Expert-opinions Based Linear Regression Model for Top-N Recommendation

Ming ZHU, Hong-tao ZHANG* and Cai-rong YAN
School of Computer Science and Technology, Donghua University, Shanghai, China
*Corresponding author

Keywords: Recommender system, Top-N recommendation, Expert opinions, Linear regression.

Abstract. Expert-opinions approaches based on CF (collaborative filtering) have demonstrated a successful means for top-N recommendation, such as Expert-CF. However, the choice of the similarity measure used for evaluation of user-expert relationships is crucial for the success of such approaches. In this paper, we present an approach to calculate user-expert similarities by formulating a regression problem which enables us to extract the similarities from the data in a problem specific way. A comprehensive set of experiments is conducted by predicting a subset of the MovieLens data set. We use ratings crawled from a web portal of expert reviews to generate high quality recommendations. The experiments show that the proposed method improves upon the standard Expert-CF model and outperforms other state-of-the-art top-N recommendation approaches in terms of achieving good balance between recommendation quality and speed.

Introduction

As the number of products conforming to the customers’ desires has dramatically increased, how to effectively and efficiently help customers identify the products that best fit their personal tastes has become the urgent problem. In particular, given the user purchase/rating profiles, recommending a ranked list of items for the user so as to encourage additional purchases has the most application scenarios. This leads to the widely used of top-N recommender systems [1] from online shopping websites [2] to video portals. They provide users with a ranked list of N items they will likely be interested in, in order to encourage views and purchases.

Several algorithms for the top-N recommendation problem have been developed [3], including approaches that use latent space models and approaches that rely on neighborhoods. The latent space methods [4] factorize the user-item matrix into lower rank user factor and item factor matrices, which represent both the users and the items in a common latent space. The neighborhood-based methods [5] (user-based or item-based) identify similar users or items. The latent-based methods have been shown to be superior for solving the rating prediction problem, whereas the neighborhood methods are shown to be better for the top-N recommendation problem [5-8]. Among them, the collaborative filtering (CF) is the current mainstream approach used to build web-based top-N recommender systems [9].

CF algorithms assume that in order to recommend items to users, information can be drawn from what other similar users liked in the past. The User-Based CF, for instance, does so by identifying, for each user, a number of similar users whose profiles can then be used to predict recommendations. However, defining similarity between users is not an easy task: the choice of the similarity measure is crucial for the success of such approaches and it's limited by the sparsity and noise in the data and is computationally expensive.

Some studies have suggested that the noise in the users’ explicit feedback can cause a significant part of the error. In recent work, we have found that the behavior of a large population can be predicted by using an extremely small expert (individual that we can trust to have produced thoughtful, consistent and reliable evaluations of items in a given domain) set. Therefore, we aim at using feedback from less noisy sources to build recommendations.

In this work we suggest a combined model that improves prediction accuracy by capitalizing on the advantages of both expert opinions based methods and the linear regression model. To our best knowledge, this is the first time that a single model has integrated the two approaches. Our
experimental evaluation shows that ExpertLR can generate top-N recommendations of high quality at a very high speed over the existing competing top-N recommendation approaches.

The remainder of this paper is organized as follows: In Section 2, a brief review on related work is provided. In Section 3 the proposed methods are described. In Section 4, the materials used for experiments and the results are presented. Finally in Section 5 are the discussions and conclusions.

Related Work

There has been extensive work in the area of top-N recommendation. Here we present a few notable works in the area that have proved to be effective. The methods for top-N recommendation can be broadly classified into two categories, neighborhood-based collaborative filtering (CF) and the model-based methods.

Neighborhood-based CF

The traditional neighborhood-based CF can usually be divided into user-based and item-based approach. The user-based [10] approach is directly used to predict user's interest, complex and unexpected patterns from user's past behaviors and recommend top-N items to other users with similar interests and preferences. Similarly, item-based collaborative filtering [11] methods first identify a set of similar items for each of the items that the user has purchased, and then recommend top-N items based on those similar items. The user/item similarity is calculated from user-item purchase/rating matrix in a collaborative filtering fashion with some similarity measures applied.

a) Expert-based CF: To solve the problem of cold start, data sparsity and expensive computation of traditional neighborhood-based CF, Amatriain et al have developed an CF-based approach [12] to recommend content, based on the opinion of a reduced number of experts. It's able to predict the ratings of a large population by considering a reduced set of expert ratings. The method’s performance is comparable to traditional CF algorithms, even when using an extremely small expert set.

Model-based Methods

Latent factor models comprise an alternative approach to Collaborative Filtering with the more holistic goal to uncover latent features that explain observed ratings. Examples include pLSA [13], neural networks [14], and Latent Dirichlet Allocation [15]. There are various Matrix Factorization (MF) based methods proposed in recent years for building such latent factor models. Cremonesi et al [16] proposed a simple Pure Singular-Value-Decomposition-based (PureSVD) matrix factorization method, which describes users and items by the most principle singular vectors of the user-item matrix. Pan et al and Hu et al proposed a Weighted Regularized Matrix Factorization (WRMF) method formulated as a regularized Least-Squares (LS) problem, in which a weighting matrix is used to differentiate the contributions from observed purchase/rating activities and unobserved ones. Rennie and Srebro proposed a Max-Margin Matrix Factorization (MMMF) method, which requires a low-norm factorization of the user-item matrix and allows unbounded dimensionality for the latent space.

Sparse Linear Method: Ning et al introduced SLIM, which was the first method to compute the item-item relations using statistical learning and has been shown to be able to generate high-quality Top-N recommendations fast. SLIM employs a sparse linear model in which the recommendation score for a new item can be calculated as an aggregation of other items. A sparse m × m aggregation coefficient matrix W is learned from SLIM to make the aggregation very fast. The recommendation score on an unrated item for user u is computed as a sparse aggregation of all the user’s past rated items:

\[
\hat{r}_{ui} = \sum_{i=1}^{m} w_i \cdot \mathbf{v}_i
\]

where \(\hat{r}_{ui}\) is the row-vector of corresponding to user u, \(\mathbf{s}_{ui}\) is the \(ith\) column vector of matrix W, that is estimated by solving the following optimization problem

\[
\text{minimize} \quad \frac{1}{2} \| \mathbf{r}_i - R \mathbf{w}_i \|_2^2 + \frac{\beta}{2} \| \mathbf{w}_i \|_2^2 + \lambda \| \mathbf{w}_i \|_1 \quad \text{subject to} \quad \mathbf{w}_i \geq 0, \mathbf{s}_{ui} = 0
\]
The constants $\beta$ and $\lambda$ are regularization parameters. The non-negativity constraint is used so that the vector estimated contains positive coefficients. The $s_{ii} = 0$ constraint makes sure that when computing the weights of an item, that item itself is not used as this would lead to trivial solutions.

**Proposed Approach**

**Getting Expert Data Sets**

As our method needs expert data sets to generate recommendations. Firstly, we require obtaining a set of ratings from a reduced population of experts. To select the expert, we first establish a definition for an expert. We define an expert as an individual that we can trust to have produced thoughtful, consistent and reliable evaluations (ratings) of items in a given domain. There are a number of approaches to populate a database of expert ratings, ranging from a manually maintained database of dedicated experts to the result of crawling and inferring quantitative ratings from online reviews. The focus of our work is not on extracting the expert ratings, but on using such an external and reduced source of ratings to generate top-$N$ recommendation. Here we give a brief introduction on some methods that have been generally used.

**Mining from Other Data Sources.** One option to obtain experts opinions is to mining data from trusted sources. There are plenty of expert evaluations like songs, movies or books online that include a quantitative rating, it is feasible to crawl the web in order to gather expert ratings. In our work, we have crawled the DoubanMovie web site which aggregates the opinions of movie critics from various media sources, to obtain expert ratings of the movies in the Movielens data set. We crawled the opinions of 2,000 experts, and the ratings correspond to 8,750 of the total of 10,677 movies in the MovieLens data set. Because the movie titles we crawled are all in Chinese, we did much manual work to establish the mapping relations between Chinese movie title in DoubanMovie and the English title in MovieLens. The missing movies had significantly different titles in both databases and were difficult to match. We removed those experts who did not contain at least $\sigma$ ratings of the MoiveLens movies because they had very few ratings and were therefore not adding any improvement to our predictions. Using a threshold of $\sigma = 200$ minimum ratings, we finally kept 342 experts. The relation between $\sigma$ and the number of selected experts is depicted in Figure 1.

![Figure 1. Relation between the minimum ratings threshold and the number of experts selected.](image_url)

**Finding Representative Users in Target System.** While the approach mentioned above relies on a set of predefined experts, representative-based model (RBM) chooses representative users automatically from the existing user population of the target system, rather than some independent third parties. Representatives are those users whose linear combinations of tastes would accurately approximate other users, once the representative users were selected can be used as experts to generate recommendation.
Establishing Regression Model for Top-N Recommendation

The ExpertLR model we proposed is able to make high quality recommendations by formulating a regression problem.

In the ExpertLR method, the recommendation score on an unrated item \( t_j \) for user \( u_i \) is calculated as an aggregation of all experts’ opinions on item \( t_j \),

\[
\tilde{a}_{ij} = w_i^T b_j
\]

(3)

it’s a linear regression model, where \( \tilde{a}_{ij} \) is the predicted rating of user \( u_i \) on un-rated item \( t_j \), \( w_i^T \) is a row vector of regression coefficients representing the similarity/relations between user \( u_i \) and all experts we selected, \( b_j \) is the \( j \)th column of experts ratings matrix \( A \). In this paper, a predicted value is denoted by having a \( \sim \) over it. For a given matrix \( M \) its \( j \)th row is represented by \( m_i^T \) and its \( j \)th column by \( m_j \). Thus, the model utilized by ExpertLR can be presented as

\[
\tilde{A} = W \times B
\]

(4)

where \( A \) is the user-item rating/purchase matrix of size \( m_u \times n \), \( n \) is the number of items, \( m_u \) is the number of users here, \( B \) represents the expert-item rating/purchase matrix of size \( m_e \times n \), whose \( j \)th column corresponds to \( b_j \) in Equation 3. \( W \) is the regression coefficients matrix of size \( m_u \times m_e \), whose \( ith \) row corresponds to \( w_i^T \) in Equation 3, \( m_e (m_e < m_u) \) is the number of experts we selected.

The matrix \( W \) saved all the similarity/relations between users and experts. From the point of view of linear algebra, users’ ratings/tastes are actually the linear combinations of experts', when user \( u_i \) is very similar to an expert \( e_j \), the model will endow \( e_j \) with a larger weight to generate recommendation, and top-N recommendation for \( u_i \) is done by sorting \( u_i \)'s non-rated/purchased items based on their predicted recommendation scores in \( \tilde{a}_i^T \) in decreasing order and recommending the top-N items. Figure 2 illustrates this process.

![Figure 2. The Top-N recommendation process.](image)

Learning User-Expert Relation Matrix \( W \)

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

In order to learn the user-expert relation matrix \( W \), we treat the users’ rating/purchase matrix \( A \) as

the ground-truth item recommendation score, learning the weights using the linear regression method. Given a user-item matrix \( A \), we learn the user-expert relation matrix \( W \) by solving the regularized optimization problem in Equation 5:

\[
\text{minimize} \frac{1}{2} \left\| A - WB \right\|_2^2 + \frac{\beta}{2} \left\| W \right\|_2^2 + \lambda \left\| W \right\|_1,
\]

(5)

the first term \( \left\| A - WB \right\|_2^2 \) measures how well the linear model fits the training data, \( \left\| \cdot \right\| _1 \) and \( \left\| \cdot \right\| _2 \) denotes the frobenius norm, and constants \( \beta \) and \( \alpha \) are \( l_2 - \text{norm} \) and \( l_1 - \text{norm} \) regularization parameters respectively. It is well known that \( l_1 - \text{norm} \) regularization introduces sparsity into the solutions and \( l_2 - \text{norm} \) leads the optimization problem to an elastic net problem. The regularization terms measure model complexity and prevents overfitting, the larger the parameters are, the more severe the regularizations are. The non-negativity constraint is applied on \( W \) such that the learned \( W \) represents positive relations between users and experts.
**Parallel Computing** \( W \). Since every row of \( W \) are independent, the optimization problem in Equation 5 can be decoupled into a set of optimization problems of the form:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| w_i^T B - a_i^T \|_2^2 + \frac{\beta}{2} \| w_i \|_2^2 + \lambda \| w_i \|_1, \\
\text{subject to} & \quad w_i \geq 0, \\
& \quad 1 \leq i \leq m_u,
\end{align*}
\]

(6)

which allows each row of \( W \) to be solved independently. In Equation 6, \( w_i^T \) is the row of \( W \) and \( a_i \) is the \( i \)th row of \( A \). \( B \) is the experts rating matrix. \( \beta_i \) and \( \lambda_i \) are regular term coefficients for user \( u_i \).

Due to the row-wise independence property of \( W \), learning \( W \) can be easily parallelized. The optimization problem of Equation 4 can be solved using coordinate descent and soft thresholding. The process is shown in Algorithm 1.

**Penalty Coefficient for Popular Items.** Hot items do not indicate that two users have similar interests. In the case of books, if both users have ever bought *Xinhua Dictionary*, it does not suggest that they have similar interests, because the vast majority of Chinese people have bought *Xinhua Dictionary* as children, but if both users have bought the *Introduction to Data Mining*, their interests may be considered similar, because only those who study data mining will buy the book. In other words, the similarity of their interests is better illustrated by the fact that two users have taken the same action on unpopular items. ExpertLR punish those popular items by adding weights to each sample to reduce the influence of popular items on user-expert similarity/relation weights, the optimization equation can be changed to the following form:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \left\| \sum_{j=1}^{n} c_j (w_i^T b_j - a_{ij}) \right\|_2^2 + \frac{\beta_i}{2} \| w_i \|_2^2 + \lambda_i \| w_i \|_1, \\
\text{subject to} & \quad w_i \geq 0, \\
& \quad 1 \leq i \leq m_u,
\end{align*}
\]

(7)

where \( c_j \) is the penalty coefficient for item \( t_j \), \( N(j) \) is the number of users who show interest in \( t_j \) in user rating/purchase matrix \( A \). The influence of popular items on the similarity coefficient of user experts during training is weakened by combining the penalty coefficient and the sum of squares residuals, so we can get more reasonable similarity coefficients between users and experts. The overview of ExpertLR is shown in Algorithm 2.

<table>
<thead>
<tr>
<th>Algorithm 1 ExpertLR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Require:</strong> Set ( W ) with random value; ( \beta_i, \lambda_i ) for every user ( u_i ); loss threshold ( \delta )</td>
</tr>
<tr>
<td>1: for user ( u_i = u_1 \rightarrow u_{m_u} ) do</td>
</tr>
<tr>
<td>2: while loss &gt; ( \delta ) do</td>
</tr>
<tr>
<td>3: computer the training loss of user ( u_i ) with penalty factor</td>
</tr>
<tr>
<td>4: update ( a_i^T )</td>
</tr>
<tr>
<td>5: end while</td>
</tr>
<tr>
<td>6: end for</td>
</tr>
</tbody>
</table>
**Experimental Evaluation**

Based on the previously described data, we measure how well the experts predict the ratings of MovieLens users. We present the performance of our methods by comparing them with other popular top-N recommendation methods. All the top-N recommendation methods use user-item rating information during learning. For all the methods, we conducted an exhaustive grid search to identify the best parameters to use, making sure that any difference in performance is due to the algorithms themselves, and not due to the implementation.

**Data Sets**

We used experts data set we crawled from DoubanMovie and MovieLens data set for recommendation in our experiments. The experts data set contains 2,000 experts with about 100 thousand ratings, we finally keep 342 experts by setting a threshold $\sigma = 200$. MovieLens data set contains over 10 million ratings by over 60,000 users on about 10,000 movies.

**Evaluation Metrics and Methodology**

We employed leave-one-out cross-validation to evaluate the performance of the proposed model. For each user, we randomly selected an item, which we placed in the test set. The rest of the data comprised the training set. We measure the performance by computing the number of times the single left-out item was in the top-N recommended items for this user and its position in that list. The quality measures used are the hit-rate (HR) and average-reciprocal hit rank (ARHR).

HR is defined as

\[
HR = \frac{\text{#hits}}{\text{#users}}
\]  

(8)

and ARHR is defined as

\[
ARHR = \frac{1}{\text{#users}} \sum_{i=1}^{\text{#hits}} \frac{1}{p_i}
\]  

(9)

where “#user” is the total number of users ($m_u$), $p$ is the position of the item in the list, where $p = 1$ specifies the top of the list, and “#hits” is the number of users whose item in the test set is present in the size-N recommendation list.

**Performance of the Proposed Methods**

**Top-N.** The results presented throughout the paper show the performance of our algorithms for a list of size 10. The recommendation list can be of different sizes. In this section, we describe how the performance of our method is affected by the size. We choose N to be quite small because users do not look past the very top presented recommendations in a list. In Figure 3, we can see the HR of ExpertLrpc, while using the parameters with the best results for the different sizes of top-N list. We can see that as N increases, the performance of our method increases as well, which is expected, as there
is higher probability that the hidden item of our test set will be in the top-N list. The impact of the size of the recommendation list N on ARHR is similar.

![Graph showing the relationship between recommended quality and recommendation size N](image)

Figure 3. Relation between recommendation size N and HR, ARHR.

**Performance Against Computing Approaches.** Table 1 presents the performance of the competing algorithms PureSVD, BPR-MF and SLIM versus the performance of our methods ExpertLR and ExpertLRpc. The table presents the best HR and ARHR achieved, along with the set of parameters for which they were achieved. We can see that Expertpc outperforms all competing approaches on MOvieLens data set, in terms of HR and ARHR. We can also see that ExpertLR, which is our simplest method, still outperforms the best competing approach, which shows that expert method helps top-N recommendation quality. Because our model uses a very small amount of expert data for training, the training process is also very fast.

<table>
<thead>
<tr>
<th>Method</th>
<th>params</th>
<th>HR</th>
<th>ARHR</th>
<th>mt</th>
<th>tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemKNN</td>
<td>25</td>
<td>-</td>
<td>0.234</td>
<td>0.109</td>
<td>2.02m</td>
</tr>
<tr>
<td>SLIM</td>
<td>5</td>
<td>2</td>
<td>0.312</td>
<td>0.151</td>
<td>51.2(h)</td>
</tr>
<tr>
<td>Expert-CF</td>
<td>10</td>
<td>0.02</td>
<td>0.182</td>
<td>0.087</td>
<td>24.3(s)</td>
</tr>
<tr>
<td>PureSVD</td>
<td>182</td>
<td>12</td>
<td>0.291</td>
<td>0.142</td>
<td>1.62(m)</td>
</tr>
<tr>
<td>BRP-MF</td>
<td>362</td>
<td>0.12</td>
<td>0.283</td>
<td>0.121</td>
<td>4.83(h)</td>
</tr>
<tr>
<td>ExpertLR</td>
<td>6</td>
<td>1.2</td>
<td>0.317</td>
<td>0.154</td>
<td>12.4(m)</td>
</tr>
<tr>
<td>ExpertRpc</td>
<td>-</td>
<td>-</td>
<td>0.326</td>
<td>0.158</td>
<td>24.5(m)</td>
</tr>
</tbody>
</table>

For each method, columns corresponding to the best HR and ARHR nd the et of parameters with which they are achieved are shown. For methods itemKNN, the parameters are number of neighbors. For SLIM, the parameters are the $l_2$ regularization parameter $\beta$ and the $l_1$ regularization parameter $\lambda$. For Expert-CF, the parameters are the confidence threshold $\tau$ and the similarity threshold $\delta$. For PureSVD the parameter the parameters are the number of singular values and the number of iterations during SVD. For BPR-MF, the parameters are the dimension of the latent space and learning rate, respectively. For ExpertLR, the parameters are the parameters are the $l_2$ regularization parameter $\beta$ and the $l_1$ regularization parameter $\lambda$. For ExpertLRpc, every user u i use efficient $l_2$ regularization parameter $\beta_u$ and the $l_1$ regularization parameter $\lambda_u$, because there are too many parameters, we’re not going to list them here. Columns corresponding to HR and ARHR present the hit rate and average reciprocal hit-rank, respectively. Columns corresponding to mt and tt present the time used by model learning and recommendation, respectively. The mt/tt numbers with (s), (m) and (h) are time used in seconds, minutes and hours, respectively.
Conclusion

In this paper, we have proposed an approach for top-N recommendation, based on the opinion of a reduced number of experts. The proposed method is able generate high quality top-N recommendation fast. ExpertLR employs a linear regression model in which the recommendation score for a new item can be calculated as an aggregation of experts’ tastes. An user-expert similarity/relation matrix $W$ is learned for ExpertLR to predict the recommendation score. $W$ is learned by solving a regularized optimization problem. Our results show that the method’s performance is comparable to competing top-N algorithms, even when using an extremely small expert set.

Acknowledgment

This research is supported by the Dongguan professional town innovation service platform construction project, Dongguan Humen garment collaborative innovation center, and the Guangdong provincial cooperation innovation and platform environment construction special funds (Grant No. 2014B090908004).

References


