JMLR: Workshop and Conference Proceedings 13: 269-284
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

Multi-task Learning for Recommender Systems

Xia Ning XNING@CS.UMN.EDU
Computer Science & Engineering Department
University of Minnesota, Twin Cities

George Karypis KARYPIS@CS.UMN.EDU
Computer Science & Engineering Department
University of Minnesota, Twin Cities

Editor: Masashi Sugiyama and Qiang Yang

Abstract

This paper focuses on exploring personalized multi-task learning approaches for collaborative filtering towards the goal of improving the prediction performance of rating prediction systems. These methods first specifically identify a set of users that are closely related to the user under consideration (i.e., *active* user), and then learn multiple rating prediction models simultaneously, one for the active user and one for each of the related users. Such learning for multiple models (tasks) in parallel is implemented by representing all learning instances (users and items) using a coupled user-item representation, and within error-insensitive Support Vector Regression (ϵ-SVR) framework applying multi-task kernel tricks. A comprehensive set of experiments shows that multi-task learning approaches lead to significant performance improvement over conventional alternatives.

Keywords: Collaborative Filtering, Multi-Task Learning

1. Introduction

The emergence and fast growth of E-commerce have significantly changed people's traditional perspective on purchasing products by providing huge amounts of products and detailed product information fast, and thus making online transactions much easier. However, as the number of products conforming to the customers' desires has dramatically increased, the problem becomes how to efficiently and effectively help customers identify the products that best fit their personal tastes. During the recent years, various recommender systems (Herlocker et al. (1999)) have been developed to address this problem, of which different collaborative filtering algorithms are widely used.

Collaborative filtering predicts a user's interest (i.e., rating) on an unseen item based on the historical profiles of that user in relation to the historical profiles of the other users. Conventional neighborhood-based collaborative filtering approaches focus on generating prediction on an unseen item as an aggregate of ratings from similar users or similar items (Sarwar et al. (2001)). However, a notable drawback of these approaches is that they do not take into account relationships that exist among the users' preferences and tastes, which cannot be captured via their user- or item-based similarities. Model-based collaborative-filtering approaches have been developed to address this problem by using various machine-learning techniques to build recommendation models from a collection of ratings (Breese et al. (1998)). However, these approaches tend to model the users or the items independently, and as such they fail to capture relationships that may exist in the combined user-item space. Matrix Factorization (MF) (Koren et al. (2009b)) represents a new trend in collaborative filtering, and it has resulted the state-of-the-art performance particularly for large-scale recommendation problems. A less notable issue with MF is that it does not address a lot user personalization, which limits its power for user experience analysis if applied commercially.

In this paper, we present multi-task learning methods for collaborative filtering. The methods first identify a set of related users for the active user based on his/her rating patterns, and then utilize rating information from all the related users together to build a multi-task regression model for predicting the active user's ratings on unseen items. In this fashion, the model for the active user is not only learned from his/her own preferences and tastes, but also from the knowledge that is transfered from other *related* users. The novelty of our methods is that by using the rating prediction models of a set of related users as the parallel learning tasks, they construct multi-task learning models that are explicitly personalized to each individual user. This significantly improves the performance that can be achieved via transfer learning approaches, as the information that is being learned is now transferred across related learning tasks. The multi-task learning methods induce a high level of model interpretability and it is able to provide explicit information for further analysis on user behavior patterns.

The rest of this paper is organized as follows. In Section 2 we briefly review some related work. In Section 3, we introduce definitions and notations used in the paper. In Section 4, we present multi-task learning model for collaborative filtering. In Section 5, we present the results of our methods on multiple benchmark datasets. In Section 7, we discuss computational considerations. In the end, we present our conclusions.

2. Related Work

Collaborative filtering (CF) has been one of the most successful approaches for building recommender systems. Conventional CF-based recommender systems that have been developed can be broadly classified into two categories. The first, referred to as neighborhood-based methods (Herlocker et al. (1999)), computes the recommendations by analyzing the ratings of a set of similar users or items. The second, referred to as model-based methods (Kitts et al. (2000)), computes the recommendations by first building machine learning models to capture the relations between users and their preferences. A review on early works of traditional collaborative filtering is available in (Breese et al. (1998)), whereas a recent review on up-to-date collaborative filtering methods can be found in (Adomavicius and Tuzhilin (2005)). In recent years, Matrix Factorization (MF) has gained great popularity and it has achieved the state-of-the-art performance particularly on large-scale recommendation tasks. A review on MF for recommender systems is available in (Koren et al. (2009b)). In Section 7 we will discuss matrix factorization in more details.

Multi-task learning (Caruana (1993)) is a transfer learning mechanism designed to improve the generalization performance of a given model by leveraging the domain-specific information contained in the training signals of related tasks. In multi-task learning, multiple related tasks are represented by a common representation, and then they are learned in parallel, such that information from one task can be transfered to another task and help achieve performance boost. In recent years, many sophisticated multi-task learning methods have emerged, which include task clustering (Thrun and O'Sullivan (1996)), and matrix regularization (Agarwal et al. (2008)), *etc*. Various studies have reported promising results with the use of multi-task learning for different problems in face recognition (Jin and Sun (2008)), and text mining (Keller and Bengio (2006)), *etc*.

A number of different CF-based rating prediction methods have been developed that directly or indirectly combine the prediction models of different users and as such can be considered as instances of multi-task learning. An example is Yu *et al.* (Yu et al. (2006)), in which they assume that the ratings from each user are generated from a latent function that is derived from a common Gaussian process prior. They formulate the rating prediction problem for each user as an ordinal regression problem. They simultaneously learn the regression models for all the users iteratively by first obtaining a multivariate Gaussian on each latent function through expectation propagation, and then collaboratively updating the Gaussian process prior based on all latent functions. Similarly, in Yu (Yu and Tresp (2005)), they formulate the collaborative filtering as a maximum-margin matrix approximation problem with regularization. They prove that under certain circumstances, such a formulation is equivalent to a kernel learning problem similar to (Yu et al. (2006)), in which each user is modeled by a prediction function, and the common structure of all prediction functions is modeled by a kernel. By doing matrix factorization on the entire user-item rating matrix, all such prediction functions are essentially learned simultaneously. The close relationship between maximum-margin matrix factorization for collaborative filtering and multi-task learning has also been observed and studied in (Argyriou et al. (2008)).

Another similar approach was developed by Abernethy *et al.* (Abernethy et al. (2009)), in which they formulate the rating prediction problem as a linear map from user Hilbert space to item Hilbert space by an operator, and they solve the problem by looking for the best operator that can achieve such a map with minimum loss and penalty. They argue that if only users (or items) have attributes available, and a certain rank penalty is applied so as to enforce information sharing across prediction operators, the operator estimation formulation results in a multi-task learning method based on low-rank representation of the prediction functions.

A different methodology was developed by Phuong *et al.* (Phuong and Phuong (2008)), in which they formulate the rating prediction problem as binary classification (i.e., like or dislike). They build one classifier for each user, and all the classifiers are learned and boosted together. The classification algorithm fits a decision stump for each user on his/her weighted ratings, whereas the boosting algorithm selects a stump for each user that gives the lowest overall error for a subset of classifiers (corresponding to a certain class), and then updates weights on items correspondingly until best performance is achieved.

3. Definitions and Notations

In this paper, the symbols u and i will be used to denote the users and items, respectively, Individual users and items will be denoted using different subscripts (i.e., u_i, i_j). The user for which a prediction needs to be computed will be referred to as the *active* user and will be denoted by u_*. Similarly, the item whose rating needs to be predicted will be referred to as the *active* item and will be denoted by i_*. The rating of user u_i on item i_j will be denoted by $r_{i,j}$. The set of items rated by user u_i will be denoted by $\mathcal{I}_i$, and the set of users that have rated i_j will be denoted by $\mathcal{U}_j$. The average rating of user u_i on items $\mathcal{I}_i$ is then denoted by $\bar{r}_{i,\cdot}$. The set of all users and items in the system will be denoted by $\mathcal{U}$ and $\mathcal{I}$, respectively, and the entire set of ratings will be represented by a user-item rating matrix $\mathbb{R}$, whose (i, j) entry stores the rating of user u_i on item i_j (i.e., $r_{i,j}$), if user u_i has ever rated item i_j, otherwise the entry is marked as empty. The size of that matrix is $|\mathcal{U}| \times |\mathcal{I}|$. For a certain active user u_*, the identified set of related users (described in Section 4) is denoted by $\mathcal{N}_*$. The set of items rated by a set of users $\mathcal{N}$ is denoted by $\mathcal{I}_{\mathcal{N}}$.

4. Multi-Task Learning for Collaborative Filtering

A characteristic of existing multi-task learning formulations for collaborative filtering is that they treat the problem of learning the CF-based rating prediction models for the individual users as different learning tasks and they build a *single* multi-task prediction model that combines all these learning tasks. A direct consequence of that is that from the point of view of a specific user (u_i), the resulting multi-task model will be influenced both from the learning tasks that correspond to users that exhibit similar rating patterns with u_i (i.e., *related* tasks) and from the learning tasks that correspond to users that exhibit highly dissimilar and uncorrelated rating patterns with u_i (i.e., *unrelated* tasks). This goes against the transfer learning principle that underlies multi-task learning approaches (Thrun and O'Sullivan (1996)), which are designed to leverage information that can be concurrently learned from a set of related tasks.

Motivated by this observation, the multi-task learning approach that we developed for solving the CF-based rating prediction problem does not build a single model for all the users, but instead builds a multi-task model that is explicitly designed for each user. Specifically, for each active user u_*, our approach first identifies m users, $\mathcal{N}_* = \{u_{*_1}, \ldots, u_{*_m}\}$, that are related to u_*, treats their corresponding rating prediction problems as the related learning tasks, and uses them to build a multi-task learning-based rating prediction model for u_* using ϵ-SVR. This model takes into account the ratings from the users in $\{u_*\} \cup \mathcal{N}_*$ and does not include any rating information from users that are unrelated to u_*. Moreover, the model is personalized for u_* since models built for different users will tend to have different sets of related users.

The multi-task model of user u_* is learned using error-insensitive Support Vector Regression (ϵ-SVR) (Smola et al. (2003)). The input to the model are tuples of the form

$$((u_i, i_j), r_{i,j})$$

, where $u_i \in \{u_*\} \cup \mathcal{N}_*$, $i_j \in \mathcal{I}_i$, and $r_{i,j}$ is the rating that u_i has given to i_j. The user-item tuples (i.e., (u_i, i_j)) represent the instances on which the model is being learned and their corresponding ratings (i.e., $r_{i,j}$) represent the target values that are estimated by the learned regression function $f(\cdot)$. Once $f(\cdot)$ has been estimated, the prediction from the active user u_* on an unrated item i_* is determined as $f((u_*, i_*))$.

Following the previously developed kernel-based approaches for multi-task learning (Erhan et al. (2006)), the multi-task kernel function $\mathcal{K}_{mt}$ on training instances (i.e., user-item tuples) for support vector regression is defined as

$$\mathcal{K}_{mt}((u_i, i_j), (u_{i'}, i_{j'})) = \mathcal{K}_u(u_i, u_{i'}) \times \mathcal{K}_i(i_j, i_{j'}), \tag{1}$$

where $\mathcal{K}_u$ and $\mathcal{K}_i$ are kernel functions defined on the users and the items, respectively. When $\mathcal{K}_u$ and $\mathcal{K}_i$ are valid kernels (i.e., positive semi-definite), $\mathcal{K}_{mt}$ is a valid kernel as well (Vapnik (1998)).

Both the training instance representation and the multi-task kernel are essential in order to enable multiple task learning in parallel coupling with information transfer. Intuitively, the kernel function $\mathcal{K}_{mt}$ compares two tasks (users) $(u_i, i_j), \forall i_j \in \mathcal{I}_i$ and $(u_{i'}, i_{j'}), \forall i_{j'} \in I_{i'}$ by computing their similarity as the tensor of corresponding user similarity (i.e., $\mathcal{K}_u$) and item similarity (i.e., $\mathcal{K}_i$). In case $u_i = u_{i'}$, the kernel regression minimizes learning errors for only user u_i (i.e, $\mathcal{K}_u$ (u_i, u_i) is constant, and thus single-task learning). In case $i_j = i_{j'}$, the kernel regression transfers user taste on item i_j across user u_i and $u_{i'}$ through $\mathcal{K}_u$ (i.e, $\mathcal{K}_i$ (i_j, i_j) is constant). Note that the above two cases are integrated within one model learning process. In the rest of this section we discuss the methods that we developed for selecting the related users $\mathcal{N}_*$ for the active user u_*, and for computing the $\mathcal{K}_u$ and $\mathcal{K}_i$ functions.

4.1 Selection of Related Users

We developed two different approaches for selecting the users that will be used as the parallel tasks in our multi-task learning framework. Both of them take advantage of the "like-minded users" insight utilized by user-based collaborative filtering approaches, that is, to look for a set of users who rate items in a way that is consistent with the active user's rating behaviors.

4.1.1 Selection of Most Similar Users

The first approach, for a given user u_*, selects the m most related users $\mathcal{N}_* = \{u_{*_1}, u_{*_2}, \ldots, u_{*_m}\}$ as those users whose historical ratings on co-rated items are the most similar to user u_*. The historical rating similarity between two users is computed using the modified version of Pearson correlation coefficient employed by user-based approaches (Herlocker et al. (1999)), which is given by

$$\mathrm{sim}_u(u_i, u_j) = p_u(u_i, u_j) \cdot \frac{\sum_{i_k \in \mathcal{I}_c} (r_{i,k} - \bar{r}_{i,\cdot})(r_{j,k} - \bar{r}_{j,\cdot})}{\sqrt{\sum_{i_k \in \mathcal{I}_c} (r_{i,k} - \bar{r}_{i,\cdot})^2} \sqrt{\sum_{i_k \in \mathcal{I}_c} (r_{j,k} - \bar{r}_{j,\cdot})^2}}, \tag{2}$$

where $\mathcal{I}_c$ is the set of items that have been co-rated by users u_i and u_j (i.e, $\mathcal{I}_c = \mathcal{I}_i \cap \mathcal{I}_j$), $\bar{r}_{i,\cdot}$ and $\bar{r}_{j,\cdot}$ are the average ratings of users u_i and u_j over all their rated items $\mathcal{I}_i$ and $\mathcal{I}_j$, respectively. p_u is a penalty factor that linearly penalizes the similarity value when the number of co-rated items is smaller than a pre-defined small constant C, and p_u is defined as $p_u(u_i, u_j) = \min(|\mathcal{I}_c|, C)/C$. Note that since the Pearson correlation coefficient can be negative, we only select the users whose similarity to the active user u_* is positive. We refer to the above method as $\mathrm{USM}_{\mathrm{sim}}$.

Note that users with consistent negative similarities can also provide meaningful information from an opposite perspective. However, we did not include such users during user selection in order to avoid the complexity during model training (i.e., handling negative signs).

4.1.2 Selection of Most Covering Users

A limitation of $\mathrm{USM}_{\mathrm{sim}}$ is that it does not take into account which items the selected users have provided ratings on. The items that are rated by the selected users but not by the active user are critical in order to generate accurate rating predictions for the active user on his/her unseen items. Note that the user similarity between users u_* and u_i is calculated on the items that are co-rated by u_* and u_i. Consider the following extreme scenario. If the active user u_* and another user u_i have rated a same set of items (i.e., $\mathcal{I}_* = \mathcal{I}_i$) and the corresponding ratings are identical, then their user similarity will be the highest, and thus by $\mathrm{USM}_{\mathrm{sim}}$ user u_i is very likely to be selected. However, such a user u_i does not bring any additional information into model training, and therefore, he/she is not a good one to select.

The above example suggests that there are two aspects that we need to consider when select users. One is how similar the users are with the active user, and this aspect is the focus of $\mathrm{USM}_{\mathrm{sim}}$. The other is how informative the selected users are. We use the number of items that are rated by a certain user u_i but not by the active user to measure u_i's informativeness, and we propose the method which selects users that collectively maximize the coverage of the unrated items for u_*, in addition to the selection of users that rated in a consistent way with u_*. We developed a rather simple greedy approach for combining these two heuristics, referred to as $\mathrm{USM}_{\mathrm{cvg}}$, as Program 1 shows. Along the user list identified by $\mathrm{USM}_{\mathrm{sim}}$ (i.e., $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$), this approach selects the user who have rated maximum number of items that are not rated by both of the active user and the users that are already selected by USM$_{\text{cvg}}$ (line 9). In the user list (i.e., $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$), the users are sorted in descending order based on their user similarity with the active user, In case that fewer users are selected by USM$_{\text{cvg}}$ than required, USM$_{\text{cvg}}$ chooses the most similar users from the rest of the user list (line 26). Note that by USM$_{\text{cvg}}$, we also increase the chances that the active item has already been rated by some selected users.

Algorithm 1: USMcvg

```
Data: $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$, $m$, $\mathcal{I}_*$
Result: $\mathcal{N}_*$
begin
    $\mathcal{N}_* = \{u_{*_1}\}$
    $i = n = 1$
    while $(i < 2m)$ do
        for $j = 1$ to WINDOWS_SIZE do
            $k = i + j$
            if $k > 2m$ then
                goto L
            else
                count the number of items that have been rated by $u_{*_k}$ but not rated by $u_*$ or $\forall u \in \mathcal{N}_*$
        look for $u_{*_{k_{max}}}$ with maximum non-zero count in this window if such user $u_{*_{k_{max}}}$ exists then
            $\mathcal{N}_* = \mathcal{N}_* \cup \{u_{*_{k_{max}}}\}$
            $n = n + 1$
            $i = k_{max}$
        else
            $i = i$+WINDOWS_SIZE
        if $n \geq m$ then
            break
    L: if $n < m$ then
        select $m - n$ most similar users that are not selected into $\mathcal{N}_*$
end
```

4.2 Kernel Functions for Users and Items

In order to construct a valid multi-task kernel $\mathcal{K}_{mt}$ as in Equation 1, a valid user kernel $\mathcal{K}_u$ and a valid item kernel $\mathcal{K}_i$ are required. For the kernel on users, we use the user-similarity function as defined in Equation 2 to construct $\mathcal{K}_u$. Similarly, we use an item-similarity function to construct the kernel $\mathcal{K}_i$ on items.

The item similarity between i_i and i_j is defined based on the adjusted cosine similarity that is used to determine the similarity between items in the context of item-based collaborative filtering algorithms for recommender systems (Sarwar et al. (2001)). It is calculated as follows:

$$\text{sim}_i(i_i, i_j) = p_i(i_i, i_j) \cdot \frac{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,\cdot})(r_{k,j} - \bar{r}_{k,\cdot})}{\sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,\cdot})^2} \sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,j} - \bar{r}_{k,\cdot})^2}}, \tag{3}$$

where $\mathcal{U}_c$ is the set of users that have rated both items i_i and i_j, $\bar{r}_{k,\cdot}$ is the average ratings of user u_k over all the items that he/she rated (i.e, $\mathcal{I}_i$), and p_i is a penalty factor that linearly penalizes the similarity value when the number of co-rating users is smaller than a pre-defined small constant C and is given by $p_i(i_i, i_j) = \min(|\mathcal{U}_c|, C)/C$.

Note that both the modified Pearson correlation coefficient in Equation 2 and the adjusted cosine similarity in Equation 3 are computed only on the co-rated and co-rating items and users, respectively, and then penalized by a factor, they may not necessarily lead to valid kernels (i.e., positive semi-definite) to be used directly as $\mathcal{K}_u$ and $\mathcal{K}_i$, respectively. In such cases, the approach described in Saigo *et al* (Saigo et al. (2004)) is used to convert a symmetric matrix into positive semi-definite by subtracting from the diagonal of the matrix its smallest negative eigenvalue. After converting user similarity matrix and item similarity matrix into valid kernel matrix and used as $\mathcal{K}_u$ and $\mathcal{K}_i$, respectively, $\mathcal{K}_{mt}$ in Equation 1 is a valid kernel.

5. Experimental Methodology

5.1 Datasets

We evaluated the performance of our methods on multiple data-
sets. The first dataset is from MovieLens[1], which contains 100,000 ratings from 943 users on 1,682 movies. Each user rates more than 20 movies. The rating scores fall in the range 1 to 5 as integers. We refer to this dataset as *ML100K*. The second dataset is from MovieRating[2]. It contains 43,865 ratings from 500 users on 1,000 movies. Each user rates more than 11 movies, and the rating scores fall in the range of 1 to 5 as integers. This dataset is referred to as *MR*. The third dataset is a subset extracted from BookCrossing[3] by selecting the users who have no less than 20 ratings. Thus, the extracted dataset contains 217,736 ratings from 3,305 users on 108,385 books. The rating scores fall in range 1 to 10 as integers. This dataset is referred to as *BX*. The next dataset is extracted from Wikilens[4] in which 150 users are selected with more than 20 ratings each, resulting in 2,599 items rated and 11,559 ratings in total. The rating scores range from 0.5 to 5.0 with granularity 0.5. The dataset is referred to as *Wiki*. The fifth dataset is extracted from Yahoo! Music user ratings of songs, provided as part of the Yahoo! Research Alliance Webscope program [5]. Totally 5,000 users are sampled with each having more than 20 ratings, and there are 675,279 ratings on 95,984 items. The rating scores range from 1 to 5 as integers. This dataset is referred to as *Music*. For all the datasets above, higher rating represents higher preferences. Characteristics of the datasets is summarized in Table 1. We did not test the multi-task learning methods on very large datasets like Netflix dataset due to time constrains, but we discuss the scalability problem in Section 7.

5.2 Model Learning

We used the publicly and freely available support vector machine tool SVM^{light} (Joachims (2002)) which implements an efficient soft margin optimization algorithm. We implemented support vector regression using SVM^{light} by specifying the ”-z” option as ”r” (i.e., support vector regression).

1. http://www.grouplens.org/node/73
2. http://www.cs.usyd.edu.au/~irena/movie_data.zip
3. http://www.informatik.uni-freiburg.de/~cziegler/BX/
4. http://www.grouplens.org/node/425
5. http://research.yahoo.com/Academic_Relations

Table 1: Dataset summary

	ML100K[1]	MR[2]	BX[3]	Wiki[4]	Music[5]
# users	943	500	3,305	150	5,000
# items	1,682	1,000	108,385	2,599	95,984
# ratings	100,000	43,865	217,736	11,559	675,279
min $\lvert\mathcal{I}_i\rvert$	20	11	20	20	20
rating range	1-5	1-5	1-10	0.5-5.0	1-5
avg rating	3.53	3.59	7.77	3.52	3.49
sparsity	93.7%	91.2%	99.9%	97.0%	99.9%

In this table, min $|\mathcal{I}_i|$ is the minimum number of ratings per user.

5.3 Evaluation Methodology and Performance Metrics

In the following sections, we will collectively refer to the multi-task learning methods as MTCF. The performance of the methods is evaluated using a five-fold cross validation framework. For each user, all of his/her ratings are randomly split into five equal-sized folds. For the active user u_*, 4 out of his/her 5 folds are used in training, and his/her remaining fold is used for evaluation. For other users, all of their 5 folds are used to compute user similarity with u_*'s 4 training folds and to select items from. This is done in order to maximize the opportunity for the active user to identify the best set of related users and items.

We used Mean Absolute Error (MAE) as the statistical accuracy metric. MAE is a widely used evaluation metric in recommender system community, which is defined as

$$MAE = \frac{\sum_{i,j} |p_{i,j} - r_{i,j}|}{n} \tag{4}$$

where n is the number of predictions that are made from users on items, $p_{i,j}$ is the predicted rating on item i_j from user i (if any), and $r_{i,j}$ is the corresponding true rating. MAE measures the average absolute deviation of recommendations from true user-specified ratings. Lower MAE indicates more accurate recommendations.

6. Results

In this section we present the experimental evaluation of the multi-task learning methods that we developed for building collaborative filtering-based recommender systems.

The experimental evaluation consists of two components in the following order. First, we evaluate the performance of the different methods for selecting related users. Second, we compare the performance achieved by our methods against those achieved by other widely used approaches for collaborative filtering-based recommender systems. The reported running times were obtained on workstations running Linux with Intel Xeon CPUs running at 2.66Ghz.

6.1 Evaluation of the Related User Selection Methods

Table 2 shows the MAE achieved by MTCF for the two related users selection schemes described in Section 4.1. These results show that the performance of the two related user selection schemes is

Table 2: MAE of the methods for selecting related users.

	ML100K			MR			BX		
method	m = 10	m = 30	m = 50	m = 10	m = 30	m = 50	m = 5	m = 10	m = 30
USM_{sim}	0.714	0.712	0.714	0.715	0.709	0.710	1.177	1.175	1.175
USM_{cvg}	0.712	0.712	0.714	0.713	0.709	0.710	1.176	1.174	1.174

	Wiki			Music		
method	m = 5	m = 10	m = 30	m = 10	m = 30	m = 50
USM_{sim}	0.809	0.801	0.802	0.969	0.963	0.964
USM_{cvg}	0.808	0.802	0.802	0.967	0.960	0.961

In this table, *m* is the number of users selected.

very similar across the different datasets and number of related users, although USM_{cvg} is slightly better than USM_{sim}. This may because the information brought by extra users selected from USM_{cvg} has already been covered by users solely from USM_{sim}. Comparing the performance of the MTCF schemes as a function of the number of related users *m*, we see their performance initially improves as *m* increases, but it subsequently degrades as *m* is further increased. However, for some datasets (e.g., ML100K) the performance differences are rather small. Table 3 shows corresponding computational complexities of different user selection methods.

6.2 Evaluation of the Training Instance Selection Methods

The recommendation performance achieved by the schemes that reduce the size of the training set by selecting a smaller subset of the ratings for each related user is shown in Table 4. Comparing the MAE achieved by these methods over those shown in Table 2, we see that by selectively reducing the size of the training set the accuracy of the resulting MTCF models actually improves (except MR).

We believe that these approaches lead to improved performance for two reasons. First, each of the related users may have provided ratings for some items that are not similar with the items rated by the active user. By removing such items and corresponding ratings from model learning, unrelated (i.e., noisy) information is eliminated, resulting in a more accurate model. Second, by restricting the set of ratings from each related user, we also reduce the total number of training instances that involve other users, increasing the importance of the active user's own ratings during the learning, and thus the model learned is more focused on the active user.

Comparing the performance of the two rating selection schemes, we see that ISM_{cvg} tends to perform better than ISM_{max} and that the best overall results are usually obtained when ISM_{cvg} is involved. Moreover, the results of Table 4 illustrate that unlike the earlier experiments in Section 6.1, USM_{cvg} tends to outperform USM_{sim} when the number of ratings is restricted, Also, the performance of the MTCF models can improve as the number of related users increases. This is especially true for the schemes using USM_{cvg} and ISM_{cvg}.

Table 5 shows the average learning time, as well as the average number of support vectors, for MTCF models with different number of selected ratings. Comparing these results with those of Table 3, we see that by reducing the number of training instances, we can achieve significant

Table 3: Computational complexities of different related user selection methods.

metric	method	MK100K			MR			BX		
		m = 10	m = 30	m = 50	m = 10	m = 30	m = 50	m = 5	m = 10	m = 30
t	USM$_{\text{sim}}$	1.551	14.863	69.094	1.208	9.878	28.424	2.028	6.040	12.767
	USM$_{\text{cvg}}$	1.740	14.825	67.804	1.208	10.261	31.205	2.909	6.125	12.425
#sv	USM$_{\text{sim}}$	1188	3491	5887	1040	2984	4939	722	1133	2154
	USM$_{\text{cvg}}$	1247	3640	6024	1092	3067	4988	898	1204	2015

metric	method	Wiki			Music		
		m = 5	m = 10	m = 30	m = 10	m = 30	m = 50
t	USM$_{\text{sim}}$	0.351	1.213	8.608	5.423	42.47	125.32
	USM$_{\text{cvg}}$	0.372	1.322	8.799	5.679	44.47	129.54
#sv	USM$_{\text{sim}}$	616	1143	2823	1710	4720	7012
	USM$_{\text{cvg}}$	636	1193	2873	1739	4743	7026

In this table, *m* is number of related users. t is model learning time in second, and #sv is the number of support vectors in the regression model.

reductions in the amount of time required to train the models, and also the number of support vectors used in the resulting models.

6.3 Comparison with Other Methods

Table 6 compares the performance achieved by the multi-task learning schemes developed in this paper against the performance achieved by some popular methods for building collaborative filtering-based recommender systems. Specifically, the performance of our methods is compared against four different schemes. The first, denoted by "ϵ-SVR", is a regression-based approach using ϵ-SVR, in which a model is learned for each user using *his/her own* ratings as the training instances. The second, denoted by "SVD", is the approach based on singular value decomposition (Sarwar et al. (2001)), which has been shown to be closely related to multi-task learning. The third, denoted by "uKNN", is the standard user-based method described in (Herlocker et al. (1999)). Finally, the fourth, denoted by "iKNN", is the standard item-based approach described in (Sarwar et al. (2001)).

The results in this table show that MTCF achieves the best performance among the different schemes in four out of the five datasets. Its MAE is on the average lower by 1.9%, 9.1%, 11.2%, and 5.4% than that of the ϵ-SVR, SVD, uKNN, and iKNN schemes, respectively. Compared to the second best-performing scheme (ϵ-SVR), its actual performance gains varies across the different datasets. For some of them the gains are rather small (e.g., ML100K and BX), whereas for the rest the gains are substantial (e.g., MR, Wiki and Music).

7. Discussion

7.1 Comparison with Matrix Factorization

Since matrix factorization methods represent the state-of-the-art in recommender systems, due to their high accuracy and suitability to large-scale problems, we compare our methodologies and

Table 4: MAE of the different methods for reducing the number of training instances.

dataset	k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
		$m = 30$	$m = 50$	$m = 70$	$m = 30$	$m = 50$	$m = 70$	$m = 30$	$m = 50$	$m = 70$	$m = 30$	$m = 50$	$m = 70$
ML100K	10	0.711	0.712	0.730	0.709	0.710	0.713	0.706	0.703	0.706	0.710	0.704	0.710
	30	0.707	0.706	0.713	0.709	0.709	0.709	0.705	0.703	0.703	0.710	**0.700**	0.707
	50	0.703	0.704	0.708	0.707	0.704	0.707	0.707	0.701	0.704	0.707	0.702	0.703
		$m = 1$	$m = 5$	$m = 10$	$m = 1$	$m = 5$	$m = 10$	$m = 1$	$m = 5$	$m = 10$	$m = 1$	$m = 5$	$m = 10$
MR	5	0.713	0.723	0.728	0.712	0.714	0.715	0.712	0.724	0.728	0.712	0.711	0.715
	10	0.713	0.727	0.732	0.711	0.718	0.719	0.712	0.728	0.730	**0.710**	0.719	0.718
	30	0.719	0.725	0.724	0.718	0.721	0.720	0.716	0.721	0.726	0.714	0.719	0.720
		$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$
BX	1	1.164	1.164	1.166	**1.161**	1.162	1.164	1.164	1.164	1.166	**1.161**	1.162	1.164
	5	1.167	1.168	1.172	1.165	1.166	1.169	1.167	1.169	1.172	1.164	1.166	1.169
	10	1.171	1.174	1.176	1.169	1.172	1.174	1.171	1.174	1.177	1.169	1.172	1.175
		$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$
Wiki	1	0.806	0.804	0.812	0.806	0.805	0.812	0.790	**0.783**	0.785	0.788	**0.783**	0.785
	5	0.802	0.804	0.809	0.801	0.803	0.809	0.790	0.785	0.791	0.786	0.784	0.790
	10	0.800	0.803	0.806	0.802	0.804	0.805	0.791	0.787	0.794	0.789	0.786	0.793
		$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$	$m = 5$	$m = 10$	$m = 30$
Music	5	0.969	0.963	0.960	0.950	0.945	0.947	0.946	0.947	0.950	0.965	0.959	0.958
	10	0.964	0.959	0.961	0.949	**0.943**	0.948	0.951	0.946	0.951	0.961	0.957	0.959
	30	0.962	0.961	0.963	0.951	0.947	0.952	0.952	0.947	0.953	0.961	0.959	0.962

In this table, m is the number of related users and k is the maximum number of ratings selected from each related user. USM_*/ISM_* is the paired user selection method and item selection method. **Bold** numbers are the best performance of each dataset.

results with MF. We implemented the MF method BRISMF as described in Takács et al. (2009), which gives a typical MF performance. Out of five datasets as described in Table 1, BRISMF outperforms MTCF on three datasets (ML100K: 0.682 vs 0.700, MR: 0.697 vs 0.710, BX: 1.146 vs 1.161, where the first number for each dataset is MAE by BRISMF and the second number is best MAE from MTCF as presented in Table 6). On the other two datasets, MTCF actually slightly outperforms BRISMF (Wiki: 0.789 vs 0.783, Music 1.002 vs 0.943). As it shows, our methods do not outperforman MF significantly. However, we still compare these two categories of methods so as to illustrate their respective values.

The intuition behind MF methods is that users' tastes can be determined and expressed by some latent factors, and users make ratings on items mainly based on the item properties according to such factors. The main idea following this is to look for a factor space with low dimensionality, which optimally describes all user in the system, and meanwhile all items can be mapped into the same space as feature vectors. Regularization is usually applied in order to avoid overfitting. One

Table 5: Computational complexities of different training set reduction schemes.

method	metric	ML100K			metric	MR		
		k = 10	k = 30	k = 50		k = 5	k = 10	k = 30
t	$\text{USM}_{\text{cvg}}/\text{ISM}_{\text{cvg}}$	0.466	1.874	5.394	$\text{USM}_{\text{cvg}}/\text{ISM}_{\text{cvg}}$	0.012	0.012	0.014
#sv	(m = 50)	473	979	1,335	(m = 1)	64	68	82

method	metric	BX			metric	Wiki		
		k = 1	k = 5	k = 10		k = 1	k = 5	k = 10
t	$\text{USM}_{\text{cvg}}/\text{ISM}_{\text{cvg}}$	0.008	0.009	0.009	$\text{USM}_{\text{cvg}}/\text{ISM}_{\text{cvg}}$	0.032	0.084	0.177
#sv	(m = 5)	47	56	68	(m = 10)	159	304	406

method	metric	Music		
		k = 5	k = 10	k = 30
t	$\text{USM}_{\text{sim}}/\text{ISM}_{\text{cvg}}$	0.038	0.052	0.155
#sv	(m = 10)	156	198	366

In this table, k is maximum number of ratings selected for each related user. m is the number of related users. t is model learing time in second. #sv is the number of support vectors.

Table 6: Performance comparison with other methods

dataset	MTCF	ϵ-SVR	SVD	uKNN	iKNN
ML100K	0.700	0.703	0.746	**0.691**	0.705
MR	**0.710**	0.724	0.789	0.730	0.727
BX	**1.161**	1.164	-	1.218	1.199
Wiki	**0.783**	0.816	0.852	0.961	0.944
Music	**0.943**	0.956	-	1.053	0.979

In this table, **bold** numbers are the best performance for each dataset.

problem with such matrix factorization or low-rank approximation, as a compromise to its high accuracy, is that the model itself suffers from poor interpretability, due to the existence of a latent space with unknown structures. Therefore, in practical applications, to reason the predictions and furthermore enhance user experience within MF framework become not easy. This disadvantage of MF has been pointed out and well discussed by Koren (Koren (2010)).

Another problem of MF, which is related to the first one, is that MF is doing global optimization, and thus personalization is to a large extent ignored. Most of MF algorithms optimize a function of prediction errors over the entire system without consideration of the intrinsic difference that naturally exists between user communities. All the users within the system are treated as homogeneous through a relatively small yet same set of factors, and therefore community bias may not be effectively modeled in the user factor space, unless the latent space has a sufficient large dimensionality to necessarily cover user community variance, which in effect goes against the initial principle of MF methodology (i.e., low-rank factorization).

Compared to MF methods, neighborhood-based methods provide a higher level of tractability for user community analysis, as well as a stronger emphasis on user personalization, in compensation to its relatively low accuracy. Considering MTCF as a formal framework, it very naturally formulates user communities through the concept of "*related*" users. Another advantage of MTCF is that it is very handy to incorporate other source of information, once available, into model learning so as to improve recommendation performance. For example, item properties or product specifications can readily be used in neighborhood-based methods, but may not be easily incorporated into MF. The simple, tractable and flexible nature of neighborhood-based methods may still distinguish themselves as a good candidate for real applications of recommender systems.

7.2 Computational Considerations

Computational scalability is a major consideration in recommender systems. In the context of model-based approaches for recommender systems, there are two relevant aspects in terms of computational scalability. The first has to do with the amount of time required to build the models and the second with the amount of time required to compute the predictions.

Since model building is an off-line process, this aspect of the system can tolerate methods that have relatively high computational requirements given that they exhibit reasonable scaling properties as the numbers of users and/or their ratings increase over time. The multi-task learning methods described in Section 4 need to learn a separate model for each user in the system through multiple steps. In our experience, the ϵ-SVR model learning step takes by far the largest amount of time. The amount of time required for updating the ϵ-SVR models will be lower because for many of them, the sets of users involved will remain the same. When new ratings are added, the regression function estimated in the earlier learning cycle can be used as the initial function to be further optimized so as to take into account the updates.

The prediction aspect of recommender systems is often a real- or a near real-time component and as such it is important to be able to compute the predictions fast. The time required for this step depends on the number of user-item tuples that make up the model's support vectors and whether or not the active item was part of the items used in training the model. If the active item was not used in the training step, then the item-to-item similarities between the active item and the items in support vectors need to be computed on the fly. Otherwise, the item-to-item similarities can be directly retrieved from the saved model.

The above discussion suggests two optimizations that can save computational costs for the MTCF system. The first is to reduce the number of models to be learned, which will lower the expense of off-line computations and space. The second is to reduce the number of items used for training.

7.2.1 Reducing the Models Learned

Our approach to reduce the number of models being learned is motivated by the following observation. If two users are among the m related users for each other and their sets of related users overlap significantly, then their multi-task models should be quite similar, due to overlapping training sets, and thus they can be used interchangeably to some extent.

We formalize this idea by using a maximal independent set (MIS) in the user-to-user graph formed during related user selection. Specifically, we construct an undirected user-to-user graph G

in which there is an edge between u_i and u_j if

$$\begin{array}{ll} \text{(i)} & u_i \in \mathcal{N}_j, \text{ and} \\ \text{(ii)} & u_j \in \mathcal{N}_i, \text{ and} \\ \text{(iii)} & |\mathcal{N}_i \cap \mathcal{N}_j|/|\mathcal{N}_i \cup \mathcal{N}_j| \geq \theta, \end{array}$$

where θ is a threshold that measures the minimum overlap between the two sets of related users. Then, using a greedy algorithm, we find a maximal independent set MIS of G, and we learn multi-task models only for the users in MIS.

During prediction, if the active user u_* is part of MIS, its prediction is determined by its own multi-task model. Otherwise, its prediction is determined by other models through two ways. The first simply selects the user $u_i \in$ MIS such that $u_i \in \mathcal{N}_*$ and $|\mathcal{N}_i \cap \mathcal{N}_*|/|\mathcal{N}_i \cup \mathcal{N}_*|$ is the highest among $\mathcal{N}_*$ in MIS, and then computes the prediction using u_i's model. Note that because MIS is maximal, there has to exist such a user u_i that satisfies the above constraint. The second computes the predictions using the models of $\mathcal{N}_*$ within MIS and determines the final rating as the weighted average of these predictions. For the predictions obtained from u_i's model ($u_i \in \mathcal{N}_*$), the weight corresponds to $|\mathcal{N}_i \cap \mathcal{N}_*|/|\mathcal{N}_i \cup \mathcal{N}_*|$ (i.e., the degree of overlap among their sets of related users).

7.2.2 Selection of Item Ratings

We developed two methods to select ratings for the related users. By doing this, we can reduce the number of training instances, and meanwhile further exclude inconsistent rating information. In the first method, for each related user, we calculate the average item similarity between each of his/her rated items and all the rated items of the active user. Based on the average similarity, we select the top k most similar items and corresponding ratings from that related user. Note that the item similarity is calculated from the entire user space (Equation 3), and thus the average item similarity can be dominated by two different subsets of users. The first subset is the selected users. If an item has high average item similarity that mainly comes from the selected users (i.e., other users seldom rated the item), then this item is very specific to the selected users and thus to the multi-task model. Inclusion of such items can provide information on user-specific rating behaviors. The second subset is the other users rather than the selected ones. If an item is similar because many non-selected users rated it in a consistent way, then such items provide background preferences that are not specific to a certain user. By incorporating such items into model training, we can regularize the rating predictions so as to avoid extreme rating behaviors.

The second method, in addition to using the ratings identified as above for each related user, we also use ratings on the items that are not rated by the active user but have been rated by that related user. The motivation of this method is that it is possible the unseen items of the active user are diverse from his/her already rated items, and thus accurate predictions on such unseen items cannot be produced simply from his/her own rated items and their most similar items from his/her related users. By incorporating related users' rated items that are not rated by the active user, we may include those items that are similar to the active user's unrated items into model learning so as to improve prediction accuracy. Note that the similarity between two items is measured using the item-similarity function defined in Equation 3.

Due to space constraints, we do not present the results of MIS-based algorithms. However, the conclusion is the MIS-based methods do not degrade recommendation performance but significantly lower down the running time.

8. Conclusions

In this paper, we presented a model for collaborative filtering based on multi-task learning. This model leverages information from a set of related users and leads to more accurate predictions. The experimental results show that the multi-task model consistently gives better predictions than other existing methods.

Acknowledgments

This work was supported in part by NSF (IIS-0905220, OCI-1048018, IOS-0820730), NIH (RLM008713A). Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

References

Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *J. Mach. Learn. Res.*, 10: 803–826, 2009.

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

Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.

John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann, 1998.

Richard A. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

Dumitru Erhan, Pierre-Jean L'Heureux, Shi Yi Yue, and Yoshua Bengio. Collaborative filtering on a family of biological targets. *Journal of Chemical Information and Modeling*, 46(2):626–635, 2006.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

Feng Jin and Shiliang Sun. A multitask learning approach to face recognition based on neural networks. In *IDEAL '08: Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31, Berlin, Heidelberg, 2008. Springer-Verlag.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

Mikaela Keller and Samy Bengio. A multitask learning approach to document representation using unlabeled data. IDIAP-RR 44, IDIAP, 2006.

Brendan Kitts, David Freed, and Martin Vrieze. Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In *In Proceedings of ACM SIGKDD International Conference*, pages 437–446. ACM Press, 2000.

Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1), 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. 42(8):30–37, 2009a.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009b.

Nguyon Du Phuong and Tu Minh Phuong. Collaborative filtering by multi-task learning. In *IEEE International Conference on Research, Innovation and Vision for the Future.*, pages 227–232, 2008.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce, 1999.

Alex J. Smola, Bernhard Schölkopf, and Bernhard Schölkopf. A tutorial on support vector regression. Technical report, Statistics and Computing, 2003.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, 2009.

Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *In International Conference on Machine Learning*, pages 489–497. Morgan Kaufmann, 1996.

V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

Kai Yu and Volker Tresp. Learning to learn and collaborative filtering. In *In Neural Information Processing Systems Workshop on Inductive Transfer: 10 Years Later*, 2005.

Shipeng Yu, Kai Yu, Volker Tresp, and Hans-Peter Kriegel. Collaborative ordinal regression. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1089–1096, New York, NY, USA, 2006. ACM.