Expert Systems with Applications 38 (2011) 5101–5109

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities

Deepa Anand *, Kamal K. Bharadwaj

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110 067, India

ARTICLE INFO

Keywords:
Collaborative filtering
Recommender systems
Similarity measures
Sparsity measures

ABSTRACT

Collaborative filtering is a popular recommendation technique, which suggests items to users by exploiting past user-item interactions involving affinities between pairs of users or items. In spite of their huge success they suffer from a range of problems, the most fundamental being that of data sparsity. When the rating matrix is sparse, local similarity measures yield a poor neighborhood set thus affecting the recommendation quality. In such cases global similarity measures can be used to enrich the neighborhood set by considering transitive relationships among users even in the absence of any common experiences. In this work we propose a recommender system framework utilizing both local and global similarities, taking into account not only the overall sparsity in the rating data, but also sparsity at the user-item level. Several schemes are proposed, based on various sparsity measures pertaining to the active user, for the estimation of the parameter α, that allows the variation of the importance given to the global user similarity with regards to local user similarity. Furthermore, we propose an automatic scheme for weighting the various sparsity measures, through evolutionary approach, to obtain a unified measure of sparsity (UMS). In order to take maximum possible advantage of the various sparsity measures relating to an active user, a scheme based on the UMS is suggested for estimating α. Experimental results demonstrate that the proposed estimates of α, markedly, outperform the schemes for which α is kept constant across all predictions (fixed-α schemes), on accuracy of predicted ratings.

1. Introduction

The explosive growth of the web has led to the problem of 'information overload'-the overwhelming plethora of choices and options available to a user, often varying in quality. The need for a solution to this abundance of information and the drive to bridge the gap between the vendor and customer in e-commerce, has led to popularity of Web personalization. Recommender systems are the most notable application of Web Personalization. Recommender systems are personalization tools which enable users to be presented information suiting his interests, which are novel, serendipitous and relevant, without being explicitly asked for, thus supporting "discovery" rather than "search". Recommender systems have become ubiquitous, with their presence everywhere from recommending books, CDs (Amazon.com, Linden, Smith, & York, 2003), music [last.fm(www.last.fm), movies [MovieLens (www.MovieLens.umn.edu)] to recommending high risk products such as mutual funds and vacations.

Among the different types of recommender systems, collaborative filtering is the most widely used and effective recommendation technique. Collaborative filtering (Goldberg, Nichols, Oki, & Terry, 1992) is the automation of "word of mouth" (Shardanand & Maes, 1995), where opinions gleaned from people, who share similar tastes as the active user, is used in the decision making process. It is based on the assumption that users who have agreed in the past tend to agree in the future. The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects being recommended, and work well for complex objects such as music and movies where variations in taste are responsible for much of the variation in preferences (Burke, 2002).

Collaborative filtering algorithms can be classified as memory-based or model-based algorithms (Breese, Heckerman, & Kadie, 1998). Memory-based algorithms (Candillier, Meyer, & Fessant, 2008; Russell & Yoon, 2008) are heuristics based algorithms, which utilize the entire rating history to arrive at predictions. These include the commonly implemented class of user-based and item-based CF methods. Model-based recommender systems (Al-Shamri et al., 2007; Bell, Koren, & Volinsky, 2007) build a user-model in an off-line learning phase and then apply this model on-line for recommendation. The accuracy offered by memory-based RS, since they examine the entire rating database for prediction, and their simplicity, lend to their popularity.

* Corresponding author. Tel.: +91 11 9810253296; fax: +91 11 26717528.
E-mail address: deepanand209@gmail.com (D. Anand).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.09.141

Sparsity of ratings data is one of the primary challenges to CF systems. The large number of users and items in any e-commerce recommendation system restricts the number of items rated by any user to a small subset of the total number of items. Effective prediction of ratings from a small number of examples is important (Adomavicius & Tuzhilin, 2005). Local similarity measures such as the Perason similarity measure (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994) and Cosine similarity measure (Sarwar, Karypis, Konstan, & Riedl, 2001), base the similarity computation step between two users, on the set of common items rated by both users. In the presence of sparseness in the data, many users do not share common items, thus rendering CF useless. Even if users are deemed similar to other users, the similarity might be based on a small set of common experiences. The user must rate a sufficient number of items to get a reasonable profile overlap in order for the system to establish his neighbors. The poor quality of recommendations at high sparsity levels is established (Gřcar, Mladenĭc, Fortuna, & Grobelnik, 2006).

Over the years, a variety of solutions to the data sparsity problem, have been proposed. (Luo, Niu, Shen, & Ullrich, 2008) propose a novel approach of combining predictions from locally similar neighbors, i.e. users who have co-rated items, and globally similar neighbors. Two users are considered to be globally similar if they are connected through locally similar neighbors. It was established experimentally that when the data is very sparse, globally similar neighbors provide better prediction, whereas in case of low sparsity, predictions from local neighbors are superior. In (Luo et al., 2008) a parameter α was utilized to balance contributions from global neighborhood over local neighborhood. The value taken by the parameter α was static, i.e. the importance given to local/global predictions remained the same across all predictions. Also the significance of local or global predictions had to be manually set to reflect the actual contribution of local/global neighborhood to accuracy. A promising extension to the above scheme would be to automate the α estimation by considering not only the overall sparsity but also the sparsity at the user/item level. Intuitively a user rating a very small subset of items or a user with unusual tastes who has rated a reasonably large number of items, might have a poor quality local neighborhood, even if the ratings matrix is dense.

In this work we propose various α estimation schemes based on the overall sparsity in the ratings data as well as sparsity based on the active user and the item whose rating needs to be predicted. The dependency of the α value on the user and item ensures that the local and global neighbors are weighed differently for every user and for every prediction. The experimental results support our ideas and demonstrate that the proposed methods are superior to the fixed α scheme (Luo et al., 2008).

The remainder of the paper is organized as follows: Section 2 provides a literature overview into the variety of techniques which have been employed to solve the data sparsity problem and an overview of the various existing local and global similarity measures. Several new α estimation based on the user and item are introduced in Section 3 while Section 4 suggests an automatic weighting scheme of combining the various sparsity measures to arrive at a unified measure of sparsity(UMS) to integrate local and global predictions. Section 5 presents an experimental evaluation of the proposed schemes and compares them with the schemes proposed in (Luo et al., 2008). Finally, Section 6 winds up with the conclusions and points out some directions for future research.

2. Literature review

In spite of more than a decade long research on recommender systems, sparsity remains the major hurdle. We first take a look at the various solutions to the sparsity problems that have been proposed in literature and then examine the various local and global similarity measures.

2.1. Sparsity

Several model-based techniques based on dimensionality reduction techniques such as SVD (Billsus & Pazzani, 1998; Rosenstein & Lochbaum, 2000) and Latent Semantic Indexing (Sarwar, Karypis, Konstan, & Riedl, 2000) have been applied in order to alleviate the sparsity problem. (Suryavanshi, Shiri, & Mudur, 2005) propose a two level model-based technique, employing relational fuzzy subtractive clustering and association rules mining at the first and second levels respectively, to produce recommendations which are less prone to effects of sparsity. To aid computation of comparability between users preferences, (Al-Shamri & Bharadwaj, 2008) proposed building a concise and representative hybrid user model, which enables quantification of user closeness irrespective of any shared experiences, and thus aid predictions which are accurate in the presence of sparsity.

Hybridization of collaborative filtering with other filtering mechanisms is another way to help solve the problem of data sparsity. Other data sources such as the item contents (Melville, Mooney, & Nagarajan, 2002; Popescul, Ungar, Pennock, & Lawrence, 2001) have been combined with collaborative filtering to enhance existing user data, and thus provide personalized suggestions.

Rules generated through Apriori algorithm (O'Sullivan, Wilson, & Smyth, 2002) have been used to address the sparsity problem by enabling similarity computation between pairs of users having no common ratings. A recursive algorithm (Zhang & Pu, 2007) is proposed, which allows nearest-neighbor users to join the prediction process even if they have not rated the given item, by recursively predicting the scores for neighbors who have not rated the item, and integrating it into the prediction for the active user.

Trustworthiness of users is an important factor, worth consideration, when assessing the neighbors who contribute to the prediction. Trust as a metric can replace or complement similarity metrics (Al-Shamri & Bharadwaj, 2008; Massa & Avesani, 2004) in order to enhance the neighborhood set, by adding contributions from users who may not have commonly rated items with the active user, but may be trustworthy.

Many criterion for similarity which allow computation of user likeness for users who may not share any common experiences, have been proposed. A similarity metric, Generalized Cosine Max, which does not need ratings of common items by users, is presented in Anand, Kearney, and Shapcott, 2007. This similarity measure which uses item similarity within the calculation of user similarity is shown to work well when the rating data is sparse. (Aggarwal, Wolf, Wu, & Yu, 1999) proposed a graph-theoretic approach to collaborative filtering, that enables users having no direct resemblance with the active user, to participate in the prediction.

2.2. Similarity measures

One of the important steps in collaborative filtering is the neighborhood formation step. Several similarity measures have been proposed in literature in order to identify users with similar inclinations. However most of the popular measures, base the similarity computation, on the local information available i.e. on ratings common to both users. Global similarity measures can complement local similarity measures in order to improve accuracy and coverage in sparse-data scenarios. This section presents an overview of the different local and global similarity measures proposed in literature and concludes with a framework for combining both in order to exploit the advantages offered by both approaches.

2.2.1. Local similarity measures

Similarity measurement between users plays an elemental role in both user-based and item-based algorithms. The most commonly used measurement techniques of the similarity between users are the Pearson correlation (Resnick et al., 1994) and Cosine similarity (Breese et al., 1998) algorithms. Typically the similarity computation is based on finding the similarity between the rating vectors, containing ratings of items rated in common by both users.

Pearson correlation coefficient defines similarity between users x and y as:

$$sim(x,y) = \frac{\sum_{i\in S_{xy}}(r_{x,i}-\bar{r}_x)(r_{y,i}-\bar{r}_y)}{\sqrt{\sum_{i\in S_{xy}}(r_{x,i}-\bar{r}_x)^2\sum_{i\in S_{xy}}(r_{y,i}-r_y)^2}} \quad (1)$$

where S_{xy} is the set of items which users x and y have co-rated and $\bar{r}_x$ is the mean rating for user x. Another popular similarity metric used for CF, the Cosine similarity is defined as:

$$sim(x,y) = \frac{\sum_{i\in S_{xy}} r_{x,i}r_{y,i}}{\sqrt{\sum_{i\in S_{xy}} r_{x,i}^2\sum_{i\in S_{xy}} r_{y,i}^2}} \quad (2)$$

When the preference information is binary i.e. like or do not like an item, then the Jaccard coefficient is used to measure user similarity. The Jaccard coefficient is defined as:

$$sim(x,y) = \frac{|R_x \cap R_y|}{|R_x \cup R_y|} \quad (3)$$

where R_x is the set of elements liked by user x.

Lathia, Hailes, and Capra (2007) use concordance based methods to measure the association between a pair of users. The closeness among a pair of users is estimated based on the number of concordant, discordant and tied pairs of common ratings. Candillier et al. (2008) introduce several weighted similarity measures for user-based and item-based collaborative filtering. The method proposed, uses Jaccard similarity as a weighting scheme and combines it with other similarity measures such as Pearson correlation coefficient to emphasize similarity of users who share appreciation on several items with the active users.

Luo et al. (2008) exploited the fact that two users whose opinion about an item is contrasting to the average attitude about the item, but comparable to each other, can be deemed to be more similar, than users who go along with the popular opinion about the item. They stressed on the discriminative power of a rating by measuring its "surprisal", which carries information about how distinct a particular rating from the average rating for the item. The rating for an item was assumed to follow the Laplacian distribution and the surprisal of a rating was computed as:

$$I(rp,i) = -\ln(f(r=rp,i|\mu^A i, b^A i)) = \ln(2b^A i) + \frac{|rp,i-\mu^A i|}{b^A i} \quad (4)$$

where $r_{p,i}$ is the rating of item 'i' by user 'p', b_i^A and μ_i^A are the maximum likelihood estimates of location and scale parameters, respectively. Given the surprisal of all ratings, the user p's surprisal vector, S_p is defined as

$$\begin{aligned} S_p &= [s_{p,1},\ldots,s_{p,N}]^T \\ &= [\mathrm{sgn}(r_{p,1}-\hat{\mu}_1)*I(r_{p,1}),\ldots,\mathrm{sgn}(r_{p,N}-\hat{\mu}_N)*I(r_{p,N})]^T, \quad p \\ &= 1,\ldots,M \end{aligned} \quad (5)$$

where sgn $(r_{p,I}-\mu_i^A)$ represents whether the preference of user p is positive or negative with respect to the average attitude for the item. The similarity calculation between two users is the vector space similarity between the users' surprisal vectors. To discount the high similarity between two users based on a small set of co-rated items, the following significance weighting scheme is proposed:

$$sim'_L = \frac{Min(|Iu_p \cap Iu_q|,\gamma)}{\gamma} sim_L(u_p,u_q) \quad (6)$$

where $|Iup \cap Iuq|$ is the number of items co-rated by users p and q. 'γ' is the minimum number of common items that needs to be rated in common by both users. This method is termed surprisal-based vector similarity with significance weighting (SVSS).

2.2.2. Global similarity measures

Global similarity measures enable similarity computation between pairs of users who may not share any common encounters. Several algorithms utilizing predictions from global neighbors have been proposed in literature. An approach to capturing transitive similarity between users, discovery hidden similarity (DHS), is proposed in Lee, Yang, and Park (2004). The similarities are captured not only through movie ratings but also through user similarities. A new technique of computing user similarities using a Markov-chain model of a random walk is introduced in Fouss, Pirotte, Renders, and Saerens (2007). Utilizing a bipartite graph with users and items as nodes, where users are connected to items they have experienced, the quantities, "average commute time" and "average first passage time" are used as similarity measure between two users. These quantities have the nice property of increasing when the number of paths connecting nodes increases and when the lengths of the paths decreases. Other item-oriented approaches using random walk based approach (Gori & Pucci, 2007; Yıldırım & Krishnamoorthy, 2008), infer correlations between items using item graphs, to recommend items to the active user. A new collaborative filtering approach (Desrosiers & Karypis, 2008), computes global similarities between pairs of items and users, based on the solution to system of linear equations relating user similarities to item similarities. The new approach helps make accurate predictions in the presence of sparsity and also takes into account content-based similarities between users.

2.2.3. Framework combining local and global similarities

Luo et al. (2008) utilized the maximin distance between users in a user graph as the global similarity between two users. Two users are connected by an edge if they have positive local similarity, where local similarity was the SVSS similarity as discussed in Section 2.2.1. A novel technique of combining the predictions from locally and globally similar neighbors (Luo et al., 2008), allowed the combined framework to attain quality predictions in both sparse as well as dense data scenarios. The final prediction was a linear combination of predictions from both sets of neighbors and was defined as follows:

$$predR = (1-\alpha)*predR_l + \alpha*predR_G \quad (7)$$

where $predR_L$ is the predicted rating obtained through the local neighbors and $predR_G$ is the predicted rating obtained through the global neighbors. α is the weight given to prediction from the global neighborhood set.

Luo et al. (2008) empirically established the dependence of α, on the sparsity of the rating data. It was shown that with the increase in the sparsity of the ratings matrix, an increase in α led to more accurate predictions. This scheme of combining local and global similarities referred to as LS & GS in Luo et al. (2008) were fixed-αschemes, since weightage of predictions from local and global neighbors was fixed and needed to be manually set depending on the data sparsity. The superiority of the LS & GS over several prediction algorithms such as effective missing data prediction (EMDP), similarity fusion Algorithm (SF), user based filtering using Pearson correlation (UPCC) and item based filtering using Pearson correlation (IPCC), was established experimentally by setting α to 0.5. This scheme will henceforth be referred to as the fixed-α scheme 1. An alternative basis for comparison would be a fixed-α,

with α set to a value which results in the least MAE. This scheme of estimating α would hereafter be referred to as the fixed-α scheme 2.

3. Novel schemes for estimating parameter α

A framework combining predictions from local and global neighbors was discussed in Section 2. The parameter α serves to adjust the weight that we give to global neighbors with regards to the weight that we give to the local neighbors. When the ratings matrix is dense then generally the local neighborhood set is rich enough to enable prediction for the active user, in which case the predictions from local neighborhood should be weighed more. However, when the ratings matrix is sparse, the meager local neighborhood set generated may lead to low quality recommendations, and therefore it needs to be enriched by the globally similar neighbors and for better predictions.

In this section we propose several sparsity measures which enable the global neighbors to be weighed according to the active user and the item whose vote needs to be predicted. To the best of our knowledge there has been no previous attempt at quantifying the various facets of sparseness in data and to use them in the prediction process. The "sparsity problem" in collaborative filtering refers to inability to find a sufficient quantity of good quality neighbors to aid in the prediction process due to insufficient overlap of ratings between the active user and his neighbors. This can happen when the ratings matrix is sparse, or the number of users participating is not large. Even when the data is dense enough to allow quality predictions for most users, some users may not have rated enough items or may have rated items not rated by most people, with the result that such users get poor quality predictions. Users whose local neighborhood set is sparse can thus be aided by using predictions from the global neighborhood set. Thus intuitively the value of α should depend on the user and the item whose rating is to be predicted. The proposed work introduces several estimates for α, which captures the various aspects of sparseness in the data.

3.1. Individual sparsity measures

In this section we introduce several sparsity measures namely, OS (overall sparsity measure), USS (user specific sparsity measure), LGR (local global ratio), UIS1 (user item based sparsity measure 1) and UIS2 (user item based sparsity measure 2).

3.1.1. Overall sparsity measure(OS)

The overall sparsity measure captures the level of sparsity in the entire rating matrix. This sparsity measure is universal i.e. the α computed is fixed for all users. The overall sparsity measure is defined as:

$$\text{Overall sparsity measure} = 1 - \frac{nR}{nUsers * nItems} \tag{8}$$

where nR is the total number of ratings which exist, $nUsers$ is the total number of users in the system and $nItems$ is the total number of items.

3.1.2. User specific sparsity measure(USS)

The user specific sparsity measure is based on the intuition that users who have rated very few items are less likely to get reliable local neighbors and thus need to depend more on global neighborhood set. The USS is user-dependent, but remains the same across all items whose rating needs to be predicted.

The USS for user $u$and item i is defined as:

$$\text{User specific sparsity measure} = 1 - \frac{n_u}{\max_{u \in U}(n_u)} \tag{9}$$

where n_u is the number of items rated by user u.

3.1.3. User and item specific sparsity measures

Sometimes the superiority of global predictions over local ones also depend on the type of items rated by the user. For example the local neighbors for a user, with eclectic tastes who has rated several items which have not been experienced by a large majority of users, may be scarce. When the local neighborhood is meager, global neighbors can contribute to improvement in accuracy of predictions. This means that the sparsity measure should take into account not only the active user, but also the item for which the rating needs to be predicted. Three sparsity measures, which capture sparsity at user-item level, are introduced below.

- Local global ratio (LGR)
  One simple measure is to find the ratio of number of local neighbors to the number of global neighbors who have rated the item. It is to be noted that the number of global neighbors always exceeds or is equal to the number of local neighbors. This is due to the fact that $sim_G(x,y) \geqslant sim_L(x,y)$. The LGR for a user u and item i is defined as:

  $$\text{LGR}(u,i) = 1 - \frac{|L_{u,i}|}{|G_{u,i}|} \tag{10}$$

  where $L_{u,i}$ is the set of local neighbors of user $\boldsymbol{u}$ who have rated item $\boldsymbol{i}$, $G_{u,i}$ is the set of global neighbors of user $\boldsymbol{u}$ who have rated item $\boldsymbol{i}$.
- User-item specific sparsity measure1 (UIS1) The UIS1 measure bases the sparsity measurement on the ratio of number of local neighbors who have rated a particular item to the total number of people who have rated the item i.e. the UIS1 for a user u and item i is defined as:

  $$\text{UIS1}(u,i) = 1 - \frac{|L_{u,i}|}{|N_i|} \tag{11}$$

  where N_i is the set of users who have rated item $\boldsymbol{i}$.
- User-item specific sparsity measure2 (UIS2) The UIS2 measure computation is founded on the ratio of number of local neighbors who have rated a particular item to the total number of users in the local neighborhood set i.e. the UIS2 for a user u and item i is defined as:

  $$\text{UIS2}(u,i) = 1 - \frac{|L_{u,i}|}{|L_u|} \tag{12}$$

  where L_u is the set of local neighbors for user $\boldsymbol{u}$.

3.2. Unified measure of sparsity(UMS)–GA approach

The various measures of sparsity, described in Section 3, encapsulate the diverse factors that come into play, when deciding the extent to which the local and global predictions should influence the final prediction. The performance of each sparsity measure is dataset dependent and hence the quality offered, varies across several datasets. The sparsity measure, which best reflects the balance between local and global predictions may differ from user to user and also may evolve over time. The idea of unifying the various sparsity measures to derive a single weight which works best for the active user, is propitious. The sparsity measures can be coalesced into a unified sparsity measure(UMS) by considering a weighted average of all the sparsity measures, where a sparsity measure more representative of the user's preference for local/global predictions, has a higher value.

$$\mathrm{UMS} = w_1\mathrm{OS} + w_2\mathrm{USS} + w_3\mathrm{LGR} + w_4\mathrm{UIS1} + w_5\mathrm{UIS2} \quad (13)$$

where $\sum_{i=1}^{5} w_i = 1$.

Genetic algorithms can be effectively employed to learn weights of each of the five proposed α-estimates for every user, in an offline learning process. Such a set of weights can then be used to compute the value of α during prediction for the active user. The next subsection discusses the genetic operators and the fitness function utilized to generate the set of weights.

3.2.1. Automatic weighting of sparsity measures using real-valued genetic algorithm

Genetic algorithms base their operation on the Darwanian principle of "survival of the fittest" and utilize artificial evolution to get enhanced solutions with each iteration. The GA process starts with a population of candidate solutions known as chromosome or genotype. Each chromosome in the population has an associated fitness and these scores are used in a competition to determine which chromosomes are used to form new ones (De Jong, 1988). As the population evolves from one generation of chromosomes to a new population by using a kind of natural selection together with the genetics inspired operations of crossover, mutation and inversion (Mitchell, 1998), the solutions tend to approach the global optimum regardless of the topography of the fitness landscape.

The traditional chromosome representation technique of using bit strings, suffer from slow convergence due to the large chromosome structure, especially in optimization problems involving several parameters Practical experience favors utilization of the most natural representation of solutions rather than enforce a binary vector representation for many problems (Bäck, Fogel, & Michalewicz, 1997). Since the weights for each of the sparsity measures are real numbers, a floating point representation of chromosomes is most suited for our purpose. Thus each chromosome consists of a vector of five real numbers representing weights corresponding to five different sparsity measures. The population then evolves over several generations to arrive at the weight vector which is the fittest.

3.2.2. Genetic operators

Genetic operators drive the search process in GA. Crossover and mutation are the two basic genetic operators. While crossover allows creation of two new individuals by allowing two parent chromosomes to exchange meaningful information, mutation is used to maintain the genetic diversity of the population by introducing a completely new member into the population. Crossover and mutation operators for real-valued GA was introduced by Michalewicz (1992). Let a chromosome be represented by n real numbers(genes) where the ith gene lying in the range $[a_i, b_i]$. Given two parent chromosomes $X = \langle x1, x2, \ldots, xn\rangle$ and $Y = \langle y1, y2, \ldots, yn\rangle$, some of the different types of crossover and mutation for real-valued GA, are discussed below (Houck, Joines, & Kay, 1995; Michalewicz, 1992).

Uniform mutation

A gene i is randomly selected and its value is modified to a uniform random number between a_i and b_i:

$$x'_j = \begin{cases} U(a_i, b_i), & \text{if } i = j \\ x_j, & \text{otherwise} \end{cases} \quad (14)$$

Arithmetic crossover

Arithmetic crossover generates two new complementary offspring as a linear combination of the parent chromosomes.

$$\begin{aligned} X' &= \tau X + (1-\tau)Y \\ Y' &= (1-\tau)X + \tau Y \end{aligned} \quad (15)$$

where $\tau = U(0,1)$ and X' and Y' are the generated offspring.

Uniform mutation and arithmetic crossover operators are used as the basic mutation and crossover operators respectively, in our experiments.

3.2.3. Fitness function

The fitness function is the factor which drives the GA process towards convergence to the optimal solution. Chromosomes which are more optimal, are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will be even better. Choosing a fitness function for finding the set of weights for the sparsity measures, requires that the weight should be evaluated via the error that the weight produces while predicting each rating in the user's training set. An individual whose error is lesser is a fitter individual. The weights are learnt for the active user by trying to predict each rating in the user's training set using the particular weight to combine the five sparsity measures, and determining how well the unified sparsity measure balances local and global predictions. The fitness function is the average error among all such predictions for the active user.

$$\text{fitness} = \frac{1}{|T|}\sum_{i \in T} |r_{a,i} - pr_{a,i}| \quad (16)$$

where T is the set of all ratings in the training set of the user, $r_{a,i}$ is the actual rating for item i by the active user, and $pr_{a,I}$ is the predicted score computed using the combined framework.. The weights to be applied for each of the measures of sparsity($w1, w2, \ldots w5$) to arrive at a unified sparsity measures, (Eq. (13)), can be computed by employing real valued GA as described in this section.

4. Proposed recommender system framework

The proposed estimates of α can be utilized to integrate predictions from locally and globally similar users and arrive at the final predicted vote for the active user. The main steps of the proposed recommender system framework are given below:

Step 1: Compute local user similarities.
Compute the SVSS similarity between all pairs of users, using the formula (6). Let $sim_L(x,y)$ refer to the local similarity between users x and y as computed in this step.

Step 2: Compute the global user similarities.
Compute the global similarities between all pairs of users, by first constructing the user graph based on local similarities, and then finding the maximin distance between users, as described in Section 2. Let $sim_G(x,y)$ refer to the global similarity between users x and y as computed in this step.

Step 3: Obtain predicted ratings using local and global neighbors.
The predicted rating for an item $\boldsymbol{i}$ for active user $\boldsymbol{u}$ is based on Resnick's prediction formula (Resnick et al., 1994).

$$pr_{i,k} = \bar{r}_i + \frac{\sum_{j \in N(i)} sim_{i,j} * (r_{j,k} - \bar{r}_j)}{\sum_{j \in N(i)} |sim_{i,j}|} \quad (17)$$

where $\bar{r}_i$ is the mean rating for user $\boldsymbol{i}$, $sim_{i,j}$ is the similarity between users $\boldsymbol{i}$ and $\boldsymbol{j}$ and $N(i)$ is the neighborhood of user $\boldsymbol{i}$.
The above formula can be used to arrive at predictions from local neighborhood by setting sim_{ij} to $sim_L(i,j)$ and $N(i)$ to the local neighborhood for user $\boldsymbol{i}$. A similar method can be adopted to arrive at predictions from global neighborhood. Let $pr^L_{i,k}$ and $pr^G_{i,k}$ be the predicted ratings using local and global similarities respectively.

Step 4 : Merge the local and global ratings.

The predicted ratings from local and global neighborhoods are combined using the following formula:

$$pr_{i,k} = (1-\alpha) * pr^{L}_{i,k} + \alpha * pr^{G}_{i,k} \quad (18)$$

to get the final prediction for the active user, where α can be set to any of the sparsity measures as given below:

- Overall sparsity measure (OS) (Eq. (8)).
- User-specific sparsity measure (USS) (Eq. (9)).
- User-item specific sparsity measures (LGR, UIS1, UIS2) (Eqs. (10)–(12)).
- Unified measure of sparsity (UMS) (Eq. (13)).

5. Experiments and results

In order to evaluate the performance of the system utilizing different α estimates, based on various proposed sparsity measures, several experiments are conducted on the vastly popular MovieLens(http://www.MovieLens.umn.edu) and Jester datasets. The aim of these experiments is to show the superior performance of the proposed techniques over the fixed-α schemes presented in (Luo et al., 2008).

5.1. Experimental setup

The MovieLens dataset consists of 100,000 ratings provided by 943 users on 1682 movies. The ratings scale is in the range 1–5 with 1-"bad" to 5-"excellent". The ratings are discrete. Each user in the dataset has rated at least 20 movies. The Jester dataset provides 4.1 million ratings by 73,421 users on 100 jokes. The ratings are continuous and lie in the range −10 to 10. MovieLens and Jester are different from each other, since the overall sparsity of MovieLens is quite high whereas the Jester dataset is quite dense. Moreover the number of items in MovieLens is much higher than the number of users, whereas in Jester the users far exceed the number of jokes. Testing the various prediction techniques on these two datasets allow for comparison under diverse data environments.

For all the experiments 300 users were randomly selected from both the MovieLens and Jester datasets, the respective datasets being denoted as ML300 and Jester300 respectively. The datasets were then divided into 200 training users and 100 active users. The ratings from the training users are utilized in order to predict ratings for the test users. The number of ratings provided by the active user is varied as 10, 15, 20 and 25 giving rise to four different configurations Given10, Given15, Given20 and Given25 respectively. The training ratings are used as explicit ratings available, thus aiding neighborhood construction as well as guiding the learning process of the GA, whereas the test ratings are considered as ratings which are unavailable and hence need to be predicted. The ratings from the Jester dataset, were discretized by rounding the rating value to the nearest integer.

To compare the prediction quality of the proposed methods, we employ two accuracy metrics namely Mean Absolute Error(MAE) and Root Mean Squared Error(RMSE). MAE measures the average absolute deviation of the predicted rating of an item from the actual rating for the item.

The MAE(u_i) for the active user u_i is as follows:

$$\mathrm{MAE}(u_i) = \frac{1}{|S_i|}\sum_{k=1}^{|S_i|} |pr_{i,k} - r_{i,k}| \quad (19)$$

where $|S_i|$ is the cardinality of the test ratings set of user u_i.

The total MAE over all the active users, N_T can be computed as:

$$\mathrm{MAE} = \frac{1}{N_T}\sum_{i=1}^{N_T} \mathrm{MAE}(u_i) \quad (20)$$

A smaller value of MAE signifies better prediction quality. A related accuracy metric is the RMSE which squares the error before summing them. The RMSE(u_i) for active user $u_i$can be defined as:

$$\mathrm{RMSE}(u_i) = \sqrt{\frac{1}{|S_i|}\sum_{k=1}^{|S_i|} (pr_{i,k} - r_{i,k})^2} \quad (21)$$

The RMSE over all active users is the average of RMSE of individual active users.

$$\mathrm{RMSE} = \frac{1}{N_T}\sum_{i=1}^{N_T} \mathrm{RMSE}(u_i) \quad (22)$$

Whereas the MAE weighs the individual differences equally, RMSE gives relatively high weight to large errors. It is useful when large errors are particularly undesirable. It has been demonstrated in a recent study that a small difference in RMSE can lead to a very significant improvement when ordering items by their predicted preferences (Koren, 2008).

It is to be noted that for any Given x (x = 10, 15, 20, 25) configurations, only users having more than x ratings contribute to the error (MAE, RMSE) computation.We run three experiments to compare the prediction methodologies under different data settings.

5.2. Experiment 1

In order to demonstrate the effectiveness of the various proposed α-estimation schemes, we conduct experiments using both datasets (ML300 and Jester300) to compare the performance of the proposed schemes with both the fixed-α schemes, under the various aforementioned configurations. The parameter γ in all experiments was set to 30. The number of nearest local and global neighbors used for prediction (K) was restricted to 30.

The prediction accuracy of the following schemes are compared:

Fixed-α schemes.

- Fixed-α scheme 1 (Luo et al., 2008).
- Fixed-α scheme 2 (Luo et al., 2008).

Proposed schemes.

- Overall sparsity (OS).
- Local global ratio (LGR).
- User-item specific sparsity measure (UIS1).
- User-item specific sparsity measure (UIS2).
- User specific sparsity measure (USS).

Tables 1 and 2 demonstrate the results of applying the various weighting schemes to ML300 and Jester 300 datasets.The results illustrate the improved prediction quality of the α-estimation schemes based on user and item sparsity measures. All the proposed schemes when used for weighting local and global predictions, outperform the fixed-α schemes under all configurations for both datasets, thus highlighting the importance of considering sparsity both at the user and item level, in order to weigh the predictions from local and global neighbors.

For the ML300 dataset, the LGR scheme performs best, with the lowest MAE among all the proposed schemes under all configurations, whereas UIS2 gives the least RMSE among all schemes for the same dataset. The measures OS and USS consistently perform worse than the other proposed schemes, but offer better prediction accuracy than the fixed-α schemes, both in terms of MAE and RMSE.

For the Jester300 dataset, the OS scheme performs the best in almost all configurations with the lowest MAE and RMSE, except for the Given15 configuration when UIS1 performs best, giving the least MAE. The USS measure performs the worst while still being better than the fixed-α schemes.

Table 1
MAE and RMSE comparison of proposed weighing schemes wit the fixed-α and best-α schemes for ML300.

Configurations		Fixed-α scheme 1	Fixed-α scheme 2	LGR	UIS1	UIS2	OS	USS	UMS
Given10	MAE	0.852652	0.846077	0.832761	0.833769	0.834540	0.837084	0.838417	**0.829230**
	RMSE	1.047549	1.046703	1.041914	1.041470	1.040415	1.042390	1.043752	**1.036936**
Given15	MAE	0.834998	0.828519	0.799258	0.799396	0.799260	0.801574	0.802726	**0.796225**
	RMSE	1.024083	1.020234	0.996773	0.995680	0.993368	0.995504	0.996690	**0.991263**
Given20	MAE	0.832258	0.825387	0.795202	0.796822	0.795784	0.800076	0.800990	**0.793316**
	RMSE	1.016667	1.014926	0.994329	0.994564	0.991338	0.995320	0.996373	**0.990454**
Given25	MAE	0.823745	0.817562	0.791298	0.793779	0.791601	0.796753	0.797741	**0.788414**
	RMSE	1.011413	1.010889	0.990690	0.991102	0.987201	0.991261	0.992035	**0.985489**

Table 2
MAE and RMSE comparison of proposed weighing schemes wit the fixed-α and best-α schemes for Jester300.

Configurations		Fixed-α scheme 1	Fixed-α scheme 2	LGR	UIS1	UIS2	OS	USS	UMS
Given10	MAE	4.218933	4.071382	3.779358	3.780444	3.779666	3.773205	3.813883	**3.769652**
	RMSE	4.938895	4.802815	4.617893	4.629019	4.623246	4.616324	4.678356	**4.612176**
Given15	MAE	4.203137	4.092861	3.808832	3.808395	3.808405	3.808868	3.832313	**3.806232**
	RMSE	4.911645	4.820675	4.593892	4.604040	4.593890	4.591937	4.644166	**4.590631**
Given20	MAE	4.235337	4.084560	3.765855	3.766902	3.772736	3.761516	3.783759	**3.759980**
	RMSE	4.923953	4.794233	4.510134	4.522585	4.519029	4.508156	4.557284	**4.507524**
Given25	MAE	4.161225	4.004565	3.689722	3.692673	3.691680	3.688199	3.705325	**3.686564**
	RMSE	4.854251	4.721695	4.429332	4.438499	4.429764	4.429282	4.461536	**4.428535**

5.3. Experiment 2

In this experiment the UMS approach is compared with OS, LGR, UIS1, UIS2, USS as well as with the fixed-α schemes. A real valued GA is used to evolve the weights representing the importance of each sparsity measure for each user, as described in Section 4. The fitness of each individual is the average MAE across all predictions made by using the particular weight. Consequently the individual having the least fitness is deemed to be the fittest individual. The best weight so evolved, is used to as weight to balance local and global predictions.

The initial population size is 30, with 30 new individuals being produced in each generation. An elitist approach is followed wherein the best 30 individuals are selected to be retained in the next generation. The maximum number of iterations is fixed at 500. The GA ends when the best fitness among all individuals in the population, remains unchanged for 20 generations. Roulette-wheel selection is employed to select individuals for the next generation, with the likelihood of getting selected for each individual being inversely proportional to its fitness (MAE). Arithmetic crossover is used in order to produce 28 new individuals. The final two individuals are created using uniform mutation. The GA begins with random genotypes which are real numbers in the range 0 and 1. Each genotype is normalized, by dividing the weight of an α-estimate by the sum of all weights assigned to each α-estimate, so that all the weights add up to unity.

Once weights are evolved for each user, these weights are used to combine the sparsity measures to provide an α, to balance the local and global predictions and arrive at a final prediction. The predicted ratings produced by the UMS method is compared to the ones produced by the proposed sparsity measures and the fixed-α schemes, as shown in Tables 1 and 2. The results clearly demonstrate that the UMS method outperforms all α-estimation schemes discussed above, under all configurations, both in terms of MAE as well as RMSE. For each user the sparsity measure giving best quality predictions may be different, depending on the type of the dataset. UMS is able to learn the significance of each sparsity measure for each user, by examining the votes for items already rated by the user and hence is able to offer better quality predictions.

5.4. Experiment 3

The impact of various sparsity levels on performance of OS, LGR, UIS1, UIS2, USS and UMS is observed in this experiment. Jester dataset was chosen for this experiment, since it is dense and hence offers scope for constructing datasets with greater variation in the sparsity levels. Different levels of sparsity were achieved by retaining 100%, 95%, 90% and so on upto 10% of the total number of training users' ratings, and discarding the rest of the ratings. This results in 19 datasets with gradually escalating sparsity levels. The Given20 configuration was used for the comparison among the various schemes, for all the datasets generated. The other parameters such as the number of nearest neighbors and γ are set to the values as in the previous two experiments.

The performance comparisons of different schemes are illustrated in Fig. 1. where MAE under the various schemes, are plotted

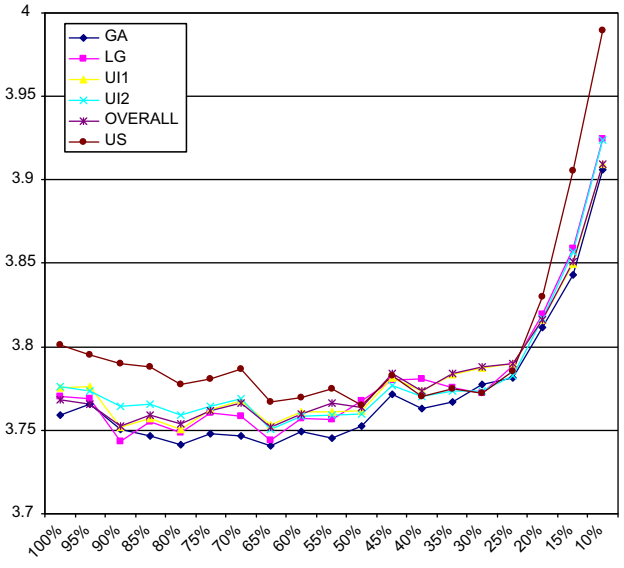

Fig. 1. Performance of UMS, LGR, UIS1, UIS2, OS and USS under different levels of sparsity.

for each dataset. The results show that for all schemes, the average error increases with the increase in sparsity. This is expected since, higher sparsity leads to poor neighborhood set and hence reduced prediction quality. As is clearly seen different sparsity measures give best results for the different datasets. The performance of the weighing sparsity measures, thus is highly dataset dependent. The UMS scheme, outperforms the other weighing schemes for almost all datasets, except the cases when 90% and 30% of the training users' ratings are retained. Another point to be noted is that the improvement that UMS offers over the other weighing schemes is more significant at lower sparsity levels. At higher sparsity levels, the difference in average error, between UMS and the best performing scheme among all sparsity measures, reduces. It may be because, with higher sparsity the GA has a limited ratings set to learn from. It may also be noted that, when the dataset is dense, the LGR scheme performs best in most cases (90% to 55%), but at higher sparsity levels, the relative performance of the method deteriorates, in comparison to the other schemes. The USS scheme almost always performs the worst with the only exception being the case when 30% of the original ratings set, is retained.

6. Conclusions and future directions

In our work we presented several schemes for estimating the parameter α, to leverage on both local and global neighbors for achieving quality predictions. The α estimates are based on the various kinds of sparsity in the data, which in turn can affect the quality of neighbors generated using local similarity measures. Taking into consideration the various factors which affect the local neighborhood quality, predictions based on our α estimates could attain considerable improvement over the state-of-the art prediction methods proposed in Luo et al. (2008), by allowing global neighbors for the active users to complement the local neighbors in varying degrees, as the sparsity situation warranted. Through the application of real-valued GA to the automatic learning of weights for various sparsity measures, it was possible to appropriately assess true significance of individual sparsity measures for an active user. The learnt weights along with the various sparsity measures allowed the computation of a unified sparsity measure(UMS), that provide α estimates, which outperforms all α estimation schemes based on individual sparsity measures, in terms of prediction quality. Even under varying sparsity levels UMS gives superior performance than the other α-estimates, in most cases.

In the current work, the local and global similarity measures were computed using SVSS and maximin techniques respectively as proposed in Luo et al. (2008). One of the future research directions would be to explore various possibilities for employing alternative techniques for the computation of local and global similarities in the proposed local/global similarity framework, for further improvement. Incorporation of Trust, Distrust and Reputation concepts (Bharadwaj & Al-Shamri, 2009; Victor et al., 2008) in the proposed system also needs to be investigated.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transaction on Knowledge and Data Engineering, 17*(6), 734–749.

Aggarwal, C. C., Wolf, J. L., Wu, K. L., & Yu, P. S. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth acm sigkdd intelligent conf. knowledge discovery and data mining* (pp. 201–212).

Al-Shamri, M. Y. H., & Bharadwaj, K. K. (2008). Fuzzy-genetic approach to recommender system based on a novel hybrid user model. *Expert Systems with Applications, 35*(3), 1386–1399.

Al-Shamri, M. Y. H., & Bharadwaj, K. K. (2007). A compact user model for hybrid movie recommender system. In *Proceedings of the seventh international conference on computational intelligence and multimedia applications (ICCIMA'07)* (pp. 519–524).

Anand, S. S., Kearney, P., & Shapcott, M. (2007). Generating semantically enriched user profiles for web personalization. *ACM Transactions on Internet Technologies, 7*(4). Article no.22.

Bäck, T., Fogel, D., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*. Oxford Univ. Press.

Bell, R. M., Koren, Y., & Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 95–108).

Bharadwaj, K. K., & Al-Shamri, M. Y. H. (2009). Fuzzy computational models for trust and reputation systems. *Electronic Commerce Research and Applications, 8*(1), 37–47.

Billsus, D., & Pazzani, M. (1998). Learning collaborative information filters. In *Proceedings of the international conference on machine learning* (pp. 46–54).

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth annual conference on uncertainty in artificial intelligence* (pp. 3–52). Madison, WI/San Francisco, CA: Morgan Kaufmann.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*, 331–370.

Candillier, L., Meyer, F., & Fessant, F. (2008). Designing specific weighted similarity measures to improve collaborative filtering systems. In *Proceedings of the eigth industrial conference on advances in data mining: medical applications, e-commerce, marketing, and theoretical aspects, LNAI, 5077* (pp. 242–255).

De Jong, K. (1988). Learning with genetic algorithms: An overview. *Machine Learning, 3*, 121–138.

Desrosiers, C., & Karypis, G. (2008). *Solving the sparsity problem: Collaborative filtering via indirect similarities*. Technical Report, University of Minnesota, TR 08-044.

Fouss, F., Pirotte, A., Renders, J. M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering, 19*(3), 355–369.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61–70.

Gori, M., & Pucci, A. (2007). ItemRank: A random-walk based scoring algorithm for recommender engines. In *Proceedings of the international joint conference on artificial intelligence* (pp. 2766–2771).

Gřcar, M., Mladenic, D., Fortuna, B., & Grobelnik, M. (2006). Data sparsity issues in the collaborative filtering framework. *Advances in Web Mining and Web Usage Analysis LNAI, 4198*, 58–76.

Houck, C. R., Joines, J. A., & Kay, M. G. (1995). *A genetic algorithm for function optimization: A matlab implementation*. Technical report NCSU-IE TR 95-09, North Carolina State University, Raleigh, NC.

Koren, Y.(2008). Tutorial on recent progress in collaborative filtering. In *Proceedings of the 2008 ACM conference on recommender systems(ACM Recsys'08)* (pp. 333–334).

Lathia, N., Hailes, S., & Capra, L.(2007). Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of the 2007 ACM conference on recommender systems* (pp. 1–8).

Lee, S., Yang, J., & Park, S.(2004). Discovery of hidden similarity on collaborative filtering to overcome sparsity problem. In *Proceedings of discovery science: Seventh international conference, DS 2004, (LNAI 3245)* (pp. 396–402).

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

Luo, H., Niu, C., Shen, R., & Ullrich, C. (2008). A collaborative filtering framework based on both local user similarity and global user similarity. *Machine Learning, 72*(3), 231–245.

Massa, P., Avesani, P. (2004). Trust-aware collaborative filtering for recommender systems. CoopIS/DOA/ODBASE (1), 492–508.

Melville, P., Mooney, R., & Nagarajan, R.(2002). Content-boosted collaborative filtering_for improved recommendations. In *Proceedings of the eighteenth national conference on articial intelligence* (pp.187–192).

Michalewicz, Z. (1992). Genetic algorithms + data structures = evolution programs. *AI Series*. New York: Springer-Verlag.

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

O'Sullivan, D., Wilson, D., & Smyth, B. (2002). Using collaborative filtering data in case-based recommendation. In *Proceedings of the fifteenth international florida artificial intelligence research society, AAAI* (pp. 121–125).

Popescul, R., Ungar, L. H., Pennock, D. M., & Lawrence, S.(2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments, In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence* (pp. 437–444).

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J.(1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 conference on computer-supported cooperative work* (pp. 175–186).

Rosenstein, M., & Lochbaum, C. (2000). Recommending from content: Preliminary results from an e-commerce experiment. In *Proceedings of CHI'00: Conference on human factors in computing, The Hague, Netherlands* (pp. 291–292).

Russell, S., & Yoon, V. (2008). Applications of wavelet data reduction in recommender systems. *Expert Systems with Applications, 34*(4), 2316–2325.

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J.(2000). Analysis of recommender algorithms for e-commerce. In *Proceedings of the second ACM E-commerce conference(EC'00), Minneapolis, MN* (pp. 158–167).

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international WWW conference, 2001*.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth. In *Proceedings of ACM CHI'95 conference on human factors in computing systems* (pp. 210–217).

Suryavanshi, B. S., Shiri, N., & Mudur, S.P. (2005). Improving the effectiveness of model based recommender systems for highly sparse and noisy web usage data. In *Proceedings of 2005 IEEE/WIC/ACM international conference on web intelligence* (pp. 618–621). IEEE Press.

Yıldırım, H., & Krishnamoorthy M.S. (2008). A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2008 ACM conference on recommender systems(RecSys'08)* (pp. 131–138). Lausanne, Switzerland.

Zhang, J., & Pu, P. (2007). A recursive prediction algorithm for collaborative filtering recommender systems. In *Proceedings of the 2007 ACM conference on recommender systems(RecSys'07)* (pp. 57-64). Minneapolis, Minnesota, USA.