Report : Local Item-Item Models for Top-N Recommendation

Abstract

Item-item based methods within neighborhood strategies have demonstrated superior performance in Top-N recommendations. These strategies, such as kNN and SLIM (Sparse Linear Method), have outperformed general user-based techniques. While SLIM estimates a single model for each user, it often falls short in delivering satisfactory results. However, SLIM can be extended to incorporate global and local considerations, where predictions are based on similarities among groups of users with shared interests. By optimizing the aggregation and assignment of users to local models, performance is enhanced, resulting in improved outcomes. In simpler terms, SLIM can be adapted to consider both overall trends and specific user groups, leading to better recommendation results.

1 Introduction

Top-N recommender systems are integral to online platforms, offering users tailored lists of N items to enhance engagement and drive purchases. These systems leverage various algorithms, including latent-space models and neighborhood-based methods, to provide personalized recommendations [3].

Latent-space methods decompose the user-item matrix to capture latent factors of users and items, while neighborhood-based methods identify similar users or items. While latent methods excel in predicting ratings, neighborhood methods perform better in Top-N recommendations [4, 8, 13, 16].

Among neighborhood methods, item-based approaches like item k-NN and SLIM have shown promise, outperforming user-based schemes [8, 16]. However, these approaches often rely on a single model for all users, failing to capture diverse preferences.

To address this, GLSLIM combines global and local SLIM models, automatically identifying user subsets. Experimental results show significant improvements in recommendation quality, up to 17% [3].

2 Literature Survey

In this paper, the author propose a sparse linear method for top-N recommendation, termed SLIM, which efficiently generates high-quality recommendations. SLIM utilizes a sparse linear model where the recommendation score for a new item is computed through an aggregation of other items. This aggregation is facilitated by learning a sparse aggregation coefficient matrix W, which is optimized using 1 and 2-norm regularization to introduce sparsity into W.

The author partitioning approach aims to achieve three objectives: reducing computation time, enabling parallel computation of predictions, and improving prediction accuracy. The author experimented with the kMetis graph partitioning algorithm, which showed promise in achieving these goals. Although the accuracy of predictions on kMetis-generated partitions was not as high as the unpartitioned base case, it outperformed random partitioning and was comparable to genre partitioning in terms of accuracy and coverage.

Partitioning the item space into smaller clusters reduces the time required for individual prediction computations. Moreover, since each partition is independent, predictions can be computed in parallel, further enhancing computational efficiency.

While the author's hypothesis suggested that clustering similarly rated movies could improve prediction accuracy, author's experiments showed inconsistent results. Only in a few cases did partitioned items exhibit higher accuracy compared to the unpartitioned base case. This discrepancy could be attributed to the nature of rating correlation, which measures rating similarity rather than content similarity. Additionally, restricting items to a single cluster may overlook items with predictive value across multiple clusters, potentially reducing overall accuracy..

3 Methods and Approaches

3.0.1 SLIM (Sparse LInear Method)

Introduced by Ning et al. [ning2011slim], pioneered the computation of item-item relations through statistical learning, emerging as a leading approach for top-N recommendation. SLIM estimates a sparse aggregation coefficient matrix S of size $m \times m$. The recommendation score for an unrated item i by user u is calculated as the sparse aggregation of the user's past rated items:

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

Here, r_u^T represents the row-vector of matrix R corresponding to user u, and s_i denotes the ith column vector of matrix S, which is estimated by solving an optimization problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2}||r_i - Rs_i||_2^2 + \frac{\beta}{2}||s_i||_2^2 + \lambda||s_i||_1 \\ &\text{subject to } s_i \geq 0 \text{ and } s_{ii} = 0 \end{aligned}$$

where β and λ are regularization parameters. The non-negativity constraint ensures that the estimated vector contains positive coefficients, while $s_{ii} = 0$ constraint ensures that an item's weight is not computed using itself, preventing trivial solutions.

This optimization framework optimizes the item-item relations to enhance recommendation quality while controlling overfitting through regularization.

3.1 To estimate the item-item models

We begin by partitioning users into subsets using either a clustering algorithm or randomly. Initially, we set the personalized weight g_u to 0.5 for all users to ensure an equal contribution of the global and local components. We then estimate the coefficient matrices S and S_{pu}, where pu ranges from 1 to k, the number of subsets. This process involves splitting the training matrix R into k training matrices R_{pu}, each corresponding to a subset pu. The local model S_{pu} is estimated using only the corresponding R_{pu} matrix. Following the approach of SLIM, the item-item coefficient matrices are calculated per column, allowing parallel estimation of columns for both global and local coefficient matrices. To estimate the

i^{th} column of S (s_i) and S_{pu} (s_{pu_i}), GLSLIM solves the optimization problem described by Equation

$$\text{minimize } s_i, \{s_{1i}, ..., s_{ki}\} \frac{1}{2}||r_i - g \circ R s_i - g_0 \circ \sum_{pu=1}^{k} R_{pu} s_{pu_i}||_2^2 + \frac{1}{2}\beta_g ||s_i||_2^2 + \lambda_g ||s_i||_1 + \sum_{pu=1}^{k} \frac{1}{2}\beta_l ||s_{pu_i}||_2^2 + \lambda_l ||s_{pu_i}||_1$$

subject to
$s_i \geq 0$,
$s_{pu_i} \geq 0$ for all $pu \in \{1, ..., k\}$,
$s_{ii} = 0$,
and $s_{pu_{ii}} = 0$ for all $pu \in \{1, ..., k\}$.
Here, r_i represents the i^{th} column of R, β_g and β_l are the l_2 regularization weights for S and S_{pu} respectively, and λ_g and λ_l control the sparsity of S and S_{pu} respectively. By incorporating different regularization parameters for the global and local sparse coefficient matrices, we introduce flexibility into the model. This allows us to control which of the two components - global or local

3.2 Evaluation 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

$$HR = \frac{\#hits}{\#users},$$

and ARHR is defined as

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

where "#users" is the total number of users (n), 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.

3.3 How we will update s_i and s_i^{pu}?

After solving the optimization problem using coordinate descent and soft thresholding, the update rules for s_i and s_i^{pu} can be derived as follows:

For s_i, the update rule is obtained by taking the derivative of the objective function with respect to s_i and setting it to zero. This results in the soft thresholding operation applied to the sum of the residuals, regularized by the appropriate weights:

For si

$$\min_{si} \frac{1}{2} \left\| r_i - gR - g' \sum_{pu=1}^{k} R^{pu} s_i^{pu} \right\|_2^2 + \quad \frac{1}{2} B_g \|s_i\|_2^2 + \lambda_g \|s_i\|_1 + \quad \sum_{pu=1}^{k} \frac{1}{2} B_1 \|s_i^{pu}\|_2^2 + \lambda_l \|s_i^{pu}\|_1$$

Subjected to

$$\begin{aligned} & s_i \geq 0, \quad \forall pu \in \{1, 2 \ldots k\} \\ & s_i^{pu} \geq 0, \quad \forall pu \in \{1, 2 \ldots k\} \\ & s_{ii} = 0 \\ & s_{ii}^{pu} = 0, \quad \forall pu \in \{1, 2 \ldots, k\} \end{aligned}$$

$$\nabla si = \left((gR)^\top gR + I\beta_g\right) s_i + \lambda_g \nabla \|s_i\| \; - (gR)^\top r_i - (gR)^\top \sum \left(g' R^{pu}\right) s_i^j \Big)$$

Put $\nabla si = 0$

$$\left((gR)^\top gR + I\beta_g\right) s_i = S\left((gR)^\top r_i - (gR)^\top \sum \left(gR^{pu}\right) s_i^j, \lambda_g\right)$$

$$\mathrm{s}_i = \left((gR)^\top gR + I\beta_g\right)^{-1} S\left((gR)^\top r_i - (gR)^\top \sum \left(g' R^{pu}\right) s_i^j, \lambda_g\right)$$

For s_i^{pu}

$$s_i^{pu} = \left((\mathrm{g}' R^{pu})^\top g' R^{pu} + IB_l\right)^{-1} S\left((g' R^{pu})^\top r_i - (g' R^{pu})^\top gRs_i \sum \left((g' R^{pu})^\top g' R^{pu}\right) s_i^j, \lambda_l\right)$$

The main role of soft thresholding function to find the difference between the two vector that is given to it (i.e. S(X,A)) and then it give the maximum of 0 and the differnce between the vectors

4 Work Done

I have carefully studied the paper and delved into the mathematical concepts underpinning Algorithm 1, GLSLIM. This algorithm focuses on generating personalized recommendations by assigning users to clusters and iteratively updating the personalized weights (g_u) for each cluster. Initially, each user's weight is set to 0.5 to ensure an equal contribution from the global and local models. The process begins by clustering users using CLUTO, a clustering algorithm. Then, the algorithm iterates until a certain criterion is met, typically until the number of users who switch clusters drops below a certain threshold. Within each iteration, the algorithm estimates the item-item coefficient matrices S and S_{pu} for each cluster using Equation 4. For each user and cluster pair, the personalized weight g_u is computed. Additionally, the training error is computed to assess the model's performance. Finally, each user is assigned to the cluster with the smallest training error, and the corresponding personalized weight g_u is updated accordingly. This iterative process continues until the stopping criterion is satisfied. By implementing this algorithm and understanding its mathematical underpinnings, I gained insights into how personalized recommendations can be efficiently generated using a combination of global and local models.

5 Data set Details

We evaluated the performance of our method on different datasets, whose characteristics are shown in Table 1.

- **Groceries Dataset:** Transactions of a local grocery store. Each user corresponds to a customer, and the items correspond to the distinct products purchased over a period of one year.

Table 1: Dataset Characteristics

Name	#Users	#Items	#Transactions	Density
groceries	63,034	15,846	2,060,719	0.21%
ml	69,878	10,677	10,000,054	1.34%
jester	57,732	150	1,760,039	20.32%
flixster	29,828	10,085	7,356,146	2.45%
netflix	274,036	17,770	31,756,784	0.65%

- **MovieLens (ml) Dataset:** Represents movie ratings from the MovieLens 10M dataset [movie$_lens$].

- **Jester Dataset:** Corresponds to an online joke recommender system [jester$_dataset$].

- **Flixster Dataset:** Subset of the original Flixster dataset [flixster$_dataset$], $consistingofmovieratingsfromasocialmovies$
$fiveusers.$

- **Netflix Dataset:** Subset of the original Netflix dataset [netflix$_dataset$], $containinganonymousmovieratings.Usersinthis$

Nature and Type of Data:

Datasets consist primarily of user-item interactions, such as ratings or purchases, treated as numerical data.

Data Pre-processing Techniques:

Pre-processing includes handling missing values, filtering out inactive users/items, and normalization.

Data Procurement:

For data procurement, I obtained the dataset from Kaggle, a platform for hosting datasets. The dataset contained a wide range of information, including user interactions with various items. To focus specifically on the user-item ratings, I selected relevant rows and columns from the dataset. This ensured that our analysis and experiments were centered around the user-item rating interactions, aligning with the objectives of our study.

Usage in Experiments:

In the experiments, I utilized the MovieLens dataset, which consists of user interactions with movies. Each entry in the dataset includes a user ID, movie ID, rating assigned by the user to the movie, and a timestamp indicating when the rating was given. As the timestamp information was not relevant to our experiment, it was omitted from the analysis. Therefore, the dataset used for our experiments primarily included user IDs, movie IDs, and corresponding ratings.

6 Experiments

6.1 Experiment Setup and Methodology

For our experiment, we initially obtained the dataset from Kaggle, consisting of user interactions with various items, such as movies. We focused on user-item rating interactions, so we selected relevant rows and columns from the dataset, which included the `User_ID`, `movie_ID`, and `rating`. The `timestamp` column was dropped as it was not relevant to our analysis.

We then transformed the dataset into a binary matrix using pandas. This matrix represented whether a user had rated a movie or not. Each row represented a user, each column represented a movie, and the cells contained binary values indicating whether a user had rated a movie or not.

Next, we performed clustering on the binary matrix using the KMeans algorithm from the scikit-learn library. We specified the number of clusters as three and obtained cluster labels for each user.

6.2 Algorithm Implementation

We implemented the GLSLIM algorithm to estimate the item-item models for recommendation. The algorithm involves assigning an initial weight to every user, clustering users based on their ratings, and iteratively updating the weights and cluster assignments until convergence.

In each iteration, we estimated the coefficient matrices S and S_{pu} for each cluster using a sparse linear model. We used coordinate descent and soft thresholding to solve the optimization problem involved in estimating the coefficient matrices.

Finally, we evaluated the performance of our recommendation model using leave-one-out cross-validation. We computed the hit rate (HR) to measure how many times the left-out item was in the top-N recommended items for each user.

7 Results

The HR (Hit Rate) of 0.19 obtained when I consider first 5000 rows of the dataset in which approx 40 users and 2081 items are present of the MovieLens dataset experiment indicates that, on average, 19 percentage of the left-out items were included in the top-N recommended items for each user during the leave-one-out cross-validation process.

This metric provides insight into the effectiveness of the recommendation model in accurately predicting items that users are likely to interact with. A higher HR value signifies better performance of the recommendation algorithm in suggesting relevant items to users based on their preferences and historical interactions.

Therefore, the HR value of 0.19 suggests that the recommendation model implemented using the GLSLIM algorithm achieved moderate success in accurately predicting items that users would likely rate highly or interact with positively, based on the MovieLens dataset

8 Future Work

In our forthcoming research, we aim to extend our investigation by implementing additional recommendation algorithms, including LSLIM and GLSLIMr0, alongside our existing GLSLIM approach. This expansion will enable a comprehensive comparative analysis of the performance and efficacy of these algorithms in generating top-N recommendations. By applying a standardized evaluation metric, such as hit rate, we intend to quantitatively assess the recommendation quality achieved by each algorithm.

Furthermore, we plan to delve deeper into the impact of varying cluster sizes on the performance of the GLSLIM algorithm. Through systematic experimentation across different cluster configurations, we aim to elucidate how the number of clusters influences the recommendation effectiveness and scalability of GLSLIM. This analysis will provide valuable insights into the optimal cluster settings for maximizing recommendation performance.

By rigorously evaluating the performance of multiple recommendation algorithms and systematically analyzing the effect of cluster sizes, our research endeavors to offer a comprehensive understanding of the strengths and limitations of each approach. Ultimately, this comparative study will contribute to advancing the field of recommendation systems by identifying the most effective algorithms and informing practitioners about their practical applicability in real-world scenarios

9 Work Done After Mid-term Review

After the midterm review I have perform the LSLIM and GLSLIMr0 algorithms, in LSLIM algorithm, Initially we randomly make clusters for users, it iteratively refines recommendations. For each user cluster, it estimates local item-item coefficient matrices, capturing diverse preferences. Users are then assigned to clusters based on minimal training error, optimizing personalized recommendations. This iterative process continues until convergence, ensuring stable user assignments. LSLIM's adaptive approach improves recommendation quality by tailoring models to user subsets, offering more effective and personalized recommendations compared to global models while in GLSLIMr0 we are refining the item-item similarity matrix not the clusters,we just make the clusters randomly or using CLUTO and then we are not refining it.

I had perform these algorithms using different number of clusters i.e. 3,4,5 clusters and use first 2200 rows of movieLens dataset in which 18 users and 1406 items are present.

9.1 Result and comparison of algorithm

Algorithm	**GLSLIM**	**LSLIM**	**GLSLIMr0**
Clusters	3 — 4 — 5	3 — 4 — 5	3 — 4 — 5
Hit-rate	0.0 — 0.0 — 0.09	0.0 — 0.0 — 0.058	0.02 — 0.025 — 0.025

Table 2: Performance Comparison of Algorithms

We can see that when we consider 5 clusters for same number of users and item of dataset, GLSLIM is performing better than LSLIM and GLSLIMr0 When we consider 3 or 4 number of clusters our GLSLIMr0 algorithm is performing better than GLSLIM and LSLIM As we are increasing number of clusters and users the algorithms are performing better than when we consider less number of users and number of clusters

10 Conclusion

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.

In addition to the proposed method's effectiveness in outperforming competing top-N recommender methods, our comparative analysis sheds light on the performance variations across different clustering configurations. When employing five clusters for datasets with the same number of users and items, GLSLIM demonstrates superior performance compared to LSLIM and GLSLIMr0. Conversely, for three or four clusters, GLSLIMr0 exhibits better performance metrics. Furthermore, our findings indicate that increasing the number of clusters and users generally enhances algorithm performance, underscoring the scalability and adaptability of our approach. These insights contribute to a deeper understanding of the algorithm's behavior and highlight its potential for recommendation systems

References

1. X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In Data Mining (ICDM). In 11th International IEEE Conference on Intelligent Systems, 2011. [Link]

2. M. Connor and J. Herlocker. Clustering items for collaborative filtering. In Proceedings of the ACM SIGIR workshop on recommender systems, volume 128. Citeseer, 1999. [Link]

3. Collaborative filtering https://en.wikipedia.org/wiki/Collaborative_filtering

4. The K-Nearest Neighbors (KNN) algorithm Introduction to K-Nearest Neighbors Algorithm on Analytics Vidhya.