Integrating Context Similarity with Sparse Linear Recommendation Model

Yong Zheng(✉), Bamshad Mobasher, and Robin Burke

School of Computing, Center for Web Intelligence, DePaul University,
243 South Wabash Ave, Chicago, IL 60604, USA
{yzheng8,mobasher,rburke}@cs.depaul.edu

Abstract. Context-aware recommender systems extend traditional recommender systems by adapting their output to users' specific contextual situations. Most of the existing approaches to context-aware recommendation involve directly incorporating context into standard recommendation algorithms (e.g., collaborative filtering, matrix factorization). In this paper, we highlight the importance of context similarity and make the attempt to incorporate it into context-aware recommender. The underlying assumption behind is that the recommendation lists should be similar if their contextual situations are similar. We integrate context similarity with sparse linear recommendation model to build a similarity-learning model. Our experimental evaluation demonstrates that the proposed model is able to outperform several state-of-the-art context-aware recommendation algorithms for the top-N recommendation task.

Keywords: Context · Context-aware recommendation · Context similarity

1 Introduction and Background

Recommender systems are an effective way to alleviate information overload in many application areas by tailoring recommendations to users' personal preferences. Context-aware recommender systems (CARS) emerged to go beyond user preferences and also take into account the contextual situation of users in generating recommendations.

The standard formulation of the recommendation problem begins with a two dimens-ional (2D) matrix of ratings, organized by user and item: *Users* × *Items* → *Ratings*. The key insight of context-aware recommender systems is that users' preferences for items may be a function of the context in which those items are encountered. Incorporating contexts requires that we estimate user preferences using a multidimensional rating function – R: *Users* × *Items* × *Contexts* → *Ratings* [1].

Context-aware recommendation algorithms, such as context-aware matrix factorization [2], seeks to incorporate context into the traditional recommendation framework without considering the relationships among contextual conditions. In this paper, we position the importance of context similarity, where

F. Ricci et al. (Eds.): UMAP 2015, LNCS 9146, pp. 370–376, 2015.
DOI: 10.1007/978-3-319-20267-9_33

the assumption behind is that the recommendation lists should be similar if their contextual situations are similar. Furthermore, we propose the Similarity-Learning Model (SLM) which attempts to integrate context similarity with the sparse linear recommendation model. We explore different ways to represent and model the context similarity in order to build SLM and evaluate these methods using multiple data sets.

2 Similarity-Based Contextual Recommendation

In this paper, we use *contextual dimension* to denote the contextual variable, e.g. "Time". The term *contextual condition* refers to a specific value in a dimension, e.g. "weekday" and "weekend" are two conditions for "Time". A *context* or *contextual situation* is, therefore, a set of contextual conditions, e.g. {*weekend, home, family*}.

We consider the context-aware recommendation task to be a top-N recommendation problem, where a ranked list of items should be recommended to a user in specific contextual conditions (i.e., the input is a pair <user, context>). Therefore, the recommendation list should be updated accordingly if the contextual conditions are changed. Previous work [7] has incorporated contextual deviation into the sparse linear recommendation model and demonstrated its effectiveness. In this paper, we switch to integrate context similarity with the sparse linear method, and propose the similarity-learning model (SLM) which avoids similarity calculation by learning context similarity directly while optimizing the ranking score in the algorithm. Previous research has found that similarity calculations relying on existing ratings, yield unreliable results if the rating space is sparse. In SLM, simply, a ranking score prediction function for <user, item, context> is required in which the similarity between contexts is an integral part. The algorithm learns context similarity by minimizing the ranking score error and provides contextual recommendations to the end user based on the predicted ranking score.

In this paper, we choose the Sparse LInear Method (SLIM) [4] as our base algorithm, where SLIM is a ranking-based recommendation algorithm aggregating users' ratings with item coefficients. And the General Contextual Sparse LInear Method (GCSLIM) [7] is our previous work which integrates the contextual deviations. Alternatively, we try to modify the GCSLIM and replace the rating deviation term by the context similarity in this work. More specifically, the goal in GCSLIM is to create a prediction function shown in Equation 1 to estimate the ranking score, $\widehat{S}_{i,j,k}$, for u_i on item t_j in contexts c_k. t_h is an item in the set of items rated by u_i ($h \neq i$), and c_m is the contextual situation where u_i placed rating on t_h (note: it is allowed that $c_m = c_k$). $R_{i,h,m}$ is one contextual rating placed by u_i on t_h under context c_m. Assume there are L contextual dimensions in total, thus $c_{m,l}$ denotes the contextual condition in the l^{th} dimension in context c_m. The function Dev measures the contextual rating deviation between two contextual conditions – it is zero if $c_{m,l} = c_{k,l}$. Meantime, $W_{j,h}$ measures the coefficient between item t_j and t_h. Simply, GCSLIM learns

the coefficients between items and the contextual rating deviations, where the optimization goal is to minimize the ranking score prediction error and the loss function can be generated accordingly by using gradient descent [7].

$$\widehat{S}_{i,j,k} = \sum_{\substack{h=1 \\ h \neq i}}^{N} (R_{i,h,m} + \sum_{l=1}^{L} Dev(c_{m,l}, c_{k,l}))W_{j,h} \tag{1}$$

Inspired by GCSLIM, it is also possible to incorporate the notion of context similarity into the model to formulate the SLM by replacing the deviation term in Equation 1:

$$\widehat{S}_{i,j,k} = \sum_{\substack{h=1 \\ h \neq j}}^{N} R_{i,h,m} \times Sim(c_k, c_m) \times W_{j,h} \tag{2}$$

In SLM, we aggregate the ranking score by the contextual rating score $R_{i,h,m}$ with the similarity between c_k and c_m multiplying by the coefficient between item t_j and t_h. Note that we set $h \neq j$ to avoid bias by using u_i's other contextual ratings on t_j. This strategy will ensure that we learn the coefficients between as many different items as possible. The loss function and parameter updating rules can be generated accordingly by using gradient descent. However, the form of the similarity term will vary if the context similarity is represented in different ways. The remaining challenge is how to represent or model the similarity of contexts in the prediction function shown in Equation 2. In this paper, we introduce four ways to represent similarity of contexts.

2.1 Independent Context Similarity (ICS)

	Time=Weekend	Time=Weekday	Loc=Home	Loc=Cinema
Time=Weekend	1	0.54	N/A	N/A
Time=Weekday	0.54	1	N/A	N/A
Loc=Home	N/A	N/A	1	0.82
Loc=Cinema	N/A	N/A	0.82	1

Fig. 1. Example of a similarity matrix

Table 1. Sample of Contexts

	Time (W_T)		Location (W_L)	
	Weekend	Weekday	Home	Cinema
c_k	1	0	1	0
c_m	0	1	1	0

An example of a similarity matrix can be seen in Table 1. With ICS, we only measure the similarity between two contextual conditions when they lie on the same dimension. Each pair of contextual dimensions are assumed to be independent. Assuming there are L contextual dimensions in total, the similarity represented by ICS can be depicted by Equation 3, where l is the index of context dimension.

$$Sim(c_k, c_m) = \prod_{l=1}^{L} similarity(c_{k,l}, c_{m,l}) \tag{3}$$

These similarity values (i.e., $similarity(c_{k,l}, c_{m,l})$) can be learned by the optimization process in SLM. The risk of this representation is that some information may be lost, if context dimensions are not in fact independent, e.g., if users usually go to cinema to see romantic movies with their partners, the "Location" (e.g. at cinema) and "Companion" (e.g. partners) may have significant correlations as a result.

2.2 Latent Context Similarity (LCS)

As noted earlier, contextual rating data is often sparse, since it is unusual to have users rate items repeatedly within multiple situations. This poses a difficulty when new contexts are encountered. For example, the similarity between a *new pair* of contexts <"Time=Weekend", "Time=Holiday"> may be required in the testing set, but it may not have been learned from the training data, while the similarity for two *existing pairs*, <"Time=Weekend", "Time=Weekday"> and <"Time=Weekday", "Time=Holiday">, may have been learned. In this case, the representation in ICS suffers from the sparsity problem. Alternatively, we represent each contextual condition by a vector of weights over a set of latent factors (we use 5 latent factors in our experiments), where the weights are initialized at the beginning and learnt by the optimization process. Even if the newly observed pair does not exist in the training data, the vectors representing the two individual conditions (i.e., "Time=Weekend" and "Time=Holiday") will be learned over *existing pairs*, and the similarity for the *new pair* can be computed by the dot product of those two updated vectors. Then context similarity can be computed as in Equation 3. We call this approach the *Latent Context Similarity* (LCS) model, and we learn the vectors representing contextual conditions, which may increase computational costs in contrast to ICS.

2.3 Weighted Jaccard Context Similarity (WJCS)

Another approach to calculate context similarity is to associate weights with each dimension. An example is shown in the Table 1, where weight W_T is associated with the dimension *Time*, and W_L is assigned to *Location*. This representation of contextual similarity is an adaptation of the differential context weighting algorithm [5].

The weight is used as part of similarity computation only when the conditions within a dimension match. The similarity can be calculated by Equation 4, where l denotes the index of contextual dimension. Given the example in Table 1, only the conditions in *Location* are the same, therefore, the similarity of c_k and c_m is calculated as $\frac{W_L}{W_T+W_L}$.

$$Sim(c_k, c_m) = \frac{\sum_{l \in c_{k,l} \cap c_{m,l}} W_l}{\sum_{l \in c_{k,l} \cup c_{m,l}} W_l} \tag{4}$$

When all the conditions in the same dimension are different, we assign a small fixed value (e.g., 0.01) to the similarity to avoid the zero values being used in the prediction function. In WJCS, the weights for each contextual dimension are the ones to be learned in the optimization process. The computational cost is correlated with the number of contexts in the data set – the more dimensions, the more weights to be learned. However, the number of contextual dimensions can be reduced by a pre-selection process.

2.4 Multidimensional Context Similarity (MCS)

In the multidimensional context similarity model, we assume that contextual dimensions form a multidimensional coordinate system. An example is depicted in Figure 2.

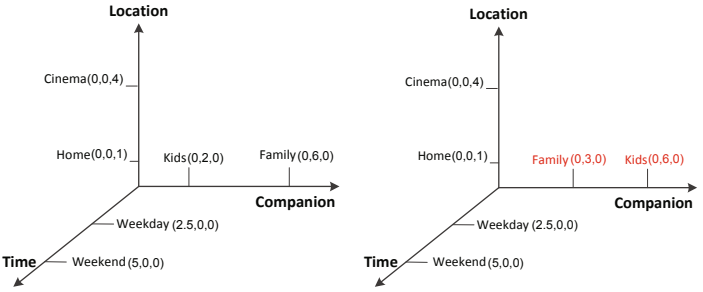

Fig. 2. Example of Multidimensional Coordinate System

Let us assume that there are three contextual dimensions: time, location and companion. We assign a real value to each contextual condition in those dimensions, so that each condition can locate a position in the corresponding axis. In this case, a contextual situation can be viewed as a point in the multidimensional space. Accordingly, the distance between two such points can be used as the basis for a similarity measure. In this approach, the real values for each contextual condition are the parameters to be learned in the optimization process. For example, the values for "family" and "kids" are updated in the right-hand side of the figure. Thus, the position of the data points associated to those two contextual conditions will be changed, as well as the distance between the corresponding two contexts. The similarity can be measured as the inverse of the distance between two data points. In our experiments, we use Euclidean distance to measure the distances, though other distance measures can also be used. The computational cost is directly associated with the number of contextual conditions in the data set, which may make this approach the highest-cost model. Again, the number of contextual conditions can be reduced by context selection, as with WJCS.

3 Experimental Evaluation

In our experiments, we use three context-aware data sets: the Restaurant, Music and Tourism data sets where the data descriptions can be found in [7]. We use a five-fold cross validation, performing top 10 recommendation task evaluated by precision and mean average precision (MAP). The algorithm proposed in this paper is build upon the GCSLIM; therefore, we choose the best performing deviation-based GCSLIM (denoted by "Deviation Model") [7] as

baseline. In addition, we compare our approach to the state-of-the-art context-aware recommendation algorithms, including context-aware splitting approaches (CASA) [6], context-aware matrix factorization (CAMF) [2] and tensor factorization (TF) [3]. We vary different parameters for those algorithms and present the best performing one in this paper, which is the same setting used in [7].

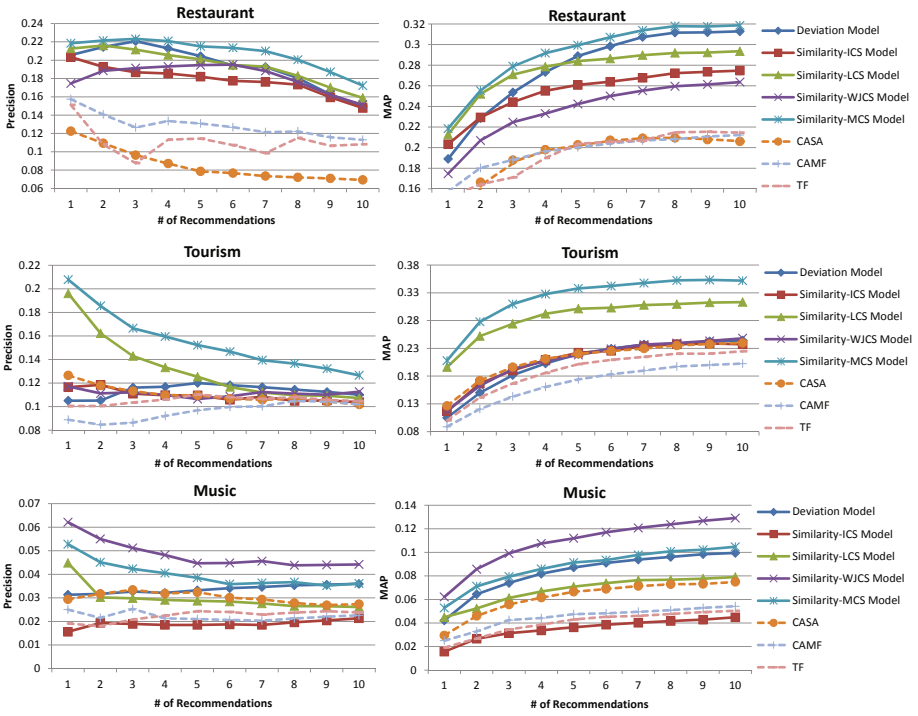

Fig. 3. Results in Precision and MAP @ Top-10 Context-aware Recommendations

The experimental results can be described by the Figure 3. We see that the MCS model outperforms all the baselines for the restaurant and tourism data, and the WJCS model is the best similarity model for the music data set. Note that the LCS model usually outperforms the ICS model. We can conclude that the latent factor approach is able to alleviate the sparsity problem suffered in ICS and improve performance.

The WJCS model has inconsistent performance. It is worse than the deviation-based baseline on the restaurant data, achieves comparable results to the deviation model with the tourism data, and is the best performer on the music data. We would expect the performance of the WJCS model to be influenced by a number of factors. The first is the number of contextual dimensions – more dimensions, more weights to be learned. This requires the rating profiles to be dense enough so that the weights for each dimension can be fully learned.

The second factor is the density of contextual ratings inferred by the data statistics reported in [7]. Based on those two factors, we can see that there are many more dimensions in the Tourism data, and the density in the restaurant data is pretty low, which explains the poor performance of WJCS model in these two data sets.

The multidimensional model seems to be an excellent representation. However, the computational cost is directly correlated with the number of conditions in this model. Context selection should be performed in preprocessing to alleviate this problem.

4 Conclusions and Future Work

We highlighted the notion of context similarity and incorporated it into the recommendation algorithm. We developed similarity-learning models in which contextual similarity is learned by optimizing the ranking score for top-N recommendations. We have shown that similarity of contexts can be represented in a variety of ways and our multidimensional context similarity approach outperforms other state-of-the-art baselines.

References

1. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Magazine **32**(3), 67–80 (2011)
2. Baltrunas, L., Ludwig, B., Ricci, F.: Matrix factorization techniques for context aware recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 301–304. ACM (2011)
3. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 79–86. ACM (2010)
4. Ning, X., Karypis, G.: SLIM: sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining, pp. 497–506. IEEE (2011)
5. Zheng, Y., Burke, R., Mobasher, B.: Recommendation with differential context weighting. In: Carberry, S., Weibelzahl, S., Micarelli, A., Semeraro, G. (eds.) UMAP 2013. LNCS, vol. 7899, pp. 152–164. Springer, Heidelberg (2013)
6. Zheng, Y., Burke, R., Mobasher, B.: Splitting approaches for context-aware recommendation: an empirical study. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 274–279. ACM (2014)
7. Zheng, Y., Mobasher, B., Burke, R.: Deviation-based contextual SLIM recommenders. In: Proceedings of the 23rd ACM Conference on Information and Knowledge Management, pp. 271–280. ACM (2014)