We observed, that for each user, MF gets similar error on similar items. To exploit it, we propose the following correction of the prediction made by MF for the (u,i)-th rating in the test set:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + \gamma \frac{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij} \left(\mathbf{p}_u^T \mathbf{q}_j - r_{uj} \right)}{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij}},$$

where s_{ij} is the similarity between items i and j. The weight of the correction term γ can be optimized via cross-validation.

The similarity s_{ij} can be defined in many different ways. We propose the following that proved to be useful for the Netflix Prize problem [5]:
(S1): Normalized scalar product based similarity.

$$s_{ij} = \left(\frac{\sum_{\ell=1}^K q_{\ell i} q_{\ell j}}{\sqrt{\sum_{\ell=1}^K q_{\ell i}^2} \cdot \sqrt{\sum_{\ell=1}^K q_{\ell j}^2}} \right)^\alpha,$$

where α is the amplification parameter. The value s_{ij} can be calculated in O(K) time.

This model can be seen as an elegant unification of factorization models and NB approaches. It is simple, scalable, and accurate. Its does not require additional training (it builds on an existing factorization model). The prediction consists of two terms: the original term and the correction. The similarities used in the NB term need not to be precomputed and stored, because they can be calculated very efficiently from the factorization model.

If neighbor correction is applied on the hybrid MF-NSVD1 approach, we get an accurate unified method that efficiently combines MF, NSVD1 and neighbor based approaches.

6. Experimental results

The experimentations have been performed on the dataset released by Netflix for NP, which is almost 2 orders of magnitude larger than previously used benchmarks (EachMovie, GroupLens). It contains about 100 million ratings from over 480k users on nearly 18k movies. Netflix also provided the date of the ratings. Netflix provides two validation sets: the Probe set and the Quiz set. The ratings of Quiz set is only known by Netflix, while the Probe set is released with ratings. For more information on the dataset see e.g. [3, 4]. The reported running time refers to tests performed on a 2 GHz Intel Pentium M (Dothan) laptop with 1 GB RAM.

In this experimentation section we evaluate the presented methods on a randomly selected 10% subset of the Probe set[3], which we refer to as Probe10. [4] We mention that we measured only a slight difference between the RMSE values on the two test sets: Probe10 and Quiz[5]. For some selected methods, we also report on Quiz RMSE values.

6.1. Results of matrix factorization approaches

We ordered the training examples user-wise and then by date. By this we model that user's taste change in time, and the most recent taste is the dominant. This coincides with the creation of train-test split of NP dataset. We performed test runs with numerous parameter settings. We report on the actual settings of the parameters at each result. Naming convention for parameters: learning rate and regularization factor for users and movies ($\eta^{(p)}, \eta^{(q)}, \lambda^{(p)}, \lambda^{(q)}$); the corresponding variables of bias features ($\eta^{(pb)}, \eta^{(qb)}, \lambda^{(pb)}, \lambda^{(qb)}$); minimum and maximum weights in the uniform random initialization of $\mathbf{P}$ and $\mathbf{Q}$: $w_{\underline{p}}, w_{\overline{p}}, w_{\underline{q}}, w_{\overline{q}}$; offset to subtract from $\mathbf{R}$ before learning: G.

[3] Probe set: dedicated for testing purpose by Netflix with known ratings

[4] A Perl script is available at our homepage, gravityrd.com, which selects the Probe10 from the original Netflix Probe set to ensure repeatability.

[5] Quiz set: to be predicted data, but only Netflix knows the ratings.

6.1.1 Comparing RISMF and BRISMF

We compare a RISMF and a BRISMF with the same parameters: $K = 40, \eta = \lambda = w_{\overline{p}} = w_{\overline{q}} = -w_{\underline{p}} = -w_{\underline{q}} = 0.01$: RISMF reaches its optimal Probe10 RMSE in the 13th epoch: Probe10 RMSE is 0.9214, while these numbers for BRISMF are: 10th and 0.9113, which is a 0.0101 improvement. We refer to this method as BRISMF#0.

The proper setting of η and λ is crucial: If $\eta = 0.007, \lambda = 0.005$, then Probe10 RMSE is 0.9056 (in the 10th epoch). We refer to this method as BRISMF#1. The running time for this MF is only 14 minutes!

6.1.2 Retraining user features

We have found a simple tweak that boosts the accuracy of MFs: after the best epoch, reset user features, and keep going the learning procedure until it gets the best RMSE on $\mathcal{V}$. We refer to this version of models by appending “UM” to the model name, e.g. BRISMF#1UM.

6.1.3 Parameter optimization

We applied a very simple but effective parameter optimization algorithm. We initialize parameters randomly, then we select a single parameter to optimize. We assign n (typically 2) distinct random values to the parameter, and choose the best performing one by evaluating MF with this parameter setting. This method is performed systematically for all parameters one-by-one.

6.1.4 Models

Here we define models of MF, NSVD1 and hybrid MF-NSVD1 that will be experimented with.

- BRISMF#250: $K = 250$, $w_{\underline{p}} = -0.01$, $w_{\overline{p}} = -0.006$, $w_{\underline{q}} = -0.01$, $w_{\overline{q}} = 0.02$, $\eta^{(p)} = 0.008$, $\eta^{(pb)} = 0.016$, $\eta^{(q)} = 0.015$, $\eta^{(qb)} = 0.007$, $\lambda^{(p)} = 0.048$, $\lambda^{(q)} = 0.008$, $\lambda^{(pb)} = 0.019$, $\lambda^{(qb)} = 0, G = 0$.
- BRISMF#800: manually parameterized MF, with 800 features. Parameters 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. Parameter optimization (PO) were done to improve on blended model. Interestingly, the PO algorithm found some exotic settings that would be very hard to find for an human expert, e.g. $\eta^{(qb)} = 10 \cdot \eta^{(pb)}$.
- MlMF#80: like MlMF#200, but with 80 features.
- NSVD1#80: NSVD1 with 80 features. Parameters: $K = 80$, $G = 3.6043$, $\eta = 0.005$, $\lambda = 0.01$, initially, weights are between -0.0005 and 0.0005. Training requires 15 epoch.
- NSVD1#80’: like NSVD1#80, but after 15 epoch, we change η to 0.0005, and run one more epoch.
- HYBRID1#440: A hybrid NSVD1-MF method. Parameters: $K_1 = 400$, $K_2 = 40$, $\beta = 0.8$, $\eta^{(p)\prime} = 0.024, \eta^{(q)\prime} = \eta^{(p)\prime\prime} = \eta^{(q)\prime\prime} = 0.008, \eta^{(pb)} = 0.024$, $\eta^{(qb)} = 0.008, \lambda^{(p)\prime} = \lambda^{(q)\prime} = 0.01$.

6.1.5 Results

Here we give the results of the mentioned models. See Table 1 for the RMSE values of the each method and their blended versions. We also applied neighbor based correction on the models.

In single models, the “w/o S1” and “with S1” columns represent result of ridge regression of 1 and 2 (respectively) columns on Probe10 data. In the combined results (3+6, 1+2, etc) we also used ridge regression. MF and NSVD1 blends well, as line 9 and 14 of the table indicates. In column “w/o S1” of line 9, linear regression found weights ~0.8 and ~0.2. Note that β is 0.8 in HYBRID1#440: we intended to use the same settings. It is interesting, that the hybrid approach gives much better result than the linear combination of an MF and an NSVD1. This justifies the proposition of our hybrid approach: it is able to capture new aspects of the data that is unable to be captured by a single MF or NSVD1.

The hybrid approach is able to improve on the combination of all the other mentioned methods, see line 16 of Table 1.

The Quiz RMSE for HYBRID1#440UM is 0.8873 (Probe10 RMSE is 0.8871). The Quiz RMSE for HYBRID1#440UM + S1 is 0.8851 (Probe10 RMSE is 0.8845). The running time for this method is 4 hours.

6.2. RMSE values reported by other authors

We compare the presented Probe10 RMSE values (which differ from Quiz RMSE values at most by 0.0006) with other RMSE values reported for the Netflix Prize dataset.

Authors often report on Probe RMSE or Quiz RMSE values. Probe RMSE values are calculated by leaving out the Probe set from the Train set, while Quiz RMSE is often computed by incorporating the Probe data into the training of the predictor. Thus, Probe RMSE is often much lower than Quiz RMSE.

Paterek in [7] introduces the use of bias features and NSVD1. He reports on Quiz RMSE $= 0.9070$ for his biased MF (called there RSVD2), and 0.9039 on his own subset of Probe. His NSVD1 gets RMSE of 0.9312 on his own subset of Probe. The results for our BRISMF#800 are: Probe10 RMSE=0.8940, Quiz RMSE=0.8939.

#	Model	w/o S1	with S1
1	BRISMF#250	0.8954	$0.8925_{(\alpha=8)}$
2	BRISMF#250UM	0.8937	$0.8913_{(\alpha=7)}$
3	BRISMF#800	0.8940	$0.8916_{(\alpha=7)}$
4	MlMF#200	0.9112	$0.9087_{(\alpha=8)}$
5	MlMF#80	0.9251	$0.9104_{(\alpha=9)}$
6	NSVD1#80	0.9344	$0.9037_{(\alpha=4)}$
7	NSVD1#80'	0.9298	$0.9011_{(\alpha=4)}$
8	HYBRID1#440UM	0.8871	$0.8845_{(\alpha=4)}$
9	3+6	0.8910	0.8862
10	1+2	0.8933	0.8904
11	1+2+3	0.8921	0.8894
12	1+2+3+4	0.8915	0.8890
13	1+2+3+4+5	0.8911	0.8881
14	1+2+3+4+5+6	0.8889	0.8844
15	1+2+3+4+5+6+7	0.8882	0.8834
16	1+2+3+4+5+6+7+8	0.8838	0.8789

Table 1. Probe10 RMSE of models without and with S1-correction. We also indicated the optimal value of parameter α.

The best results in the NP literature are reported in [3], where the authors briefly describe their approach for the Netflix Progress Prize 2007. They report only on Quiz RMSE values. Here we summarize the best results of that paper:

- Best stand-alone MF: Quiz RMSE $= 0.8998$
- Best stand-alone MF with "adaptive user factors": Quiz RMSE $= 0.8955$
- Best restricted Boltzmann machine (RBM): Quiz RMSE=0.8998
- Best neighbor method on RBM: Quiz RMSE=0.8888
- Best neighbor method on MF: Quiz RMSE $= 0.8953$

Our hybrid approach (HYBRID1#440UM) gives the following results: Probe10 RMSE is 0.8873, Quiz RMSE is 0.8871). Our unified approach (HYBRID1#440UM + S1) gives: Probe10 RMSE is 0.8845, Quiz RMSE is 0.8851, running time is 4 hours.

Clearly, our matrix factorization method (BRISMF#800) and our hybrid and unified approaches outperform the single methods reported in [3].

7. Conclusions

This paper presented factorization based approaches for rating based collaborative filtering problems. We described an efficient way for training Paterek's asymmetric factor model called NSVD1. We also introduced a hybrid approach that alloys MF and NSVD1, and a unified approach that alloys the hybrid approach (or more generally, factor models) and a neighbor based approach. We pointed out that the hybrid and unified approaches outperform all known single methods (on July 8, 2008) on the Netflix Prize problem dataset, while they are single methods as well.

References

[1] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 7–14, San Jose, CA, USA, 2007.

[2] 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.

[3] 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`.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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, at 6th ACM SIGKDD*, Boston, MA, USA, 2000.

[9] 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.

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

[11] 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.