Noname manuscript No.
(will be inserted by the editor)

Speeding up Memory-based Collaborative Filtering with Landmarks

Gustavo R. Lima · Carlos E. Mello · Geraldo Zimbrao

the date of receipt and acceptance should be inserted later

Abstract Recommender systems play an important role in many scenarios where users are overwhelmed with too many choices to make. In this context, Collaborative Filtering (CF) arises by providing a simple and widely used approach for personalized recommendation. Memory-based CF algorithms mostly rely on similarities between pairs of users or items, which are posteriorly employed in classifiers like k-Nearest Neighbor (kNN) to generalize for unknown ratings. A major issue regarding this approach is to build the similarity matrix. Depending on the dimensionality of the rating matrix, the similarity computations may become computationally intractable. To overcome this issue, we propose to represent users by their distances to preselected users, namely landmarks. This procedure allows to drastically reduce the computational cost associated with the similarity matrix. We evaluated our proposal on two distinct distinguishing databases, and the results showed our method has consistently and considerably outperformed eight CF algorithms (including both memory-based and model-based) in terms of computational performance.

Keywords Recommender system · Collaborative filtering · Memory-based algorithms · Landmarks · Data reduction · Dimensionality reduction · Non-linear transformations

1 Introduction

The continuously improving network technology and the exponential growth of social networks have been connecting the whole world, putting available a huge volume of content, media, goods, services, and many other different kinds of items on Internet (Pasinato et al 2015). However, this phenomenon leads to the paradox of choice (Schwartz 2004). It addresses the problem that people overwhelmed with too many choices tend to be more anxious, and eventually give up to proceed with the order.

To tackle this issue, a massive effort has been made towards the development of data mining methods for recommender systems (Ricci et al 2011). This promising technology aims at helping users search and find items that are likely to be consumed, alleviating the burden of choice.

In this context, many recommender systems have been designed to provide users with suggested items in a personalized manner. A well-known and widely used approach for this kind of recommendation is Collaborative Filtering (CF) (Adomavicius and Tuzhilin 2005). It consists in considering the history of purchases and users' tastes to identify items that are likely to be acquired. In general, this data is represented by a rating matrix, where each row corresponds to a user, each column is assigned to an item, and each cell holds a rating given by the corresponding user and item. Thus, CF algorithms aim at predicting the missing ratings of the matrix, which are posteriorly used for personalized item recommendations.

Gustavo R. Lima · Geraldo Zimbrao
Federal University of Rio de Janeiro
E-mail: grlima@cos.ufrj.br; zimbrao@cos.ufrj.br

Carlos E. Mello
Federal University of the State of Rio de Janeiro
E-mail: mello@uniriotec.br

CF algorithms may be divide into two main classes: *memory-based* and *model-based* algorithms. The former class uses k-Nearest Neighbors (kNN) methods for rating predictions, and therefore relies on computing similarities between pairs of users or items according to their ratings (Koren and Bell 2011). The latter class employs matrix factorization techniques so as to obtain an approximation of the rating matrix, in which the unknown cells are filled with rating predictions (Koren et al 2009). Both memory-based and model-based algorithms provide advantages and disadvantages.

In this work, we are interested in memory-based algorithms. This class of CF algorithms remains widely used in many real systems due to its simplicity. It provides an elegant way for integrating information of users and items beyond the ratings for refining similarities (Shi et al 2014). In addition, memory-based CF algorithms allow *online* recommendations, something required in many practical applications as data is arriving constantly, new users are signing up, and new products are being offered (Abernethy et al 2007). So, incorporating such information in a *online* fashion is very desired to make up-to-date predictions on the fly by avoiding to re-optimize from scratch with each new piece of data.

The major issue regarding to memory-based CF algorithms lies in its computational scalability associated with the growth of the rating matrix (Shi et al 2014). As users are often represented by vectors of items (*i.e.* rows of the rating matrix), it turns out that the larger the number of items is, the higher the computational cost to compute similarities between users. Consequently, memory-based CF may become computationally intractable for a large number of users or items.

In this paper, we propose an alternative to improve the computational scalability of memory-based CF algorithms. Our proposal consists in representing users by their distances to preselected users, namely landmarks. Thus, instead of computing similarities between users represented by large vectors (often sparse) of ratings, our method calculates similarities through vectors of distances to fixed landmarks, obtaining an approximate similarity matrix for posterior rating predictions. As the number of landmarks required for a good approximation is mostly much smaller than the number of items, the proposed method drastically alleviates the cost associated with the similarity matrix computation.

The results show that our proposal consistently and considerably outperforms the evaluated CF algorithms (including both memory-based and model-based) in terms of computational performance. Interestingly, it achieves accuracy results better than the original memory-based CF algorithms with few landmarks.

The main contributions of this work are the following:

- A rating matrix reduction method to speed up memory-based CF algorithms.
- The proposal and investigation of 5 landmark selection strategies.
- An extensive comparison between our proposal and 8 CF algorithms, including both memory-based and model-based classes.

The work is organized in five sections, where this is the first one. Section 2 reviews the literature and presents the related work. Section 3 describes the recommendation problem definitions. It also introduces our proposal and presents the landmark selection strategies. Section 4 starts with the description of databases and metrics employed in experiments, follows by detailing the parameter tuning of the proposed method, and finishes by comparing our proposal against other CF algorithms. Finally, Section 5 points out conclusions and future work.

2 Related Work

Collaborative Filtering (CF) approach consists in predicting whether a specific user would prefer an item rather than others based on ratings given by users (Adomavicius and Tuzhilin 2005). For this purpose, CF uses only a rating matrix R, where rows correspond to users, columns correspond to items, and each cell holds the rating value r_{uv} given by user u to item v. Thus, the recommendation problem lies in predicting the missing ratings of R, which is often very sparse.

Interestingly, although there are many algorithms in Supervised Learning (SL) for data classification and regression, these are not properly suitable to CF, since ratings are not represented in a shared vector space $\mathbb{R}^d$. This happens because most users do not consume the same items by preventing their representation in the same vector space $\mathbb{R}^d$. Consequently, CF problem is slightly different from SL.

To overcome this issue, Braida et al. propose to build a vector space of latent factors to represent all item ratings given by users, and then apply SL techniques to predict unknown ratings. The authors use Singular Value Decomposition (SVD) to obtain user and item latent factors, and then build a vector space which contains all item ratings given by users. Their scheme consistently outperforms many state-of-the-art algorithms (Braida et al 2015).

Sarwar et al. also apply SVD on the rating matrix to reduce its dimensionality and transform it in a new feature vector space. Thus, predictions are generated by operations between latent factor matrices of users and items (Sarwar et al 2000).

Generally, dimensionality reduction techniques based on Matrix Factorization (MF) for CF are more efficient than other techniques, for instance Regularized SVD (Paterek 2007), Improved Regularized SVD (Paterek 2007), Probabilistic MF (Salakhutdinov and Mnih 2011) and Bayesian Probabilistic MF (Salakhutdinov and Mnih 2008). They have received great attention after Netflix Prize and are known as model-based CF algorithms (Breese et al 1998).

In contrast, memory-based CF algorithms are an adapted k-Nearest Neighbors (kNN) method, in which similarity is computed considering only co-rated items between users, *i.e.* the similarity between users are computed only for the vectors of co-rated items (Adomavicius and Tuzhilin 2005). Although model-based CF algorithms usually provide higher accuracy than the memory-based ones, the latter has been widely used (Beladev et al 2015; Elbadrawy and Karypis 2015; Li and Sun 2008; Pang et al 2015; Saleh et al 2015). This is due to its simplicity in providing an elegant way for integrating information of users and items beyond the ratings for refining similarities (Shi et al 2014). Additionally, memory-based algorithms allow *online* recommendations, making up-to-date predictions on the fly, which avoids to re-optimize from scratch with each new piece of data(Abernethy et al 2007). For these reasons, many authors seek to improve memory-based CF accuracy and performance, for example in (Bobadilla et al 2013; Gao et al 2012; Luo et al 2013).

A well-known problem present in memory-based CF algorithms lies in applying distance functions to users for calculating their similarities, which are computationally expensive. Often, the algorithm runtime increases with the number of users/items, becoming prohibitive to apply it on very large databases. Furthermore, finding a sub-matrix of R which contains all users and also is not empty might be impossible due to data sparsity, *i.e.* it is difficult to find an item vector subspace in which all users are represented.

To tackle these issues, we propose a method to reduce the size rating matrix via landmarks. It consists in selecting n users as landmarks, and then representing all users by their similarities to these landmarks. Thus, instead of representing users in item vector space, we propose to locate users in landmark vector space whose dimensionality is much smaller.

The landmark technique is useful to improve algorithm runtime and it was proposed by Silva and Tenenbaum in Multidimensional Scaling (MDS) context (Silva and Tenenbaum 2002). In this case, the authors propose a Landmark MDS (LMDS) algorithm, which uses landmarks to reduce the computational costs of traditional MDS. LMDS builds a landmark set by selecting few observations from data – the landmark set represents all observations. Then, it computes the similarity matrix for this set to obtain a suitable landmark representation in d-dimensional vector space. Finally, the other observations are mapped to this new space, considering their similarities to the landmarks (De Silva and Tenenbaum 2004).

The main advantage of using LMDS instead of other techniques is to adjust accuracy and runtime. If one needs to decrease runtime, it is possible to sacrifice accuracy by reducing the size of the landmark set. Otherwise, if one needs to improve the algorithm's accuracy, it is also possible to increase the number of landmarks up to the database limit. Therefore, a good LMDS characteristic is to manage this trade off between runtime and accuracy (Platt 2004).

Lee and Choi (Lee and Choi 2009) argue that noise in database harms LMDS accuracy, and then propose an adaptation for this algorithm, namely Landmark MDS Ensemble (LMDS Ensemble). They propose applying LMDS to different data partitions, and then combine individual solutions in the same coordinate system. Their algorithm is less noise-sensitive but maintains computational performance of LMDS.

Another pitfall of landmark approach is to choose the most representative observation as landmarks, once the data representation depends on the similarity to these points. Several selection strategies are proposed in literature (Chen et al 2006; Chi and Crawford 2013, 2014; Orsenigo 2014; Shi et al 2016, 2015; Silva et al 2005), most of them related to select landmarks for Land-

mark Isomap, which is a nonlinear reduction method variation to improve scalability (Babaeian et al 2015; Shang et al 2011; Silva and Tenenbaum 2002; Sun et al 2014).

Finally, Hu et al. (Hu et al 2009) tackle the problem of applying Linear Discriminant Analysis (LDA) on databases where the number of samples is smaller than the data dimensionality. They propose joining MDS and LDA in an algorithm, named as Discriminant Multidimensional Mapping (DMM), and also employ landmarks in DMM (LDMM) to improve scalability and turn it feasible to very large databases.

3 Proposal

We now present some basic definitions about memory-based Collaborative Filtering (CF) algorithms and discuss how it scales with the rating matrix size. Then, we follow by describing our proposal, which uses landmarks to improve the computational performance for computing the similarity matrix, and analyze its complexity. We also propose some selection strategies for the problem of choosing landmarks.

3.1 Problem Definition

For making predictions, memory-based CF algorithms consider similarities computed between pairs of users or pairs of items. Here, we assume that similarity is obtained between pair of users, namely used-based CF. The same can be done for pairs of items, namely item-based CF.

User-based CF considers only the co-rated items to compute similarities between users, and predicts ratings for not yet rated items given a particular user (Adomavicius and Tuzhilin 2005). Thus, the items with highest predicted ratings are recommended.

In order to formally define the rating prediction problem, let U, P, R be the set of users, the set of items and the rating matrix, respectively. Yet, let V be the set of possible rating values in the recommender system. Thus, the rows of R represent users and the columns represent items. If a user $u \in U$ rated an item $v \in P$ with the value $r_{uv} \in V$, then the cell at row u and column v of the matrix R holds the value r_{uv}, otherwise it is empty. Consequently, the matrix R dimension is $|U| \times |I|$ and, because most of the ratings are not provided, it is typically very sparse (Adomavicius and Tuzhilin 2005).

Let P_u denote the item subset rated by a particular user u, and $P_{uu'} = P_u \cap P_{u'}$ the subset of items co-rated by users u and u'. Note that, the recommender system aims at finding for a particular user u the item $v \in P \setminus P_u$ to which the user u is likely to be most interested. In other words, it estimates a function $f : U \times P \to V$ that predicts the rating $f(u, v)$ for a user u and an item v. We denote the predicted rating by $\hat{r}_{uv}$ (Ricci et al 2011).

To estimate this function, User-based CF employs a similarity measure $S : U \times U \to \mathbb{R}$ to determine the similarity $s_{uu'}$ between users u and u'. Thus, the predicted rating $\hat{r}_{uv}$ is obtained with the k-Nearest Neighbors (kNN) rule given by (1):

$$\hat{r}_{uv} = \frac{\sum\limits_{u' \in U \setminus \{u\}} s_{uu'} * (r_{u'v} - \bar{u'})}{\sum\limits_{u' \in U \setminus \{u\}} s_{uu'}} + \bar{u}, \tag{1}$$

where $\bar{u}$ and $\bar{u'}$ represents the mean rating value of users u and u', respectively.

The most costly procedure in user-based CF is to compute the user-user similarity matrix. As the similarity measure must be applied to each pair of users in the system, which are represented by item ratings, a typical user-based CF must perform two nested loops, as one may see in algorithm 1. These loops iterate over the user set and select a pair of users u and u' to compute their similarity. Thus, the algorithm complexity reaches $O(|U| \times |U| \times d)$, where d indicates the similarity measure complexity, in case the number of items.

Different similarity measures may be employed to build user-user similarity matrix and their complexity obviously depends on the number of operations performed. To compute the similarity between users u and u', the measure must iterates over the item set. The algorithm 2 computes the Cosine similarity.

Algorithm 1: Algorithm to build user-user similarity matrix

Data: user set U, similarity measure d
Result: user-user similarity matrix S
for $u \in U$ **do**
  for $u' \in U \setminus \{u\}$ **do**
    $S_{uu'} \leftarrow d(u, u')$
  end
end

Algorithm 2: The algorithm calculates the Cosine similarity between users u and u'

Data: users u and u', item set P, rating matrix R
Result: Cosine similarity $d_{uu'}$
$x, y, z \leftarrow 0$
if $|P_{uu'}| > 1$ **then**
  for $v \in P_{uu'}$ **do**
    $z \leftarrow z + r_{uv} * r_{u'v}$
    $x \leftarrow x + r_{uv}^2$
    $y \leftarrow y + r_{u'v}^2$
  end
  $d_{uu'} \leftarrow z/(\sqrt{x} * \sqrt{y})$
else
  $d_{uu'} \leftarrow -\infty$
end

Note that, algorithm 2 has a loop that iterates over co-rated items of users u and u'. Therefore, user-based CF algorithm performs three nested loops and, consequently, their complexity is $O(|U| \times |U| \times |P|)$, which explains why its performance quickly decreases as the number of users increases.

3.2 Building the New User Space

In user-based CF, users are represented in item vector space, where components are the corresponding item ratings. Therefore, the user space dimensionality is $|P|$.

To improve user-based CF performance, we propose representing users in a space whose dimensionality is much smaller than the original one. The new vector space basis consists of preselected users from the rating matrix R, namely landmarks. The new user vector components are composed of the similarities to each corresponding landmark.

We select n users from R, according to some criterion, like the number of item ratings. These n users constitute the landmark set. Each landmark belongs to the original item vector space, with dimensionality $|P|$.

To build the new user space, one applies for each user $u \in U$ (including the landmarks) a non-linear transformation, which provides a smaller dimensional space. This transformation consists in computing the similarities between users and landmarks. These values forms the components of the new user vector representation. Therefore, to improve user-based CF performance, one must choose $n \ll |P|$. Additionally, the most representative landmarks should be preferred.

Figure 1 illustrates the proposed method for a toy example. In Figure 1(a), the rating matrix R contains the item ratings given by users A, B, C, D, E and F. Missing ratings are indicated with '-'. Users B and C are selected as landmarks, considering the number of their given ratings (highlighted in red).

In Figure 1(b), the user-landmark similarity matrix is computed with Euclidean distance as a similarity measure. Note that, the rows of this matrix represent users, the two columns correspond to the landmarks B and C, and its cells to corresponding similarities between the user and each landmark.

At this step, users are totally represented by their distance to landmarks. In other words, the landmarks constitute the basis of the new vector space and each user is positioned on according to the landmarks.

An advantage of the new representation may be seen in the toy example: users A and D are co-rated at only one item, and therefore computing their similarity would produce a low accurate

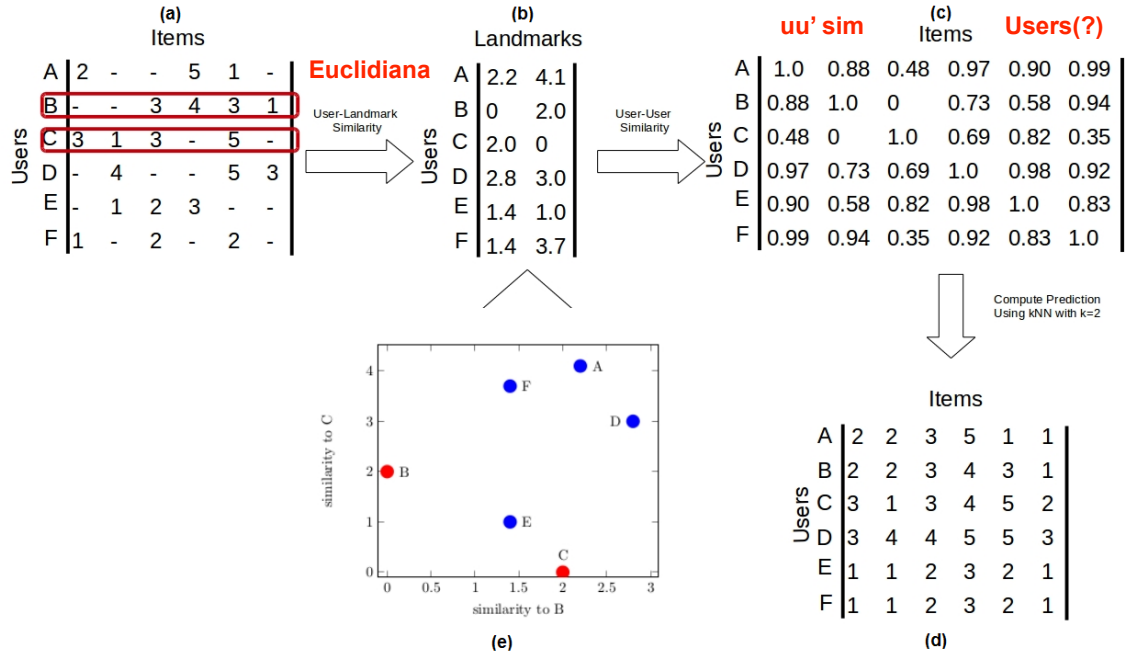


Fig. 1 The figure shows the procedures to apply the proposed algorithm to a toy example. The algorithm input is the rating matrix illustrated in (a). The first step of algorithm is select landmarks, and then represent users in new vector space, as showed in (b) and (e). Then, similarities between users are computed in (c), and ratings predicted in (d).

value. However, the new space locates users A and D according to their distance to landmarks B and C, which is here computed with more than one co-rated item. Therefore, users A and B are positioned near each other, as may be seen on Figure 1(e).

Once computed the new reduced user space, the next step is to compute the user-user similarity matrix based on this new representation. In Figure 1(c), one applied the Cosine similarity measure. Finally, it predicts the missing ratings with kNN approach, using k=2, as shown in Figure 1(d).

explicacao Alg 3

The computation of the user-user similarity matrix via landmarks is presented in algorithm 3. The first part of this algorithm consists in selecting n landmarks based on some criterion defined by the function $selectLandmarks$. After choosing the landmarks, users are represented in the landmark vector space by calculating their similarities to landmarks – the similarity measure d_1. Finally, the last step builds the user-user similarity matrix for the new landmark representation – the similarity measure d_2.

Algorithm 3: Algorithm to build user-user similarity matrix using landmarks.

Data: user set U, number of landmarks n, user-landmark similarity measure d_1, user-user similarity measure d_2 (Alg 2, Alg 4)
Result: user-user similarity matrix S
$L \leftarrow selectLandmarks(n)$
$H \leftarrow zeroMatrix(|U|, n)$
for $u \in U$ **do**
  for $l \in L$ **do**
    $H_{ul} \leftarrow d_1(u, l)$ Alg 2
  end
end
for $u \in U$ **do**
  for $u' \in U \setminus \{u\}$ **do**
    $S_{uu'} \leftarrow d_2(u, u')$ Alg 4
  end
end

The difference between similarity measures d_1 and d_2 is that the former considers user ratings to compute similarities, while the latter considers user-landmark similarities to achieve its purpose. Algorithms 2 and 4 illustrates both cases, respectively.

Algorithm 4: Algorithm to compute Cosine similarity between user u and u' which are represented in landmark vector space

Data: users u and u', landmark set L, user-landmark similarity matrix H
Data: Cosine similarity $S_{uu'}$
$x, y, z \leftarrow 0$
for $l \in L$ **do**
  if H_{ul} != $-\infty$ *and* $H_{u'l}$!= $-\infty$ **then**
    $z \leftarrow z + H_{ul} * H_{u'l}$
    $x \leftarrow x + H_{ul}^2$
    $y \leftarrow y + H_{u'l}^2$
  end
end
$S_{uu'} = z/(\sqrt{x} * \sqrt{y})$

3.3 Choosing Landmarks

The choice of landmarks is a critical task, once it directly affects the resulting vector space, and, consequently, influences the accuracy of user-based CF. To investigate this effect we propose five landmark selection strategies, described as follows:

- *Random* - n users are randomly chosen as landmarks.
- *Dist. of Ratings* - randomly chooses n users as landmarks by considering the distribution of ratings, *i.e.* users with more ratings are more likely to be selected.
- *Coresets* - chooses at random n landmark candidates taking into account the rating distribution. Then, it computes the user similarity to these candidates and removes half of the most similar users. From the remaining users, n new landmark candidates are again randomly chosen and the half most similar users to the candidates are removed. The algorithm proceeds until no remaining user exists in database. This strategy is based on *coresets*(Feldman et al 2011).
- *Coresets Random* - does similar to *Coresets*, but without considering the rating distribution, instead it just samples users uniformly at random.
- *Popularity* - ranks users/items in descend order by their number of ratings and selects the first n users as landmarks.

The proposed landmark selection strategies have underlying different criterion. Three of them – *Dist. of Ratings*, *Coresets* and *Popularity* – consider the number of ratings as criterion to select landmarks. Thus, we expect to select more representative landmarks compared with the other two strategies – *Random* and *Coresets Random*.

Additionally, *Random* and *Dist. of Ratings* are both the simplest, and consequently the fastest strategies, since landmarks are selected with no data preprocessing. *Popularity* is an intermediate case, as one needs to sort users/items by their number of ratings, which requires more computations than random sampling.

Finally, *Coresets* and *Coresets Random* are the most complex strategies. One should compute user similarities to n landmark candidates and remove the half most similar users from the entire set. This process proceeds until no user remains. Thus, both strategies require more computations than *Popularity*.

3.4 Complexity Analysis

In our proposal, users are represented in the landmark vector space. Once landmarks are chosen, the proposed algorithm performs three nested loops to compute the user-landmark similarity matrix. The first loop iterates over all users and the second one over n landmarks. Then, for each user u

and landmark l, the loop inside the similarity measure d_1 iterates over the co-rated items of u and l and computes the corresponding similarity value. Therefore, to build the user-landmark similarity matrix, one requires $O(|U| \times n \times |P|)$ steps.

To compute user-user similarity matrix, the algorithm performs three nested loops. The first two iterate over all users and the one inside the similarity measure d_2 iterates over n landmarks. Thus, it results in $O(|U| \times |U| \times n)$ steps.

Accordingly, the proposed algorithm complexity is $O(|U| \times n \times |P| + |U| \times |U| \times n) = O(|U| \times n \times (|P| + |U|))$.

Note that, this complexity becomes smaller as the number n of landmarks decreases. By comparing this result with the original complexity of user-based CF, $O(|U| \times |U| \times |P|)$, it turns out that $O(|U| \times n \times (|P| + |U|)) \leq O(|U| \times |U| \times |P|)$ if $n \leq \frac{|U| \times |P|}{|U| + |P|}$, what is a very realistic assumption.

4 Experiments

4.1 Methodology

In order to analyze the proposed method performance, we conduct experiments on two well-known databases: MovieLens (Harper and Konstan 2016; Miller et al 2003) and Netflix (Bennett and Lanning 2007). Our objective was twofold: (1) parameter investigation and settings – to investigate the functioning of the proposed method with regards to its parameter settings such as the number of landmarks, the similarity measure to build the landmark-user matrix, and the landmark selection strategy; and, (2) comparative analysis – to compare the proposal with different state-of-the-art algorithms.

In (1), the parameter investigation and settings, we start by varying the number of landmarks from 10 up to 100 for each selection strategy. The idea is to evaluate the algorithm prediction accuracy with different parameter settings and how these may be affected. Still, we also evaluate different similarity measures either to build the landmark-user/item matrix or the similarity matrix for rating predictions. The similarity measures were Euclidean, Cosine, and Pearson.

In (2), the comparative analysis, there were compared several algorithms from both memory-based and model-based algorithms. These are listed as follows:

1. Memory-based algorithms:
   (a) k-Nearest Neighbor (kNN) with Euclidean (Adomavicius and Tuzhilin 2005);
   (b) kNN with Cosine (Adomavicius and Tuzhilin 2005);
   (c) kNN with Pearson (Adomavicius and Tuzhilin 2005);
2. Model-based algorithms:
   (a) Regularized Singular Value Decomposition (SVD) (Paterek 2007);
   (b) Improved Regularized Singular Value Decomposition (Paterek 2007);
   (c) Probability Matrix Factorization (MF) (Salakhutdinov and Mnih 2011);
   (d) Bayesian Probability Matrix Factorization (Salakhutdinov and Mnih 2008); and
   (e) SVD++ (Koren 2008; Koren et al 2009).

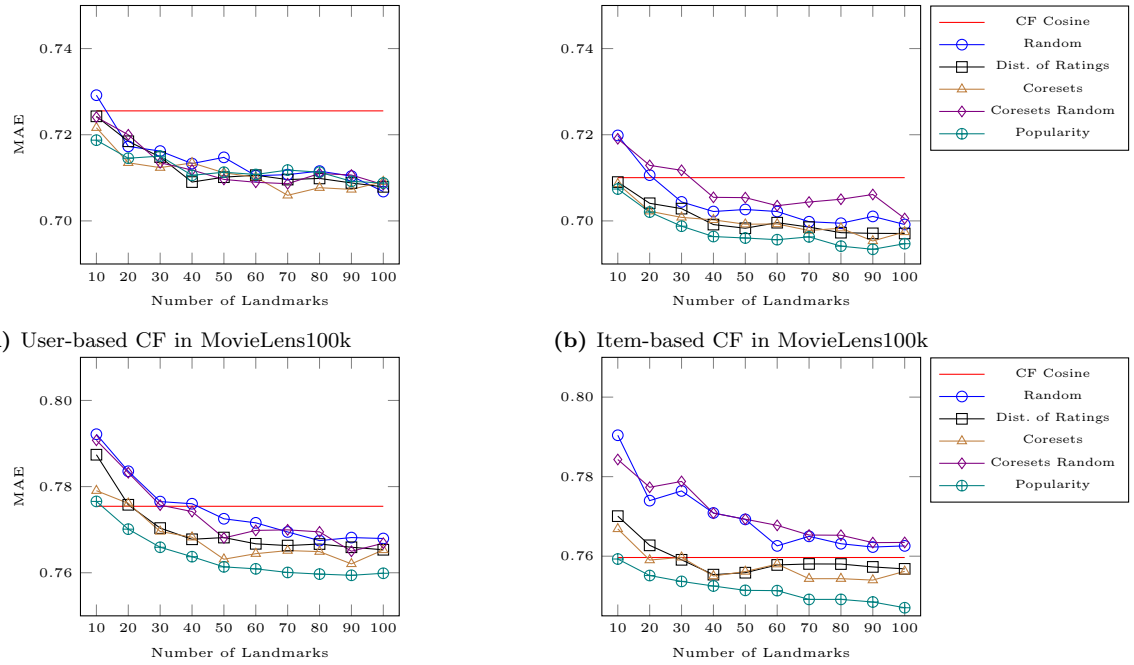
All experiments were carried out with 10-fold cross validation. There were considered the mean absolute error (MAE) to measure accuracy of rating predictions and the runtime in seconds to assess computational performance (Herlocker et al 2004).

4.2 Data sets

In order to pursue a fair evaluation, we consider two different data set sizes from MoviesLens and Netflix databases. In the former database, there were already two available data sets: MovieLens100k and MovieLens1M; each one with one hundred thousand (100k) and one million (1M) ratings, respectively(Herlocker et al 1999). In the latter database, Netflix, it was necessary to cut out the original database by generating data sets with amounts of ratings equal to the two MovieLens data sets. Thus, there were extracted 100k and 1M ratings in chronological recording order from the original database, obtaining two data sets: Netflix100k and Netflix1M, respectively. The idea of these cuts was to preserve temporal characteristics. Table 1 shows the number of users and items, and the sparsity of the rating matrix in the four data sets.

Table 1 Data set characteristics.

	#ratings	#users	#items	sparsity(%)
MovieLens100k	100k	943	1,682	6.3
Netflix100k	100k	1,490	2,380	2.82
MovieLens1M	1M	6,040	3,952	4.19
Netflix1M	1M	8,782	4,577	2.48



(a) User-based CF in MovieLens100k

(b) Item-based CF in MovieLens100k

(c) User-based CF in Netflix100k

(d) Item-based CF in Netflix100k

Fig. 2 The figure shows how MAE behaves as landmark increases for CF in 100k rating data sets.

4.3 Parameter investigation and settings

We here aim to analyze and discuss how the proposed algorithm works with different parameter values.

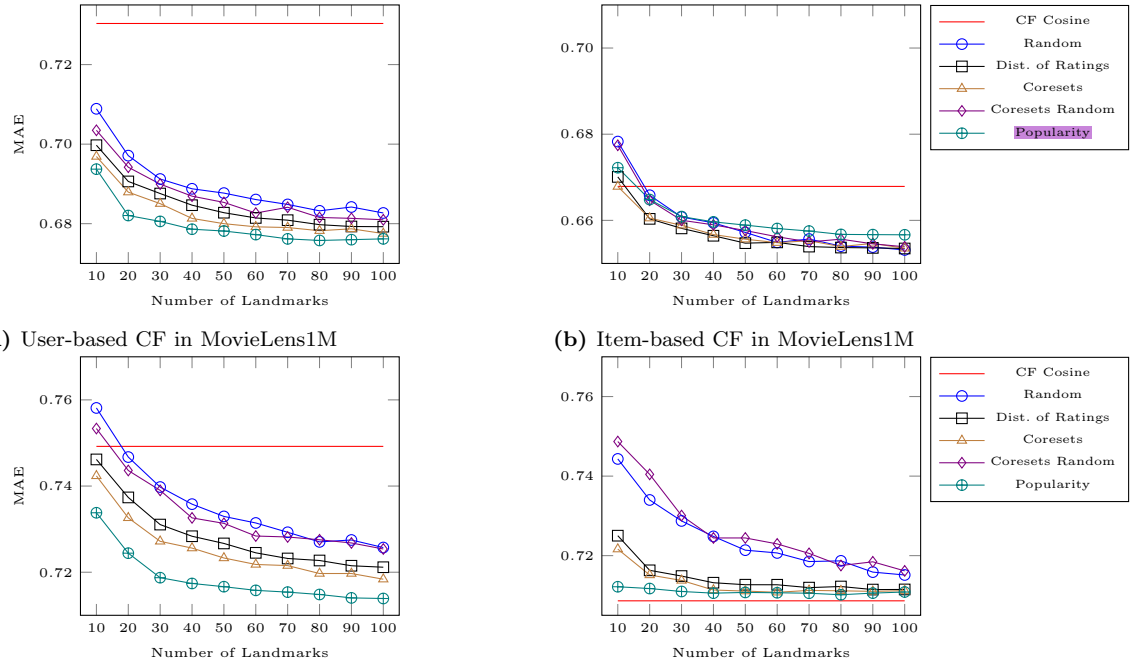
4.3.1 Accuracy Analysis

The graphs in Figure 2 and 3 present MAE per number of landmarks on the data sets of 100k and 1M ratings, respectively. The set of landmarks was varied from 10 up to 100, and, at each 10 landmarks, we compute MAE. This procedure was conducted for five landmark selection strategies: Random, Dist. of Ratings, Coresets, Coresets Random, and Popularity. In addition, these results were compared with the original version of the memory-based CF algorithm, *i.e.* user/item-based Collaborative Filtering (CF) with Cosine similarity, namely *baseline* algorithm.

In this experiment, the proposed method employed Euclidean distance to build the user-landmark matrix, and then obtained the user-user similarity matrix by Cosine distance, for reducing the dimensionality of user-based CF. Analogously, the same settings were adopted for item-based CF.

As one may observe, MAE decreases as the number of landmarks increases, and the proposed algorithm has outperformed their corresponding *baseline* algorithms with very few landmarks. This behavior was expected, once the more landmarks, the more information are supposedly retained throughout user/item representation.

By comparing landmark selection strategies, *Popularity* has produced the highest accuracies in most of the times. There were few cases in which *Popularity* did not outperform the others as one



(a) User-based CF in MovieLens1M

(b) Item-based CF in MovieLens1M

(c) User-based CF in Netflix1M

(d) Item-based CF in Netflix1M

Fig. 3 The figure shows how MAE behaves as landmark increases for CF in 1M rating data sets.

may observe for user-based CF on MovieLens100k and item-based CF on MovieLens1M. However, in these cases, no other strategy has also consistently outperformed the others.

Random and *Coresets Random* have shown the worst performance. Their corresponding MAE values were greater than the other strategies, despite these still remains below the *baseline* algorithms.

We consider that all strategies have performed similarly, especially on MovieLens database. Interestingly, in this database, the MAE difference between landmark selection strategies was very tight, less than 10^{-2} for both user-based and item-based CF. From a recommender system perspective, this difference may indicate no greater improvements for the final recommendation list.

One may also note two distinct groups of landmark selection strategies in both Figure 2d and 3d. One group uniformly selects landmark at random: *Random* and *Coreset Random*; and another group composed of those strategies that take into account the number of ratings: *Dist. of Ratings*, *Coresets*, and *Popularity*. The difference in accuracy between these groups was higher for few landmarks. This leads us to claim that it is preferred to choose landmarks with more ratings, thus one obtains more co-rated items, and consequently, more 'representativeness' in the new user space.

In Tables[1] 2, 3, 4 and 5, we present the results obtained by the proposed method with different combinations of similarity measures to build both the user-landmark matrix and the user-user similarity matrix. There were evaluated three distance measures: Euclidean, Cosine and Pearson. We fixed the number of landmarks in 20 for MovieLens data sets, and 30 for Netflix.

One should note that, *Popularity* with Cosine distance to build both matrices (user-landmark and user-user) in user-based CF has achieved the best accuracy overall. In item-based CF, the best measure combination was Cosine and Pearson distances to build item-landmark and item-item matrices, respectively. Nevertheless, *Popularity* has performed with relatively similar accuracy to the other combinations of similarity measures. This similar behavior holds for other selection strategies by differing in MAE about 10^{-2} and 10^{-3}, what is insignificant.

Accordingly, we conclude that accuracy increases with the number of landmarks, the most accurate landmark selection strategies are those based on rating distribution, and the choice of similarity measures does not bring significant advantages for accuracy. Additionally, it is possible

[1] The best result for each landmark selection strategy is highlighted in bold and is marked with an asterisk ('*'). The best result overall is also in bold but marked with double asterisk ('**').

Table 2 MAE of the user/item-based CF on MovieLens100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	0.725	**0.708***	0.724	0.705	0.724	0.705	0.723	**0.711***	0.723	0.706
	Cosine	**0.719***	0.710	0.718	0.704	0.717	0.704	0.719	0.715	0.715	0.702
	Pearson	0.721	0.711	0.717	0.705	0.714	0.704	0.718	0.715	0.713	0.700
Cosine	Euclidean	0.724	0.715	0.715	**0.703***	0.710	**0.698***	**0.716***	0.717	0.705	0.690
	Cosine	0.720	0.717	**0.711***	0.705	**0.707***	0.699	0.719	0.717	**0.701****	0.690
	Pearson	**0.719***	0.715	0.713	0.705	0.708	0.702	0.719	0.718	0.703	**0.688****
Pearson	Euclidean	0.723	0.721	0.716	0.711	0.709	0.704	0.721	0.718	0.703	0.696
	Cosine	0.722	0.718	0.715	0.707	0.712	0.703	0.722	0.725	0.703	0.696
	Pearson	0.728	0.717	0.722	0.711	0.716	0.708	0.729	0.726	0.709	0.698

Table 3 MAE of the user/item-based CF on Netflix100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	0.786	**0.770***	0.784	0.767	0.785	0.766	0.787	**0.771***	0.786	0.764
	Cosine	0.784	0.771	0.774	0.758	0.776	0.758	0.779	0.778	0.770	0.754
	Pearson	0.784	0.778	0.776	0.761	0.776	0.760	0.781	0.777	0.770	0.746
Cosine	Euclidean	0.776	0.773	**0.763***	**0.750***	0.761	**0.743***	0.774	0.779	0.757	0.740
	Cosine	0.775	0.774	0.764	0.754	0.759	0.747	**0.772***	0.779	**0.752****	0.739
	Pearson	**0.774***	0.775	0.767	0.753	**0.758***	0.742	0.776	0.782	0.755	**0.730****
Pearson	Euclidean	0.781	0.784	0.777	0.761	0.769	0.752	0.779	0.786	0.763	0.745
	Cosine	0.787	0.784	0.777	0.762	0.771	0.751	0.783	0.788	0.768	0.743
	Pearson	0.791	0.786	0.782	0.763	0.780	0.753	0.787	0.794	0.775	0.747

Table 4 MAE of the user/item-based CF on MovieLens1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	0.704	0.670	0.700	0.671	0.699	0.672	0.701	0.671	0.697	0.677
	Cosine	0.698	**0.666***	0.691	**0.660***	0.688	0.661	0.694	**0.665***	0.682	0.665
	Pearson	**0.695***	0.669	0.691	0.661	0.688	0.662	0.695	0.666	0.681	0.666
Cosine	Euclidean	0.698	0.677	**0.689***	0.665	0.684	0.660	**0.692***	0.671	0.676	0.659
	Cosine	0.697	0.681	**0.689***	0.665	**0.682***	0.659	0.694	0.675	**0.673****	0.651
	Pearson	0.703	0.679	0.690	0.663	0.684	**0.656***	0.695	0.675	**0.673****	**0.648****
Pearson	Euclidean	0.706	0.681	0.693	0.670	0.688	0.663	0.698	0.679	0.679	0.658
	Cosine	0.704	0.684	0.694	0.668	0.689	0.661	0.698	0.679	0.679	0.655
	Pearson	0.712	0.684	0.699	0.668	0.692	0.662	0.705	0.682	0.680	0.657

to outperform the corresponding *baseline* algorithms with very few landmarks – 10 to 40 landmarks quickly improve accuracy.

4.3.2 Computational Performance Analysis

Table 6, 7, 8 and 9 show the corresponding time required by the proposed algorithm for different parameter settings and data sets. The idea is to investigate the impact on the computational performance with regards to the number of landmarks, the distance measures, and the selection strategies. Thus, we consider the runtime in seconds to build the similarity matrix and to compute rating predictions for the test set.

As one would expect, the time increases almost linearly with the number of landmarks. As the dimensionality of user-landmark matrix grows, more computations are necessary to calculate the user-user similarity matrix. Analogously, the same behavior is observed for item-based CF.

Table 5 MAE of the user/item-based CF on Netflix1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	0.741	0.734	0.738	0.730	0.737	0.730	0.740	0.735	0.731	0.728
	Cosine	0.738	**0.727***	0.730	0.716	0.728	0.714	0.738	0.730	0.719	0.711
	Pearson	0.739	0.728	0.731	0.715	0.726	0.713	0.739	**0.728***	0.717	0.709
Cosine	Euclidean	0.736	0.738	0.723	0.705	0.717	0.702	**0.729***	0.740	0.711	0.701
	Cosine	0.736	0.737	**0.720***	**0.701***	**0.715***	0.696	**0.729***	0.737	**0.708****	0.693
	Pearson	**0.733***	0.731	0.722	**0.701***	0.716	**0.695***	0.731	0.735	0.710	**0.691****
Pearson	Euclidean	0.739	0.738	0.728	0.704	0.721	0.699	0.734	0.740	0.714	0.697
	Cosine	0.742	0.734	0.728	0.705	0.722	0.696	0.737	0.740	0.714	0.692
	Pearson	0.745	0.740	0.733	0.707	0.725	0.700	0.741	0.738	0.716	0.697

Table 6 User/Item-based CF runtime, in seconds, on MovieLens100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Landmarks	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
10	0.4	0.7	0.4	0.9	1.5	2.2	1.4	1.9	0.5	1.0
20	0.8	1.3	0.8	1.6	2.6	3.9	2.6	3.2	0.9	1.8
30	1.2	2.0	1.2	2.5	3.7	5.7	3.6	4.5	1.3	2.6
40	1.6	2.7	1.6	3.2	4.7	7.1	4.6	5.8	1.7	3.5
50	2.0	3.4	2.0	4.0	5.7	8.8	5.6	7.2	2.1	4.4
60	2.4	4.1	2.5	4.9	6.7	10.3	6.5	8.5	2.6	5.3
70	2.8	4.8	2.9	5.7	7.8	11.8	7.7	9.5	3.0	6.2
80	3.2	5.5	3.3	6.5	8.7	13.3	8.4	10.7	3.4	7.0
90	3.6	6.2	3.7	7.3	9.5	14.7	9.4	11.7	3.8	7.8
100	4.0	6.9	4.1	8.1	10.6	16.2	10.2	13.1	4.2	8.6

Table 7 User/Item-based CF runtime, in seconds, on Netflix100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Landmarks	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
10	0.9	1.2	0.9	1.5	3.0	3.6	2.8	3.0	1.0	1.7
20	1.6	2.2	1.8	2.7	5.5	6.5	5.1	5.3	1.8	3.2
30	2.4	3.2	2.6	4.1	8.0	9.4	7.5	7.7	2.8	4.6
40	3.2	4.3	3.5	5.4	10.4	12.4	9.7	10.2	3.6	6.1
50	4.0	5.2	4.4	6.7	12.7	15.2	12.1	12.5	4.5	7.6
60	4.8	6.5	5.2	8.2	15.2	18.2	14.0	14.8	5.4	9.0
70	5.5	7.4	6.0	9.4	17.2	20.8	16.3	17.1	6.3	10.5
80	6.4	8.5	6.8	11.0	19.5	24.6	18.4	19.6	7.2	12.2
90	7.1	9.7	7.8	12.2	21.7	26.7	20.2	22.0	8.1	13.3
100	8.0	10.6	8.6	13.3	23.7	29.3	22.4	24.5	9.0	14.8

In terms of landmark selection strategy, one should note that the simpler strategies outperform the more complex ones. *Random* was away the fastest selection strategy, followed in order by *Dist. of Ratings*, *Popularity*, *Coresets Random*, and *Coresets*.

The times spent by the *baseline* algorithms are presented in Table 10. Thus, our proposal may achieve a reduction up to 99.22% in terms of runtime.

Random, *Dist. of Ratings* and *Popularity* have performed faster than the corresponding *baseline* algorithms for any number of landmarks between 10 and 100 on both data sets of 100k ratings. *Coresets* and *Coresets Random* have made the proposed algorithm slower due to their own complexities. Consequently, the proposal becomes slower than the *baseline* algorithms after 80 landmarks on MovieLens100k for user-based CF, and after adding 100 landmarks for item-based CF.

Interestingly, all *baseline* algorithms on 1M-sized data set have taken more time to perform than the proposed algorithm for any parameter setting (number of landmarks and selection strategy). Thus, the proposal may succeed well on very large databases.

Table 8 User/Item-based CF runtime, in seconds, on MovieLens1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Landmarks	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
10	14.4	7.7	15.3	8.6	32.0	23.5	30.8	21.6	15.9	8.9
20	27.2	14.6	28.8	16.3	61.0	46.5	59.5	42.6	29.7	16.8
30	41.2	21.8	42.8	24.7	92.4	69.4	88.2	64.9	44.7	25.5
40	54.8	30.0	57.2	32.9	123.0	92.7	119.9	84.8	59.7	33.9
50	68.5	36.9	72.0	41.0	152.7	114.4	148.5	105.6	75.6	42.2
60	82.1	45.2	88.2	49.2	185.2	136.2	176.0	125.0	88.6	51.8
70	95.3	53.3	99.9	58.0	210.7	157.0	203.2	143.5	102.6	59.6
80	107.9	60.7	114.3	66.5	241.3	178.6	233.1	163.0	117.7	67.9
90	126.2	68.3	131.3	74.5	271.7	199.0	261.2	179.7	134.0	76.4
100	136.8	75.5	142.8	82.6	295.3	220.1	286.1	200.7	147.3	84.5

Table 9 User/Item-based CF runtime, in seconds, on Netflix1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Landmarks	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
10	26.8	9.8	29.8	11.2	58.9	35.6	56.0	30.0	31.1	12.0
20	50.4	18.1	55.5	21.6	113.2	69.2	105.5	60.5	57.7	23.4
30	74.9	27.2	82.2	32.2	170.9	103.3	160.1	88.4	86.7	33.8
40	103.0	36.2	110.7	42.6	227.2	137.3	216.8	118.2	114.1	45.0
50	127.5	45.4	138.6	53.5	285.6	165.4	268.7	144.2	143.7	54.9
60	153.1	54.3	169.3	63.9	340.2	197.9	316.3	173.1	170.6	66.4
70	177.7	63.9	194.7	74.2	394.0	231.8	366.6	204.9	199.8	79.8
80	201.5	74.2	220.9	86.2	447.9	263.5	415.6	232.5	231.6	91.6
90	227.3	83.6	250.3	97.5	499.7	286.9	462.3	250.5	257.0	98.1
100	253.9	89.8	277.0	104.3	552.7	313.5	508.6	278.6	286.1	111.6

Table 10 The runtime, in seconds, of user/item-based CF with Cosine in databases.

	CF Type	MovieLens100k	Netflix100k	MovieLens1M	MovieLens1M
Time (s)	User-based	7.9	24.3	1250.9	3219.8
	Item-based	15.2	42.6	758.2	1260.0

Table 11 User/Item-based runtime, in seconds, on MovieLens100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	0.70	**1.19****	0.71	**1.39***	2.59	3.66	2.50	3.08	0.75	**1.47***
	Cosine	0.82	1.37	0.86	1.64	2.69	3.96	2.56	3.15	0.86	1.79
	Pearson	1.20	2.03	1.25	2.55	3.17	4.88	3.02	3.77	1.30	2.82
Cosine	Euclidean	**0.67****	1.21	**0.70***	**1.39***	**2.28***	**3.35***	**2.20***	**2.81***	**0.71***	**1.47***
	Cosine	0.78	1.32	0.82	1.64	2.40	3.63	2.31	2.86	0.84	1.81
	Pearson	1.16	2.05	1.21	2.56	2.81	4.56	2.72	3.54	1.25	2.81
Pearson	Euclidean	1.01	1.39	1.03	1.67	2.88	3.90	2.78	3.24	1.08	1.81
	Cosine	1.11	1.53	1.19	1.90	3.07	4.19	2.95	3.33	1.24	2.11
	Pearson	1.49	2.16	1.59	2.76	3.47	5.14	3.29	3.86	1.68	3.08

Tables[1] 11, 12, 13 and 14 present the computational performance with regards to different distance measures for similarity. To build user-landmark and user-user matrices, the fastest distance combinations were Euclidean-Euclidean and Cosine-Euclidean. The same behavior can be observed for the item-based CF. Person has presented the worst performance due to its nonlinear computational complexity, which becomes worse on the data sets of 1M ratings.

Therefore, any of these distance measures may be applied to small databases without great influence on the computational performance. Nevertheless, Euclidean and Cosine should be preferred

Table 12 User/Item-based CF runtime, in seconds, on Netflix100k. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	**1.44****	**2.92****	**1.55***	**3.46***	**5.39***	8.61	**5.08***	7.34	**1.59***	**3.78***
	Cosine	1.65	3.07	1.84	3.88	5.64	9.20	5.31	7.36	1.92	4.45
	Pearson	2.35	4.11	2.67	5.84	6.61	11.15	5.99	8.48	2.81	6.73
Cosine	Euclidean	2.27	2.96	2.38	3.51	7.66	**8.60***	7.17	**7.24***	2.47	3.80
	Cosine	2.52	3.08	2.79	3.98	7.90	9.12	7.31	7.45	2.94	4.49
	Pearson	3.71	4.10	4.11	5.83	9.30	11.24	8.60	8.31	4.29	6.74
Pearson	Euclidean	1.94	3.30	2.15	4.11	5.97	9.30	5.57	7.67	2.27	4.56
	Cosine	2.11	3.42	2.42	4.39	6.31	9.68	5.93	7.67	2.59	5.03
	Pearson	2.77	4.15	3.19	5.90	7.09	11.32	6.48	8.46	3.48	6.80

Table 13 User/Item-based CF runtime, in seconds, on MovieLens1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	**22.44***	13.58	**23.43****	14.56	55.21	44.98	54.03	41.87	**23.68***	14.81
	Cosine	26.95	15.28	28.88	16.70	61.24	47.32	59.00	43.15	29.66	17.09
	Pearson	42.55	19.67	44.80	22.49	77.83	53.14	74.95	48.52	46.42	23.26
Cosine	Euclidean	22.93	**12.52****	23.47	**13.52***	**53.58***	**40.34***	**53.67***	**37.60***	24.16	**13.59***
	Cosine	27.87	13.16	29.69	15.02	62.05	41.78	59.87	38.35	29.80	15.57
	Pearson	42.16	17.98	45.60	20.50	76.33	47.99	73.77	43.94	47.09	21.46
Pearson	Euclidean	29.71	19.12	31.55	21.38	64.25	52.21	62.16	47.79	32.72	22.64
	Cosine	33.61	20.14	36.72	23.46	69.50	54.25	66.20	48.47	38.36	24.80
	Pearson	48.09	25.52	52.28	29.31	85.78	60.48	81.24	53.62	55.14	30.85

Table 14 User/Item-based CF runtime, in seconds, on Netflix1M. 'UCF' and 'ICF' stand for User-based CF and Item-based CF, respectively.

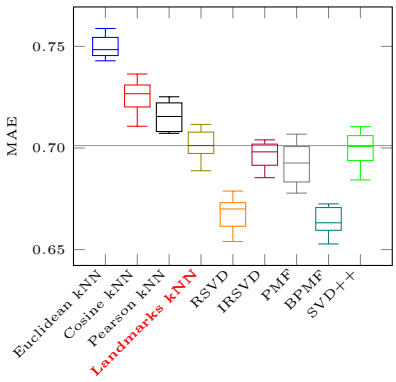
Users-Landmarks	User-User	Random		Dist. of Ratings		Coresets		Coresets Random		Popularity	
		UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF	UCF	ICF
Euclidean	Euclidean	64.06	**25.07****	67.69	**27.85***	153.63	**96.31***	146.65	**84.85***	68.62	**28.92***
	Cosine	75.96	26.21	81.72	30.56	170.05	99.91	159.74	86.23	84.92	32.87
	Pearson	114.99	31.49	126.05	40.08	215.21	109.13	196.14	92.30	132.18	42.85
Cosine	Euclidean	**60.80****	25.93	**64.15***	28.97	**145.08***	97.36	**138.86***	85.60	**66.03***	30.00
	Cosine	71.33	26.40	78.23	31.39	159.55	101.11	148.97	87.01	81.33	33.58
	Pearson	108.43	33.81	121.19	41.11	203.26	110.32	189.93	91.68	126.64	43.99
Pearson	Euclidean	80.55	33.13	87.49	42.02	174.98	111.45	165.95	92.20	91.35	44.94
	Cosine	89.80	34.73	101.65	43.92	189.86	114.41	174.55	92.30	107.27	47.98
	Pearson	126.18	38.95	145.39	52.65	238.83	122.86	219.20	97.15	156.31	57.40

in case of very large databases. It is also interesting to highlight that our proposal has considerably reduced the runtime compared to the *baseline* algorithms, even with Pearson distance.

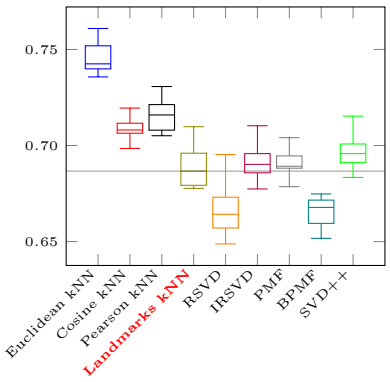
4.4 Comparative Analysis

We here aim at comparing the proposed algorithm in terms of accuracy and performance to CF algorithms of the state-of-the-art. In this experiment, our proposal is set with *Popularity* for landmark selection, and Cosine similarity to build both reduced and similarity matrix. Additionally, one used 20 landmarks for MovieLens data sets and 30 for Netflix. Yet, both user-based and item-based CF were set with 13 neighbors for rating prediction.

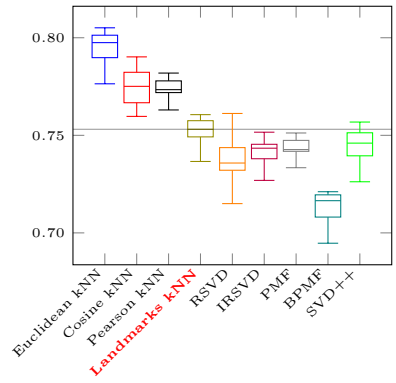
The comparison considers 8 recommender algorithms: kNN with Euclidean, kNN with Cosine, kNN with Pearson, Regularized Singular Value Decomposition (RSVD), Improved Regularized Singular Value Decomposition (IRSVD), Probability Matrix Factorization (PMF), Bayesian Prob-



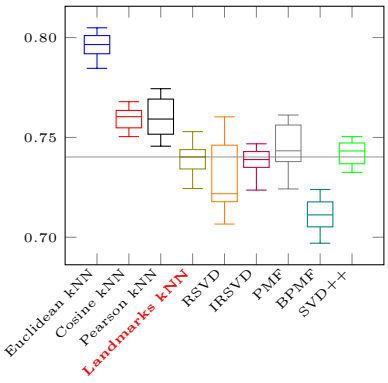
(a) User-based CF in MovieLens100k



(b) Item-based CF in MovieLens100k



(c) User-based CF in Netflix100k



(d) Item-based CF in Netflix100k

Fig. 4 The figure compares proposal MAE against the chosen baselines in 100k rating data sets.

ability Matrix Factorization (BPMF), and SVD++. The proposed algorithm is here referred as *Landmarks kNN*.

4.4.1 Accuracy Analysis

Figures 4 and 5 show accuracies achieved by the proposal and the other CF algorithms. The horizontal line crossing the boxplot graph indicates the MAE median of 10-fold cross validation obtained with *Landmarks kNN*.

As one should note, most of memory-based CF algorithms have higher MAE than model-based ones, mainly on the data sets of 1M ratings. This was expected, once model-based CF has been often superior in the sense of accuracy.

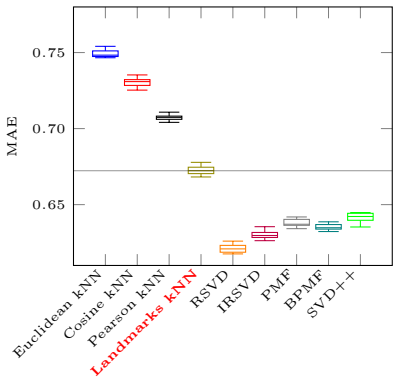
In spite of *Landmarks kNN* has been classified as memory-based, it outperformed some model-based ones like IRSVD, PMF and SVD++, and yielded higher accuracy for the item-based CF on the data sets of 100k ratings, as one may see in Figure 4. However, this behavior does not stand on the data sets of 1M ratings, wherein model-based CF algorithms have outperformed our proposal.

Among memory-based CF algorithms, Euclidean kNN has achieved the highest MAE, while Cosine kNN and Pearson kNN have reached intermediate values.

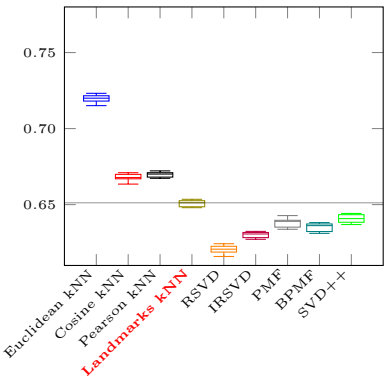
In case of model-based CF approach, BPMF has shown the lowest MAE on the data sets of 100k ratings, while RSVD has outperformed all model-based CF algorithms on the data sets of 1M ratings.

Figure 5 clearly distinguishes the two groups of CF approaches, *i.e.* memory-based and model-based. The first one has provided higher MAE than the proposed method, *Landmarks kNN*. On the other hand, the second group has beaten our proposal with relative margin.

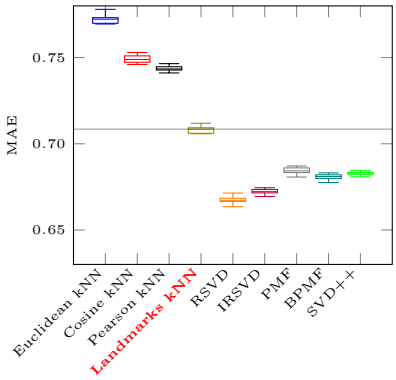
Therefore, we may note that our proposal outperformed memory-based CF algorithms on all data sets. Additionally, it has yielded similar accuracy to some model-based techniques on the data sets of 100k ratings.



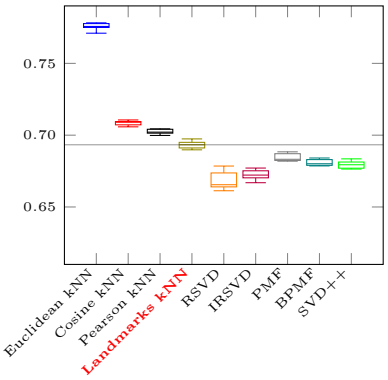
(a) User-based CF in MovieLens1M



(b) Item-based CF in MovieLens1M



(c) User-based CF in Netflix1M



(d) Item-based CF in Netflix1M

Fig. 5 The figure compares proposal MAE against the chosen baselines in 1M rating data sets.

Table 15 The table presents how many times the corresponding algorithm is slower than the proposal, which appears in bold. For instance, Euclidean kNN is 8.7 times slower than *Landmarks kNN* on MovieLens100k.

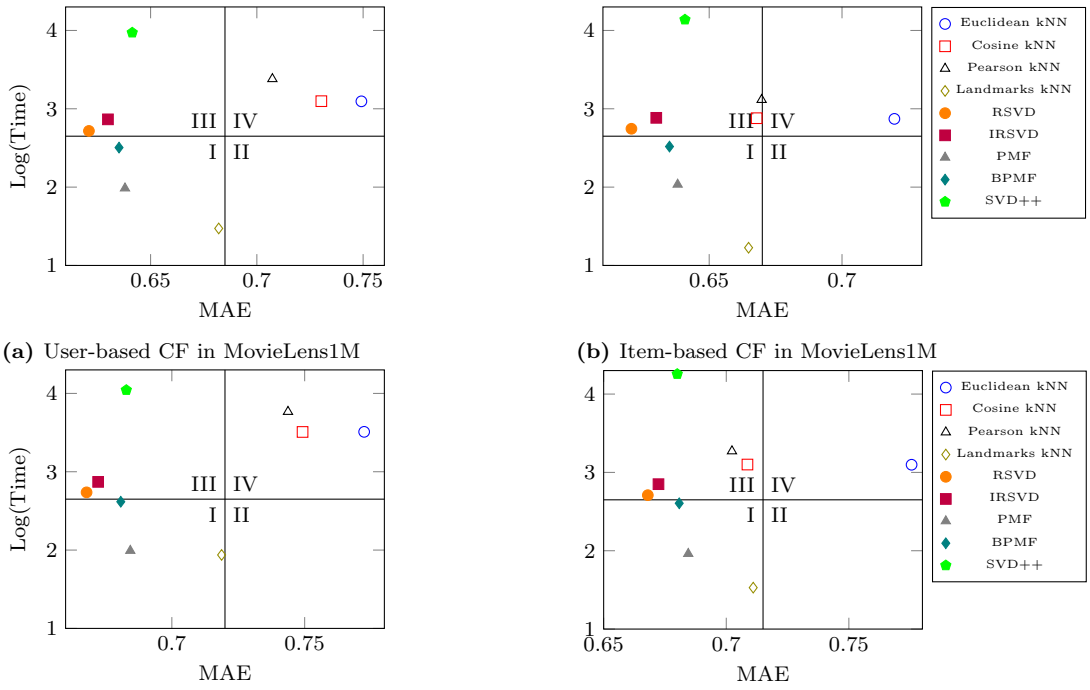
CF Technique	User-based				Item-based			
	MovieLens		Netflix		MovieLens		Netflix	
	100k	1M	100k	1M	100k	1M	100k	1M
Euclidean kNN	8.7	39.5	8.9	39.2	8.3	44.8	9.1	37.6
Cosine kNN	8.8	39.7	9.0	39.1	8.4	45.7	9.2	37.8
Pearson kNN	17.1	76.3	15.7	70.8	14.2	78.5	13.1	56.1
Landmarks kNN	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
RSVD	49.2	16.6	15.8	6.6	23.3	33.5	9.8	15.4
IRSVD	70.9	23.3	22.8	9.0	33.9	46.2	13.8	21.2
PMF	8.3	3.1	2.8	1.2	4.1	6.5	1.7	2.7
BPMF	50.3	10.1	24.2	5.0	24.8	19.8	14.5	12.1
SVD++	437.1	297.8	177.9	134.0	161.1	828.6	85.6	541.2

4.4.2 Computational Performance Analysis

Table 15 presents the comparative performance among the CF algorithms. The values indicates how many times each CF algorithm is slower than our proposal (in bold).

By comparing our proposal with memory-based CF algorithms it is at least 8 times faster than the fastest memory-based. This difference becomes smaller when compared with model-based CF algorithms, wherein PMF performs 1.2 times slower than the proposed method on Netflix1M. However, the other model-based CF algorithms are away slower.

Therefore, we conclude that our proposal consistently and considerably outperformed the compared state-of-the-art CF algorithms, in the sense of computation performance.



(a) User-based CF in MovieLens1M

(b) Item-based CF in MovieLens1M

(c) User-based CF in Netflix1M

(d) Item-based CF in Netflix1M

Fig. 6 This figure illustrates Accuracy per Time of CF algorithms. Each CF algorithm is represented by a point in the graphs which were divided in four quadrants, in order to better discern algorithm performances. Note that, user/item-based CF with Landmarks (*Landmarks kNN*), PMF and BPMF are located at the first quadrant, which means these perform faster than others and have comparable accuracy. The graphs were generated using 1M ratings data sets.

4.4.3 Compromise Between Accuracy and Runtime

In order to compare CF algorithms from different standpoints, we plot the accuracies of each algorithm for the data sets of 1M ratings by their corresponding runtime. As one may see in Figure 6, x and y axes indicates the accuracy (measured with MAE) and the runtime logarithm (in log-seconds), respectively. Each algorithm is therefore represented by a point in 2D space.

Each graph in Figure 6 has been divided into four quadrants so as to better discern the algorithm performances. The first quadrant is the desired one, since it indicates the lowest values for both MAE and runtime, *i.e.* the best compromise between accuracy and computational performance.

Note that, the proposed algorithm, PMF and BPMF are within the first quadrant in all graphs. Additionally, our proposal has performed faster than any other algorithm, including PMF and BPMF.

Regarding to accuracy, one should observe that model-based CF algorithms have clearly yielded higher accuracy than the other techniques based on similarity. Regularized SVD, Improved Regularized SVD and Bayesian Probabilistic MF have performed similarly, in the sense that, their accuracies and computational performances were comparable. The same might be verified for both Cosine and Pearson (Cosine kNN and Pearson kNN, respectively). This behavior is also observed in Figure 6a and Figure 6c, which correspond to user-based CF, and, analogously, in Figure 6b and Figure 6d for item-based CF.

It is remarkable how our proposal consistently and considerably outperforms CF algorithms in computational performance. Furthermore, it provides rating predictions as accurate as any other memory-based techniques, and may offer an interesting compromise between accuracy and computational performance when compared with model-based CF algorithms.

5 Conclusion

In this paper, we presented a proposal to improve memory-based CF computational performance via rating matrix reduction with landmarks. It consists in representing users by their similarities to few preselected users, namely landmarks. Thus, instead of modeling users by rating vectors and building a user-user similarity matrix, we proposed to locate users by their similarities to landmarks, resulting in a new user-user similarity matrix. Thus, small numbers of landmarks leads to great reduction of the rating matrix. Consequently, it decreases the time spent to compute the posterior user-user similarity matrix.

The proposed method has three parameters that influence the new space representation, and consequently the CF algorithm accuracy and runtime. These are 1)the number of landmarks, 2)the distance measure to compute the user-landmark matrix, and 3)the distance measure to compute user-user similarity matrix. After investigating different parameter settings, we found out that accuracy and runtime increase with the number of landmarks. Besides, all evaluated distance measures (Euclidean, Cosine and Pearson) have yielded similar accuracies.

Another important component of the proposed algorithm is how to select landmarks. There were proposed five different ways of selecting landmarks: *Random*, *Dist. of Ratings*, *Coresets*, *Coresets Random*, and *Popularity*. The most accurate strategy was *Popularity*, while *Random* and *Dist. of Ratings* were the fastest ones.

In order to conduct a fair comparison of the proposed algorithm, we selected eight CF algorithms – both memory-based and model-based approaches – to compare their performance on MovieLens and Netflix databases. There were considered two different cuts of each database (100k and 1M ratings). We have taken MAE as an accuracy measure, and the runtime in seconds as a computational performance measure.

The results have shown that our proposal has consistently outperformed the other algorithms in terms of time consuming. It has also improved CF scalability, once its runtime increased almost linearly with the number of landmarks. Furthermore, one yielded the highest accuracy overall with regards to memory-based CF algorithms.

Concluding, our proposal offers a very simple and efficient manner to reduce the cost of similarity computations for memory-based CF algorithms by conferring great speedup without loss of accuracy.

As future work, a theoretical investigation should be addressed so as to determine the number of landmarks that guarantee accuracy bounds. We want to determine lower bounds for the number of landmarks given an approximation error ϵ. Besides, it also important to investigate how landmarks can be used to improve model-based CF algorithms.

Acknowledgements Our thanks to CNPq/CAPES for funding this research.

References

Abernethy J, Canini K, Langford J, Simma A (2007) Online collaborative filtering. University of California at Berkeley, Tech Rep

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering 17(6):734–749

Babaeian A, Babaee M, Bayestehtashk A, Bandarabadi M (2015) Nonlinear subspace clustering using curvature constrained distances. Pattern Recognition Letters 68:118–125

Beladev M, Shapira B, Rokach L (2015) Recommender systems for product bundling

Bennett J, Lanning S (2007) The netflix prize. In: Proceedings of KDD cup and workshop, vol 2007, p 35

Bobadilla J, Ortega F, Hernando A, Glez-de Rivera G (2013) A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors algorithm. Knowledge-Based Systems 51:27–34

Braida F, Mello CE, Pasinato MB, Zimbrão G (2015) Transforming collaborative filtering into supervised learning. Expert Systems with Applications 42(10):4733–4742

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., pp 43–52

Chen Y, Crawford M, Ghosh J (2006) Improved nonlinear manifold learning for land cover classification via intelligent landmark selection. In: 2006 IEEE International Symposium on Geoscience and Remote Sensing, IEEE, pp 545–548

Chi J, Crawford MM (2013) Selection of landmark points on nonlinear manifolds for spectral unmixing using local homogeneity. IEEE Geoscience and Remote Sensing Letters 10(4):711–715

Chi J, Crawford MM (2014) Active landmark sampling for manifold learning based spectral unmixing. IEEE Geoscience and Remote Sensing Letters 11(11):1881–1885
De Silva V, Tenenbaum JB (2004) Sparse multidimensional scaling using landmark points. Tech. rep., Technical report, Stanford University
Elbadrawy A, Karypis G (2015) User-specific feature-based similarity models for top-n recommendation of new items. ACM Transactions on Intelligent Systems and Technology (TIST) 6(3):33
Feldman D, Faulkner M, Krause A (2011) Scalable training of mixture models via coresets. In: Advances in neural information processing systems, pp 2142–2150
Gao Y, Pan J, Ji G, Yang Z (2012) A novel two-level nearest neighbor classification algorithm using an adaptive distance metric. Knowledge-Based Systems 26:103–110
Harper FM, Konstan JA (2016) The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5(4):19
Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 230–237
Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) 22(1):5–53
Hu X, Yang Z, Jing L (2009) An incremental dimensionality reduction method on discriminant information for pattern classification. Pattern Recognition Letters 30(15):1416–1423
Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 426–434
Koren Y, Bell R (2011) Advances in collaborative filtering. In: Recommender systems handbook, Springer, pp 145–186
Koren Y, Bell R, Volinsky C, et al (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
Lee S, Choi S (2009) Landmark mds ensemble. Pattern Recognition 42(9):2045–2053
Li H, Sun J (2008) Ranking-order case-based reasoning for financial distress prediction. Knowledge-Based Systems 21(8):868–878
Luo X, Xia Y, Zhu Q, Li Y (2013) Boosting the k-nearest-neighborhood based incremental collaborative filtering. Knowledge-Based Systems 53:90–99
Miller BN, Albert I, Lam SK, Konstan JA, Riedl J (2003) Movielens unplugged: experiences with an occasionally connected recommender system. In: Proceedings of the 8th international conference on Intelligent user interfaces, ACM, pp 263–266
Orsenigo C (2014) An improved set covering problem for isomap supervised landmark selection. Pattern Recognition Letters 49:131–137
Pang G, Jin H, Jiang S (2015) Cenknn: a scalable and effective text classifier. Data Mining and Knowledge Discovery 29(3):593–625
Pasinato MB, Mello CE, Zimbrão G (2015) Active learning applied to rating elicitation for incentive purposes. In: European Conference on Information Retrieval, Springer, pp 291–302
Paterek A (2007) Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, vol 2007, pp 5–8
Platt JC (2004) Fast embedding of sparse music similarity graphs. Advances in neural information processing systems 16:571,578
Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. Springer
Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 880–887
Salakhutdinov R, Mnih A (2011) Probabilistic matrix factorization. In: NIPS, vol 20, pp 1–8
Saleh AI, El Desouky AI, Ali SH (2015) Promoting the performance of vertical recommendation systems by applying new classification techniques. Knowledge-Based Systems 75:192–223
Sarwar BM, Karypis G, Konstan JA, Riedl JT (2000) Application of dimensionality reduction in recommender system–a case study. In: IN ACM WEBKDD WORKSHOP
Schwartz B (2004) The paradox of choice. Ecco New York
Shang F, Jiao L, Shi J, Chai J (2011) Robust positive semidefinite l-isomap ensemble. Pattern Recognition Letters 32(4):640–649
Shi H, Yin B, Zhang X, Kang Y, Lei Y (2015) A landmark selection method for l-isomap based on greedy algorithm and its application. In: 2015 54th IEEE Conference on Decision and Control (CDC), IEEE, pp 7371–7376
Shi H, Yin B, Bao Y, Lei Y (2016) A novel landmark point selection method for l-isomap. In: Control and Automation (ICCA), 2016 12th IEEE International Conference on, IEEE, pp 621–625
Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. ACM Computing Surveys (CSUR) 47(1):3
Silva J, Marques J, Lemos J (2005) Selecting landmark points for sparse manifold learning. In: Advances in neural information processing systems, pp 1241–1248
Silva VD, Tenenbaum JB (2002) Global versus local methods in nonlinear dimensionality reduction. In: Advances in neural information processing systems, pp 705–712
Sun W, Halevy A, Benedetto JJ, Czaja W, Liu C, Wu H, Shi B, Li W (2014) Ul-isomap based nonlinear dimensionality reduction for hyperspectral imagery classification. ISPRS Journal of Photogrammetry and Remote Sensing 89:25–36