Pairwise Preference Regression for Cold-start Recommendation

Seung-Taek Park*
Samsung Advanced Institute of Technology
Mt. 14-1, Nongseo-dong, Giheung-gu
Yongin-si, Gyunggi-do 446-712, South Korea
park.seungtaek@gmail.com

Wei Chu
Yahoo! Labs
4401 Great America Parkway
Santa Clara, CA 95054, USA
chuwei@yahoo-inc.com

ABSTRACT

Recommender systems are widely used in online e-commerce applications to improve user engagement and then to increase revenue. A key challenge for recommender systems is providing high quality recommendation to users in "cold-start" situations. We consider three types of cold-start problems: 1) recommendation on existing items for new users; 2) recommendation on new items for existing users; 3) recommendation on new items for new users. We propose predictive feature-based regression models that leverage all available information of users and items, such as user demographic information and item content features, to tackle cold-start problems. The resulting algorithms scale efficiently as a linear function of the number of observations. We verify the usefulness of our approach in three cold-start settings on the MovieLens and EachMovie datasets, by comparing with five alternatives including random, most popular, segmented most popular, and two variations of Vibes affinity algorithm widely used at Yahoo! for recommendation.

Categories and Subject Descriptors

H.1.0 [**Models and Principles**]: General; H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Recommender System, cold-start problems, user and item features, ranking, normalized Discounted Cumulative Gain

*This work was conducted while Seung-Taek Park was at Yahoo! Labs.

RecSys'09, October 23–25, 2009, New York, New York, USA.

1. INTRODUCTION

Recommender systems automate the familiar social process of friends endorsing products to others in their community. Widely deployed on the web, such systems help users explore their interests in many domains, including movies, music, books, and electronics. Recommender systems are widely applied from independent, community-driven web sites to large e-commerce powerhouses like Amazon.com. Recommender systems can improve users' experience by personalizing what they see, often leading to greater engagement and loyalty. Merchants, in turn, receive more explicit preference information that paints a clearer picture of their customers. Two different approaches are widely adopted to design recommender systems: *content-based filtering* and *collaborative filtering*.

Content-based filtering generates a profile for a user based on the content descriptions of the items previously rated by the user. The major benefit of this approach is that it can recommend users new items, which have not been rated by any users. However, content-based filtering cannot provide recommendations to new users who have no historical ratings. To provide new user recommendation, content-based filtering often asks new users to answer a questionnaire that explicitly states their preferences to generate initial profiles of new users. As a user consumes more items, her profile is updated and content features of the items that she consumed will receive more weights. One drawback of content-based filtering is that the recommended items are similar to the items previously consumed by the user. For example, if a user has watched only romance movies, then content-based filtering would recommend only romance movies. It often causes low satisfaction of recommendations due to lack of diversity for new or casual users who may reveal only small fraction of their interests. Another limitation of content-based filtering is that its performance highly depends on the quality of feature generation and selection.

On the other hand, collaborative filtering typically associates a user with a group of like-minded users, and then recommends items enjoyed by others in the group. Collaborative filtering has a few merits over content-based filtering. First, collaborative filtering does not require any feature generation and selection method and it can be applied to any domains if user ratings (either explicit or implicit) are available. In other words, collaborative filtering is content-independent. Second, collaborative filtering can provide "serendipitous finding", whereas content-based filtering cannot. For example, even though a user has watched

only romance movies, a comedy movie would be recommended to the user if most other romance movie fans also love it. Collaborative filtering captures this kind of hidden connections between items by analyzing user consumption history (or user ratings on items) over the population. Note that content-based filtering uses a profile of individual user but does not exploit profiles of other users.

Even though collaborative filtering often performs better than content-based filtering when lots of user ratings are available, it suffers from the *cold-start* problems where no historical ratings on items or users are available. A key challenge in recommender systems including content-based and collaborative filtering is how to provide recommendations at early stage when available data is extremely sparse. The problem is of course more severe when the system newly launches and most users and items are new. However, the problem never goes away completely, since new users and items are constantly coming in any healthy recommender system. We consider three types of cold-start setting in this paper: 1) recommending existing items for new users, 2) recommending new items for existing users, and 3) recommending new items for new users.

We realize that there are additional information on users and items often available in real-world recommender systems. We can request users' preference information by encouraging them to fill in questionnaires or simply collect user-declared demographic information (i.e. age and gender) at registration. We can also utilize item information by accessing the inventory of most on-line enterpriser. These legally accessible information is valuable for both recommending new items and serving new users. To attack the cold-start problem, we propose new hybrid approaches which exploit not only user ratings but also user and item features. We construct tensor profiles for user/item pairs from their individual features. Within the tensor regression framework, we optimize the regression coefficients by minimizing pairwise preference loss. The resulting algorithm scales efficiently as a linear function of the number of observed ratings. We evaluate our approach with two standard movie data sets: MovieLens and EachMovie. We cannot use the Netflix data since it does not provide any user information. Note that one of our goals is providing reasonable recommendation to even new users with no historical ratings but only a few demographic information.

We split user ratings into four partitions. We randomly select half of users as new users and the rest as existing users. Similarly, we randomly split items as new and existing items. Figure 1 illustrates data partition. Then we use partition I for training and partition II, III, and IV for test. We summarize available techniques for each partition in the following:

- Partition I (recommendation on existing items for existing users): This is the standard case for most existing collaborative filtering techniques, such as user-user, item based collaborative filtering, singular value decomposition (SVD), etc.;
- Partition II (recommendation on existing items for new users): For new users without historical ratings, "most popular" strategy that recommends the highly-rated items to new users serves as a strong baseline;
- Partition III (recommendation on new items for existing users): Content-based filtering can effectively recommend new items to existing users based on the users' historical ratings and features of items;
- Partition VI (recommendation on new items for new users): This is a hard case, where "random" strategy is the basic means of collecting ratings.

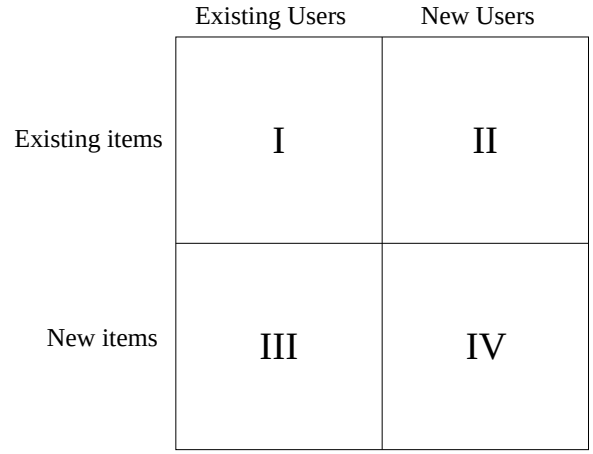

Figure 1: Data partition: We randomly select 50% users as new users and the rest as existing users. Similarly half of items are randomly selected as new items and the others as existing items. We use the partition I for training and the partition II, III, and IV for test.

We evaluate performance of recommender systems based on the correctness of ranking rather than prediction accuracy, the normalized Discounted Cumulative Gain ($nDCG$), widely used in information retrieval (IR), as performance metric. We compare against five recommendation approaches including Random, Most Popular (MP), Segmented Most Popular (SMP), and two variations of Vibes Affinity algorithm (Affinity) [21] widely used at Yahoo!.

The paper is organized as follows: We describe our algorithm in Section 2; We review related work in Section 3; We report experimental results with comparison against five existing competitive approaches in Section 4; We conclude in Section 5.

2. METHODOLOGY

In this section, we propose a regression approach based on profiles (TR) for cold-start recommendation. In many real recommender systems, users sometimes declare their demographical information, such as age, gender, residence, and etc., whereas the recommender systems also maintain information of items when items are either created or acquired, which may include product name, manufacturer, genre, production year, etc. Our key idea is to build a predictive model for user/item pairs by leveraging all available information of users and items, which is particularly useful for cold-start recommendation including new user and new item recommendation. In the following, we describe our approach in three subsections. The first subsection presents profile construction, and the last two subsections cover algorithm design.

2.1 Profile Construction

It is crucial to generate and maintain profiles of items of interest for effective cold-start strategies. For example, we collect item contents (i.e. genre, cast, manufacturer, production year etc.) as the initial part of the profile for movie recommendation. In addition to these static attributes, we also estimate items' popularity/quality from available historical ratings in training data, e.g. indexed by averaged scores in different user segments, where user segments could be simply defined by demographical descriptors or advanced conjoint analysis.

Generally we can construct user profiles as well by collecting legally usable user-specific features that effectively represent a user's preferences and recent interests. The user features usually consist of demographical information and historical behavior aggregated to some extent.

In this way, each item is represented by a set of features, denoted as a vector $\mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^D$ and D is the number of item features. Similarly, each user is represented by a set of user features, denoted as $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^C$ and C is the number of user features. Note that we append a constant feature to the user feature set for all users. A new user with no information is represented as $[0, \ldots, 0, 1]$ instead of a vector of zero entries.

In traditional collaborative filtering (CF), the ratings given by users on items of interest are used as user profiles to evaluate commonalities between users. In our regression approach, we separate these feedbacks from user profiles. The ratings are utilized as targets that reveal affinities between user features to item features.

We have collected three sets of data, including item features, user profiles and the ratings on items given by users. Let index the u-th user as $\mathbf{x}_u$ and the i-th content item as $\mathbf{z}_i$, and denote by r_{ui} the interaction between the user $\mathbf{x}_u$ and the item $\mathbf{z}_i$. We only observe interactions on a small subset of all possible user/item pairs, and denote by $\mathbb{O}$ the index set of observations $\{r_{ui}\}$.

2.2 Regression on Pairwise Preference

A predictive model relates a pair of vectors, $\mathbf{x}_u$ and $\mathbf{z}_i$, to the rating r_{ui} on the item $\mathbf{z}_i$ given by the user $\mathbf{x}_u$. There are various ways to construct joint feature space for user/item pairs. We focus on the representation via outer products, i.e., each pair is represented as $\mathbf{x}_u \otimes \mathbf{z}_i$, a vector of CD entries $\{x_{u,a} z_{i,b}\}$ where $z_{i,b}$ denotes the b-th feature of $\mathbf{z}_i$ and $x_{u,a}$ denotes the a-th feature of $\mathbf{x}_u$.

We define a parametric indicator as a bilinear function of $\mathbf{x}_u$ and $\mathbf{z}_i$ in the following:

$$s_{ui} = \sum_{a=1}^{C} \sum_{b=1}^{D} x_{u,a} z_{i,b} w_{ab}, \tag{1}$$

where C and D are the dimensionality of user and content features respectively, a, b are feature indices. The weight variable w_{ab} is independent of user and content features and characterizes the affinity of these two factors $x_{u,a}$ and $z_{i,b}$ in interaction. The indicator can be equivalently rewritten as

$$s_{ui} = \mathbf{x}_u \mathbf{W} \mathbf{z}_i^\top = \mathbf{w}^\top (\mathbf{z}_i \otimes \mathbf{x}_u), \tag{2}$$

where $\mathbf{W}$ is a matrix containing entries $\{w_{ab}\}$, $\mathbf{w}$ denotes a column vector stacked from $\mathbf{W}$, and $\mathbf{z}_i \otimes \mathbf{x}_u$ denotes the outer product of $\mathbf{x}_u$ and $\mathbf{z}_i$, a column vector of entries $\{x_{u,a} z_{i,b}\}$.

The regression coefficients can be optimized in regularization framework, i.e.

$$\arg\min_{\mathbf{w}} \sum_{ui \in \mathbb{O}} (r_{ui} - s_{ui})^2 + \lambda \|\mathbf{w}\|_2^2 \tag{3}$$

where λ is a tradeoff between empirical error and model complexity. Least squares loss coupled with 2-norm of $\mathbf{w}$, is widely applied in practice due to computational advantages.[1] The optimal solution of $\mathbf{w}$ is unique and has a closed form of matrix manipulation, i.e.

$$\mathbf{w}^* = \left(\sum_{ui \in \mathbb{O}} \mathbf{z}_i \mathbf{z}_i^\top \otimes \mathbf{x}_u \mathbf{x}_u^\top + \lambda \mathbf{I} \right)^{-1} \left(\sum_{ui \in \mathbb{O}} r_{ui}\, \mathbf{z}_i \otimes \mathbf{x}_u \right) \tag{4}$$

where $\mathbf{I}$ is an CD by CD identity matrix. By exploiting the tensor structure, the matrix preparation costs $\mathcal{O}(NC^2 + MC^2D^2)$ where M and N are the number of items and users respectively. The matrix inverse costs $\mathcal{O}(C^3D^3)$, which becomes the most expensive part if $M < CD$ and $N < MD^2$.

The tensor idea can be traced back to the Tucker family [33] and the PARAFAC family [16]. Recently exploratory data analysis with tensor product has been applied to image ensembles [34], DNA microarray data intergration [22] and semi-infinite stream analysis [32] etc. To our best knowledge, tensor regression hasn't been applied to cold-start recommendation yet.

In recommender systems, users may enjoy different rating criteria. Thus the ratings given by different users are not comparable due to user-specific bias. We can lessen the effect by introducing a bias term for each user in the above regression formulation, however it not only enlarges the problem size dramatically from CD to $CD + N$ where N denotes the number of users and usually $N \gg CD$, but also increases uncertainty in the modelling. Another concern is that the least squares loss is favorable for RMSE metric but may result in inferior ranking performance. Pairwise loss is widely used for preference learning and ranking, e.g. RankRLS [23] and RankSVM [17], for superior performance.

In this paper, we introduce a personalized pairwise loss in the regression framework. For each user $\mathbf{x}_u$, the loss function is generalized as

$$\frac{1}{n_u} \sum_{i \in \mathbb{O}_u} \sum_{j \in \mathbb{O}_u} ((r_{ui} - r_{uj}) - (s_{ui} - s_{uj}))^2 \tag{5}$$

where $\mathbb{O}_u$ denotes the index set of all items the user $\mathbf{x}_u$ have rated, $n_u = |\mathbb{O}_u|$ the number of ratings given by the user $\mathbf{x}_u$, and s_{ui} is defined as in eq(1). Replacing the squares loss by the personalized pairwise loss in the regularization framework, we have the following optimization problem:

$$\min_{\mathbf{w}} \sum_{u} \left(\frac{1}{n_u} \sum_{i \in \mathbb{O}_u} \sum_{j \in \mathbb{O}_u} ((r_{ui} - r_{uj}) - (s_{ui} - s_{uj}))^2 \right) + \lambda \|\mathbf{w}\|_2^2 \tag{6}$$

where u runs over all users. The optimal solution can be

[1] Other loss functions could be applied as well, e.g. the hinge loss in support vector machines, while advanced quadratic programming has to be applied.

computed in a closed form as well, i.e.

$$\mathbf{w}^* = \left(\mathbf{A} + \frac{\lambda}{2}\mathbf{I}\right)^{-1} \mathbf{B} \tag{7}$$

$$\mathbf{A} = \sum_u \sum_{i \in \mathbb{O}_u} \mathbf{z}_i (\mathbf{z}_i - \tilde{\mathbf{z}}_u)^\top \otimes \mathbf{x}_u \mathbf{x}_u^\top \tag{8}$$

$$\mathbf{B} = \sum_u \sum_{i \in \mathbb{O}_u} r_{ui} (\mathbf{z}_i - \tilde{\mathbf{z}}_u) \otimes \mathbf{x}_u \tag{9}$$

$$\tilde{\mathbf{z}}_u = \frac{1}{n_u} \sum_{i \in \mathbb{O}_u} \mathbf{z}_i \tag{10}$$

The size in matrix inverse is still CD and the matrix preparation costs $\mathcal{O}(NC^2 + MC^2D^2)$ same as that of the least squares loss.

When matrix inversion with very large CD becomes computationally prohibitive, we can instead apply gradient-descent techniques for a solution. The gradient can be evaluated by $\mathbf{Aw} - \mathbf{B}$. There is no matrix inversion involved in each evaluation, and the most expensive step inside is to construct the matrix $\mathbf{A}$ once only. Usually it would take hundreds of iterations for a gradient-descent package to get close to the minimum. Note that this is a convex optimization problem with a unique solution at the minimum.

3. RELATED WORK

Two different approaches have been widely used to build recommender systems: content-based filtering and collaborative filtering. Content-based filtering uses behavioral data about a user to recommend items similar to those consumed by the user in the past while collaborative filtering compares one user's behavior against a database of other users' behaviors in order to identify items that like-minded users are interested in. The major difference between two approaches is that content-based filtering uses a single user information while collaborative filtering uses community information.

Even though content-based filtering is efficient in filtering out unwanted information and generating recommendations for a user from massive information, it often suffers from lack of diversity on the recommendation. Content-based filtering requires a good feature generation and selection method while collaborative filtering only requires user ratings. Content-based filtering finds few if any coincidental discoveries while collaborative filtering systems enables serendipitous discoveries by using historical user data. Hundreds of collaborative filtering algorithms have been proposed and studied, including K nearest neighbors [30, 18, 28], Bayesian network methods [10], classifier methods [9], clustering methods [35], graph-based methods [4], probabilistic methods [19, 26], ensemble methods [13], taxonomy-driven [36], and combination of KNN and SVD [8].

Although collaborative filtering provides recommendations effectively where massive user ratings are available such as in the Netflix data set, it does not perform well where user rating data is extremely sparse. Several linear factor models have been proposed to attack the data sparsity. Singular Value Decomposition (SVD), Principal Component Analysis (PCA), or Maximum Margin Matrix Factoriation (MMMF) has been used to reduce the dimensions of the user-item matrix and smoothing out noise [9, 27, 14]. However, those linear factor models do not solve the cold-start problems for new users or new items. Several hybrid methods, which often combine information filtering and collaborative filtering techniques, have been proposed. Fab [5] is the first hybrid recommender system, which builds user profiles based on content analysis and calulates user similarities based on user profiles. Basu et al. [7] generated three different features: collaborative features (i.e. users who like the movie X), content features, and hybrid features (i.e. users who like comedy movies). Then, an inductive learning system, Ripper, is used to learn rules and rule-based prediction was generated for the recommendation. Claypool et al. [12] built an online newspaper recommender system, called Tango, that scored items based on collaborative filtering and content-based filtering separately. Then two scores are linearly combined: As users provide ratings, absolute errors of two scores are measured and weights of two algorithms are adjusted to minimize error. Good et al. [15] experimented with a number of types of filterbots, including including Ripper-Bots, DGBots, GenreBots and MegaGenreBot. A filterbot is an automated agent that rates all or most items algorithmically. The filterbots are then treated as additional users in a collaborative filtering system. Park et al. [24] improved the scalability and performance of filterbots in cold-start situations by adding a few global bots instead of numerous personal bots and applying item-based instead of user-user collaborative filtering. Melville et al. [20] used content-based filtering to generate default ratings for unrated items to make a user-item matrix denser. Then traditional user-user collaborative filtering is performed using this denser matrix. Schein et al. [29] extended Hofmann's aspect model to combine item contents and user ratings under a single probabilistic framework. Even though hybrid approaches potentially improve the quality of the cold-start recommendation, the main focus of many hybrid methods is improving prediction accuracy over all users by using multiple data rather than directly attacking the cold-start problem for new users and items. Note that all above approaches only lessen the cold-start problem where a target user has rated at least few ratings but do not work for new user or new item recommendation.

There are a few existing hybrid approaches which are able to make new user and new item recommendation. Pazzani [25] proposed a hybrid method that recommends items based on vote of four different algorithms: user-user collaborative filtering, content-based, demographic-based, and collaboration via content. This approach can provide new user recommendation by assembling several independent models. Our approach provides a unified framework of learning user-item affinity from all heterogeneous data simultaneously. Basilico and Hofmann [6] proposed an online perceptron algorithm coupled with combinations of multiple kernel functions that unify collaborative and content-based filtering. The resulting algorithm is capable of providing recommendations for new users and new items, but the performance has not been studied yet. The computational complexity in the proposed kernel machine scales as a quadratic function of the number of observations, which limits its applications to large-scale data sets. Agarwal and Merugu [3] proposed a statistical method to model dyadic response as a function of available predictor information and unmeasured latent factors through a predictive discrete latent factor model. Even though the proposed approach can potentially solve the cold-start problems, its main focus is improving quality of recommendation in general cases and its performance in cold-start settings is not fully studied yet. Chu and Park [11] proposed a predictive bilinear regression model in "dynamic content

environment", where the popularity of items changes temporally, lifetime of items is very short (i.e. few hours), and recommender systems are forced to recommend only new items. This work suggests to maintain profiles for both contents and users, where temporal characteristics of contents, e.g. popularity and freshness, are updated in real-time. In their setting, only tens of items are available at each moment and the goal is recommending the best among these active items to users. In our setting, item space is rather static but the number of items available at any moment is much larger (i.e. few thousands), and the user attributes are limited to demographic information only. Recently, Stern et al. [31] proposed a probabilistic model that combines user and item meta data with users' historical ratings to predict the users' interaction on items. Agarwal and Chen [2] also independently proposed a regression-based latent factor model for cold-start recommendation with the same spirit. In these models, dyadic response matrix $\mathbf{Y}$ is estimated by a latent factor model such as $\mathbf{Y} \approx \mathbf{U}^T\mathbf{V}$, where latent factor matrices, $\mathbf{U}$ and $\mathbf{V}$, are estimated by regression such as $\mathbf{U} \approx \mathbf{FX}$ and $\mathbf{V} \approx \mathbf{MZ}$. $\mathbf{X}$ and $\mathbf{Z}$ denote user attribute and item feature matrices, and $\mathbf{F}$ and $\mathbf{M}$ are weight matrices learnt by regression. These approaches have similar spirit to our model, while the key difference lies on methodology to estimate the weights. In our approach, we estimate the weight matrix $\mathbf{W}$ as in eq(2) by solving a convex optimization, whereas in the above works the weight matrix is approximated by a low-rank matrix decomposition, such as $\mathbf{W} \approx \mathbf{F}^T\mathbf{M}$, and latent components are then estimated by either approximate Bayesian inference or expectation-maximization techniques. We note the latent components are rotation-invariant in their own space, that means there are numerous local minima in the solution space which may make the estimation complicated.

4. EXPERIMENTS

In this section we report experimental results on two movie recommendation data sets, MovieLens and EachMovie. We first describe existing competitive algorithms we implemented for comparison purpose. Then we describe our testing procedure and report empirical results with the MoiveLens and EachMovie data.

4.1 Competitive approaches

We implemented three alternative recommendation approaches for comparison.

4.1.1 Most popular

Most Popular (MP) provides the same recommendation to all users based on global popularity of items. The global popularity of an item is measured as following:

$$s_i = \frac{\overline{r_i} * n_i + \overline{r} * \alpha}{n_i + \alpha} \tag{11}$$

where the average rating $\overline{r_i}$ is defined as $\frac{1}{n_i}\sum_{u \in \mathbb{O}_i} r_{ui}$, the support $n_i = |\mathbb{O}_i|$ is the number of users who have rated the i-th item, $\overline{r}$ denotes the average of all ratings and α denotes a shrinkage parameter which is inspired by [8]. Here $\overline{r} = 3.6$ for the MovieLens data and $\overline{r} = 4.32$ for the EachMovie data, which were measured in the partition I, shown in the figure 1. When $\alpha = 0$, s_i is purely based on its average rating $\overline{r_i}$. When $\alpha > 0$, s_i will be close to the overall average $\overline{r}$ if only few users have rated the item i. We set $\alpha = 2$ both on MovieLens and EachMovie, which yields the best performance in validation.

4.1.2 Segmented most popular

Segmented Most Popular (SMP) divides users into several user segments based on user features (i.e. age or gender) and computes predictions of items based on their local popularity within the user segment which a target user belongs to:

$$s_{ci} = \frac{\overline{r_{ci}} * n_{ci} + \overline{r} * \alpha}{n_{ci} + \alpha} \tag{12}$$

where $\overline{r_{ci}}$ and n_{ci} denote the average rating of an item i within a user segment c and the number of users who have rated the item i within the user segment c. We have tested three different segmentation methods based on gender only, age only, and the combination of age and gender (age*gender). There are two gender groups, male and female (one additional age group "unknown" for EachMovie due to missing entries), and seven age groups, i.e. under 18, 18-24, 25-34, 35-44, 45-49, 50-55, and above 56 (one additional "unknown" age group for EachMovie). The parameter α was determined by validation. We found the best SMP model is with $\alpha = 9$ on MovieLens and $\alpha = 5$ for EachMovie.

4.1.3 Vibes Affinity

The Vibes Affinity algorithm [21] is used at several Yahoo properties including Yahoo! Shopping, Yahoo! Auto and Yahoo! Real Estate. The algorithm computes item-item affinity based on conditional probability such as

$$f(i \to j) = Pr(j|i) = \frac{N_{ij}}{N_i} \tag{13}$$

where N_i and N_{ij} denote the number of users who consumed (e.g. clicked) an item i and the number of users who consumed the item i and j. Then preference score of each item j for a user u is computed as following:

$$s_{uj} = \sum_{i \in I_u} f(i \to j) \tag{14}$$

where I_u denotes a set of items the user u has consumed.

To provide cold start recommendation we slightly modified the algorithm. For the partition II (existing item recommendation for new users), we modified the equation (13) and (14) to measure user attribute-item affinity such as

$$f(a \to j) = \frac{N_{aj;like}}{N_{aj}} \tag{15}$$

$$s_{uj} = \sum_{a \in A_u} f(a \to j) \tag{16}$$

where N_{aj} denotes the number of users who have an attribute a and have consumed an item j. A_u is a set of attributes the user u has. $N_{aj;like}$ denotes the number of users who like j among the users who have an attribute a and have consumed an item j. We consider that a user like an item if she rated it higher than the average rating, shown in the Table 1. We call this variation of the affinity model as *Affinity1* hereafter. For the partition III and IV, we measure user attribute-item feature affinity such as

$$f(a \to f) = \frac{N_{af;like}}{N_{af}} \tag{17}$$

$$s_{uj} = \sum_{a \in A_u} \sum_{f \in F_j} f(a \to f) \tag{18}$$

Table 1: Basic statistics of the MovieLens and EachMovie data.

	MovieLens	EachMovie
Range of ratings	1∼ 5	1 ∼ 6
# of users	6,040	61,131
# of items	3,706	1,622
# of ratings	1,000,209	2,559,107
Average rating	3.58	4.34

where F_j denotes a set of features the item j has. N_{af} is a number of user-item pairs in the training data, which contain both a user attribute a and an item feature f. $N_{af;like}$ denotes the number of positive preference pairs (e.g. rating higher than the average rating) in N_{af}. We call this variant affinity model as *Affinity2* hereafter.

4.2 Testing Procedure & Metric

We used two movie rating data sets: MovieLens and EachMovie. In the EachMovie data, we first removed all "sounds awful" ratings since those ratings (which have a weight less than one) were not real ratings but represented users' impressions on items. In addition to rating data, both data sets also contain some user attributes and movie information. As described in Section 2.1, we collected user and movie features for the MovieLens and EachMovie datasets respectively. The features we collected are summarized in Table 2. The age/gender categories are same as those defined in Segmented Most Popular, see Subsection 4.1.2. In MovieLens, there are 21 occupation categories for users and 18 genre categories for movies. The movie-released year was categorized into 13 groups, {>=2000, 1999, 1998, 1997, 1996, 1995, 1994-1990, 80's, 70's, 60's, 50's, 40's, <1940}. In EachMovie, there are two "status" categories for movies, "theater-status" and "video-status". We also grouped users into three location categories based on zip code, including "US", "international" and "unknown".

In addition to item features from data, we used fourteen filterbots [24] as item features for our proposed approach. These bots rate all or most of the items algorithmically according to attributes of the items or users. For example, an actor bot would rate all items in the database according to whether a specific actor was present in the movie or not. The MPBot rates items based on their global popularity computed by the equation (11). The VTBot rates items according to their "user support" such as $r_i = \log V_i/\beta$, where V_i is the number of users who have rated an item i (or user support on the item i) and β is a normalization factor that caps ratings at the maximum rating (5 for MovieLens and 6 for EachMovie). The GenreBot first calculates average ratings of each genre. Then it rates items based on the average rating of the genre which a movie belongs to. If a movie has more than one genre, GenreBot rates the item based on average of genre ratings. We also built eleven SMPBots, which rates items based on their popularity in eleven segments (three gender and eight age-group segments) computed by the equation (12).

We split user ratings into four partitions. We randomly selected half of users as new users and the rest as existing users. Similarly, we randomly split items as new and existing items. Figure 1 illustrates data partition. Then we used partition I for training and partition II, III, and IV for test.

Table 2: User attributes and movie features in MovieLens and EachMovie we used.

		MovieLens	EachMovie
User Attributes	# of gender	2	3
	# of age	7	8
	# of occupation	21	0
	# of location	0	3
	constant feature	1	1
Movie Features	# of year	13	0
	# of genre	18	10
	# of status	0	2
	# from filterbots	12	14
	constant term	1	1

We generated 20 partitions with different random seeds. We used the following test procedure: for each user in the partition II, III, and IV, we clustered items based on ratings given by each user. For example, if a user rated an item a and b five and c and d three, then the user would have two item clusters; one containing a and b and the other containing c and d. We considered only the items rated by each user. Then we randomly sampled one item for each cluster. We filtered out users who have only one item cluster. In such a way, each test user is associated with a set of sampled items with size from two to five and with different ratings. Then we measured $nDCG_k$ as following:

$$nDCG_k = \frac{1}{|U_T|} \sum_{u \in U_T} \frac{DCG_k^u}{IDCG_k^u} \tag{19}$$

$$DCG_k^u = \sum_{i=1}^{k} \frac{2^{R_{ui}} - 1}{log_2(1+i)} \tag{20}$$

where R_{ui}, U_T, and $IDCG_k^u$ are defined as the real rating of a user u on the i-th ranked item, a set of users in the test data, and the best possible DCG_k^u for the user u. We measured $nDCG_k$ where $k = 1$ and 5 and observe similar results.

One potential question might be why not to measure $nDCG_1$ for all items that a user has rated in the test data. The reason is that there might be lots of 5 rated items for certain users in the test data and any algorithm that places one of those 5 rated items at the first place would have the best possible performance. If the number of 5 rated items for a user is larger, then the test becomes easier since algorithms just need to place one of those many 5 rated items at the first place. To remove performance bias to heavy or generous users, we sampled one item for each rating cluster to have only one best item in the test item set. For each of 20 partition sets, we sampled 500 times for each user and average $nDCG_1$ over those 500 runs. Then we reported the mean $nDCG_1$ and the standard deviations over 10,000 runs. All fourteen filterbots were imported as item features when our approach was tested on the partition II (existing item recommendation for new users). For the partition III and IV, only GenreBot was imported.

4.3 Empirical Results

We conducted experiments on three types of cold-start recommendation tasks: (1) recommendation on existing items for new users, (2) recommendation on new items for existing users, and (3) recommendation on new items for new

Table 3: Test results: Average $nDCG_1$ and standard deviation (STD) over 10,000 runs with twenty partitions are measured. Random, Most Popular (MP), Segmented Most Popular (SMP: Age segmentation), two variations of the Affinity model (Affinity1 and Affinity2), and Tensor Pairwise Regression (Pairwise) approaches are tested on three cold-start settings. Bold-face represents the best.

Cold-start setting	Algorithm	MovieLens		EachMovie	
		$nDCG_1$	STD	$nDCG_1$	STD
Existing item recommendation for new users	Random	0.4163	0.0068	0.4553	0.0055
	MP	0.6808	0.0083	0.6798	0.0166
	SMP	0.6803	0.0078	**0.6868**	0.0146
	Affinity1	0.6800	0.0077	0.6698	0.0134
	Affinity2	0.4548	0.0091	0.5442	0.0154
	Pairwise	**0.6888**	0.0078	0.6853	0.0149
New item recommendation for existing users	Random	0.4158	0.0059	0.4539	0.0052
	Affinity2	0.4489	0.0094	0.5215	0.0149
	Pairwise	**0.4972**	0.0145	**0.5821**	0.0176
New item recommendation for new users	Random	0.4154	0.0065	0.4540	0.0046
	Affinity2	0.4439	0.0102	0.5212	0.0145
	Pairwise	**0.4955**	0.0141	**0.5821**	0.0172

users. The first type of cold-start recommendation task is executed when new users request recommendation at any system while the second and third cold-start recommendation usually happens in News domain or newly-launched systems where a recommender is always forced to recommend new items.

In the first recommendation task, we compared our approach to five alternative recommendation methods: Random, Most popular, Segmented Most Popular, and two variations of the Affinity algorithm. We found that SMP and our approach perform better than others but performance differences among MP, SMP, Affinity1 and our approach are not significant. Our results show that item popularity features such as global popularity (MP) or popularity within a segment (SMP) provide much stronger signals than any other item features and it makes MP and SMP hard-to-beat baselines, which is also shown in [1, 11].

In the second and third tasks, since all items we can recommend are new items without any historical ratings, MP, SMP, and Affinity1 cannot work. With absent of item popularity features, we clearly see our approach significantly outperforms over random and Affinity2. We would like to note that our approach can be used to estimate initial guess of item popularity for new items in online recommender systems such as Yahoo! Front Page Today Module [1].

5. CONCLUSIONS

In many real recommender systems, great portion of users are new users and converting new users to active users is a key of success for online enterprisers. We developed hybrid approaches which exploit not only user ratings but also features of users and items for cold-start recommendation. We constructed profiles for user/item pairs by outer product over their individual features, and built predictive models in regression framework on pairwise user preferences. The unique solution is found by solving a convex optimization problem and the resulting algorithms scale efficiently for large scale data sets. Although the available features of items and users are simple and sometimes incomplete in our experiments, our methods performed consistently and significantly better than two baseline algorithms, random and Affinity2, on new user and new item recommendation. As for future work, we would like to extensively compare with other existing variants along the direction of feature-based modeling on ranking quality in cold-start situations.

6. ACKNOWLEDGMENTS

We thank our colleagues, Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Liang Zhang for many useful discussions. We thank MovieLens and EachMovie for publishing their valuable data.

7. REFERENCES

[1] D. Agarwal, B. Chen, P. Elango, N. Motgi, S. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *Advances in Neural Information Processing Systems 21*, 2009.

[2] D. Agarwal and B.-C. Chen. Regression based latent factor models. In *KDD*, 2009.

[3] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *KDD*, 2007.

[4] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In *ACM KDD*, pages 201–212, 1999.

[5] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[6] J. Basilico and T. Hofmann. A joint framework for collaborative and content filtering. In *ACM SIGIR*, 2004.

[7] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[8] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, 2007.

[9] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, pages 46–54, 1998.

[10] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[11] W. Chu and S.-T. Park. Personalized recommendation on dynamic contents using probabilistic bilinear models. In *Proceedings of the 18th international conference on World wide web*, 2009.

[12] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems*, 1999.

[13] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix f actorization. In *ICML*, 2006.

[14] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[15] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.

[16] R. A. Harshman. Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22:30–44, 1972.

[17] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Proc. of the Ninth International Conference on Artificial Neural Networks*, pages 97–102, 1999.

[18] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR*, pages 230–237, 1999.

[19] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[20] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *AAAI*, 2002.

[21] B. Nag. Vibes: A platform-centric approach to building recommender systems. *IEEE Data Eng. Bull.*, 31(2):23–31, 2008.

[22] L. Omberg, G. H. Golub, and O. Alter. A tensor higher-order singular value decomposition for integrative analysis of dna microarray data from different studies. *PNAS*, 104(47):18371–18376, 2007.

[23] T. Pahikkala, E. Tsivtsivadze, A. Airola, T. Boberg, and T. Salakoski. Learning to rank with pairwise regularized least-squares. In *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

[24] S.-T. Park, D. M. Pennock, O. Madani, N. Good, and D. DeCoste. Naíve filterbots for robust cold-start recommendations. In *KDD*, 2006.

[25] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[26] D. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *UAI*, pages 473–480, 2000.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[28] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *ACM SIGIR*, 2002.

[30] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI*, 1995.

[31] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, 2009.

[32] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proc. of The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[33] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[34] H. Wang and N. Ahuja. Efficient rank-r approximation of tensors: A new approach to compact representation of image ensembles and recognition. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[35] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*, 2005.

[36] C. Ziegler, G. Lausen, and L. Schmidt. Taxonomy-driven computation of product recommendations. In *CIKM*, 2004.