Feature Based Informative Model for Discriminating Favorite Items from Unrated Ones

Bing Cheng, Tianqi Chen, Diyi Yang, Weinan Zhang, Yongqiang Wang, and Yong Yu

Computer Science and Engineering,
Shanghai Jiao Tong University,
Dongchuan Road, Minhang District, Shanghai, China
{chengb,tqchen,yangdiyi,wnzhang,wangyq,yyu}@apex.sjtu.edu.cn

Abstract. In this paper, we describe a feature based informative model to the second track of this year's KDD Cup Challenge[1]. The goal is to discriminate songs rated highly by the user from ones never rated by him/her. The informative model is used to incorporate different kinds of information, such as taxonomy of items, item neighborhoods, user specific features and implicit feedback, into a single model. Additionally, we also adopt ranking oriented SVD and negative sampling to improve prediction accuracy. Our final model achieves an error rate of 3.10% on the test set with a single predictor, which is the best result of single predictors in all the publicized results on this task, even better than many ensemble models.

Keywords: Feature Based Informative Model, Ranking oriented SVD, Negative Sampling.

1 Introduction

The explosive growth of available choices from content providers has given great prominence to recommendation systems. In the past years, recommendation systems have shown great potential to help users find interesting items from large item space[1,2]. Due to its great benefits to both users and content providers, recommendation systems have been actively researched since it was introduced[3,4].

For this year's KDD Cup Challenge, Yahoo! Labs released a large music rating dataset. The contest consists of two tracks. The first track is a rating prediction problem that aims at minimizing RMSE (Root Mean Square Error). It is similar to the famous Netflix Prize Challenge[2]. The task of the second one is to discriminate the 3 songs rated highly by the user from the 3 ones never rated by her. In this task "rate highly" means a rating greater than or equal to 80. We tackle this problem as a top-n recommendation problem. That is, the three songs with higher prediction scores are regarded as the user's favorite songs, while the other 3 songs are considered to be unrated.

[1] http://kddcup.yahoo.com/
[2] www.netflixprize.com

In this paper, we use ranking oriented SVD to solve this problem. A negative sampling technique is utilized to further improve prediction accuracy. Most importantly, we propose to use a feature based informative model to incorporate different kinds of information into a single model. The ensemble of many algorithms is a useful approach to improve the overall performance. This has been proved by the winners of the Netflix Prize[5]. Different algorithms capture different information of the dataset, so they blend well. All the publicized results on KDD Cup Track2 also adopt ensemble techniques to boost their final predictions. However, ensemble usually needs extra computation cost. There is no doubt that a single model with comparable performance to ensemble models is more acceptable. Here we propose such a model. Different kinds of information, such as taxonomy of items (more on this in Section 2.2), item neighborhoods, user specific features and implicit feedback, are integrated into a single model. With this model, we achieve an error rate of 3.10% on the test set. This is the best result of single predictors among all publicized results on this task, even better than the performance of many ensemble models.

The reminder of this paper is organized as follows. In Section 2 we present the preliminaries, stating our task and giving some key properties of the dataset. Symbols that will be used later are also defined in this section. Ranking oriented SVD and the negative sampling strategies are detailed in Section 3. Section 4 focuses on the feature based informative model. In Section 5 we give our experimental results. Related works are summarized in Section 6. Finally in Section 7 we conclude our work.

2 Preliminaries

2.1 Problem Statement

For each user in the test set, six songs are given. Three of them are selected randomly from the songs rated highly by the user, and the other three are selected with probability proportional to the number of high ratings the song receives. The motivation for this setting is, for popular songs, user may have already heard about them. If he/she still doesn't rate these songs, it is likely that he/she doesn't like them. Participants need to separate the three positive examples from the three negative ones. The evaluation metric is error rate. That is, the ratio of wrongly classified test cases with respect to the total number of test cases.

2.2 Key Properties of the Yahoo! Music Dataset

For the task of binary user preference prediction, Yahoo! Labs released a dataset consisting of 67 million ratings from 24 thousand users on 30 thousand items. An important property of this dataset is the taxonomy information. That is, items have four categories: artist, album, song and genre. As we will show in Section 5, the taxonomy information can decrease error rate significantly.

2.3 Notation Definition

We consider the whole rating dataset $\mathbf{R} = [r_{u,i}]$ to be a sparse matrix. The letter u and i are used to denote user and item respectively. $r_{u,i}$ is user $u's$ rating on item i, $0 \leq r_{u,i} \leq 100$. Bold letters are used for matrices and vectors, non bold for scalars. The inner product of two vectors $\mathbf{p}$ and $\mathbf{q}$ is $< \mathbf{p}, \mathbf{q} >$. The letter $\mathbb{U}$ is used to represent the whole user set and the letter $\mathbb{I}$ for the whole item set. Predicted rating for (u, i) pair is $\hat{r}_{u,i}$. The set of users who rated item i is $\mathbb{U}(i)$ and the set of items rated by user u is $\mathbb{I}(u)$. $N(i; k)$ is the set of top-k neighborhoods of item i computed by Pearson Correlation Coefficient.

3 Prediction Models

In this section we will elaborate the ranking oriented SVD model with negative sampling for the binary user reference prediction problem. In Section 3.1 we briefly present classical SVD models that try to minimize RMSE. Section 3.2 focuses on the ranking oriented SVD models and the strategies to pair rating records. Finally in Section 3.3 we introduce the negative sampling approach used in this paper.

3.1 Classical *SVD* Models

Classical SVD [6] models mainly focus on approximating the rating of user u on item i by

$$\hat{r}_{u,i} = \mu + b_u + b_i + < \mathbf{p}_u, \mathbf{q}_i > \tag{1}$$

Here μ is the global average of the rating dataset $\mathbf{R}$, b_u and b_i are user and item bias respectively. The two vectors $\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^f$ are f dimensional vectors used to capture the latent attributes of users and items. The parameters are learnt by minimizing RMSE.

$$L = \sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2 + regularization\ terms \tag{2}$$

We denote SVD models that try to minimize loss function (2) by *ReSVD* (Regression SVD) .These models gained great success in the Netflix Prize. However, as shown by [7], they usually don't work as well in the choice prediction problem.

3.2 Ranking Oriented SVD

A natural solution to the choice prediction problem is learning to rank. Eigen-Rank [8] and pLPA (probabilistic Latent Preference Analysis) [9] are two such models. Ranking oriented SVD models are first proposed by Nathan N. Liu et al.[10]. The key technique in these models is to turn either implicit or explicit feedback into user dependent pairwise preference regarding items. The pairwise preference is denoted by the variable δ_{uij}, which takes value of $+1$ if user u

prefers item i to item j, and -1 if the opposite holds. For our task of binary user preference prediction, we derive the pairwise preference by two ways. The first is based on the gap between two ratings, and the second uses two boundaries. These two ways correspond to the definition of δ_{uij} in Equation (3) and (4). We denote them as *GAP_PAIR* and *BOUND_PAIR* respectively.

$$\delta_{uij} = \begin{cases} +1 & i, j \in \mathbb{I}(u) \; and \; r_{u,i} - r_{u,j} \geq t \\ -1 & i, j \in \mathbb{I}(u) \; and \; r_{u,j} - r_{u,i} \geq t \end{cases} \tag{3}$$

$$\delta_{uij} = \begin{cases} +1 & i, j \in \mathbb{I}(u) \; and \; r_{u,i} \geq t_{lb} \; and \; r_{u,j} \leq t_{ub} \\ -1 & i, j \in \mathbb{I}(u) \; and \; r_{u,j} \geq t_{lb} \; and \; r_{u,i} \leq t_{ub} \end{cases} \tag{4}$$

Here t, t_{ub} *and* t_{lb} are pre-defined thresholds. In Section 5 we give our experimental results on these two definitions of δ_{uij}. Given a certain definition for δ_{uij}, we denote the set of triples (u, i, j) with $\delta_{uij} = +1$ as the set $\mathbb{D}$. $\mathbb{D}$ is used as the input for our ranking oriented SVD models.

We follow the work of [10] to use Bradley-Terry model [11] to design the loss function for the preference prediction problem. Under this model, each user u is associated with $|\mathbb{I}|$ parameters $\gamma_{u,i}$ indicating the utilities of these items to the user. The higher $\gamma_{u,i}$ is compared to $\gamma_{u,j}$, the more likely that $\delta_{uij} = +1$. This relation can be described using a sigmoid function:

$$P(\delta_{uij} = +1) = \frac{1}{1 + e^{-(\gamma_{u,i} - \gamma_{u,j})}} \tag{5}$$

To adopt the this model to SVD, we take $\gamma_{u,i} = \hat{r}_{u,i}$. This parametrization leads to the following maximum likelihood training procedure:

$$L = \sum_{(u,i,j) \in \mathbb{D}} ln(1 + e^{-(\hat{r}_{u,i} - \hat{r}_{u,j})}) + regularization \; terms \tag{6}$$

We denote SVD models that optimize the loss function defined in (6) as *RaSVD* (Ranking oriented SVD).

3.3 Negative Sampling for *SVD* Models

Negative sampling for CF is proposed by Rong Pan et al. [12] to solve the OCCF problem. In OCCF, only positive examples are available. Such situations include users' visiting history to news page recommendation, users' clicking and trading history on online shopping site.

The negative sampling strategy we employ is user-oriented pairwise sampling which proves to work well in [12]. For every (u, i) pair in $\mathbf{R}$ with a rating higher than a pre-define threshold θ, we select k items not rated by u as the negative examples for this user. The negative examples are selected using the same approach as the negative examples in the test data. That is, the probability of an item being selected as negative example is proportional to the number of high ratings it receives in the rating set $\mathbf{R}$. The score of negative example is 0. Such positive/negative item pairs are used as the training data for our ranking oriented SVD models. We will show the impact of k and θ on prediction accuracy in Section 5.

4 Feature Based Informative Model

In this section we elaborate the informative model we use in this paper. Section 4.1 presents a feature based collaborative filtering framework (abbreviate to *FCFF*)[13], which serves as the basis of our informative model. Following sections will focus on how to incorporate different kinds of information into the framework.

4.1 Feature Based Collaborative Filtering Framework

Our informative model is based on the feature based collaborative filtering framework proposed by Chen at al. [13]. The input format of *FCFF* is similar to LibSVM[14], users just need to define features to implement new model. The model equation of *FCFF* is:

$$y(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \mu + \left(\sum_j b_j^g \alpha_j + \sum_j b_j^u \beta_j + \sum_j b_j^i \gamma_j \right) + \left(\sum_j \beta_j \mathbf{p}_j \right)^T * \left(\sum_j \gamma_j \mathbf{q}_j \right) \tag{7}$$

We can see from Equation (7) that in *FCFF*, the features can be divided into three groups: $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$. They represent global, user and item dependent features respectively. The corresponding $\mathbf{b}^g, \mathbf{b}^u, \mathbf{b}^i$ are the weights of these features, which need to be learnt by minimizing the loss function (2) or (6). Another important property of *FCFF* is that global features are not involved in factorization. They are linear features. Incorporating these features into SVD models can help us capture information from different sources. We will show this in the following sections.

For the basic SVD model defined in Equation (1), the features for *RaSVD* are defined in Equation (8):

$$\beta_h = \begin{cases} +1 & h = index(userId) \\ 0 & otherwise \end{cases} \qquad \gamma_h = \begin{cases} +1 & h = index(posId) \\ -1 & h = index(negId) \\ 0 & otherwise \end{cases} \tag{8}$$

Here *index* is a function used to map feature to a unique ID. In Equation (8) *posId* is the ID of the positive example and *negId* is the ID of the negative example. For *ReSVD* , features can be defined in a similar way. We omit the details due to space limitation.

4.2 Taxonomy-Aware *SVD* Models

As we mentioned in Section 2.2, an important property of the Yahoo! Music dataset is the taxonomy information. In particular, each song belongs to one album and one artist. Each album belongs to one artist. Moreover, every song or album has zero or multi genres. Intuitively, items that are closely correlated in the taxonomy may receive similar ratings from the same user. For example, given that a user rates highly on an artist(album), we can conclude that it's likely that the user will also rate highly on the songs belonging to the artist(album). In this section we integrate this information into *FCFF*.

To capture the taxonomy relationship, for each item i in $\mathbb{I}$, we introduce a new bias term b'_i and latent vector $\mathbf{q}'_i$. The original bias term b_i and latent vector $\mathbf{q}_i$ are used to predict the score of the item itself, while the new bias term b'_i and latent vector $\mathbf{q}'_i$ are used to help the prediction of the children of item i. If i is artist, its children include the albums and songs of artist i. If i is album, its children are all the songs belonging to album i. This leads to the following formulation of the estimation of $r_{u,i}$:

$$\hat{r}_{u,i} = \mu + b_u + b_i + b'_{Al(i)} + b'_{Ar(i)} + < \mathbf{p}_u,\ \mathbf{q}_i + \mathbf{q}'_{Al(i)} + \mathbf{q}'_{Ar(i)} > \tag{9}$$

Here $Al(i)$ is the album of item i, and $Ar(i)$ is the artist of item i. For rank models, the features of this taxonomy-aware SVD can be defined as follows:

$$\gamma_h = \begin{cases} +1 & h = index(posId)\ or\ h = index(Al(posId))\ or \\ & h = index(Ar(posId)) \\ -1 & h = index(negId)\ or\ h = index(Al(negId))\ or \\ & h = index(Ar(negId)) \\ 0 & otherwise \end{cases} \tag{10}$$

$\boldsymbol{\beta}$ is defined the same way as Equation (8).

4.3 Integrating Implicit Feedback

As pointed out by early works [15,3], implicit feedback has great potential to improve the performance of recommendation systems. Compared to explicit feedback where explicit ratings are required, implicit feedback emphasizes more on which items the user rates. After integrating implicit feedback, prediction score for unknown (u, i) pair is:

$$\begin{aligned} \hat{r}_{u,i} = \mu + b_u + b_i + b'_{i,Al(i)} + b'_{i,Ar(i)} + < \mathbf{p}_u + \tfrac{1}{\sqrt{|\mathbb{I}(u)|}} \textstyle\sum_{j \in \mathbb{I}(u)} \mathbf{q}''_j, \\ \mathbf{q}_i + \mathbf{q}'_{Al(i)} + \mathbf{q}'_{Ar(i)} > \end{aligned} \tag{11}$$

In this model, each item i is associated with a new latent vector $\mathbf{q}''_i$, which is used to uncover information embedded in users' implicit feedback.

4.4 Integrating Neighborhood Information

In this section we will show how to integrate neighborhood information into the feature based informative model. Combining neighborhood and SVD model into a single predictor is first proposed by Y. Koren et al. [3] with the following model:

$$\begin{aligned} \hat{r}_{u,i} = \mu + b_i + b_u + < \mathbf{p}_u, \mathbf{q}_i > + \tfrac{1}{\sqrt{|\mathbb{R}(u,i;k)|}} \textstyle\sum_{j \in \mathbb{R}(u,i;k)} (r_{u,j} - \hat{b}_{u,j}) w_{i,j} + \\ \tfrac{1}{\sqrt{|\mathbb{N}(u,i;k|}} \textstyle\sum_{j \in \mathbb{N}(u,i;k)} c_{i,j} \end{aligned} \tag{12}$$

Here $\hat{b}_{u,j}$ is a baseline estimator composed only by user bias and item bias. $w_{i,j}$ and $c_{i,j}$ are correlation coefficient between two items i and j. $\mathbb{R}(u,i;k) = \mathbb{N}(u,i;k) = \mathbb{I}(u) \bigcap \mathbb{N}(i;k)$.

For our binary user preference prediction problem, we find that implicit feedback is much more useful than explicit feedback. Integrating implicit neighborhood information into Equation (11) we get:

$$\begin{aligned} \hat{r}_{u,i} = \mu + b_u + b_i + {b'}_{Al(i)} + {b'}_{Ar(i)} + < \mathbf{p}_u + \tfrac{1}{\sqrt{|\mathbb{I}(u)|}} \textstyle\sum_{j \in \mathbb{I}(u)} \mathbf{q}''_j, \\ \mathbf{q}_i + \mathbf{q}'_{Al(i)} + \mathbf{q}'_{Ar(i)} > + \tfrac{1}{\sqrt{|\mathbb{N}(u,i;k)|}} \textstyle\sum_{j \in \mathbb{N}(u,i;k)} c_{ij} \end{aligned} \tag{13}$$

To implement the model described in (13) under *FCFF*, we just need to add new global features.

$$\alpha_h = \begin{cases} \frac{1}{\sqrt{\mathbb{N}(u,posId;k)}} & g \in \mathbb{N}(u,posId;k) \ and \ h = index(posId,g) \\ -\frac{1}{\sqrt{\mathbb{N}(u,negId;k)}} & g \in \mathbb{N}(u,negId;k) \ and \ h = index(negId,g) \\ 0 & otherwise \end{cases} \tag{14}$$

Here $index(posId,g)/index(negId,g)$ is the feature ID of the neighborhood pair $(posId,g)/(negId,g)$.

4.5 Integrating Taxonomy Based Classifier

As we mentioned in Section 4.2, users' preferences on artist and album have great impact on their attitudes towards the corresponding songs. In this section, we integrate a taxonomy based classifier into our informative model. The classifier uses four user dependent features:

– User's rating on the artist of the song. If not rated, this value is 0;
– Whether the rating on the artist of the song is higher than 80;
– User's rating on the album of the song. If not rated, this value is 0;
– Whether the rating on the album of the song is higher than 80;

If we denote these four features by $b_{u,j}$ $(0 \le j \le 3)$, the new model is:

$$\begin{aligned} \hat{r}_{u,i} = \mu + b_u + b_i + {b'}_{Al(i)} + {b'}_{Ar(i)} + < \mathbf{p}_u + \tfrac{1}{\sqrt{|\mathbb{I}(u)|}} \textstyle\sum_{j \in \mathbb{I}(u)} \mathbf{q}''_j, \\ \mathbf{q}_i + \mathbf{q}'_{Al(i)} + \mathbf{q}'_{Ar(i)} > + \tfrac{1}{\sqrt{|\mathbb{N}(u,i;k)|}} \textstyle\sum_{j \in \mathbb{N}(u,i;k)} c_{ij} + \textstyle\sum_{j=0}^{3} b_{u,j} \end{aligned} \tag{15}$$

To incorporate the taxonomy based classifier into *FCFF*, we just need to define new global features for $b_{u,j}$ in a similar way as Equation (14).

5 Experimental Results and Analysis

In this section we give our experimental results on the test set provided by Yahoo! Labs. In the test set, there are 101172 users. For each user, 6 songs are given. The total number of test cases is 607032. For each test case, we need to label the song as either positive or negative example for the user. The evaluation metric is the ratio of wrongly labeled test cases with respect to the total number of test cases.

To encourage further research on the problem, we open source all the code and implementations of our experiments. The implementation of *FCFF* is available at our laboratory page[3]. The code used to generate features for all the models in Section 4 is also released[4].

5.1 Performance of Informative Model on Error Rate

In this section we show the effectiveness of informative model. Before giving the experimental results, we first introduce some abbreviations that will be used in later sections in Table 1.

Table 1. Explanation of abbreviations

(a) Loss function

ReSVD	Loss function (2)
RaSVD	Loss function (6)
+GAP_PAIR	δ_{uij} Equation (3)
RaSVD	Loss function (6)
+BOUND_PAIR	δ_{uij} Equation (4)

(b) Prediction models

BSVD	Basic SVD, Equation (1)
Tax-SVD	Taxonomy-Aware SVD, Equation (9)
+IMFB	Implicit feedback, Equation (11)
+Item-10NN	Top 10 item neighborhood, Equation (13)
+Tax-CLF	Taxonomy based classifier, Equation (15)

The "+" symbol in the table means "recursive combining". For example, the "+" in the last line of Table 1(b) means combining taxonomy-aware SVD, implicit feedback, item neighborhood and taxonomy based classifier into a single model. This is also our best prediction model. We denote this model as *InfoSVD*(Informative SVD).

Table 2 shows the error rate of different models with triple negative sampling. The results show that incorporating extra information into *FCFF*, such as taxonomy of items, item neighborhoods, user specific features and implicit feedback, can always help lower error rate. Taking *RaSVD+BOUND_PAIR* as an example, incorporating item taxonomy into $BSVD$ decreases error rate by 10.8%. After integrating implicit feedback, a decrease of 7.5% in error rate is achieved. Item neighborhood and taxonomy based classifier bring in even larger improvement. By incorporating these two sources of information, error rate decreases by 25.1% and 20.3% respectively. In Table 3 we give the performance of

[3] http://apex.sjtu.edu.cn/apex_wiki/svdfeature
[4] http://apex.sjtu.edu.cn/apex_wiki/kddtrack2

Table 2. Prediction error rate(%) of informative model with triple negative sampling. Sampling threshold $\theta = 40$.

	ReSVD	RaSVD+GAP_PAIR	RaSVD+BOUND_PAIR
	$f = 100$	$t = 20, f = 100$	$t_{lb} = 20, t_{ub} = 0, f = 100$
BSVD	6.52	*6.42*	6.76
Tax-SVD	6.15	*6.03*	*6.03*
+IMFB	5.71	*5.52*	5.58
+Item-10NN	*3.98*	4.17	4.18
+Tax-CLF	3.76	3.40	*3.33*

Table 3. Prediction error rate(%) of informative model with no negative sampling

	ReSVD	RaSVD+GAP_PAIR	RaSVD+BOUND_PAIR
	$f = 100$	$t = 20, f = 100$	$t_{lb} = 20, t_{ub} = 0, f = 100$
BSVD	*23.99*	25.20	29.40
Tax-SVD	*20.79*	22.74	26.92
+IMFB	17.09	17.03	*16.51*
+Item-10NN	*14.23*	15.70	15.63
+Tax-CLF	*9.63*	12.19	13.78

the same models as Table 2 with no negative sampling. The results confirm our analysis above. To sum up, compared to *BSVD*, *InfoSVD* can at least decrease error rate by 42.3%.

5.2 Best Models

In this section we give the results of the best models for both regression SVD and ranking oriented SVD. The parameter settings are also presented. After this we give a comparison of our results and all the publicized results on this task.

For both regression SVD and ranking oriented SVD, the best prediction models are *InfoSVD* defined in Equation 15. This again confirms the effectiveness of *FBCF* in our user preference prediction problem. By integrating different sources of information into a single model, we can indeed lower error rate significantly.

Table 4. Best results for regression SVD and ranking oriented SVD

	Best Results	Parameters
ReSVD	**3.78**	$\theta = 60, k = 3, f = 100$
RaSVD+BOUND_PAIR	**3.16**	$\theta = 60, k = 5, t_{lb} = 20, t_{ub} = 0, f = 100$
RaSVD+GAP_PAIR	**3.10**	$\theta = 60, k = 5, t = 40, f = 300$

Table 4 shows the best results of *ReSVD* and *RaSVD*. θ is sampling threshold and k is the number of negative examples generated for each triple $(u, i, r_{u,i})$

Table 5. Comparison of error rate(%) between *InfoSVD* and all publicized results

Team name	Best single predictor	Ensemble model
InfoSVD	**3.10**	
National Taiwan University	4.04	2.47
The Art of Lemon	3.49	2.48
commendo	4.28	2.49
The Thought Gang	3.71	2.93
The Core Team		3.87
False Positives	5.70	3.89
Opera Solutions	4.45	4.38
MyMediaLite	6.04	4.49
KKT's Learning Machine	5.62	4.63
coaco		5.20

with $r_{u,i} \geq \theta$. An error rate of *3.10%* is achieved by *InfoSVD* with ranking oriented loss function defined in Equation 6.

Table 5 gives a comparison of the results of *InfoSVD* and the top 10 teams on the leaderboard. We can see from the table that *InfoSVD* achieves the lowest error rate among all the single predictors, outperforming the second one of 3.49% by 11.2%. Additionally, the performance of *InfoSVD* is even better than some ensemble models.

5.3 Impact of Negative Sampling and Ranking Oriented *SVD*

Comparing Table 2 and Table 3 we find that negative sampling can improve recommendation accuracy significantly. With triple negative sampling, the error rate of the best model decreases from 9.63% to 3.33%. Another interesting observation from Table 2 and Table 3 is the performance comparison of *ReSVD* and *RaSVD*. As we can see from these two tables, without negative sampling, *ReSVD* performs better in most cases. However, after bringing in extra negative examples, the performance of *RaSVD* grows faster and outperforms *ReSVD*.

Table 6 shows the effect of parameter k in negative sampling. We can see from Table 6 that increasing k can always bring in improvement in prediction accuracy, but the gain becomes less as k increases. Since the impact of θ on error rate is not so obvious when θ varies from 20 to 80, we omit the experiment results on different values of θ.

6 Related Work

CF algorithms fall into two categories: neighborhood model [16,3] and matrix factorization model [3,17]. For choice prediction problems, learning to rank [8,9,10] and negative sampling [12] have been proved to work well. EigenRank[8] and pLPA[9] are two typical learning to rank models. These models address the ranking problem directly without a rating prediction step. Other learning to

Table 6. Impact of k in negative sampling on error rate(%) with θ fixed to 60

	ReSVD	RaSVD+GAP_PAIR	RaSVD+BOUND_PAIR
	$f = 100$	$t = 20, f = 100$	$t_{lb} = 20, t_{ub} = 0, f = 100$
k=1	4.48	*4.09*	4.30
k=2	4.14	3.66	*3.65*
k=3	4.05	3.54	*3.40*
k=4	3.96	3.45	*3.22*
k=5	3.92	3.40	*3.16*

rank models[10,18,19] assign scores to items as traditional approaches, but the scores are just used to rank items, instead of approximating the real ratings. Rendel et al.[18] propose to use Bayesian probabilistic ranking model for top-n recommendation. Shi et al.[19] propose a list-wise learning to rank model with matrix factorization.

As we can see from experiments, our proposed feature-based informative model lowers error rate significantly compared to basic SVD. LibFM[20] also uses idea of feature-based factorization model. Compared to their model, our model distinguishes features types, which allows us to incorporate useful information such as neighborhood and taxonomy more naturally.

7 Conclusions and Future Work

In this paper, we mainly study the feature based CF framework for discriminating uses' favorite songs from ones unrated by her. Under this framework, new models can be implemented easily by defining features in the input data. With this framework, we achieve an error rate of 3.10% with a single predictor, which is the best performance of all single predictors on this task. The effectiveness of ranking oriented models and negative sampling are also presented.

For future work, we plan to investigate the performance of our informative model in a more general and practical scenario: top-n recommendation. In this case, metrics like recall and precision in information retrieval can be used.

References

1. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 271–280. ACM, New York (2007)
2. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing 7, 76–80 (2003)
3. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 426–434. ACM, New York (2008)

4. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. 22, 89–115 (2004)
5. Andreas, T., Jahrer, M., Bell, R.M., Park, F.: The bigchaos solution to the netflix grand prize, pp. 1–52 (2009)
6. Funk, S.: Try this at home (2006), `http://sifter.org/∼simon/journal/20061211.html`
7. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 39–46. ACM, New York (2010)
8. Liu, N.N., Yang, Q.: EigenRank: a ranking-oriented approach to collaborative filtering. In: SIGIR 2008: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, pp. 83–90 (2008)
9. Liu, N.N., Zhao, M., Yang, Q.: Probabilistic latent preference analysis for collaborative filtering. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, pp. 759–766. ACM (2009)
10. Liu, N.N., Cao, B., Zhao, M., Yang, Q.: Adapting neighborhood and matrix factorization models for context aware recommendation. In: Proceedings of the Workshop on Context-Aware Movie Recommendation, pp. 7–13. ACM, New York (2010)
11. Marden, J.I.: Analyzing and Modeling Rank Data. Chapman & Hall (1995)
12. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: ICDM 2008 (2008)
13. Chen, T., Zheng, Z., Lu, Q., Zhang, W., Yu, Y.: Feature-based matrix factorization. Technical Report APEX-TR-2011-07-11, Apex Data & Knowledge Management Lab, Shanghai Jiao Tong University (July 2011)
14. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), `http://www.csie.ntu.edu.tw/∼cjlin/libsvm`
15. Oard, D., Kim, J.: Implicit feedback for recommender systems. In: Proceedings of the AAAI Workshop on Recommender Systems, pp. 81–83 (1998)
16. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001 (2001)
17. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: 13th ACM Int. Conf. on Knowledge Discovery and Data Mining, Proc. KDD Cup Workshop at SIGKDD 2007, pp. 39–42 (2007)
18. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. Machine Learning (2009)
19. Shi, Y., Larson, M., Hanjalic, A.: List-wise learning to rank with matrix factorization for collaborative filtering. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010 (2010)
20. Rendle, S.: Factorization machines. In: Proceedings of the 10th IEEE International Conference on Data Mining. IEEE Computer Society (2010)