Clustering Based Diversity Improvement in Top-N Recommendation

Tevfik Aytekin · Mahmut Özge Karakaya

Abstract The major aim of recommender algorithms has been to predict accurately the rating value of items. However, it has been recognized that accurate prediction of rating values is not the only requirement for achieving user satisfaction. One other requirement, which has gained importance recently, is the diversity of recommendation lists. Being able to recommend a diverse set of items is important for user satisfaction since it gives the user a richer set of items to choose from and increases the chance of discovering new items. In this study, we propose a novel method which can be used to give each user an option to adjust the diversity levels of their own recommendation lists. Experiments show that the method effectively increases the diversity levels of recommendation lists with little decrease in accuracy. Compared to the existing methods, the proposed method, while achieving similar diversification performance, has a very low computational time complexity, which makes it highly scalable and allows it to be used in the online phase of the recommendation process.

Keywords recommender systems · collaborative filtering · recommendation diversity · clustering

1 Introduction

Recommender systems help users find items of interest (e.g., movies, books, or restaurants.) based on various information such as explicit ratings of users on items, past transactions, or item content. Many successful techniques have been

Tevfik Aytekin

Department of Computer Engineering, Bahçeşehir University, Çırağan Caddesi, 34353, Beşiktaş, İstanbul, Turkey
Tel.: +90-212-3810580
Fax: +90 212 3810550
E-mail: tevfik.aytekin@bahcesehir.edu.tr

Mahmut Özge Karakaya
Department of Computer Engineering, Bahçeşehir University, Çırağan Caddesi, 34353, Beşiktaş, İstanbul, Turkey

developed up until today [3, 6, 10, 26] and these techniques have been applied to different domains such as music [12], movies [17], travel [28], e-commerce [27], and e-learning [7]. The success of a recommender algorithm is typically measured by its ability to accurately predict ratings of items. The accuracy of predictions is no doubt an important property of recommender algorithms. And naturally, most of the research in recommender systems have focused on improving accuracy. However, there are other factors which play valuable roles in user satisfaction. One such factor, which gained importance recently, is the diversity of recommendation lists [11, 20, 29]. For example, think of a system which suggests movies to its users. The system might be very accurate, that is, it might predict the user ratings on items very well. However, if the recommendation list of a user consists of one type of movies (e.g., only science fiction movies), it might not be very satisfactory. A good system should also recommend a diverse set of movies to its users (e.g., movies from different genres). However, there is a trade-off between accuracy and diversity. That is, most of the time diversity can only be increased at the expense of accuracy. But this decrease in accuracy might be preferable if the user satisfaction increases. Moreover, this trade-off can be implemented as a tunable parameter which users can adjust according to their needs. In this way users themselves decide how much to sacrifice accuracy for an increase in diversity. This also lets users experiment with the recommendations produced by the system and to discover a diverse set of items.

In this paper we will describe a novel method (named ClusDiv) which can be used to increase the diversity of recommendation lists with little decrease in accuracy. Our idea basically is to cluster items into groups and build the recommendation list by selecting items from different groups, such that, recommendation diversity is maximized without decreasing accuracy too much. The details of ClusDiv will be explained in subsequent sections. In summary, the main strengths of ClusDiv are as follows:

– ClusDiv is applied after unknown ratings are predicted by the prediction algorithm. So, real world recommender systems can use ClusDiv without altering their existing prediction algorithms.
– In order for a recommender system to allow users to adjust the diversity levels of their recommendation lists, the online time complexity of the recommendation algorithm should be very low. ClusDiv has a very low online (and also offline) time complexity, which makes it highly scalable.
– ClusDiv naturally involves a tunable parameter which users can use to adjust the diversity levels of their own recommendation lists. They can do this adjustment independent of other users.
– No content information (such as the genre or the director of a movie) about items is necessary. Rating information about items is enough to diversify the recommendation lists.

The paper is organized as follows: in Section 2, we discuss related work on diversity in recommender systems. In Section 3, we describe ClusDiv in detail. In Section 4, we give experimental results and evaluate them. Finally, in Section 5, we conclude the paper and point out new directions for research.

2 Related Works

It has been a while since researchers in recommender systems research realized that prediction accuracy is not the sole property a successful recommender system needs to have. For example, [25] argues that the evaluation of recommender systems should move beyond the conventional accuracy metrics. [19] discusses novelty and serendipity as important dimensions in evaluating recommender systems. The notions of novelty and serendipity are closely related to diversity, since, increasing the diversity of a recommendation list increases the chance of recommending novel and serendipitous items to the user.

Several strategies have been proposed to address the problem of diversity. [9] and [29] propose a greedy selection algorithm. In this method, the items are first sorted according to their similarity to the target query, and then the algorithm begins to incrementally build the retrieval set (or the recommendation list if we use the terminology of recommender systems research) such that both similarity and diversity are optimized. This is achieved as follows: in the first iteration, the most similar item to the target query is put in the retrieval set, in the next iteration the item which has the maximum combination of similarity to the target query and diversity with respect to the retrieval set already built is selected. Iterations continue until the desired retrieval set size is achieved. As noted in [29], this greedy selection algorithm is quite inefficient, so the authors proposed, in the same paper, a bounded version of the greedy selection algorithm. In this version, the algorithm first selects the most similar b number of items to the target query, and then the greedy selection method is applied to this set of items instead of the entire set of items. As b approaches n (the number of items), the complexity of this bounded version approaches the complexity of the greedy selection method. Another optimization based method is proposed in [31] and [20]. Here the authors represent the trade-off between similarity and diversity as a quadratic programming problem. Then they offer several solution strategies to solve this optimization problem. They also introduce a control parameter which determines the importance of the level of diversification in the recommendation lists. As the authors have reported, they achieve near-identical results in terms of similarity vs. diversity results with the bounded greedy selection method proposed in [29], and the time-efficiency of their algorithm is slightly worse. In order to show the effectiveness of ClusDiv, we will compare it only with the bounded greedy method proposed in [29], since the other method proposed in [20] has similar levels of diversification performance and has slightly worse computational time complexity. We will see in Section 4.5 and Section 4.6 that ClusDiv is much faster than the bounded greedy method, while achieving similar diversification performance.

[32] uses a clustering approach to better diversify recommended items according to the taste of users. To do this, they cluster items in the user profile and recommend items that match well to these individual clusters rather than the entire user profile. ClusDiv also clusters items into groups. However, as will be discussed in detail below, it clusters all the items in the system rather than just the items in the user profile. That is, the aim of ClusDiv is not to recommend items which match users' tastes, but rather to recommend a diverse set of items while maintaining accuracy as much as possible. This gives users the opportunity to meet serendipitous items.

[33] defines a similarity metric based on classification taxonomies according to which the intra-list similarity is calculated. The authors propose a heuristic algorithm for diversification of recommendation lists based on this similarity metric. Similar to other studies, the proposed method increases diversity with some negative effects on accuracy. One important contribution of this work is to show empirically that overall user satisfaction increases with diversified recommendation lists. This result supports the claim that the accuracy of recommendation lists is not the only requirement for user satisfaction, but other properties, such as being able to suggest diverse items are also important.

[8] states that most of the recommendation algorithms implement some kind of threshold parameter to balance the trade-off between diversity and accuracy. They think that threshold parameters are problematic because their tuning for a particular dataset is time consuming and is no longer effective when the data changes. So, they propose a diversification method which does not involve threshold parameters. Our method also involves a threshold parameter, however, as we will explain in detail, this parameter is tuned by the user (not by the system administrator) and can be tuned online according to the taste of the user. So, it is not needed to be tuned for a particular dataset.

Finally, we would like to mention the work in [18], where the authors propose an axiomatic approach to characterize and design diversification systems. They provide a set of natural axioms, which a diversification system is expected to satisfy, and show that all of these axioms cannot be satisfied by a diversification algorithm. They indicate that the choice of axioms from this set helps in characterizing diversification objectives independent of the algorithms used.

3 Clustering Based Diversification

In this section we will describe ClusDiv in detail. But prior to this, we first describe the diversity measure we use.

3.1 Diversity Measure

There can be different metrics for measuring different dimensions of diversity in a recommendation system. One possible metric for measuring the diversity of a recommendation list of a particular user is described in [20] and [29]. This metric measures the diversity as the average dissimilarity of all pairs of items in a user's recommendation list. Let $\mathcal{I}$ be the set of all items, and $\mathcal{U}$ be the set of all users, then the diversity of a recommendation list of a particular user, $D(L(u))$, can be defined as follows:

$$D(L(u)) = \frac{1}{N(N-1)} \sum_{i \in R} \sum_{j \in R, j \neq i} d(i,j), \tag{1}$$

where $L(u) \in \mathcal{I}$ is the recommendation list of user $u \in \mathcal{U}$ and $N = |L(u)|$. $d(i,j)$ is the dissimilarity of items $i, j \in \mathcal{I}$, which is defined as one minus the similarity of items i and j.

Although this is a reasonable metric, we think that it has two problems. One is due to the sparse nature of the recommender system datasets. These datasets typically have lots of missing ratings [4]. For example the sparsity of the MovieLens[1] (1M) dataset is about 95.75%. This means that there are many missing ratings. Generally some default value is set for these missing ratings [10]. For example, if cosine similarity measure is used, these missing ratings are generally set to 0 [15]. However, using a default value for the missing ratings creates a difficulty for measuring diversity using the formula in Eq. (1). For example, suppose that you use a default value of 0 for the missing ratings. If you use cosine similarity (as we did in the experiments described in this paper) then similarity values between items will be very close to 0, and, therefore, diversity values will be very close to 1. This does not create a problem for rating predictions because what is important most of the time is the relative (not absolute) values of the similarities. However, recommendation lists whose diversity values are close to 1 will look superficially very diverse. In other words, the diversity value generated using the formula in Eq. (1) will be dominated by the default values used for the missing ratings and will be misleading.

The second problem with Eq. (1) is that it conceals the difficulty of the diversification problem. Independent of the diversification method used, some datasets are inherently more difficult with respect to diversification. Table 1 shows the mean and standard deviations of the pairwise similarities of items in the datasets that we use in this study (using cosine as the similarity metric and 0 as the default value for missing ratings). Standard deviations of pairwise similarities show that Jester[2] dataset has the highest variance, Book-Crossing[3] dataset has the lowest variance, and MovieLens dataset's variance is in between. This shows that among these three datasets, Book-Crossing dataset is the one which is most difficult to diversify, since, pairwise similarities of items in it has the lowest variance.

We think that the best way to resolve these issues is to use z-scores of the diversity values, which we call z-diversity, instead of using absolute diversity values defined by Eq. (1). Formally z-diversity of a recommendation list is defined as:

$$ZD(L(u)) = \frac{D(L(u)) - D(\mathcal{I})}{SD(\mathcal{I})}, \tag{2}$$

where $\mathcal{I}$ is the set of all items in the dataset, and $D(L(u))$ and $D(\mathcal{I})$ are the diversity of items in $L(u)$ and $\mathcal{I}$, respectively. $SD(\mathcal{I})$ is the standard deviation of the dissimilarities of all pairs in $\mathcal{I}$ as defined below:

$$SD(\mathcal{I}) = \sqrt{\frac{1}{N(N-1)} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}, j \neq i} (d(i,j) - D(\mathcal{I}))^2},$$

where $N = |\mathcal{I}|$ and $D(\mathcal{I})$ is the diversity of items in $\mathcal{I}$. $d(i,j)$ is the dissimilarity of items $i, j \in \mathcal{I}$, which is defined as one minus the similarity of items i and j. As we point out above, we use cosine similarity for measuring the similarity between two items and use a default value of 0 for the missing ratings.

---

[1] http://www.grouplens.org/node/73/

[2] http://goldberg.berkeley.edu/jester-data/

[3] http://www.informatik.uni-freiburg.de/ cziegler/BX/

This metric does not have the two problems mentioned above. Since the diversity values are given as z-diversity values, they will not be very close to 1 even though the missing ratings are set to 0 and will give a better idea about the increase in diversity. Secondly, as we mention above, if the standard deviation of pairwise similarities of items in a dataset is low, then giving the diversity values as defined in Eq. 1 will lead us to underestimate the success of the diversification algorithm. However, if we use the z-diversity values as defined in Eq. 2, then this will help us evaluate the algorithm's performance better by showing the increase in diversity values independent of the variance of the pairwise item similarities in the dataset. Apart from solving these two problems, giving the diversity values as z-diversity also makes it easier to analyse the performance of algorithms, such as the random method, which we will discuss in Section 4.5.

3.2 ClusDiv

In this section we will describe ClusDiv in detail. Similar to many recommendation algorithms, ClusDiv has offline and online phases. In the offline phase, apart from model building, we build N (where N is the size of the recommendation list $L(u)$) item clusters $C = \{C_1, C_2, \ldots, C_n\}$. The item clusters are built using the standard k-means clustering algorithm [30]. Items are clustered based on their ratings given by the users of the system. No content information about items is used. However, if content information about items is available and if it is possible to define similarities between items based on this content information, then one can also use these similarities in clustering items.

At the heart of our algorithm lies the construction of cluster weights (CW). CW is a matrix whose $(u, i)^{th}$ entry, CW_{ui}, holds the number of items which cluster C_i will contribute to the recommendation list of user u. Users have their own CW_u vector. So, for example, if $CW_{ui} = 5$, then it means that cluster C_i will contribute 5 items (those which have the highest-predicted ratings in C_i) to the recommendation list of user u. It follows that the sum of cluster weights for any user should be equal to N (the recommendation list size).

Algorithm 1 ClusterWeights

Input: $topN_u$: top-N list of user u, C: Item clusters
Output: CW_u: Cluster weights of user u.
 1: $CW_{ui} = |C_i \cap topN_u|$
 2: **for all** $C_i \in C$ **do**
 3:     **while** $|C_i| > threshold$ **do**
 4:         $c = \operatorname*{argmin}_j |CW_{uj}|$
 5:         $CW_{uc} = CW_{uc} + 1$
 6:         $CW_{ui} = CW_{ui} - 1$
 7:     **end while**
 8: **end for**

Algorithm 1 computes the cluster weights for a particular user. It takes two parameters: the top-N recommendation list for the target user u, which is generated by the prediction algorithm, and the set of clusters $C = \{C_1, C_2, \ldots, C_k\}$.

In line 1 we initialize cluster weights. Each cluster weight CW_{ui} is assigned the number of items in C_i, which are also in $topN_u$. In other words, for each user u initially the set of items contributed by clusters is exactly the same as the $topN_u$ recommendation list generated by the prediction algorithm. This guarantees that it is always possible to get exactly the same recommendation list generated by the original prediction algorithm, as we will explain below. In line 2, the algorithm begins to distribute the cluster weights of those clusters whose weights are larger than a given threshold. The threshold value allows us to control the trade-off between accuracy and diversity. As threshold gets larger, the user gets more accurate but less diverse results and as it gets smaller, the user gets more diverse but less accurate results. In the extreme case if threshold is equal to N then the recommendation list generated by the prediction algorithm is not altered at all. The algorithm distributes the weights of clusters whose weights are larger than the threshold value to other clusters which have the smallest weights as shown in lines 3-7 of the algorithm. If more than one cluster is found in line 4, one of them can be chosen randomly without affecting the performance of the algorithm. In this way, the algorithm tries to distribute the weights to different clusters as much as possible in order to increase the diversity of the resulting recommendation list.

Other more complex methods can be used in order to distribute cluster weights. For example, to distribute the weight from a source cluster, one can select those clusters which are far away from the source using a suitable metric to measure the distance between clusters. The rationale underlying this approach is that if we choose items from distant clusters, then we can get more diversified recommendation lists. Or a combination of this approach and the one described in algorithm 1 can be used. We have also tried these approaches but we get no significant changes in the results. So, in order not to increase the complexity of the algorithm without getting significant improvements, we decide to present the simplest version of the algorithm here.

After we generate the cluster weights of a user u, we build the recommendation list of u as follows: we go over the items in the recommendation list of u from top to bottom and take an item to the top-N list of u if the weight of the cluster which that item belongs to is larger than zero and subtract one from that cluster weight. We continue to go over the recommendation list in this fashion until all cluster weights are zero. When all the cluster weights are zero, the final recommendation list is ready.

4 Experimental Results and Evaluation

In this section we describe the experimental methods we use and present the results we get. We also give a performance analysis of ClusDiv.

4.1 Testing Procedure

For evaluating accuracy we measure the recall performance. We use a similar methodology given in [13]. We randomly sub-sample 2% of ratings in order to create a probe set. The rest of the ratings is put into the training set. Test set, T,

is formed by selecting all 5-star ratings in the probe set. After the model is trained on the training set recall is measured as follows:

For each item i rated 5-stars by user u in the test set:

- Select randomly 300 items unrated by user u.
- Predict the ratings of these 300 items and the test item i.
- Form a top-N recommendation list by selecting the N items from the list of 301 items, which have the highest predicted ratings. If the test item i occurs among the top-N items, then we have a hit, otherwise we have a miss.

Recall is then defined as follows:

$$Recall(N) = \frac{\#hits}{|T|}$$

This testing methodology assumes that all randomly selected 300 items are non-relevant to user u. As stated in [13], this assumption tends to underestimate the true recall.

4.2 Prediction Algorithms

In our experiments, we use three different collaborative filtering algorithms: item and user based collaborative filtering [14, 15, 21, 23, 24] and SVD [5, 16, 22]. Although there are many variations, the basic methodologies of these algorithms can be stated as follows: in item based collaborative filtering, for predicting the rating of a user u on an item i first most similar k items to item i, which are rated by user u, are found. Then the rating is predicted by taking the average ratings of those k items. In the user-based case, for predicting the rating of a user u on an item i, this time, first the most similar k users to user u, which have rated item i, are found. Then the rating is predicted by taking the average ratings given to item i by those k users.

In SVD the rating matrix is factorized into a product of two lower ranked matrices, which are known as user-factors and item-factors. Each row of the user-factors matrix represents a single user, and similarly each row of the item-factors matrix represents a single item. Prediction of a rating by a user u on an item i is done by taking the dot-product of the corresponding rows of the factor matrices.

4.3 Datasets

We test and evaluate ClusDiv on the following commonly used three datasets:

MovieLens Dataset: this dataset contains 1,000,000 integer ratings (from 1 to 5) of 6,040 users on 3,900 movies.

Book-Crossing Dataset: contains 1,149,780 integer ratings (from 1 to 10) of 278,858 users on 271,379 books. Due to memory limitations, we reduced this dataset by selecting those users which rated more than or equal to 20 items and those items which were rated by more than or equal to 10 users. The reduced dataset contains 60,133 ratings, 6,358 users, and 2,177 items. We have not used the implicit ratings provided in the dataset.

Jester Joke Dataset: contains 620,000 continuous ratings (-10.00 to +10.00) of 24,938 users who have rated between 15 and 35 on 101 jokes. Again, due to memory limitations, we reduced this dataset by selecting randomly 5,000 users. The reduced dataset contains 123,926 ratings.

The mean and standard deviations, given in Table 1, are calculated using the reduced Book-Crossing and Jester datasets and the original Movielens dataset.

4.4 Diversity Analysis

In this section we give the diversity results we get as a function of the threshold value for all three datasets. As we point out above, there is a trade-off between diversity and recall performance. That is, most of the time, diversity can only be increased with some loss in recall performance. So, in Fig. 1, Fig. 2, and Fig. 3, we show diversity values alongside the recall values.

Fig. 1a depicts the change in diversity against the threshold values for the MovieLens dataset. We see that as we decrease threshold, that is, as we force the algorithm to use more clusters, diversity values increase for all the three prediction algorithms. The largest increase in diversity is in the SVD method. The reason for this is apparent. The top-20 list generated using the SVD method has a lower initial diversity compared to other methods. Also, the z-diversity value is below 0, which means that the diversity of the list is below the mean diversity of the dataset, which makes it easier to increase the diversity. On the other hand, the top-20 lists generated by item-based and user-based methods already have diversities above the mean diversity of the dataset. So, the increase in diversity for these methods is more limited. As also expected, for all three prediction methods, the maximum diversities achieved have similar values. This shows that ClusDiv, whatever the initial diversity values are, carries the diversity values to a similar level. This is because as the threshold values decrease, recommendation lists are forced to be generated from similar number of clusters.

Fig. 2a shows the diversity as a function of threshold for the Book-Crossing dataset. Here again we see that as the threshold decreases, diversity increases. For Book-Crossing dataset, all the prediction algorithms initially generate recommendation lists with negative z-diversities, that is, initially the diversities of the recommendation lists are below the mean diversities of the datasets. But as shown in Fig. 2a, ClusDiv manages to increase the diversity for all prediction algorithms.

Diversity results for the Jester dataset is shown in Fig. 3a. Here again diversity values increase as the threshold decreases for all prediction algorithms. Since there are only 101 items in the Jester dataset, we build a recommendation list of size 10, and, hence, build 10 item clusters instead of 20.

For all datasets, as we decrease threshold, recall values also decrease as expected. For practical purposes, large and small threshold values seem not to be usable. Large threshold values cause little or no change in diversity, and small threshold values, while having the largest increase in diversity, have low recall values. But there are enough threshold levels between the extremes, which can give users enough flexibility to adjust the diversity vs. accuracy trade-off according to their tastes. Note that the tuning of the threshold parameter is not system wide, users can adjust their own parameters independent of other users.

4.5 Comparison

In this section we compare the diversification performance of ClusDiv with two other methods. Although the primary aim of ClusDiv is to be a fast diversification algorithm, it is also important to see its diversification performance relative to other methods.

One simple and intuitive technique, which might be used to increase the diversity of recommendation lists, is to incorporate some randomization into the recommendation process. This technique, which we call random method, in order to increase the diversity of a top-N recommendation list, ranks the items according to their predicted ratings and selects N items randomly from B top ranked items, where $B > N$. In the experiments that follow, we always take the value of N (the recommendation list size) to be equal to 20.

Fig. 4, Fig. 5, and Fig 6 show the diversity and recall values as a function of B for all three datasets. Diversity values are z-diversity values as defined in Eq. (2) averaged over all the users. As expected, for all datasets, the recall values decrease as we increase B. This is because we are leaving some items which have the highest predicted ratings and including some items whose predicted ratings are lower. However, the diversity results are interesting. For example, for the MovieLens dataset, random method either causes a small increase (in the case of SVD) or decrease (in the case of item and user based methods) in diversity as shown in Fig. 4. For the other two datasets we see some small increases in diversity values. The explanation of these diversity curves can be given as follows. As we increase the size of the set B, the diversity values of the recommendation lists tend to approach the average diversity of the dataset. Since we are showing the z-diversity values, the diversity values of the recommendation lists tend to approach 0. This explains the shape of the diversity curves in Fig. 4. The diversity curves in Fig. 5 can also be explained in the same way, they tend to approach 0. For the Jester dataset the situation is a bit different, even though the initial recommendation lists have positive z-diversity values, they still continue to increase slightly. But if you continue to increase the value of B (as we did but do not show here), they also converge to zero. The initial increase in the diversity values probably is the result of some property of the dataset. But the important point to draw from these results is that contrary to first impressions, the random method is not effective in increasing the diversity values of recommendation lists, and moreover, if the initial diversities are above the average diversity of the dataset, then it may even lead to a decrease in the diversity values.

Next we compare ClusDiv with the bounded greedy (BG) method proposed in [29]. As we explain in Section 2, this method first selects the most similar b number of items to the target query and considers only this set of items to build the recommendation list. In Fig. 7, we show recall as a function of z-diversity for ClustDiv and the BG method, using three different values for b (50, 100, 200, and 400). Here we show the results for all the three datasets, using only the SVD method (in user-based and item-based methods, we get similar results). In the case of ClusDiv, the recall vs. z-diversity curves are generated by varying the threshold parameter, and in the case of the BG method, the curves are generated by varying the relative weight of similarity and diversity factors in the optimization objective defined in [29].

When we look at curves in Fig. 7, we see that for all the three datasets, the z-diversity vs. recall performance of ClusDiv is as good as the BG method which is developed primarily for optimizing diversity and recall values of the recommendation lists. The real superiority of ClusDiv emerges when the issue of time complexity enters the scene, which we consider in the next section. Note that since there are only 101 items in the Jester dataset, we only show the results of BG (50) and BG (100). Also note that the maximum diversity levels achieved by the BG method is higher than ClusDiv (significantly higher in the case of Jester and Book-Crossing datasets). However, at these high diversity levels, recall values are very low, which means that these diversity levels are not useful in practice, since the recommendation lists will be very inaccurate. So, we can conclude that the recall vs. diversity performance of ClusDiv is as good as the BG method when recall values are within acceptable limits.

4.6 Computational Complexity Analysis

ClustDiv is quite efficient with respect to both time and space complexities. In the offline phase, ClusDiv needs to build the item clusters. As we pointed out above, we use k-means algorithm to build the clusters. The complexity of k-means is $O(INmn)$, where I is the number of iterations, N is the number of clusters, m is the number of data points (number of items in our case), and n is the dimensionality of the data points (number of users in our case). Since I can safely be bounded [30], and K is significantly smaller than m, k-means and hence the offline clustering step takes $O(mn)$ time.

In the online phase, typically, a recommender system generates a recommendation list for a particular user using the data structures built in the offline phase. After this recommendation list for a particular user is generated, ClusDiv enters the scene and diversifies the list as follows. First, using algorithm 1, cluster weights of the user for a specific threshold value are calculated. A simple amortized analysis of algorithm 1 shows that this step takes $O(N)$ time where N is the number of clusters and can be assumed to take a constant time. Then we scan through the items in the recommendation list from top to bottom by taking one item to the top-N list if the weight of the cluster, which that item belongs to, is larger than zero and subtract one from that cluster weight. We continue to scan the recommendation list until all cluster weights are zero. So, in the worst case, diversification of the top-N list only requires $O(m)$ time, where m is the number of items. However, $O(m)$ is the worst case scenario. We expect that in practice, all cluster weights become zero much before scanning all the items. To test this, we count the number of items scanned during the experiments. Fig. 8 shows the average number of items scanned for different datasets and for different algorithms.

Recall that, the number of items in the datasets that we use in the experiments are 3900 for MovieLens, 6358 for Book-Crossing, and 101 for Jester datasets. Looking at Fig. 8, we can say that the average number of items scanned in practice is much less than the total number of items, even for very small values of the threshold.

Our algorithm is also much faster than the algorithm described in [29]. Recall that the bounded version of the greedy algorithm needs to make N (the retrieval set or recommendation list size) sorts of a list of size $O(b)$. Even if we take b to

be a small value such as 50, the algorithm needs to make $50x20 = 1000$ iterations (if we take $N = 20$ as we did in our experiments). Moreover, in each iteration, the algorithm needs to calculate the value of the quality metric over all items currently in the recommendation list. In contrast, ClusDiv makes a simple scan over the items.

We also perform several experiments to compare the time it takes to diversify a recommendation list. In Table 2, we give the time (in milliseconds) it takes to generate a diversified recommendation list of size N for a single user. We only show the results on the MovieLens dataset (we get similar results in Book-Crossing and Jester datasets). As expected, the results show that ClustDiv is much faster than the BG method. And as discussed in Section 4.5, this efficiency is achieved without sacrificing recommendation quality.

The space requirement of ClusDiv is also very modest. Only $O(m)$ additional space is needed for storing item cluster information in the offline phase and $O(N)$ additional space for storing cluster weights in the online phase, where m is the number of items, and N is the number of clusters.

5 Conclusion

In this article we have proposed a fast method for increasing the diversity of recommendation lists. As the experiments show, the method can be used to effectively increase the diversity levels of recommendation lists with little decrease in accuracy. One important property of our method is that it gives each user the ability to adjust the diversity levels of their own recommendation lists, independent of other users. As we have discussed above, the low computational time complexity of our method allows this personalization of diversity. Note that if the proposed method would like to be incorporated into a real world recommender system, then a mechanism needs to be added to the user interface (e.g., a scroll bar) in order to get the desired level of diversity from each user.

In the experiments discussed above, we use the rating patterns of items in grouping items into clusters. However, if content information about items is available, which can be used to define a similarity measure between items, then items can be clustered based on content information, and our method can still be used to diversify recommendation lists based on the content of items.

Apart from the diversity of user lists, there is another notion of diversity which is called aggregate diversity. Aggregate diversity can be defined in different ways but one simple and useful definition is to define aggregate diversity as the number of distinct items recommended to all users [1, 2]. Aggregate diversity is especially important with respect to increasing product sales. As a future work, we plan to investigate the effect of our method on aggregate diversity and try to develop new methods which can be used for optimising both the recommendation list diversity and aggregate diversity.

In the method proposed here, users are expected to manually adjust their own diversity levels. As we mention above, this gives users flexibility in controlling the diversity levels of their recommendation lists according to their tastes and moods. However, some users might prefer to give the control to the system, that is, they might prefer to let the system to decide the right level of diversity for themselves.

We believe that developing methods for automatic learning of user preferences for diversity is an important area for future research.

References

1. Adomavicius G, Kwon Y (2011) Maximizing aggregate recommendation diversity: a graph-theoretic approach. In: Proceedings of Workshop on Novelty and Diversity in Recommender Systems, Chicago, Illinois, USA, pp 3–10
2. Adomavicius G, Kwon Y (2012) Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans Knowl Data Eng 24(5):896–911
3. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
4. Ahn HJ (2008) A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Inf Sci 178(1):37–51
5. Bell RM, Koren Y, Volinsky C (2007) Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, pp 95–104
6. Billsus D, Pazzani MJ (1998) Learning collaborative information filters. In: Shavlik JW (ed) ICML, Morgan Kaufmann, pp 46–54
7. Bobadilla J, Serradilla F, Hernando A (2009) Collaborative filtering adapted to recommender systems of e-learning. Knowl-Based Syst 22(4):261–265
8. Boim R, Milo T, Novgorodov S (2011) Diversification and refinement in collaborative filtering recommender. In: Macdonald C, Ounis I, Ruthven I (eds) CIKM, ACM, pp 739–744
9. Bradley K, Smyth B (2001) Improving recommendation diversity. In: Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science
10. Breese JS, Heckerman D, Kadie CM (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin, USA, pp 43–52
11. Castells P, Wang J, Lara R, Zhang D (2011) Workshop on novelty and diversity in recommender systems - divers 2011. In: Mobasher B, Burke RD, Jannach D, Adomavicius G (eds) RecSys, ACM, pp 393–394
12. Chen H, Chen A (2005) A music recommendation system based on music and user grouping. J Intell Inf Syst 24(2):113–132
13. Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the 4th ACM Conference on Recommender Systems, Barcelona, Spain, pp 39–46
14. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22(1):143–177
15. Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: [26], pp 107–144
16. Funk S (2006) Netflix Update: Try This at Home. `http://sifter.org/~simon/journal/20061211.html`, [accessed on February 2012]

17. Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In: Stølen K, Winsborough WH, Martinelli F, Massacci F (eds) Proceedings of the 4th International Conference on Trust Management, Pisa, Italy, Springer, Lecture Notes in Computer Science, vol 3986, pp 93–104

18. Gollapudi S, Sharma A (2009) An axiomatic approach for result diversification. In: Quemada J, León G, Maarek YS, Nejdl W (eds) WWW, ACM, pp 381–390

19. Herlocker JL, Konstan JA, Terveen LG, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53

20. Hurley N, Zhang M (2011) Novelty and diversity in top-n recommendation - analysis and evaluation. ACM Trans Internet Techn 10(4):14

21. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J (1997) Grouplens: Applying collaborative filtering to usenet news. Commun ACM 40(3):77–87

22. Koren Y, Bell RM (2011) Advances in collaborative filtering. In: [26], pp 145–186

23. Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80

24. Luo X, Xia Y, Zhu Q (2012) Incremental collaborative filtering recommender based on regularized matrix factorization. Knowl-Based Syst 27:271–280

25. McNee SM, Riedl J, Konstan JA (2006) Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: Olson GM, Jeffries R (eds) CHI Extended Abstracts, ACM, pp 1097–1101

26. Ricci F, Rokach L, Shapira B, Kantor PB (eds) (2011) Recommender Systems Handbook. Springer

27. Rosaci D, Sarnè GML (2012) A multi-agent recommender system for supporting device adaptivity in e-commerce. J Intell Inf Syst 38(2):393–418

28. Shih DH, Yen DC, Lin HC, Shih MH (2011) An implementation and evaluation of recommender systems for traveling abroad. Expert Syst Appl 38(12):15,344–15,355

29. Smyth B, McClave P (2001) Similarity vs. diversity. In: Aha DW, Watson I (eds) Proceedings of the 4th International Conference on Case-Based Reasoning, Vancouver, Canada, Springer, Lecture Notes in Computer Science, vol 2080, pp 347–361

30. Tan PN, Steinbach M, Kumar V (2005) Introduction to Data Mining. Addison-Wesley, (Chapter 8), Boston

31. Zhang M, Hurley N (2008) Avoiding monotony: improving the diversity of recommendation lists. In: Proceedings of the 2nd ACM Conference on Recommender Systems, Lausanne, Switzerland, pp 123–130

32. Zhang M, Hurley N (2009) Novel item recommendation by user profile partitioning. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Milan, Italy, pp 508–515

33. Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web, Chiba, Japan, pp 22–32

Table 1 Mean and standard deviations (SD) of pairwise similarities of items in the datasets.

	Mean	SD
MovieLens	0.077	0.075
Jester	0.154	0.170
Book-Crossing	0.019	0.031

Table 2 Time (ms) to generate a diversified recommendation list for a single user.

N	ClustDiv	BG (50)	BG (100)	BG (200)	BG (400)
5	0.001	0.068	0.127	0.338	0.539
10	0.003	0.225	0.322	1.091	1.664
15	0.003	0.313	0.619	2.121	4.315
20	0.004	0.481	1.072	3.523	6.983
25	0.005	0.636	1.516	5.129	10.075
30	0.006	0.859	2.028	7.027	14.647
35	0.008	0.994	2.607	8.773	20.364
40	0.011	1.163	3.219	11.007	24.021
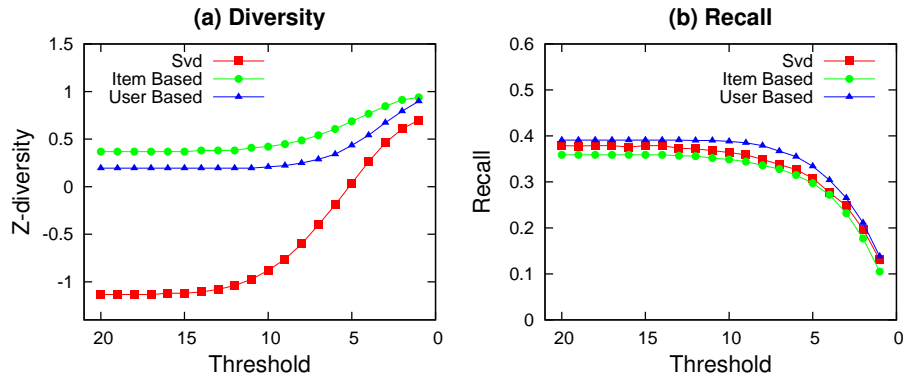
Fig. 1 MovieLens: (a) diversity and (b) recall results of ClusDiv.
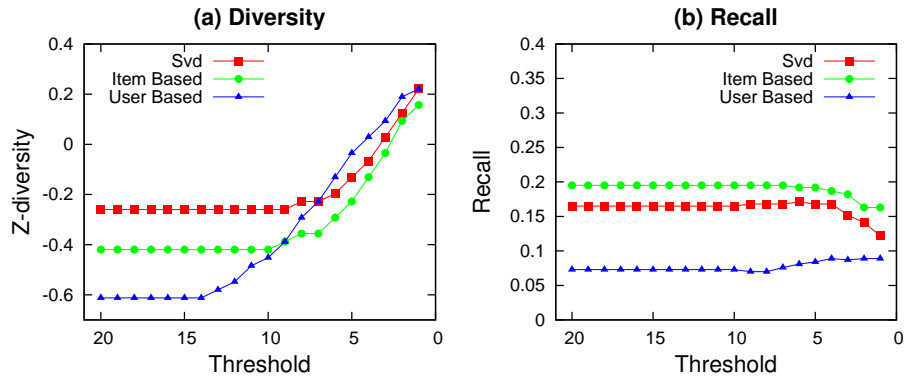
Fig. 2 Book-Crossing: (a) diversity and (b) recall results of ClusDiv.
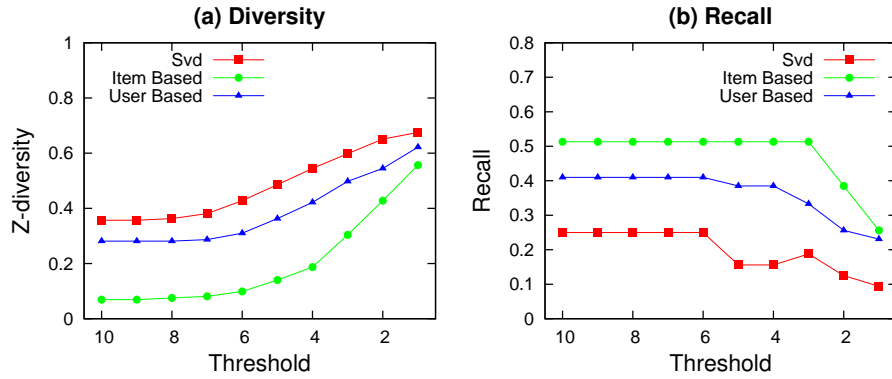
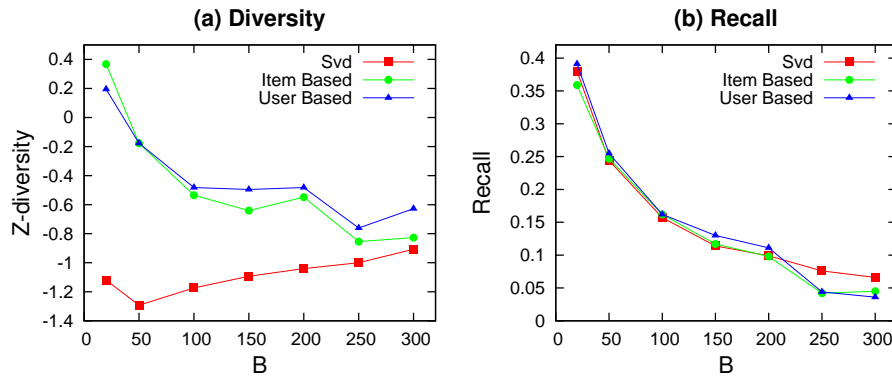**Fig. 3** Jester: (a) diversity and (b) recall results of ClusDiv.

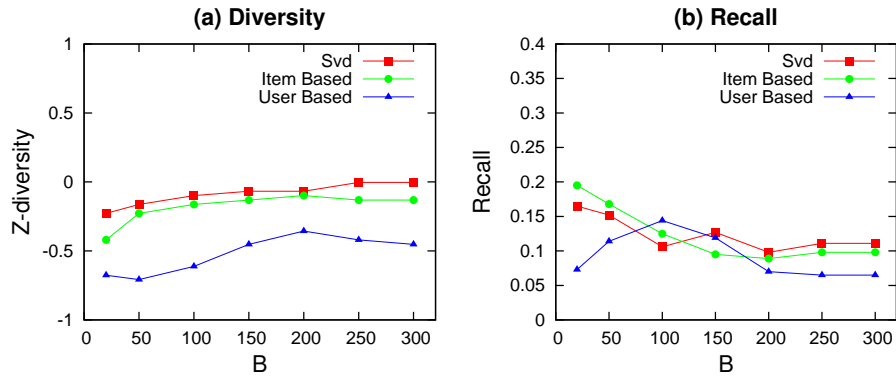**Fig. 4** MovieLens: (a) diversity and (b) recall results of random method.

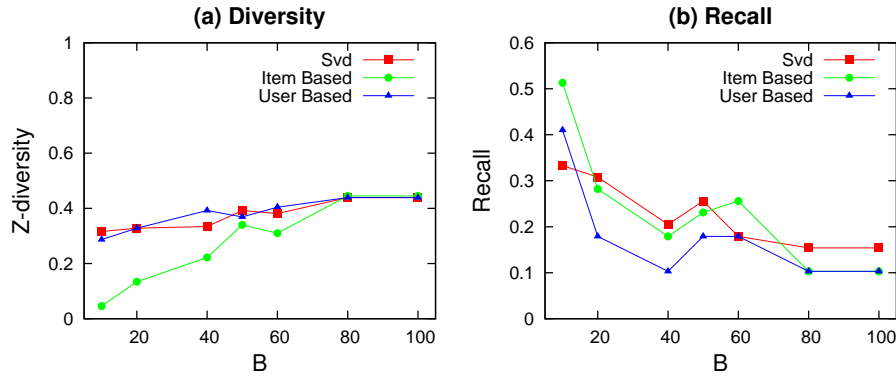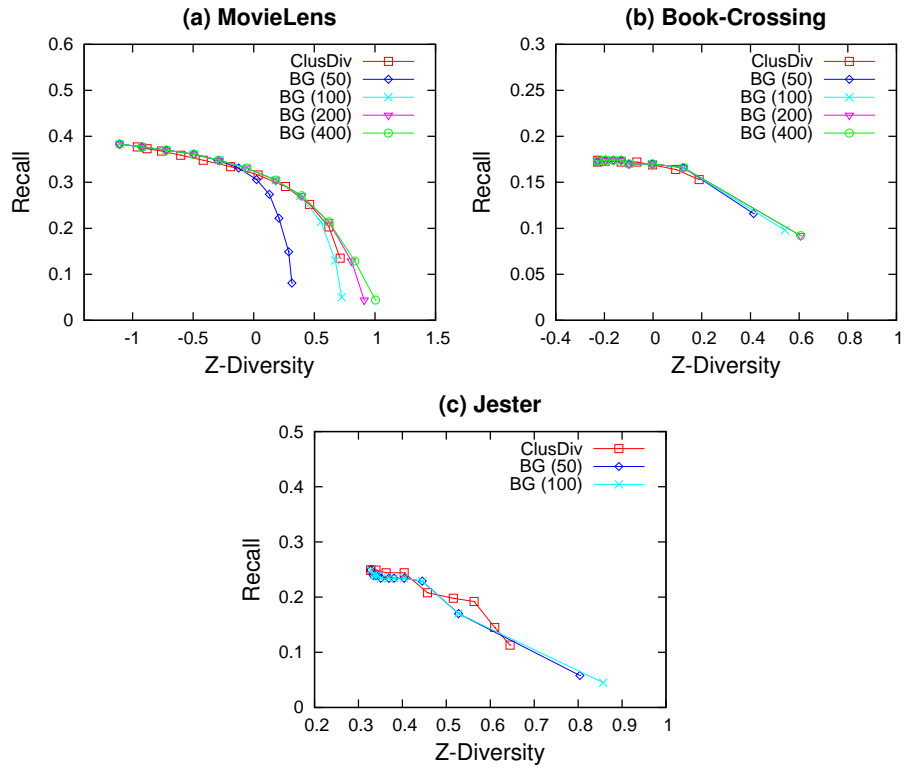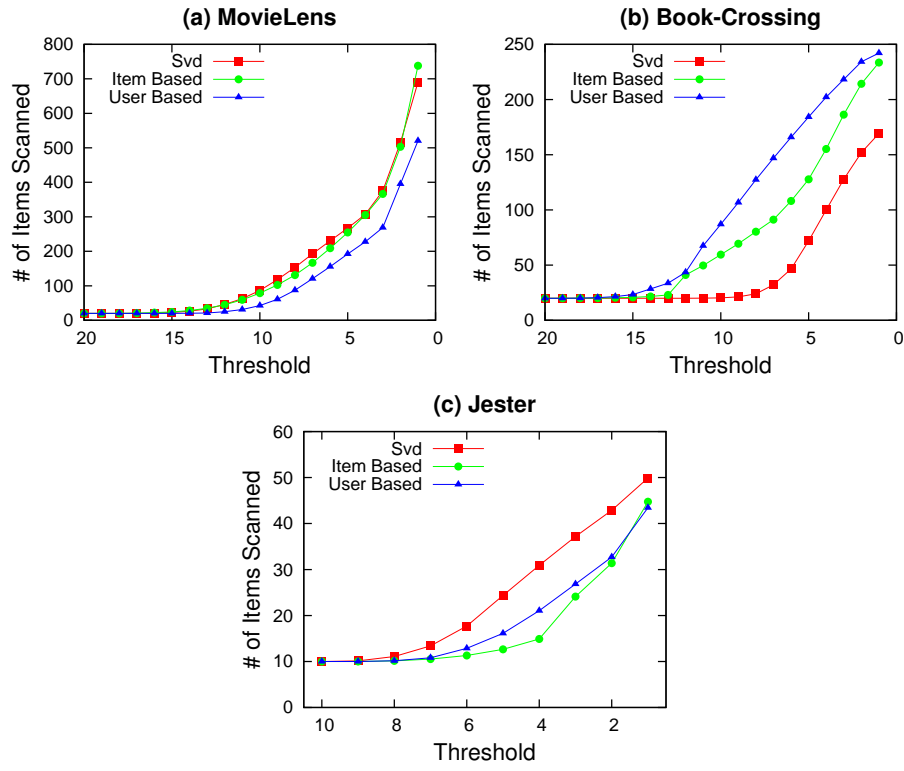**Fig. 5** Book-Crossing: (a) diversity and (b) recall results of random method.

Fig. 6 Jester: (a) diversity and (b) recall results of random method.

Fig. 7 Diversity vs. recall comparison of ClusDiv with the BG method in (a) MovieLens, (b) Book-Crossing, and (c) Jester datasets. SVD is used for rating prediction.

Fig. 8 Average number of items scanned for building the recommendation lists using (a) MovieLens, (b) Book-Crossing, and (c) Jester datasets as a function of threshold.