

Differential Context Modeling in Collaborative Filtering*

Yong Zheng, Robin Burke, Bamshad Mobasher
Center for Web Intelligence, School of Computing, DePaul University
243 South Wabash Ave, Chicago, Illinois 60604
{yzheng8, rburke, mobasher}@cs.depaul.edu

Abstract

Context-aware recommender systems (CARS) try to adapt their recommendations to users’ specific contextual situations. In many recommender systems, particularly those based on collaborative filtering (CF), the additional contextual constraints may lead to increased sparsity in the user preference data, thus fewer matches between the current user context and previous situations. Our earlier work proposed two approaches to deal with this problem – differential context relaxation (DCR) and differential context weighting (DCW) and we have successfully examined them using user-based collaborative filtering (UBCF). In this paper, we put DCR and DCW into one framework called differential context modeling (DCM). As a general framework, DCM is able to be applied to other recommendation algorithms other than UBCF. We expand the application of DCM to the other two CF approaches: item-based CF and slope one recommender. Predictive performances are evaluated based on two real-world data sets and experimental results demonstrate that applying DCM to those two algorithms is able to improve predictive accuracy compared with our baselines: context-free CF algorithms and contextual pre-filtering algorithms.

1 Introduction

Context-aware recommender systems (CARS) have been demonstrated to be able to improve the performance of recommendation in many recommendation tasks, such as travel accommodation [1, 2], food menus [3, 4], and movie recommendation [5, 6]. Traditional recommendation problem can be modeled as a two-dimensional (2D) prediction – \( R: \text{Users} \times \text{Items} \rightarrow \text{Ratings} \), where the recommender system’s task is to predict that user’s rating for that item. Context-aware recommender systems try to additionally incorporate contexts to estimate user preferences, which turns the prediction into a “multidimensional” rating function – \( R: \text{Users} \times \text{Items} \times \text{Contexts} \rightarrow \text{Ratings} \) [7], where context is defined as “any information that can be used to characterize the situation of an entity” [8].

Sparsity of contexts is a common issue in CARS, especially for those recommendation algorithms based on collaborative filtering. For example, in the movie domain, “companion” is clearly an important contextual variable, but “time of day” perhaps is not. Although additional contextual information fine-tunes the recommendation process, every variable included also fragments the data, with the result that most researchers tend to stick with a single contextual variable unless their data is particularly dense. This suggests that ratings for context \( c_1 \) may be of limited value when predicting for context \( c_2 \). However, if we partition all ratings by their contexts, the purpose of collaborative recommendation is defeated – no two individuals are in exactly the same situation and so prior ratings cannot be used to make predictions.

*SOCRS-2013, May 31, Chicago, IL USA
Copyright is held by the author or owner(s).
In our previous work, we proposed two approaches – *differential context relaxation* (DCR) [2, 4] and *differential context weighting* (DCW) [6] to deal with this problem, and they have been successfully evaluated with the user-based collaborative filtering (UBCF) algorithm. Our previous experimental results show that DCR and DCW are able to alleviate the sparsity of contexts by context relaxation and context weighting respectively, and they can provide more accurate predictions even the data sets are sparse in contexts. In this paper we put those two approaches into one framework – *differential context modeling* (DCM). As a general context-aware framework, DCM is able to be applied to other recommendation algorithms, where we evaluate its application to item-based collaborative filtering and slope one recommender to demonstrate its applicability in CARS.

2 Related and Prior Work

Adomavicius, et al. [7] introduce a two-part classification of contextual information. Their taxonomy is based on two considerations: what a recommender system knows about a contextual factor and how the contextual factor changes over time. System knowledge can be subdivided into *fully observable*, *partially observable* and *unobservable*. The temporal aspect of a factor can be categorized as *static* or *dynamic*. This analysis yields six possible classes for contextual factors, which characterizes possible contextual situations in the applications of CARS. In this paper, we are concerned with *static* and *fully observable* factors – we have a set of known potential contextual variables at hand which remain stable over time.

Existing solutions for the sparsity of contexts can be summarized into three branches: context selection [9, 10, 11], context relaxation [12, 2, 4, 13] and context weighting [6]. Context selection is usually performed based on surveys or statistical analysis; however, it relies on extra work and the data may be sparse, which makes the statistics unreliable. Context relaxation and context weighting were developed by us via two approaches – *differential context relaxation* (DCR) where contexts are relaxed for each algorithm component, and *differential context weighting* (DCW) where contexts are weighted instead of relaxed.

In this paper, we put DCR and DCW into one framework called *differential context modeling* (DCM). The idea behind DCM includes two parts: “*differential*” and “*modeling*”, where the “*differential*” part tries to separate algorithms into different functional components and each component should be applied with optimal contextual constraints so that the contextual effect for each component can be maximized; meanwhile, the “*modeling*” part can be performed by either context relaxation or context weighting as described above and the purpose of modeling is to find optimal context relaxation or weighting solutions to keep a balance between maximizing contextual effects and also alleviating the problem of sparsity of contexts.

<table>
<thead>
<tr>
<th>User</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>Weekend</td>
<td>Home</td>
<td>Girlfriend</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>Weekday</td>
<td>Home</td>
<td>Girlfriend</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>Weekday</td>
<td>Theater</td>
<td>Sister</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>Weekday</td>
<td>Home</td>
<td>Sister</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 1: Example: users’ contextual ratings for movie *Titanic*

Take the movie recommendation as an example shown in Table 1. If we want to predict whether U1 will like the movie *Titanic* at home (Location) with his sister (Companion) on a weekday
(Time), we could use only peers who had rated the movie under exactly the same circumstances – but maybe there are no neighbors that meet this constraint – as shown in Table 1, no users rated *Titanic* in such contexts. Context relaxation is to use a relaxed contextual constraints, e.g. relaxed contexts as \{weekday, sister\} or \{weekday, home\}. Or, there would be more matched ratings if we just consider one dimension, e.g. \{weekday\}. Context weighting is a finer-grained solution compared with context relaxation – it assigns weights to different dimensions and uses the notion of similarity of contexts to select influential rating profiles for further calculations. The assumption at background is that more similar the contexts of two ratings were given, the more valuable those ratings will be useful in making predictions. The contextual conditions are not required to be exactly matched, which further alleviates the problem of sparsity contexts. For example, the contexts for the first and fourth records in Table 1 only have one dimension matched – the location is all “home”, but that’s enough if the dimension “location” is very influential – in other words, it is significantly weighted higher than other dimensions, which results in a higher value of similarity of contexts. Thus the remaining work is to find the optimal context relaxation or context weighting solutions, we have successfully introduced particle swarm optimizers (PSO) as efficient solutions in our previous work [4, 6].

3 Collaborative Filtering Algorithms

Collaborative filtering (CF) has been demonstrated as one of the most successfully algorithms in traditional recommender systems, where it can be categorized into memory based CF (such as neighborhood-based CF [14, 15]), model-based CF (such as matrix factorization [16]) and hybrid algorithms which fuse different kinds of recommenders [17]. To adapt context-aware recommendations, context has been introduced to CF, including both neighborhood-based CF (NBCF) and matrix factorization (MF). Our previous work has evaluated DCM on user-based CF (UBCF). In this paper, we focus on NBCF and introduce our DCR and DCW techniques to them. More specifically, we apply DCR and DCW to two more NBCF: item-based collaborative filtering (IBCF) and slope one recommender which can be formalized as follows.

3.1 Item-based Collaborative Filtering

Item-based collaborative filtering (IBCF) [15] can be represent in Equation 1, Where $N_i$ is the nearest neighborhood for item $i$. And we use the adjusted cosine similarity as the similarity function $\text{sim}(i, j)$ which is shown in Equation 2, Where $N$ is a list of users who have rated item $i$ and item $j$ on profiles. IBCF assumes user $u$’s rating on item $i$ is similar to $u$’s ratings on the neighbors of item $i$, where the neighbors are found by the item-item similarity function.

$$P_{a,i} = \frac{\sum_{j \in N_i} r_{a,j} \times \text{sim}(i, j)}{\sum_{j \in N_i} \text{sim}(i, j)} \quad (1) \quad \text{sim}(i, j) = \frac{\sum_{u \in N} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in N} (r_{u,i} - \bar{r}_u)^2 \sum_{u \in N} (r_{u,j} - \bar{r}_u)^2}} \quad (2)$$

3.2 Slope One Recommender

Slope one recommender [18] can be considered as a simple item-based CF, but it can only be applied to rating-based data sets compared with UBCF and IBCF. It is a memory-cost algorithm
which does not require the data set to be dense – it can make predictions even if users have limited ratings. Usually a weighted slope one recommendation algorithm can be described as follows:

\[ D_{i,j} = \sum_{u \in S_{i,j}(T)} \frac{r_{u,i} - r_{u,j}}{\text{card}(S_{i,j}(T))} \]  

\[ P_{u,i} = \frac{\sum_{j \in I(u) - \{i\}} (D_{i,j} + r_{u,j}) \times \text{card}(S_{i,j}(T))}{\sum_{j \in I(u) - \{i\}} \text{card}(S_{i,j}(T))} \]

Where \( D_{i,j} \) denotes the rating deviation between two items \( i \) and \( j \). \( S_{i,j}(T) \) is the set of users who have rated items \( i \) and \( j \), where \( T \) is the training set. \( \text{card}(S_{i,j}(T)) \) represents the size of this set of users. \( I(u) \) is the set of items user \( u \) have rated; \( I(u) - \{i\} \) excludes the item \( i \) from this set. It calculates the rating deviation between the ratings of items for each user (for every item the user has rated). Then, it creates and averages the difference for every pair of items. To make a prediction of the item \( i \) for an user \( u \) for example, it would get the ratings that \( u \) has given to other items and add to it the difference between each item. With this we could obtain an average and the final prediction function is as shown in Equation 4.

4 Differential Context Modeling

With the two recommendation algorithms described as above, we can formulate the application of DCR and DCW to IBCF and slope one recommender.

4.1 Differential Context Relaxation

To introduce DCR, let us define a context \( c \) as a vector of values \( \langle f_1, f_2, ..., f_n \rangle \), one for each contextual variable known to a recommendation application. Let \( s \) be a binary vector \( \langle s_1, s_2, ..., s_n \rangle \). The projection function \( \pi(c, s) \) projects \( c \) to a smaller set of contextual features by applying \( s \). The contextual value \( f_i \) is included in the projection if \( s_i = 1 \). Two contextual profiles in two ratings, \( c \) and \( d \), match subject to the contextual constraint \( s \) if \( \pi(c, s) = \pi(d, s) \). A relaxation of \( s \), \( s' \) is defined as any vector containing fewer values of 1. Contexts that match under the constraint \( s \) will also match under any relaxed version of \( s \). The model applying DCR to IBCF can be represent by Equation 5 and 6:

\[ P_{u,i,c} = \frac{\sum_{j \in N_{C_1}} r_{u,j,C_2} \times \text{sim}_c(i, j, C_3)}{\sum_{j \in N_{C_1}} \text{sim}_c(i, j, C_3)} \]

\[ \text{sim}_c(i, j, C_3) = \frac{\sum_{u \in N} (r_{u,i,C_3} - \bar{r}_{u,C_3})(r_{u,j,C_3} - \bar{r}_{u,C_3})}{\sqrt{\sum_{u \in N} (r_{u,i,C_3} - \bar{r}_{u,C_3})^2 \sum_{u \in N} (r_{u,j,C_3} - \bar{r}_{u,C_3})^2}} \]

There are three components involved in IBCF:

1. **Neighborhood selection.** The original neighborhood selection component of Equation 1 selects the \( k \) nearest neighbors from the items this user rated before, subject to a minimum similarity threshold. In DCR, we select only item neighbors as \( N_{C_1} \) who was rated in a context matching \( c \) under relaxation \( C_1 \).
2. Neighbor contribution. Similarly, \( r_{a,j,C_2} \) is computed using a filtered set of ratings by the constraint \( C_2 \).

3. Item similarity. According to Equation 6, in DCR we will consider only ratings where \( \pi(d, C_3) = \pi(e, C_3) \), where \( d \) and \( e \) are the two contextual situations. In other words, when comparing two rating profiles, we require that the ratings be issued in matching contexts relative to \( C_3 \).

Compared with UBCF and IBCF, slope one recommender is less complex, as it can be decomposed into only two components followed by the model formula in Equation 7:

1. Neighborhood selection. A contextual slope one recommender may select item candidates from the ones this user have rated in a matched context by the constraint \( C_1 \).

2. Rating Deviation. The deviation is usually calculated based on ratings from \( < u, i > \) and \( < u, j > \) pairs; additionally, a contextual constraint \( C_2 \) can be added so that it restricts the pairs should be rated in a matched context as shown in Equation 8.

\[
P_{u,i,c} = \frac{\sum_{j \in I(u,C_1) - \{i\}} (Dev_{i,j,C_2} + r_{u,j,C_1}) \times card(S_{i,j,C_2}(T))}{\sum_{j \in I(u,C_1) - \{i\}} card(S_{i,j,C_2}(T))} (7)
\]

Where the rating deviation can be represent as follows.

\[
Dev_{i,j,C_2} = \sum_{u \in S_{i,j,C_2}(T)} \frac{r_{u,i,C_2} - r_{u,j,C_2}}{card(S_{i,j,C_2}(T))} (8)
\]

4.2 Differential Context Weighting

Similarly, it is also feasible to apply DCW to both IBCF and the slope one recommender. Differs from DCR, DCW tries to further alleviate the sparsity problem and provide a finer-grained modeling approach by introducing contextual weighting and similarity of contexts. More specifically, the idea behind it is the assumption that more similar the contexts given by two ratings, they are more influential in making predictions. As a result, it is not required to select or relax contextual dimensions and match them exactly; instead, all dimensions could be included, and a weighted Jaccard similarity measure [6] was proposed where contextual weights were assigned to each dimension for each algorithm component. In other words, we keep the algorithm components the same as the ones in DCR, and we introduce similarity of context and set similarity thresholds for each component in order to filter out ratings with not similar enough contexts. See more details in our previous work [6].

The model applying DCW to IBCF can be represent by Equation 9, and Equation 10 provides the similarity formulation.

\[
P_{a,i,\sigma} = \frac{\sum_{j \in N_i,\sigma_1,\epsilon_1} r_{a,j,\sigma_2,\epsilon_2} \times sim_w(i, j, \sigma_3, \epsilon_3)}{\sum_{j \in N_i,\sigma_1,\epsilon_1} sim_w(i, j, \sigma_3, \epsilon_3)} (9)
\]
\[
\text{sim}_w(i, j, \sigma_3, \epsilon_3) = \frac{\sum_{(u,c,d) \in T_{\epsilon_3}} (r_{u,i,c} - \bar{r}_u)(r_{u,j,d} - \bar{r}_u)J(c, d, \sigma)}{\sqrt{\sum (r_{u,i,c} - \bar{r}_u)^2 \sum (r_{u,j,d} - \bar{r}_u)^2 \sum_{(u,c,d) \in T_{\epsilon_3}} J(c, d, \sigma)^2}}
\]

There are \( \sigma \) and \( \epsilon \) variables in Equation 9, where \( \sigma_k \) denotes a weighting vector which is filled with real numbers, and \( k \) is the component index. \( \epsilon_k \) is a real number indicating the similarity threshold – only ratings with a higher similarity than this threshold can be included in the calculation of this component. For the item similarity component, we weight each comparison between ratings. We create the set \( T_{\epsilon_3} \) (as Equation 11) by collecting all users \( u \) and pairs of contexts \( c \) and \( d \) who have rated items \( i \) and \( j \), respectively, such that items were rated in that context and \( J(c, d, \sigma) > \epsilon_3 \), where \( J(c, d, \sigma) \) is the same weighted Jaccard similarity measure in [6].

\[
T_{\epsilon_3} = \{(u, c, d) \ni \exists r_{u,i,c}, r_{u,j,d} \land J(c, d, \sigma) > \epsilon_3\}
\]

Similarly, applying DCW to slope one recommender can be formulated by Equation 12.

\[
P_{u,i,\sigma} = \frac{\sum_{j \in I(u,\sigma_1,\epsilon_1) - \{i\}} (Dev_{i,j,\sigma_2,\epsilon_2} + r_{u,j,\sigma_1,\epsilon_1}) \times \text{card}(S_{i,j,\sigma_2,\epsilon_2}(T))}{\sum_{j \in I(u,\sigma_1,\epsilon_1) - \{i\}} \text{card}(S_{i,j,\sigma_2,\epsilon_2}(T))}
\]

Where the rating deviation can be described as follows:

\[
Dev_{i,j,\sigma_2,\epsilon_2} = \sum_{u \in S_{i,j,\sigma_2,\epsilon_2}(T)} \frac{r_{u,i,\sigma_2,\epsilon_2} - r_{u,j,\sigma_2,\epsilon_2}}{\text{card}(S_{i,j,\sigma_2,\epsilon_2}(T))}
\]

The specific operations are similar as the process applying DCW to IBCF. Compared with DCR, the filtering process in DCW is not controlled by the context relaxation and matching – DCW uses the weighted Jaccard similarity measure and the similarity threshold to determine whether this rating should be included or filtered out for further calculations in each component.

### 4.3 Optimization

Seeking optimal relaxation and weighting vectors as well as relevant optimal parameters (e.g. the optimal similarity of contexts) for DCR and DCW is the remaining work. Our previous work [4, 6] has demonstrated the efficiency of particle swarm optimization (PSO) as an optimizer, where binary PSO provides an optimal solution for DCR and constriction-factor PSO optimizes DCW. In this paper, we continue to use them as the optimization solutions in our experiments for evaluations.

### 5 Experiments

We continue to use the same **AIST Food Data** and **Movie Data** in our previous work [6]. A brief description of the data sets is described in Table 2, where the food data is dense with contextual ratings, and the movie data is sparse with contextual information. More specifications about the contexts in the data sets, see [6].

The predictive performance was measured by root-mean-square error (RMSE) evaluated using 5-fold cross validation. We also measured coverage for each evaluation run, measured as the
Table 2: Description ofDatasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of users</th>
<th># of items</th>
<th># of ratings</th>
<th># of dimensions</th>
<th>Rating Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food Data</td>
<td>212</td>
<td>20</td>
<td>6360</td>
<td>6</td>
<td>1 - 5</td>
</tr>
<tr>
<td>Movie Data</td>
<td>69</td>
<td>176</td>
<td>1010</td>
<td>5</td>
<td>1 - 13</td>
</tr>
</tbody>
</table>

percentage of predictions for which we can find at least one neighbor. We use a threshold \( \epsilon \) to guarantee a minimum degree of coverage. The reason for this is to avoid a solution that works well in terms of RMSE but only fits a limited number of users. The reason why we do not evaluate models on precision or recall is that the datasets are relatively small and collected from surveys – it is possible users were required to rate all items. Previous research \([3, 5, 19]\) on these two data sets used prediction errors, also. To simplify the optimization in DCW, we use the same threshold values for the similarity of contexts, i.e. all \( \epsilon \) values are the same, and we iterated this threshold value from 0.0 to 1.0 with 0.1 increment in each step.

5.1 Predictive Performances

We choose two kinds of baselines – context-free recommendation algorithms (i.e. standard IBCF and standard slope one recommender) and contextual pre-filtering approach which is a popular context-aware algorithms – it only applied contextual constraints to the neighbor selection component in collaborative filtering. The comparison of experimental results can be shown by Figure 1 and Figure 2.

Figure 1: Application of DCR and DCW to IBCF

Figure 1 provides the results based on IBCF. DCW works the best in terms of RMSE, where DCR works better than standard IBCF but worse than contextual pre-filtering. Note that pre-filtering works better than DCR at price of coverage, because filtering operation usually hurts coverage if the constraint is too strict. From the perspective of RMSE and coverage together, DCW works the best, where it helps achieve the lowest RMSE with a decent coverage compared with the standard IBCF algorithm.

\[ \text{This threshold value is empirically selected based on the coverage values by context-free CF algorithms. It is because we hope DCR and DCW can provide more accurate predictions within an acceptable decrease in coverage. Thus this value differs from data to data and they are also dependent with specific recommendation algorithms used in the experiments.} \]
Similar patterns can be found in Figure 2 where DCR and DCW were applied to slope one recommender. DCW works the best if we consider RMSE and coverage together. DCR works better than standard slope one and pre-filtering. Pre-filtering even works worse than the standard slope one recommender for the movie data.

In a summary, DCW performs the best among context-free algorithm, contextual pre-filtering and DCR when it is applied to three NBCF algorithms: IBCF, slope one recommender and UBCF which was demonstrated in [6]. The performance of DCR depends on the specific data set and which recommendation algorithms will be applied to. This conclusion is consistent with our conjectures in our previous work [6] – DCW is considered as a finer-grained DCM solution which is able to compensate the drawbacks of DCR.

5.2 Performance of Optimizers

We use BPSO and CFPSO as the optimizers for DCR and DCW respectively. We try different number of particles (such as 1, 3, 5, 10, 15, 20) in our experiments. The performance is measured by the usage of running environment to reach the optimal solution, including three factors: the number of required iteration, the number of required particle, and the running time in seconds. Comparisons of running performances can be shown in Table 3 and Table 4.

Table 3: Comparison of Running Performances Between BPSO and PSO (IBCF)

<table>
<thead>
<tr>
<th></th>
<th>Food Data</th>
<th></th>
<th>Movie Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iteration</td>
<td>Time (s)</td>
<td># of particles</td>
<td>Iteration</td>
</tr>
<tr>
<td>DCR via BPSO</td>
<td>54</td>
<td>14.8</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>DCW via PSO</td>
<td>10</td>
<td>30.0</td>
<td>5</td>
<td>38</td>
</tr>
</tbody>
</table>

Table 4: Comparison of Running Performances Between BPSO and PSO (Slope One)

<table>
<thead>
<tr>
<th></th>
<th>Food Data</th>
<th></th>
<th>Movie Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iteration</td>
<td>Time (s)</td>
<td># of particles</td>
<td>Iteration</td>
</tr>
<tr>
<td>DCR via BPSO</td>
<td>2</td>
<td>58.6</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>DCW via PSO</td>
<td>72</td>
<td>775.8</td>
<td>3</td>
<td>19</td>
</tr>
</tbody>
</table>
Typically, more particles used in experiment may help find the optimum in less iteration but the running time will be also increased. Thus there should be a balance between the running time and the number of convergence iteration. From the two tables above, we can see that generally DCW may cost more time, it is because DCW requires more calculations. Our experiments find that the run time performance of DCW via PSO depends on three factors: one is the number of contextual dimensions used – more dimensions, more parameters require to be learned; the second one is the density of contextual ratings – more dense, the weighting calculations require more time; the last one is which algorithm is used in the experiment. In Table 4, DCW requires much more time to converge for the food data, it is because the food data has more contextual dimensions and it is very dense data, where slope one recommender performed much more calculations to construct the rating deviation values in the memory, thus the running time is significantly increased for the food data when slope one recommender is applied.

6 Conclusions

DCM as a general context-aware framework can be applied to other recommendation algorithms other than UBCF which was examined in our previous work. In this paper, we extend it to IBCF and slope one recommender. Our experiments show that DCW typically works the best but may bring more costs, especially when it comes to a data set with dense contextual ratings, more contextual dimensions and being applied to slope one recommender. In our future work, it is necessary to explore practical solutions to let DCW work efficiently on large and dense data sets. One possible solution is to parallel the PSO process; another idea is to put the DCW model against Apache Hadoop * using the MapReduce strategy.

References


*http://hadoop.apache.org/


