An Incremental Approach for Collaborative Filtering in Streaming Scenarios

Rama Syamala Sreepada and Bidyut Kr. Patra

Department of Computer Science and Engineering, National Institute of Technology Rourkela, Rourkela, Odisha, India – 769008.
{515cs1002, patrabk}@nitrkl.ac.in

Abstract. The crux of a recommendation engine is to process users ratings and provide personalized suggestions to the user. However, processing the ratings and providing recommendations in real time still remains challenging, when there is a perpetual influx of new ratings. Traditional approaches fail to accommodate the new streamlined ratings and update the users' preferences on the fly. In this paper, we address this challenge of streaming data without compromising accuracy and efficiency of recommender system. We identify the affected users and incrementally update their vital statistics after each new rating. We propose an incremental similarity measure for finding neighbors who play an important role in personalizing recommendations for active user. Experimental results on real-world datasets show that the proposed approach outperforms the state-of-the-art techniques in terms of accuracy and execution time.

Keywords. Collaborative filtering, Personalized recommendation, Streamlined ratings, Tendency based approach, Incremental updates.

1 Introduction

Collaborative filtering (CF) techniques have been very successful and popular in providing recommendations to the users. CF techniques are broadly categorized into two types: Neighborhood based approach (User based CF and Item based CF) and Model based approach. The user-based (UB) and item-based (IB) CF approaches are one of the earliest methods in recommender systems [1]. On the other hand, model based CF techniques are proven to be more accurate in learning the user's and item's features through building a model using machine learning and other techniques [4]. In [3], Cacheda *et al.* proposed a tendency-based technique to further improve the recommendation accuracy and computational cost.

The traditional techniques are not equipped to process the incremental ratings in real time. As the new ratings arrive, the users' taste/ interests can change *w.r.t.* time *i.e.* the users' preferences calculated earlier could become obsolete, leading to a concept drift in the recommender system. A feasible solution is to incrementally update the preferences with the arrival of each new rating. In this direction, Huang *et al.* proposed an item-neighborhood based approach which prunes the probable dissimilar items to reduce the number of similarity computations [2]. Subbian *et al.* proposed a probabilistic neighborhood based approach to compute the similarity between the items [5]. This approach approximates similarity using Locality Sensitive Hashing where hash functions are applied on the user indices. The approaches proposed in [2] and [5] compute only an approximation but not the actual similarity which can affect the accuracy of recommender system.

In this paper, we introduce an incremental approach to update the tendency of user and item in real-time. Subsequently, we propose an incremental user similarity measure to personalize the item tendency *w.r.t* the active user. We simulated a streaming environment on two real-world datasets (Yahoo Music and Movielens 10M) and the experimental results show that the proposed technique outperforms tendency based approach and the state-of-the-art techniques in terms of accuracy and execution time.

2 Background and Proposed Approach

As discussed in the previous section, tendency based approach is found to be more accurate and efficient compared to the traditional CF approaches [3]. However, it cannot address the challenges posed by streaming environment. In this paper, we address this drawback by proposing an incremental approach which is effective in streaming scenarios. Tendency based approach computes two important statistics namely, *user tendency* and *item tendency*. The user tendency (τ_u) is computed in terms of the aggregate rating deviation from the user's rating to each rated item's average rating. Likewise item tendency (τ_i) is computed (Equation 1).

$$\tau_u = \frac{\sum_{i \in I_u}(r_{ui} - \bar{r}_i)}{|I_u|} \qquad \tau_i = \frac{\sum_{u \in U_i}(r_{ui} - \bar{r}_u)}{|U_i|} \tag{1}$$

where r_{ui} is the rating of user u on item i, $\bar{r}_u$, $\bar{r}_i$ are the average ratings of the user and item respectively, I_u is the set of items rated by the user, and U_i is the set of users who rated the item i. The final rating of an unrated item is calculated based on the tendency statistics and average rating of the active user. Although this approach needs lesser computational time, it does not update the tendencies in streaming environment. It can be observed from Equation 1 that the item tendency (τ_i) remains unchanged across the users. This leads to "non-personalized" recommendations.

2.1 Proposed Approach

As discussed in the previous section, tendency based approach has two major shortcomings: (a) inability to accommodate the streamlined ratings (b) generalized item tendency computation leading to non-personalized recommendations. We overcome the first shortcoming by incrementally updating user and item tendencies. Secondly, we introduce an incremental similarity update approach to dynamically personalize the item tendency *w.r.t.* the active user.

Incremental Tendency Approach: To accommodate the streamlined ratings and to update the preferences of the users (and items), we propose to incrementally update the tendency of users and items. Equation of user u's tendency (τ_u) is split as shown in Equation 2.

$$\tau_u = \frac{\sum_{i \in I_u} (r_{ui} - \bar{r}_i)}{|I_u|} = \frac{\sum_{i \in I_u}(r_{ui}) - \sum_{i \in I_u}(\bar{r}_i)}{C_u(u)} = \frac{A_u - B_u}{C_u(u)} \tag{2}$$

where, $C_u(u)$ is the number of items rated by user u. To accommodate incremental updates, we store the item i's mean rating ($\bar{r}_i$), the number of users who rated item i ($C_i(i)$) , the aggregate rating value of user u ($A = \sum_{i \in I_u}(r_{ui})$) and aggregate mean rating of the items rated by user ($B = \sum_{i \in I_u}(\bar{r}_i)$). In a streaming setting, the system might face two scenarios: 1) user u newly rates item x with rating r_{ux} (2) user v rates item x which has been previously rated by user u. In order to update user u's tendency (τ_u), we need to update the terms A, B and $C_u(u)$. In the first scenario, the aggregate rating value of user u is updated as $A'_u = A_u + r_{ux}$. Cardinality of item x ($C_i(x)$) and user u ($C_u(u)$) are incremented by 1. Subsequently, item x's mean rating ($\bar{r}_x$), aggregate mean of the items rated by user u (B_u) and finally the user tendency τ'_u are incrementally updated as shown below.

$$\bar{r}'_x = \frac{C_i(x) * \bar{r}_x + r_{ux}}{C_i(x) + 1}, \qquad B'_u = B_u + \bar{r}'_x, \qquad \tau'_u = \frac{A'_u - B'_u}{C'_u(u)}$$

In the second scenario, user v's tendency (τ'_v) is updated by modifying item x mean rating ($\bar{r}'_x$), A'_v, B'_v, $C'_u(v)$, $C'_i(x)$ (as discussed above). Apart from user v's tendency, user u's tendency (τ'_u) needs to be updated. The term B_u is modified as $B'_u = B_u + \bar{r}'_x - \bar{r}_x$, where $\bar{r}_x$ is the mean of item x before arrival of rating r_{vx} and $\bar{r}'_x$ is the new mean of item x. Subsequently, user u tendency is updated as: $\tau'_u = (A_u - B'_u)/C_u(u)$. It can be noted that A_u, $C_u(u)$ remain unchanged as this item has already been rated by user u. Likewise, the tendency of all the users who previously rated item x is incrementally updated. In a similar fashion, computation of item tendency (τ_i) is split and updated incrementally[1]. In the following section, we address the problem of non-personalization in tendency based approach.

Personalized Item Tendency with Incremental Similarity Update:

Let U_i be the set of users who rated target item i. We find the set of neighbors of active user u_a who rated target item. Let U_k be the neighbor set ($U_k \subset U_i$). We utilize the neighbors' ratings to personalize the item tendency *w.r.t.* the active user. In real–time scenarios, Pearson Correlation, Jaccard similarity measure cannot be used as these are computationally cumbersome. Therefore, we propose an incremental approach (inspired from Jaccard similarity) which captures the

[1] Due to space constraint, we omit the incremental item tendency computation in this paper.

Algorithm 1 Incremental Neighborhood based Similarity Measure

Input: Rating Stream R, Liked Rating Threshold θ, User Preference Matrix P, User Co–rated Matrix $I_{corated}$, Liked Cardinality of each User LCU, Disliked Cardinality of each User DCU, Liked Cardinality of each Item LCI, Disliked Cardinality of each Item DCI, List of users who rated liked each item L_i, List of users who rated disliked each item D_i

Output: Updated user and item matrices

```
 1: loop: for each $r_{ui} \in R$
 2: if $r_{ui} \geq \theta$ then
 3:     $U_i^l = L_i[i]$ // List of users who liked item $i$
 4:     loop: for each user $v \in U_i^l$
 5:     Update the similarity between the user $u$ and the users in $U_i^l$ using Equation 4
 6:     $LCU[u] = LCU[u] + 1$ // Update User Cardinality
 7:     $LCI[i] = LCI[i] + 1$ // Update Item Cardinality
 8:     $L_i[i][LCI[i]] = u$ // Add User $u$ to list of users who liked item $i$
 9:     end:loop
10: else
11:     $U_i^d = D_i[i]$ // List of users who disliked item $i$
12:     loop: for each user $v \in U_i^d$
13:     Update the similarity between the user $u$ and the users in $U_i^d$ using Equation 4
14:     $DCU[u] = DCU[u] + 1$ // Update User Cardinality
15:     $DCI[i] = DCI[i] + 1$ // Update Item Cardinality
16:     $D_i[i][DCI[i]] = u$ // Add User $u$ to list of users who disliked item $i$
17:     end:loop
18: end if
19: end:loop
```

interest/ disinterest of the users. The proposed similarity measure between two users, u and v is computed as shown in Equation 3.

$$sim(u,v) = \frac{|L_{sim}(u,v)| + |D_{sim}(u,v)|}{|I_u \cup I_v|} = \frac{P(u,v)}{C(u) + C(v) - I_{corated}(u,v)} \tag{3}$$

where $L_{sim}(u,v)$ is the set of items liked by users u and v, $D_{sim}(u,v)$ is the set of items disliked by users u and v. I_u, I_v are the set of items rated by users u and v, respectively and $I_{corated}(u,v)$ is the number of co-rated items. In the case of streamlined ratings, recomputing the set of liked/ disliked items for each user pair is expensive. Therefore, we propose an incremental approach in updating the similarity of only those users who are affected with the new ratings. Let r_{ui} be newly liked item i ($r_{ui} > \theta$) and a user set V who liked this item in the past. The similarity value between user u and user v ($\forall v \in V$) is computed by incrementally updating the terms: $P'(u,v) = P(u,v) + 1$, $C'_u(u) = C_u(u) + 1$

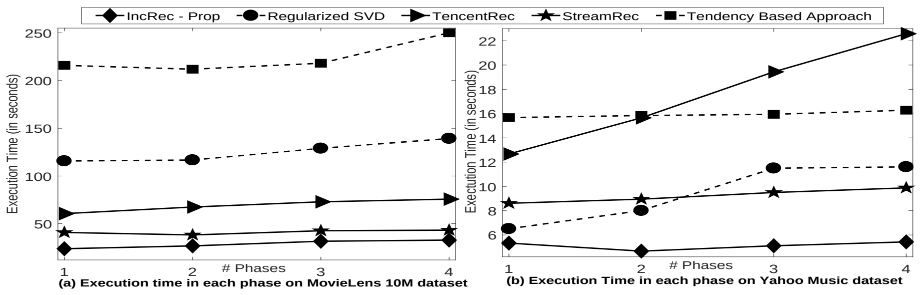

Fig. 1. Execution Time in each phase on YM and ML 10M datasets

and $I'_{corated} = I_{corated} + 1$ (as shown in Equation 4). In the similar fashion, the similarity can be updated if $r_{ui} < \theta$. Therefore, for each new rating in the streamline, the similarity values of the affected users are incrementally updated (detailed Algorithm shown in 1). To support the incremental similarity updates, we only store the most relevant information such as the 1) list of users who liked (and disliked) each item 2) cardinality of each user (and each item). In order to reduce the number of accesses and processing time, we store the liked items (users) cardinality and disliked items (users) cardinality of each user (item) separately.

$$sim'(u,v) = \frac{P(u,v)+1}{(C_u(u)+1)+C_u(v)-(I_{corated}(u,v)+1)} \tag{4}$$

After obtaining the neighbors (U_k) of the active user, we utilize the ratings of these neighbors to compute the personalized item tendency (τ_i^p) *w.r.t.* active user as shown below. Likewise, the mean of the target item ($\bar{r}_i^p$) is personalized *w.r.t.* the active user (as shown below).

$$\tau_i^p = \frac{\sum_{\acute{u}\in U_k}(r_{\acute{u}i} - \bar{r}_{\acute{u}})}{|U_k|} \qquad\qquad \bar{r}_i^p = \frac{\sum_{\acute{u}\in U_k} r_{\acute{u}i}}{|U_k|} \tag{5}$$

where $r_{\acute{u}i}$ is the rating provided by a neighbor ($\acute{u}$) on the target item (i) and $\bar{r}_{\acute{u}}$ is the mean rating of the neighbor $\acute{u}$. Finally, the overall rating of the active user on the target item is predicted using the active user's tendency and target item's personalized tendency as discussed in [3].

3 Experimental Results

In this paper, real–world datasets, MovieLens 10M (ML 10M) and Yahoo! Music (YM) are used to evaluate our approach. To replicate a streamlined environment, we shuffled the ratings set and divided it into five equal parts. In this first phase, the first part of the dataset is used on training and the second part is used for testing. In the second phase, the ratings in the second part are incrementally

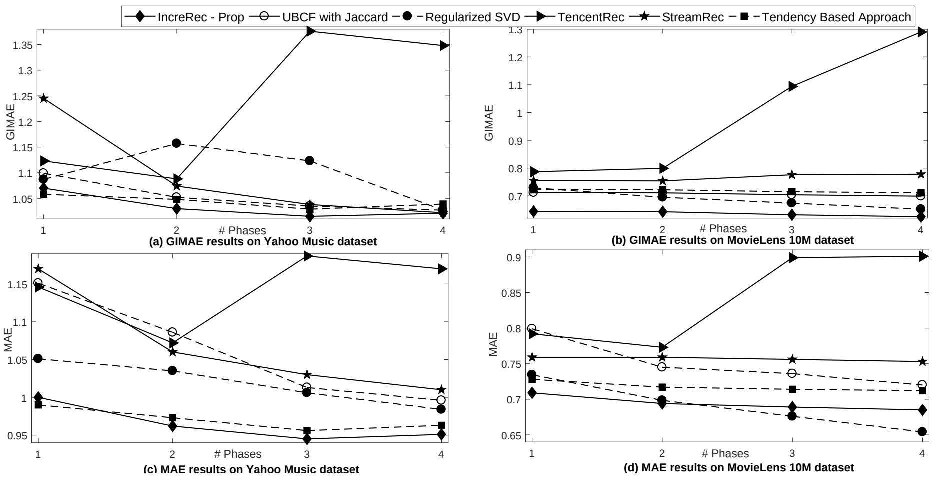

Fig. 2. Accuracy results on MovieLens YM and ML 10M datasets

added to the training set and the third part of the dataset is used for testing purpose. This process is repeated in remaining phases. We used 5% of the samples in each phase for parameter setting. Mean Absolute Error (MAE) and recently proposed Good Items MAE (GIMAE) [3] are used to evaluate the approaches. We compare our approach (IncRec) with Tendency based CF [3], Regularized SVD [4], traditional neighborhood based approach (UBCF with Jaccard), CF approaches for streaming data (StreamRec [5] and TencentRec [2]). It can be noted that all the experiments are conducted on a single CPU system.

Execution Time in Streaming Scenarios: We report the execution time of all the approaches including IncRec in Figure 1(a) and 1(b) on ML 10M and YM datasets, respectively. Results of UBCF with Jaccard are not reported as this technique takes significantly longer execution time (three order of magnitude) than the proposed approach. It can be noted (from Fig. 1) that recently proposed CF approaches (TencentRec and StreamRec) have lesser execution time than tendency based CF. However, the proposed approach outperforms all the approaches in each phase.

GIMAE and MAE results: GIMAE focuses on the error incurred on the predictions of good items (relevant) ratings [3]. The results of ML 10M and YM are plotted in Fig. 2(a) and Fig. 2(b), respectively. On YM dataset, StreamRec performs better than the existing techniques in the first two phases. Regularized SVD performs better than the existing approaches in third and forth phases. However, the proposed approach outperforms all the existing techniques in all the phases. Similar trends are found on ML 10M dataset. The MAE results of all the approaches are shown in Fig.2(c) and Fig.2(d). On YM dataset, tendency based approach incurred least MAE in the first phase. As the ratings increase, the proposed approach performs significantly better than tendency based CF and other existing techniques. On ML 10M dataset, the proposed approach outperforms the existing techniques in the first and second phase. Regularized SVD is found to be superior than all the approaches in the third and forth phases. However, Regularized SVD cannot be used in a streamlined setting as the execution time is two order of magnitude slower than the proposed approach. To summarize, the proposed approach performs significantly better than the existing techniques in terms of execution time, GIMAE and MAE.

4 Conclusion

In this paper, we proposed an incremental CF approach to tackle real–time streaming ratings. Existing streaming algorithms in CF framework usually sacrifice accuracy for speed-up. However, experimental results validate that our approach is faster and more accurate compared to many existing techniques on real–world datasets.

References

1. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, “Recommender Systems Handbook”, Springer-Verlag NewYork, Inc., 2nd edition, 2015.
2. Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. “Tencentrec: Real-time stream recommendation in practice.” In Proceedings of the SIGMOD International Conference on Management of Data, pp. 227-238. ACM, 2015.
3. Fidel Cacheda, Vctor Carneiro, Diego Fernndez, and Vreixo Formoso. “Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems.” ACM Transactions on the Web, vol. 5, no. 1, pp. 2:1:2.33. 2011.
4. Arkadiusz Paterek. “Improving regularized singular value decomposition for collaborative filtering.” In Proceedings of KDD cup and workshop, pp. 5-8. ACM, 2007.
5. Karthik Subbian, Charu Aggarwal, and Kshiteesh Hegde. “Recommendations for streaming data.” In Proceedings of the International on Conference on Information and Knowledge Management, pp. 2185-2190. ACM, 2016.