Local Item-Item Models for Top-N Recommendation

Dheeraj Singh, Kuldeep Singh, and Shubham Goyal

Indraprastha Institute of Information Technology, Delhi
dheeraj17011@iiitd.ac.in, kuldeep17022@iiitd.ac.in,
shubham17055@iiitd.ac.in

Abstract. Item-item based methodologies among neighborhood strategies have given great exhibitions for the Top-N proposal. A few neighborhood strategies incorporate kNN, and SLIM(Sparse Linear Method) beats general user based techniques. The SLIM approach appraises a solitary model for every one of the user and subsequently not ready to accomplish great outcomes. SLIM can be additionally reached out to Global and local SLIM where the predictions will be made considering the factor of similarly invested user sets. Users combination and assignment to local models are streamlined together to enhance the performance and accomplish better outcomes.

Keywords: SLIM · Clustering · GLSLIM.

1 Introduction

Top-N recommender frameworks[1] are generally utilized in online portals and shopping sites. Clients are given the best n items which they are likely to keen on empowering on the web buys. Calculations for best N proposal issue can be separated into a latent space model and neighborhood-based techniques. Latent space methods[2] factorize the user item lattice into lower rank item factor and item factor matrices, which speaks to both the users and items in common latent space. Then again, neighborhood-based methods[3] distinguish comparative items or users. From the trials performed, it is seen that latent space strategies perform better to rate prediction while neighborhood-based techniques perform better for Top-N proposal problem[4][3].
Neighborhood-based methods can be classified into user based schemes and item based schemes(kNN and SLIM Sparse Linear Methods[16]). The item-item techniques have been shown to outperform the user based schemes. However, for item based approaches based on SLIM estimates a single model for all the users. In many cases, there are differences in users behavior, which cannot be captured by a single model. There could be a pair of items that are incredibly similar for a specific user subset, while they have low similarity for another user subset. Thus, a global method needs to be incorporated which tend to move towards some average value.

The paper presents GLSLIM method, an extension of SLIM combines both local and global SLIM in a personalized way that identifies the appropriate user subsets.

2 Methods

2.1 SLIM

It estimates a sparse matrix m x m aggregation coefficient matrix S. The recommendation score on an unrated item i for user u is computed as a sparse aggregation of all the users past rated items:

$$\tilde{r}_{ui} = r_u^T s_i \tag{1}$$

r_{ui} :is the row-vector of R corresponding to user u
s_i :is the ith column vector of matrix S
which is estimated by solving an optimization problem.

In contrary to SLIM, GLSLIM focuses on the item-item model, computes global model and has personalization factor for each user determining the interplay between global and local information. It also updates the assignment of users to subsets.

3 Proposed Method

3.1 Motivation

A global item-item mode is not sufficient to capture the preference of different set of users. Especially in cases where different user subsets have diverse and sometimes opposing preferences. For example look at Figure 1. This example shows the training matrix R of two different datasets that both contain two different set of users. Item i is the target item. In this example, item-item cosine similarity is used to compute predictions.

In figure 1(a), there are some items that are rated by only one of the subset, whereas some are rated by both sets of the users. So, item c and i will have different similarities when rated for subset A than when estimated for subset B than for all users. The similarity between c and i will be of average value when computed globally i.e. the diverse preference of different user subsets namely A and B would be missed. Local item-item similarities are used to capture the diverse preference of the users. Also, in dataset 2 shown in figure 1(b) have no item rated by both of users so this dataset will no be affected much by using local item-item similarity model than by using the global model.

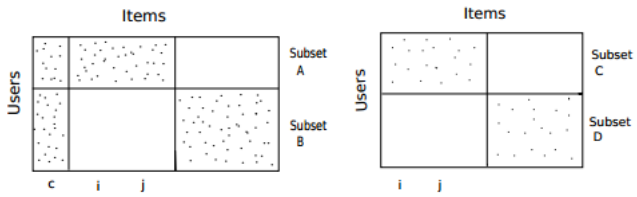

Fig. 1. (a) Local item-item models improve upon global item-item model. (b) Global item-item model and local models yield the same results.

3.2 Overview

GSLIM method uses user-specific models along with a global model to compute top N recommendations. First, we estimate a global-global item-item coefficient matrix S and k local item-item similarities matrices S^{p_u}, where k is the number of user subsets and $p_u \in \{1, ...k\}$ is the index of user subset.

The rating of item i for user u belonging to p_u will be predicted by using following equation:

$$\tilde{r}_{ui} = \sum_{l \in R_u} g_u S_{li} + (1 - g_u) S_{li}^{p_u}$$

Where, S_{li} represents the global item-item similarity between item l rated by user u and target item i, $S_{li}^{p_u}$ represents the local item-item similarity between item l and target item corresponding to the local model of user subset p_u. The term g_u lies in interval $[0, 1]$ and control the weight of local and global model. For top n recommendation items are sorted based on $\tilde{r}_{ui}$ for every unrated item.

3.3 Estimating the item-item models

The users are separated into subsets by using clustering algorithm or randomly. Initially, g_u is set to be 0.5 for all users,and then the coefficient matrices S and $S^{p_u} with \mathrm{p}_u \in 1, ...k\}$ are estimated. Two vectors g and g' each of size n is used. g contains the personalized weight g_u for every user u and the vector g' contains the complement of the personalized weight (1 - g_u) for every user u. The users are assigned into k subsets, the training matrix R is split into k training matrices R^{p_u} of size $n \times m$. Every row u of R^{p_u} will be the uth row of R, if the user u who corresponds to this row belongs in the p_uth subset. If the user u does not belong to the p_uth subset, then the corresponding row of R^{p_u} will be empty, without any ratings. When estimating the local model ^{p_u}, only the corresponding R^{p_u} will be used. Following SLIM, the item-item coefficient matrices can be calculated per column, which allows for the different columns (of both the global and the local coefficient matrices) to be estimated in parallel.

After estimating the models, GLSLIM assign each user u to all possible cluster, and compute weights g_u that the user would have if got assigned to different clusters. Then, for every cluster p_u and user u, the training error is computed.

The cluster for which this error is the smallest is the cluster to which the user is assigned.g_u is equated using the following equation:

$$g_u = \frac{\sum_{i=1}^{m}(\sum_{l\in R_u} S_{li} - \sum_{l\in R_u} S_{li}^{p_u})(r_{ui} - \sum_{l\in R_u} S_{li}^{p_u})}{\sum_{i=1}^{m}(\sum_{l\in R_u} S_{li} - \sum_{l\in R_u} S_{li}^{p_u})^2} \tag{2}$$

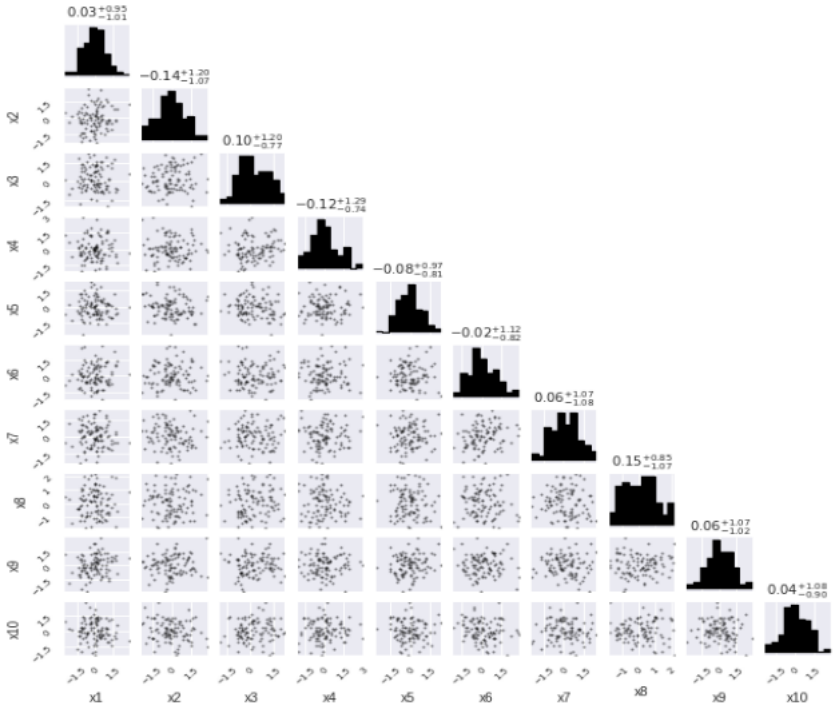

Fig. 2. Data visualization for number of clusters = 10 on Ml(100k) dataset.

4 Evaluations

4.1 Datasets

ML dataset corresponds to MovieLens 100k dataset, which represents movie ratings. The jester dataset corresponds to an online joke recommender system. Flickscore dataset depicts likes and dislikes towards different movies given by 924 users. FilmTrust dataset also represents ratings of movies.

Dataset	#users	#items	Density(%)
$Movielens(100k)$	943	1682	6.30%
$Jester$	23500	100	21.34%
$Flickscore$	924	2850	0.007%
$FilmTrust$	1508	2071	1.14%

Table 1. Dataset used

4.2 Methodology

We evaluated the performance of the model on Movielens(100 k), Jester, FilmTrust and FlickScore data.
We deployed leave one out cross validation to assess the performance of the model. The performance can be measured by computing the number of times the single left-out item was in the top-N recommended items for the user and its position in the list. Quality measures used are hit rate(HR) and average reciprocal hit rank(ARHR)

$$HR = \frac{\#hits}{\#users} \tag{3}$$

$$ARHR = \frac{1}{\#users} \sum_{i=1}^{\#hits} \frac{1}{p_i} \tag{4}$$

Here, p = 1 specifies the top of list.

5 Results

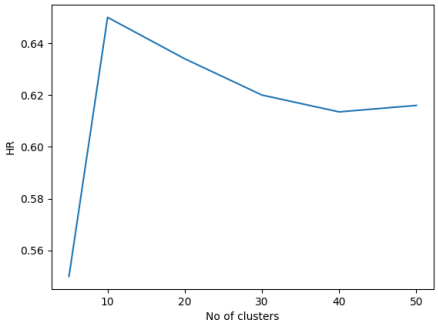

Fig. 3. Number of clusters vs performance(HR) on jester dataset.

As we can see in the figure 3, it depicts the performance of HR vs. the varied number of clusters on Jester Dataset. From this, one can draw the conclusion that the maximum performance is achieved by fixing the cluster group to 10 and by increasing clusters we only increase the error.

Visualization of data in a high-dimensional space is always a difficult problem. One solution that is commonly used is to inspect all of the 1D and 2D projections

of the data. So we generated a plot with all of the 2D projections as scatter plots and histograms of the 1D projections. For Movielens Dataset 100k, figure 2 depicts the data visualization for number of clusters(K) = 10.
All the results shown in the table are for top N = 20. The performance of the method proposed by [6] is evaluated on the datasets mention above and show in the table 5.

Dataset	K = 10		K = 20	
	HR	ARHR	HR	ARHR
Movielens	0.31	0.045	0.28	0.042
Jester	0.65	0.08	0.62	0.074
Flickscore	0.22	0.056	0.205	0.054
FilmTrust	0.538	0.096	0.57	0.093

Table 2. Comparison between approaches used with number of clusters, K = 10, 20

All associated resources for the project can be found on [7]

6 Conclusion and Limitations

The paper proposes an item-based Top-N Recommendation model. It makes use of the preference bias of different user subset. There is also a personalized weight associated with each user for global and local similarity matrices. The main limitation of the paper is the quality of the clusters formed. The quality of clusters formed are not considered in the paper[6]. Randomly selected clusters are chosen without checking whether chosen clusters were closer to optimal clusters selection or not.

References

1. G. Adomavicius and A. Tuzhilin,Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734749, 2005.
2. P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of the fourth ACM conference on Recommender systems, pages 3946. ACM, 2010.
3. M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS), 22(1):143177, 2004.
4. F. Aiolli. A preliminary study on a recommender system for the million songs dataset challenge. PREFERENCE LEARNING: PROBLEMS AND APPLICATIONS IN AI, page 1, 2012.
5. X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In Data Mining (ICDM), 2011 IEEE 11th International Conference on, pages 497506. IEEE, 2011.
6. Evangelia Christakopoulou and George Karypis, Local Item-Item Models for Top-N Recommendation, In proceedings of 10th ACM Conference on Recommender Systems (RecSys), 2016
7. `https://github.com/imshubhamgoyal/top_n_recommendation`