Information Sciences 180 (2010) 602–612

Contents lists available at ScienceDirect

Information Sciences

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

Improving memory-based collaborative filtering via similarity updating and prediction modulation

Buhwan Jeong [a], Jaewook Lee [b,*], Hyunbo Cho [b]

[a] Data Mining Team, Daum Communications Corp., 1730-8 Odeung, Jeju 690-150, South Korea
[b] Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), San 31, Hyoja, Pohang 790-784, South Korea

ARTICLE INFO

Article history:
Received 20 May 2008
Received in revised form 18 October 2009
Accepted 25 October 2009

Keywords:
Collaborative filtering
Mean absolute error (MAE)
Message passing
Recommendation accuracy
Recommender system
Similarity measure

ABSTRACT

Memory-based collaborative filtering (CF) makes recommendations based on a collection of user preferences for items. The idea underlying this approach is that the interests of an active user will more likely coincide with those of users who share similar preferences to the active user. Hence, the choice and computation of a similarity measure between users is critical to rating items. This work proposes a similarity update method that uses an iterative message passing procedure. Additionally, this work deals with a drawback of using the popular mean absolute error (MAE) for performance evaluation, namely that ignores ratings distribution. A novel modulation method and an accuracy metric are presented in order to minimize the predictive accuracy error and to evenly distribute predicted ratings over true rating scales. Preliminary results show that the proposed similarity update and prediction modulation techniques significantly improve the predicted rankings.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

Efforts to develop automated recommender systems are made difficult by the ongoing expansion of information content. A recommender system filters information items and removes those that are not relevant to the interests of an active user, suggests the top-*N* candidate items, and/or gives rating estimates for items of interest. Applications of such systems include spam filtering [11] and making recommendations regarding books and CDs [26], news articles [21], and movies [27]. As described in [1], recommender system algorithms can be classified as content-based, collaborative, or hybrid approaches based on the procedure used to make recommendations. Memory-based collaborative filtering (MeCF), which predicts preferences based on preferences of neighbor users and/or items, has become the preferred recommendation method. Although researchers, both in academia and industry, have expended considerable efforts to improve and stabilize MeCF, there is still room for improvement, especially in regard to the two principal metrics employed, that is, measures of similarity between users (or between information items) and error estimates that quantify recommendation accuracy. The classic MeCF algorithm predicts ratings for unrated items by incorporating ratings of similar neighbor users. Therefore, the choice of similarity measure is critical to rating accuracy. Most similarity measures proposed to date are designed to solve specific problems (e.g., the cold-starting problem for new users/items [2] and item-based prediction [29]). These measures can be further improved through an iterative updating process. Second, the choice of good error measures that prevent MeCF from incorrectly ratings items and improve the rating accuracy is also important [12]. Most existing performance measures focus on minimizing the gap between real and predicted ratings (mean absolute error, MAE). However, predicted ratings tend to cluster

* Corresponding author. Tel.: +82 54 279 2209.
E-mail address: jaewookl@postech.ac.kr (J. Lee).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.10.016

around the average of the true ratings (see Fig. 2) resulting in small MAE, regardless of the distribution of true ratings. This leads to unsatisfactory rating of items, and hence to user distrust in the recommended ratings.

The present work aims to assess the similarity and accuracy of these performance error metrics and to propose more accurate ones. First, we propose a novel method to solidify similarity measures through iterative messages passing framework proposed in [8]. Second, we highlight the problem of overusing the MAE as the primary evaluation metric and present a novel evaluation one that overcomes the shortcomings of the MAE.

2. Related works

In this section, we review previous research on MeCF. We begin by giving a brief review of MeCF methods. We review two important metrics used in MeCF: (1) similarity measures between users or items; and (2) performance measures for evaluating the accuracy of the recommender system.

2.1. Memory-based collaborative filtering

MeCF [32] is a simple but powerful heuristic method that estimates ratings of unrated items relative to a collection of items previously rated by users. The conventional MeCF algorithm finds "neighbor" users $u' \in \widehat{U}$ that have similar preferences to an active user u, and then aggregates their ratings on items that are unrated by the active user. In other words, the value of the unknown rating $r_{u,s}$ for user u and item s is computed as an aggregate of the ratings for the same item s by other users u'. The popular rating aggregates are given as

$$r_{u,s} = \frac{1}{|\widehat{U}|}\sum_{u'\in\widehat{U}} r_{u',s}, \tag{1}$$

$$r_{u,s} = \kappa \sum_{u'\in\widehat{U}} sim(u,u') \times r_{u',s}, \tag{2}$$

$$r_{u,s} = \bar{r}_u + \kappa \sum_{u'\in\widehat{U}} sim(u,u') \times (r_{u',s} - \bar{r}_{u'}), \tag{3}$$

where the normalization factor κ is usually given by $\kappa = 1/\sum_{u'\in\widehat{U}}|sim(u,u')|$, and $\bar{r}_u$ indicates the average rating of the user u. As mentioned, the formulas take a weighted average of ratings by other users having similar preferences, $u' \in \widehat{U}$. Some recent CF algorithms incorporate item-driven aggregations (e.g., $r_{u,s} = \sum_{s'\in\widehat{S}} r_{u,s'}/|\widehat{S}|$) in a similar way [29]. Unlike the user-driven approaches that first find users that are similar to the active user and then recommend items commonly rated by those other users, item-driven approaches look for a set of candidate items similar to a chosen item and then recommend items from that set that are commonly preferred by others. This method accommodates similarities between items. A mixture of both the user-driven and item-driven approaches often performs better than the individual approaches [31]. In [31], two parameters λ and δ are introduced to combine individual user- and item-similarities, and to integrate the similarity combined similarity into user-item similarity, respectively.

2.2. Similarity measures

As shown in Eqs. (2) and (3), the similarity measure between users u and u', $sim(u,u')$, plays an essential role in rating prediction. Increased similarity between users u and u' implies an increased weight of the rating $r_{u',s}$ for the prediction of $r_{u,s}$. User similarity is often based on ratings of items that both users have rated. The Pearson correlation coefficient and cosine similarity are the most popularly used measures of similarity. Let S_{uv} be the set of all items co-rated by both users u and v, $S_{uv} = \{s \in S | r_{u,s} \neq \emptyset \text{ and } r_{v,s} \neq \emptyset\}$. The Pearson correlation coefficient is defined as

$$sim(u,v) = \frac{\sum_{s\in S_{uv}}(r_{u,s} - \bar{r}_u)(r_{v,s} - \bar{r}_v)}{\sqrt{\sum_{s\in S_{uv}}(r_{u,s} - \bar{r}_u)^2 \sum_{s\in S_{uv}}(r_{v,s} - \bar{r}_v)^2}}. \tag{4}$$

The cosine similarity treats the two users u and v as two $|S_{uv}|$-dimensional vectors such that similarity is measured by the cosine of the angle between the vectors,

$$sim(u,v) = \cos(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|_2 \times \|\vec{v}\|_2} = \frac{\sum_{s\in S_{uv}} r_{u,s} r_{v,s}}{\sqrt{\sum_{s\in S_{uv}} r_{u,s}^2}\sqrt{\sum_{s\in S_{uv}} r_{v,s}^2}}. \tag{5}$$

Euclidean distance can be used to estimate similarity as well. In addition to these popular measures, several researchers have proposed novel similarity measures such as the adjusted item similarity [29], random walk counting [6] , the PIP (Proximity-Impact-Popularity) measure [2], rank correlation [2], user-class similarity [33], and UNION similarity [30]. These measures have their own strengths and purposes in different applications and/or situations (e.g., cold-starting problem, rating sparsity, and explicit or implicit ratings). Descriptions of all previously reported recommender systems – recommendation methods

(e.g., the hybrid approach [3], ItemRank [10], latent class model [13], linear regression model [20], semi-explicit rating approach [17]), related metrics [14–16], applications, usages, and arising issues – are outside of the scope of this paper. For a review, please see [1] as a starting reference. We would, however, like to mention that many of the proposed measures concentrate on specific problems, for example PIP for the cold-starting problem, adjusted item similarity for item-driven computation, and user-class similarity only when class information is available. A main purpose of the present work is to develop a more generic approach that updates (and thereby improves) existing state-of-art similarity measures.

2.3. Performance evaluation measures

Recommender systems are designed to assist users in identifying desirable information. One widely used performance metric for recommendation evaluation is *accuracy*, which quantifies the degree of errors between real and predicted ratings. Accuracy can be roughly classified into *predictive* accuracy, *decision-support* accuracy, and *rank* accuracy [12]. First, predictive accuracy metrics measure how closely the predictions of a recommender system conform to true user ratings. MAE is the metric typically used for this purpose, and is much more widely used than the other metrics listed below. The MAE is defined as the average absolute deviation between predicted ratings and true ratings, $\mathrm{MAE} = \sum_{v=1}^{V} |t_v - p_v|/V$ for V pairs of true and predicted ratings $\langle t_v, p_v \rangle$. Variant metrics such as *mean squared error*, *root mean square error*, and *normalized mean absolute error* fall into this category.

Decision-support accuracy metrics measure the acceptance rate by the user. Typical metrics of this type are *reversal rate*, *ROC curve*, and *precision and recall*. Reversal rate is a measure of the frequency with which the recommender system makes big mistakes (or poor recommendations) that might undermine a user's confidence in the system. A high reversal rate implies that the system frequently makes poor recommendations regarding whether a user will strongly desire or dislike an item. The ROC (receiver operating characteristics) curve in signal detection theory plots the sensitivity (or specificity) for a test of whether a randomly selected good (or bad) item is accepted (or rejected) by a filtering system. Precision and recall are measures of the degree to which the system presents relevant information: *precision* is defined as the ratio of relevant items selected to the number of items selected, while *recall* is the ratio of relevant items selected to the total number of relevant items available. *F-Measure* is also often used to combine precision and recall [12].

Rank accuracy metrics measure the ability of a recommendation algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. Therefore, such metrics are appropriate for evaluating algorithms that will be used to present a ranked recommendation list to users. Exemplary metrics include *correlation coefficient*, *half-life utility*, and the normalized distance-based performance measures (*NDPM*). The correlation coefficient indicates the strength and the direction of a linear relationship between two random variables; three of the best-known coefficients are the *Pearson's* coefficient, *Spearman's* ρ, and *Kendall's* τ. The half-life utility metric evaluates a ranked list from the user in terms of the difference between the user rating and default rating for an item. For weakly ordered rankings, NDPM is also used. It is important to note that, according to [12], these accuracy metrics are closely correlated. See [12] and [28] for detailed descriptions of the metrics.

In summary, accuracy metrics account for the mean difference, correction rate, and/or rank ordering. An intrinsic drawback of these metrics is that they may ignore the distribution of ratings. For example, MAE [12] computes the correlation between every pair of true and predicted ratings, but it gives the same result as the difference between the mean values of the true and predicted ratings. The predicted ratings, therefore, tend to cluster around the mean values of true ratings (see Fig. 2). This tendency makes a recommender system have a low MAE, but the predicted ratings by the system are likely on mean rating. Although the system still gives the correct top-N items, users are hard to discriminate which items are more likely the ones they want. Users want a high rating for the one they really want. In that, they are confident of the recommender system. Detailed examples and explanations are provided in the following section.

3. Proposed similarity solidification and prediction modulation

This section first describes a generic method to improve existing similarity measures by means of an iterative updating method. Specifically, we employed a message propagation algorithm that recursively interpolates pairwise similarities. We address some drawbacks of existing accuracy metrics and propose a way to evenly distribute predicted ratings and to incorporate rating distributions into prediction errors.

3.1. Similarity update through message passing

The similarity measures that are currently in widespread use – correlation and cosine similarity – consider only individual pairs of users (and pairs of items), leaving no room for improvement by using correlations with other users or items. Here, we seek to apply similarity updates through an iterative procedure of passing messages between data points (users and/or items). For this purpose, we adopt the affinity propagation algorithm described in [8] and [7]. This algorithm is a variant of the belief propagation (BP) algorithm (also called the sum-product algorithm) from probabilistic graphical models [18], and was originally intended to cluster data points using pairwise proximity to obtain an exemplar point that serves to describe or represent each cluster. Here, BP in general, intending to solve an inference problem, begins by constructing

a graphical model consisting of random variables and their conditional probabilities. The algorithm iteratively computes marginal distributions of the graphical model. For readers unfamiliar with the underlying ideas about graphical models, see [4] and [18].

Given a set of points and a set of similarity values between the points, affinity propagation finds clusters of similar data points and respective exemplars. Specifically, affinity propagation exchanges two types of messages between data points, designated *responsibility* and *availability*, using a similarity matrix that can be sparse. According to the original definition [8], "the responsibility $res(i,k)$ reflects the accumulated evidence for how well-suited a point k is to serving as the exemplar, the centroid of a cluster of real data points, for point i, taking into account other potential exemplars for point i, while the availability $av(i,k)$ reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar". Fig. 1 illustrates affinity propagation, the passing of responsibility messages and availability messages.

Given initial availabilities $av(i,k) = 0$, responsibilities and availabilities are iteratively and recursively updated using the following rules:

$$res(i,k) \leftarrow sim(i,k) - \max_{k' \text{ s.t. } k' \neq k} \{av(i,k') + sim(i,k')\}, \tag{6}$$

$$av(i,k) \leftarrow \min\left\{0, res(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, res(i',k)\}\right\}, \tag{7}$$

where $sim(i,k)$ is the similarity between points i and k, and the self-availability $av(k,k)$ is updated differently as $av(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, res(i',k)\}$. After the messages become stable – where stability is determined after a fixed number of iterations, after changes in the messages fall below a threshold, or after the local decisions are constant for some number of iterations – the updated similarity is set by $sim(i,k) = res(i,k) + av(i,k)$, followed by a normalization onto [0 1]. In summary, the affinity propagation algorithm first constructs a network in which data points are nodes and pairwise similarity values are interconnecting edges. Real-valued messages are iteratively transmitted along the edges of the network until the network becomes stable, that is, the energy is minimized. The original discussions of affinity propagation [8,7] argued that the algorithm has a time complexity of less than $O(N^2)$, implying that the similarity update has this time complexity.

Note that message passing/propagation theoretically implements a computation similar to the random walk in a Markov model [6,10]. The convergence of the iterative message passing procedure is the same as the convergence of the stationary state matrix after repeated application of the transition matrix. The computation differ in their input graphs: the random walk model in [6] uses a bipartite graph consisting of different types of nodes (users and items), whereas ItemRank [10] and our method use a graph of identical nodes. In addition, ItemRank focuses on the correlation index between each pair of items, but our proposed method can be used in conjunction with any similarity matrix. (Kernel-based clustering approaches [19,22–25] might be alternative methods for similarity updates of this kind when an appropriate similarity metric is provided.)

3.2. Prediction modulation

3.2.1. Misleading of memory-based CF by the MAE

The previous section showed that MeCF performs well in rating unrated items relative to rated items based on a collection of user preferences. However, the ratings predicted by MeCF tend to cluster near the mean values of the true ratings, regardless of the aggregation weights (similarity between users or between items) used. For example, Fig. 2 shows 10,000 unknown ratings randomly selected from the MovieLens (ML) data, where the default experimental configuration is the simple mixture of CF and cosine similarity measures. Note that the standard deviation of true ratings σ_t^{ML} is about three times larger than that of the predicted ratings σ_p^{ML}, i.e., $\sigma_t^{\mathrm{ML}} \simeq 2.9 \times \sigma_p^{\mathrm{ML}}$. The recommender systems not only give a list of top-N

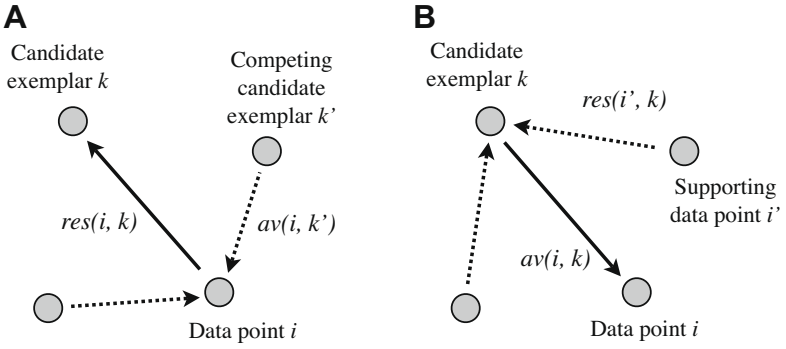

Fig. 1. Affinity propagation: sending (A) responsibilities and (B) availabilities [8].

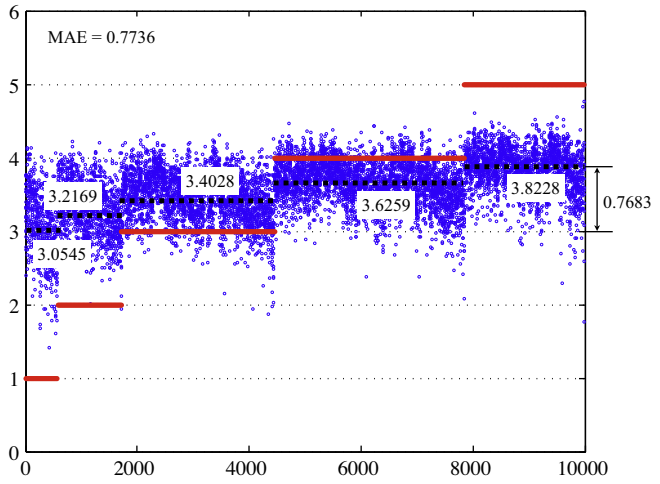

Fig. 2. Predicted ratings (ML) by the memory-based CF.

recommendations in the descending order of the predicted rating, but also support the active user's decision by showing predicted ratings of items. In other words, the predicted ratings are used not only to select the top-*N* items, but also to support user decisions. Using the clustered ratings may not pose any problem in regard to selecting the top-*N* items, but it can cause critical problems if used to support a user's decision. In the case of the *Find Good Items* strategy [12], which recommends items whose predicted ratings are above a threshold, the recommender system cannot properly recommend items. For example, a classic MeCF is used to rate unrated items, as shown in Fig. 2 (circles). In this figure, solid lines indicate the true user rating values, and the dotted lines indicate average values of predicted ratings. The MAE of 0.7736 is a fair value for the ML data. Nonetheless, if the threshold is set to 4.5, for example, then there are no recommendations (or few, if available) suggested by the recommender system. Although lowering the threshold gives users more alternatives, users cannot be confident in the suggested items with low rating values. Additionally, the number of sample points whose true ratings are either 1 or 2 is smaller than the number of true ratings of 3, 4, or 5; hence the MAE remains near 0.77. If the dataset has even numbers of sample points from each rating, the MAE can increase near 1.05. In summary, MeCF, as well as possibly the model-based CF [1], predicts ratings near the average. Using the MAE only to evaluate recommender systems may lead to undesirable consequences, especially with the *Find Good Items* strategy. Because most evaluation metrics listed above are linearly related [12], other metrics will also produce the same misleading results.

3.2.2. Prediction modulation and dispersion-driven MAE

From the observations listed above, we require (i) the recommender system to spread predicted ratings evenly over the rating scale, and (ii) the evaluation metric to incorporate the degree of dispersion. For the first requirement, we modulate the predicted ratings r_p so that they are distributed similarly to the true ratings r_t, i.e., $\sigma_p \approx \sigma_t$. We assume, based on the central limit theorem, that both ratings are normally distributed, $r \sim N(\mu, \sigma)$. Using the standardization equation $z = (x - \mu)/\sigma$, we deploy the prediction modulation, Eq. (10), as follows:

$$z_t \approx z_p, \tag{8}$$

$$\frac{r_t - \mu_t}{\sigma_t} \approx \frac{r_p - \mu_p}{\sigma_p}, \tag{9}$$

$$r_{u,s} \approx \frac{\sigma_t}{\sigma_p} \cdot (r_p - \mu_p) + \mu_t, \tag{10}$$

$$r_{u,s} = \alpha \cdot \frac{\sigma_t}{\sigma_p} \cdot (r_p - \mu_p) + \mu_t, \tag{11}$$

$$r_{u,s} = \beta \cdot (r_p - \mu_p) + \mu_t, \tag{12}$$

where the ratio of standard deviations σ_t/σ_p serves as a dispersion or modulation factor. The multiplier α in Eq. (11) generalizes the modulation. Furthermore, we may rewrite Eq. (11) as Eq. (12) because the standard deviations are nearly fixed. The modulation disperses the predicted ratings fairly over the rating scale as shown in Fig. 3, for example, so that the final ratings $r_{u,s}$ can be used as the reference values to support the active user's decision. Although the modulation seemingly distorts the rating values as marked by circles in Fig. 3, it never changes the order of recommendations from the original MeCF. Therefore, the top-*N* recommendations after rating modulation are the same as those made by MeCF.

For the second requirement of a dispersion-driven evaluation metric, we have modified existing predictive accuracy metrics. The new metrics are defined as the predictive accuracy multiplied by the difference between the degrees of

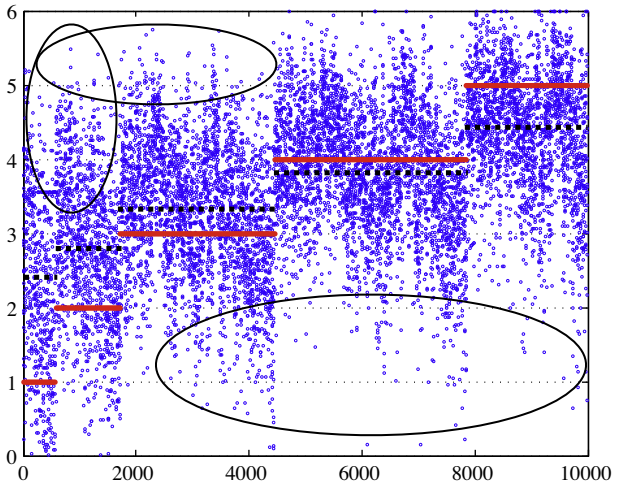

Fig. 3. Exemplar rating dispersion/modulation.

scatter of the true and predicted ratings. For instance, a new MAE metric, namely dispersion-driven MAE (DMAE), is defined as

$$\mathrm{DMAE} = \mathrm{MAE} \times \Delta(d_t, d_p), \tag{13}$$

where d_t and d_p represent the degree of scatter for true and predicted ratings, respectively, and the function $\Delta(,)$ quantifies the difference in the degree of scatter. The degree of scatter d can be measured in terms of the standard deviation of the ratings and the difference between the means of predicted ratings of those items having the highest true rating and of those having the lowest true rating. In Fig. 2, for example, the differences between the true rating averages and between the predicted rating averages are 4 and 0.7683, respectively. The scatter difference function can be any convex (e.g., second-order polynomial) function (loss function) whose minimum appears near $d_t = d_p$. An example function is defined as follows:

$$\Delta(d_t, d_p) = \rho \cdot \left(\frac{d_t - d_p}{d_t}\right)^2 + \eta, \tag{14}$$

where the positive constants ρ and η balance this dispersion factor with the predictive accuracy factor (MAE). The dispersion factor is secondary to the new metric. It is suggested that the constant η be set to no less than the minimum MAE.

4. Numerical experiment

4.1. Experimental setting

To evaluate the proposed similarity update, we used ML data [29] and Book-Crossing (BX) data [34]. The ML data contains 100,000 ratings (on a 1–5 scale) from 943 users and 1682 items. The BX data contains 1,149,790 ratings (433,681 explicit ratings on a 1–10 scale and 716,109 implicit ratings) from 278,858 users and 271,379 books. We used two smaller subsets, BX1 and BX2, which have 19,886 and 40,620 explicit ratings, respectively (see Table 1). The BX data are rescaled onto [1 5]. The use of subsets BX1 and BX2 demonstrates the proposed algorithm's performance at different levels of sparsity and size, in terms of accuracy and time complexity. Constraints for selection of each subset are that (i) each row or column has enough ratings to avoid severe sparsity (BX1 has rating data about movies with more than 20 ratings, and raters rated more than five movies; and BX2 has 10 and 5, respectively) and (ii) our implementation must be able to support the size of the resulting rating matrix. In addition, we tried to perform the same simulation on large-scale data (Netflix Prize data, which contains

Table 1
Brief description of the ML/BX1/BX2 data sets.

Description	ML	BX1	BX2
No. of explicit ratings	100,000	19,886	40,620
Matrix size	943 × 1682	1441 × 616	1880 × 2142
Sparsity level (%)[a]	93.69	97.76	98.99

[a] Sparsity level = 100 × (total entries − nonzero entries)/(total entries).

over 100 million ratings), but our implementation on MATLAB could not support such a huge data set. We will discuss this point below, at the end of the simulation results.

Next, we used the cosine similarity 'COS', Pearson correlation coefficient 'PCC', Euclidean distance 'EUC' (which is transformed into a similarity via EUC $= 1 - dist./dist._{max}$), and modified cosine similarity 'MOD'. 'MOD' incorporates the proportion of shared items rated by both users (or users who rated both items) into the standard cosine similarity, MOD $= \cos(\vec{u}, \vec{v}) \times |S_u \bigcap S_v|/|S_u \bigcup S_v|$, for two item- or user-oriented rating sets S_u and S_v. Other general-purpose similarity measures (chebyshev, jaccard, and union, but not problem-specific measures such as the PIP, adjusted item similarity, and user-class similarity) were subjected to our analysis. The results are not presented here because they showed behavior similar to that observed for the similarity measures presented here, and our proposed update method improves their prediction accuracy. The main purpose of our work is not to propose an improved similarity measure, but to show that the message passing method can regulate and update many existing similarity measures, improving the prediction performance of existing CFs.

To demonstrate this improvement, we summarize the experimental results obtained using four popular similarity measures. We used a mixture of both user-driven and item-driven predictions as described in [31], because this approach gives better predictions than do the individual user- or item-driven predictions. The mixture model assigns equal importance to user-driven and item-driven prediction, but has no interaction effect, that is, $\lambda = 0.5$ and $\delta = 0$. Rather than fixing the number of similar neighbors, we used all neighbors whose similarity exceeded an arbitrary similarity cutoff. If there were no users/items above the cutoff, we selected up to N most similar users/items (N = 10 in this work). As shown in Fig. 4, the recommender system becomes worse (MAE approaches 0.81) as the cutoff is increased up to 0.3, then improves and stabilizes around MAE = 0.75. Because the overall similarity average is 0.15, the number of neighbors above the cutoff gradually decreases, and above a cutoff of 0.3, the effect of the fixed N neighbors appears. A cutoff of 0.5 not only produces a mean result, but also prevents the method from being dominated by the effect of the top-N neighbors. The updated similarity was observed to give better and more robust predictions for all cutoffs.

MAE and RMSE (root mean squared error) [28] are used as performance metrics. The use of the RMSE in addition to the MAE accentuates the improvement afforded by the similarity update method. Finally, each experiment was replicated 10 times with different training (80%) and test (20%) sets. The similarity threshold was set to 0.5 in this experiment. To our knowledge, no guideline exists for the selection of an optimal threshold, and empirical pretests are required for each application of interest. Although a smaller similarity threshold (a larger set of neighbors) tends to perform better, in practical applications a trade-off must be made between accuracy and complexity. All experiments are performed on a 2.5 GHz Intel Core 2 Duo CPU PC with 2 GB RAM. Sample implementations of affinity propagation [8] are available online at http://www.psi.toronto.edu/affinitypropagation.

4.2. Experimental results of similarity update

The experimental results for similarity update are summarized in Table 2. Bold font indicates the best performance among various similarity measures. The use of message passing significantly improved the recommendation accuracy (0.48–19.97%) for all similarity measures except for Euclidean distance, 'EUC'. The ANOVA P-values indicate that the null hypothesis, $h_0 : \mu_o = \mu_u$, can be rejected with a 99% confidence interval for all measures except 'COS' and 'MOD' for the ML data. The lack of improvement after updating the 'EUC' similarity is unexpected. It is possible that the affinity propagation algorithm [8] interpolates unknown similarities between unconnected data points in the Euclidean space. This possibility is supported by [6], which found that the square root of the average commute time (which is the sum of the availability

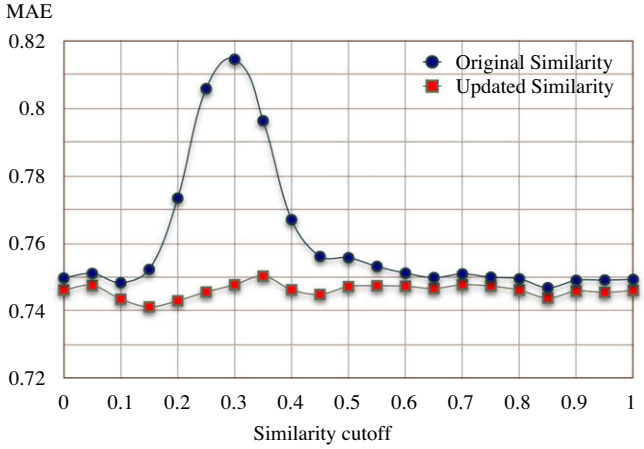

Fig. 4. Prediction sensitivity as a function of cutoff value.

Table 2
Summary of experimental results with cosine similarity, Pearson correlation coefficient, Euclidean distance, and modified cosine similarity.

		COS	PCC	EUC	MOD
ML	Original MAE	0.7486	0.7628	0.7584	0.7379
	(RMSE)	(0.9578)	(0.9804)	(0.9553)	(0.9451)
	Update MAE	0.7451	0.7535	0.7584	**0.7354**
	(RMSE)	(0.9413)	(0.9569)	(0.9553)	**(0.9305)**
	Improvement rate (%)	0.48	1.23	0	0.47
	(RMSE)	(1.73)	(2.39)	(0)	(1.55)
	P-value	0.0191	4.75×10^{-5}	–	0.1139
	(RMSE)	(4.19×10^{-4})	(1.13×10^{-8})	(–)	(1.04×10^{-4})
BX1	Original MAE	0.5788	0.6176	**0.5173**	0.5759
	(RMSE)	(0.8714)	(0.7847)	**(0.6780)**	(0.8531)
	Update MAE	0.5388	0.5533	**0.5173**	0.5382
	(RMSE)	(0.7097)	(0.7279)	**(0.6780)**	(0.7083)
	Improvement rate (%)	6.54	10.41	0	6.55
	(RMSE)	(18.6)	(7.23)	(0)	(16.98)
	P-value	3.87×10^{-9}	1.87×10^{-16}	–	8.47×10^{-11}
	(RMSE)	(1.29×10^{-12})	(6.83×10^{-14})	(–)	(4.16×10^{-14})
BX2	Original MAE	0.6836	0.6167	**0.5195**	0.6684
	(RMSE)	(1.1991)	(0.7781)	**(0.6763)**	(1.1119)
	Update MAE	0.5471	0.5514	**0.5195**	0.5470
	(RMSE)	(0.7184)	(0.7181)	**(0.6763)**	(0.7175)
	Improvement rate (%)	19.97	10.59	0	18.16
	(RMSE)	(40.09)	(7.71)	(0)	−(35.47)
	P-value	4.29×10^{-17}	3.92×10^{-17}	–	3.66×10^{-17}
	(RMSE)	(2.80×10^{-21})	(3.80×10^{-14})	(–)	(1.29×10^{-19})

and responsibility of the affinity propagation) is equivalent to a Euclidean distance. Other similarity/distance measures (chebyshev, jaccard, union, cosine, and correlation) display significantly improved CF performance. The best predictions were obtained using 'MOD' for the ML data and 'EUC' for the BX data. No single similarity measure outperforms all others for all applications or data sets. It is unusual that the simple Euclidean distance performs best for the BX data. This may result from the properties of the BX data itself. To our knowledge, the EUC characteristics of the BX data have not been theoretically verified. The use of RMSE affords the same results as MAE. The updated similarity also increases the rank accuracy, that is, the correlation between the predicted and true ratings (data not shown).

The time complexity of the similarity update is expected to be less than $O(N^2)$, because affinities need only be propagated between relevant pairs [8,7]. Simulations performed on the BX data show that the update time, on average, increases from 10.4 s to 35.6 s for BX1 and BX2 data, respectively. The update time corresponds to the time required to update similarities between both items and users, according to the mixture approach. For reference, the ML data requires 16.6 s for the similarity update alone. Because similarity updates are performed off-line, the computation time may not pose a problem in practical applications. Nonetheless, efficient ways to deal with large-scale data are urgently needed. We failed to achieve meaningful results in simulations on very large-scale data (the Netflix Prize data) due to insufficient memory on the MATLAB program and very long computation times. The use of local similarity updates is a feasible strategy. To accomplish this, we need to group users and associated items simultaneously (co-clustering [5]), and update the similarities in dense user-item groups. Final recommendations can also be made within a corresponding user-item group [9]. The computation time for computing (cosine) similarities of all pairs of users and items, and for predicting 20,000 unknown ratings, is much smaller than the time required for updating similarities: 1.1 s and 0.05 s for the ML data, respectively. The BX2 data required 1.2 s and 0.0065 s, respectively.

4.3. Experimental results of DMAE

We first predicted unknown ratings for the ML data using MeCF [31], then modulated them by varying α on [02]. Table 3 summarizes the true ratings, predicted ratings, and modulated ratings at $\alpha = 0.61$ (where the MAE reached a minimum). Table 3 also provides the following metadata: average, differences in degree of scatter (in terms of average rating gap and standard deviation), dispersion, MAE, and DMAE. The table shows that average of all ratings does not differ greatly between MeCF-predicted and modulated ratings ($\mu_{\text{MeCF}} = 3.572,\ \mu_{\text{Mod}} = 3.578$), but the scatter differences are large ($\Delta_{\text{MeCF}} = 0.77,\ \Delta_{\text{Mod}} = 1.97$). Figs. 2 and 5 show the same observation as the Table 3 shows. More importantly, modulation provides better predictive accuracy ($\text{MAE}_{\text{MeCF}} = 0.774$ and $\text{MAE}_{Mod} = 0.736$).

Fig. 6 shows the sensitivity of the MAE, scatter difference, and DMAE to the value of α. To compute the difference in the degree of scatter, we used the standard deviations of the true and predicted/modulated ratings. The control constants ρ and η in Eq. (14) were set to 1 and 0.7408, respectively. As expected, the dispersion factor reaches a minimum when α is near 1.1,

Table 3
Comparison between original memory-based CF and prediction modulation at $\alpha = 0.61$.

	True rating	Memory-based CF	Prediction modulation
Rating average at each true rating	1	3.0545	2.4173
	2	3.2169	2.8320
	3	3.4028	3.2892
	4	3.6259	3.8399
	5	3.8228	4.3830
Average	3.5394	3.5722	3.5782
Difference	4	0.7683	1.9657
St. Dev.	1.1241	0.3867	0.6711
Dispersion	0	1.1711	0.9032
MAE	0	0.7736	0.7359
DMAE	0	0.9060	0.6647

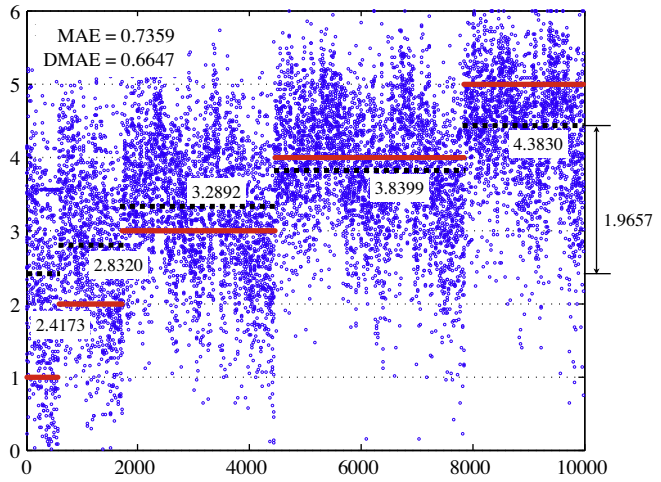

Fig. 5. ML ratings after modulation ($\alpha = 0.61$ or $\beta = 1.78$).

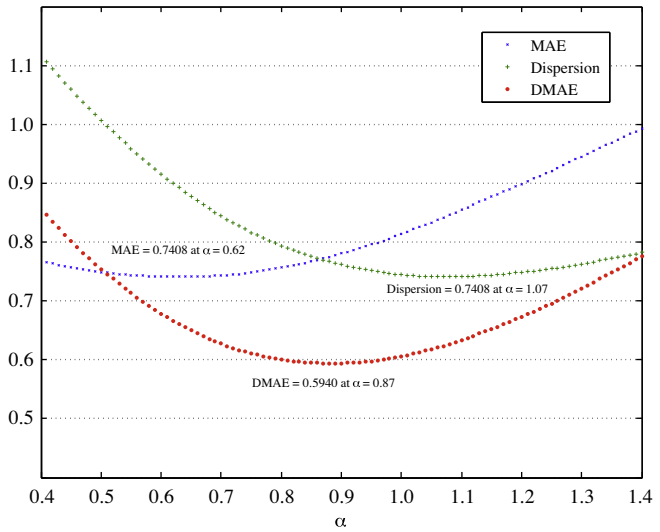

Fig. 6. Sensitivity of the MAE, scattering difference, and DMAE by α.

whereas the MAE reaches a minimum at $\alpha = 0.6$. Hence, the DMAE reaches a minimum (0.5940) at $\alpha = 0.87$. The experimental results using the BX data for they gave similar results, (data not shown).

5. Conclusions

Recommender systems play a crucial role in the provision of user-specific services by filtering the massive quantity of available information to extract information on user preferences. The rankings afforded by recommender systems can be used in contexts such as recommending products to purchase or news articles to read. In particular, collaborative filtering (CF), which utilizes the information of user preferences for items, has recently gained attention.

In the present work, we proposed two principal metrics for similarity and accuracy. First, we proposed a similarity update method that solidifies naïve similarities between users/items, computed, for example, using the cosine similarity and correlation coefficient. The similarities are updated by an iterative process of message passing between data points (users or items). We adopted an affinity propagation algorithm that, using a proximity matrix, computes *responsibility* and *availability* messages between data points. Although the original propagation algorithm used these messages to check whether two corresponding data points belonged to the same cluster, we defined a normalized value of the summation of messages between data points as their corresponding similarity to improve performance. Use of the MAE and other performance metrics often misleads MeCF to predict ratings near an average rating. This is because MeCF does not consider the distribution of ratings. Hence, we proposed a prediction modulation method that distributes condensed predicted ratings evenly over the rating scale. The modulation uses the ratio of deviations of true and predicted ratings. We also presented dispersion-driven MAE (DMAE) that comprises both the predictive accuracy (MAE) and the degree of rating dispersion. Experimental results showed that both proposed methods significantly improve the classic MeCF. In this work, we experimentally evaluated each method and applied the methods only to explicit ratings for movies and books. Integrating both methods and evaluating their combination in applications requires further investigation. Future studies will consider local similarity updates with the co-clustering approach to efficiently cope with large-scale data.

Acknowledgments

The authors wish to thanks to Shyong Lam and Jon Herlocker for cleaning up and generating the MovieLens (ML) data set, and to Cai-Nicolas Ziegler and Ron Hornbaker for the Book-Crossing (BX) data set. This work was supported partially by the Korea Research Foundation under the Grant No. KRF-2008-314-D00483 and partially by the KOSEF under the Grant No. R01-2007-000-20792-0.

References

[1] G. Adomavicius, A. Tuzhilin, 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) (2005) 734–749.
[2] H.J. Ahn, A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, Information Sciences 178 (1) (2008) 37–51.
[3] J. Basilico, T. Hofmann, Unifying collaborative and content-based filtering, in: Proceedings of the 21st International Conference on Machine Learning, 2004, pp. 65–72.
[4] C. Bishop, Graphical models, in: Pattern Recognition and Machine Learning, Springer-Verlag, New York, 2006, pp. 359–422.
[5] I.S. Dhillon, S. Mallela, D.S. Modha, Information-theoretic co-clustering, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 89–98.
[6] F. Fouss, A. Pirotte, J.-M. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering 19 (3) (2007) 355–369.
[7] B. Frey, D. Dueck, Mixture modeling by affinity propagation, Advances in Neural Information Processing Systems 18 (2006) 379–386.
[8] B. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[9] T. George, S. Merugu, A scalable collaborative filtering framework based on co-clustering, in: Proceedings of the 5th IEEE International Conference on Data Mining, 2005, pp. 625–628.
[10] M. Gori, A. Pucci, ItemRank: A random-walk based scoring algorithm for recommender engines, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 2766–2771.
[11] S. Han, Y. Ahn, S. Moon, H. Jeong, Collaborative blog spam filtering using adaptive percolation search, in: Proceedings of the 15th International Conference on World Wide Web (WWW 2006), 2006.
[12] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.
[13] T. Hofmann, J. Puzicha, Latent class models for collaborative filtering, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI), 1999, pp. 688–693.
[14] B. Jeong, D. Lee, H. Cho, J. Lee, A novel method for measuring semantic similarity for XML schema matching, Expert Systems with Applications 34 (2008) 1651–1658.
[15] B. Jeong, D. Lee, J. Lee, H. Cho, Support for seamless data exchanges between web services through information mapping analysis using kernel methods, Expert Systems with Applications 36 (1) (2008) 358–365.
[16] B. Jeong, J. Lee, H. Cho, User credit-based collaborative filtering, Expert Systems with Applications 36 (3) (2009) 7309–7312.
[17] B. Jeong, J. Lee, H. Cho, An iterative semi-explicit rating method for building collaborative recommender systems, Expert Systems with Applications 36 (3) (2009) 6181–6186.
[18] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul, An introduction to variational methods for graphical models, Machine Learning 37 (1999) 183–233.
[19] H.-C. Kim, J. Lee, Clustering based on gaussian processes, Neural Computation 19 (11) (2007) 3088–3107.
[20] J. Kunegis, S. Albayrak, Adapting ratings in memory-based collaborative filtering using linear regression, in: Proceedings of IEEE International Conference on Information Reuse and Integration (IRI 2007), 2007, pp. 49–54.
[21] H. Lee, S. Park, MONERS: a news recommender for the mobile web, Expert Systems with Applications 32 (1) (2007) 143–150.
[22] D. Lee, J. Lee, Equilibrium-based support vector machine for semi-supervised classification, IEEE Transactions on Neural Networks 18 (2) (2007) 578–583.
[23] D. Lee, K.-H. Jung, J. Lee, Constructing sparse kernel machines using attractors, IEEE Transactions on Neural Networks 20 (4) (2009) 721–729.

[24] J. Lee, D. Lee, An improved cluster labeling method for support vector clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 461–464.
[25] J. Lee, D. Lee, Dynamic characterization of cluster structures for robust and inductive support vector clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 461–464.
[26] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Computing 7 (1) (2003) 76–80.
[27] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, J.T. Riedl, MovieLens unplugged: experiences with an occasionally connected recommender system on four mobile devices, in: Proceedings of the 2003 International Conference on Intelligent User Interfaces (IUI 2003), 2003, pp. 263–266.
[28] B.M. Sarwar, J.A. Konstan, A. Borchers, J. Herlocker, B. Miller, J. Riedl, Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system, in: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCS'98), 1998, pp. 345–354.
[29] B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[30] P. Symeonidis, A. Nanopoulos, A.N. Papadopoulos, Y. Manolopoulos, Collaborative filtering: fallacies and insights in measuring similarity, in: Proceedings of 10th PKDD Workshop on Web Mining (WEBMine'2006), 2006, pp. 56–67.
[31] J. Wang, A.P. de Vries, M.J.T. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 501–508.
[32] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H.-P. Kriegel, Probabilistic memory-based collaborative filtering, IEEE Transaction on Knowledge and Data Engineering 16 (1) (2004) 56–69.
[33] C. Zeng, C.-X. Xing, L.-Z. Zhou, X-H. Zheng, Similarity measure and instance selection for collaborative filtering, International Journal of Electronic Commerce 8 (4) (2004) 115–129.
[34] C.-N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International World Wide Web Conference (WWW '05), 2005, pp. 22–32.