Recommendation with Differential Context Weighting

Yong Zheng, Robin Burke, and 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) adapt their recommendations to users' specific situations. In many recommender systems, particularly those based on collaborative filtering, the contextual constraints may lead to sparsity: fewer matches between the current user context and previous situations. Our earlier work proposed an approach called *differential context relaxation* (DCR), in which different subsets of contextual features were applied in different components of a recommendation algorithm. In this paper, we expand on our previous work on DCR, proposing a more general approach – *differential context weighting* (DCW), in which contextual features are weighted. We compare DCR and DCW on two real-world datasets, and DCW demonstrates improved accuracy over DCR with comparable coverage. We also show that particle swarm optimization (PSO) can be used to efficiently determine the weights for DCW.

Keywords: recommender systems, collaborative filtering, context, context-aware recommendation.

1 Introduction

Researchers in the domain of recommender systems started to realize that it is useful to take contexts into account when making recommendations. Context-aware recommender systems (CARS) have shown improved accuracy in many recommendation tasks, such as travel accommodation [22,14,29], food menus [24,30], movie recommendation [3,13], music recommendation [25,20], and mobile applications [28,7].

The fundamental assumption of CARS is that a rating for an item is a function not just of the user and item but also of the context in which the item is evaluated or used. This suggests that ratings for context c_1 may be of limited value when predicting for context c_2. However, if we partition all of our ratings by their contexts, the purpose of collaborative recommendation is defeated – no two individuals are ever in exactly the same situation and so prior ratings cannot be used to extrapolate future preferences.

Applying contexts in recommendation is therefore a matter of separating those aspects of the context that are relevant to the recommendation from those that are not. In the movie example, "companion" is clearly an important contextual variable, but "time of day" perhaps is not. Selecting contextual variables is clearly a blunt instrument. 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.

S. Carberry et al. (Eds.): UMAP 2013, LNCS 7899, pp. 152–164, 2013.

Therefore, how to select contextual variables, and how to use and apply selected contexts in recommendation algorithms are really serious problems, especially when contextual information is not dense in the data. In our previous research, we sought to escape from the dilemma of contextual variable selection through *differential context relaxation* (DCR) [29,30]. DCR divides the recommendation algorithm into components and allows each part to treat context differently – this is the "differential" part. Rather than choose a priori among the possible contextual variables, we use an optimization approach to select the best set of variables for each component. Because we think of the contextual variables as constraints on the operation of the components, we consider this a matter of finding the optimal relaxation of the context.

Although DCR does allow a recommender to take context into account and is an improvement on global context selection, we found that in some cases, it did increase sparsity more than was desirable. In this paper, we show that the DCR approach can be extended to an algorithm – differential context weighting (DCW) in which the contribution of each contextual variable is weighted. Below, we introduce DCR and DCW and demonstrate the relative benefits of DCW in two real-world datasets.

Optimization is an important part of this research. The DCR algorithm requires an optimal set of contextual variables for each component; DCW requires an optimal set of weights for each contextual variables for each component. We show that particle swarm optimization can be used to solve this non-linear optimization problem for our sample datasets.

2 Related Work

Traditional recommendation problem can be modeled as a two-dimensional (2D) prediction – R: $Users \times Items \rightarrow Ratings$, 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: $Users \times Items \times Contexts \rightarrow Ratings$ [2].

Adomavicius, *et al.* [2] 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.

The problem of sparse contexts is a well-known one in CARS research. Users may rate items in different contexts, but it is not guaranteed that we can find dense contextual ratings under the same contexts, i.e. there may be very few users who have rated the items in the same contexts. The solutions for this problem can be categorized into two branches: *context selection* and *context relaxation*.

Context selection identifies the most influential contextual variables and then applies them into recommendation algorithms. There are several different means of discovering

influential contextual variables [6,27,15]. Odic *et al.* [5] summarized and categorized those approaches into the relevancy assessment from the user survey and the relevancy detection with statistical testing. However, each of these techniques has its drawbacks. Survey assessment requires a lot of user effort. Statistical testing are not reliable unless the data set is dense – items have been rated multiple times in different contexts.

The notion of context relaxation was first applied by De Pessemier *et al.* [9] in 2010. The researcher sought to alleviate sparisty by removing one of the contextual variables in a set of contextual features. We propose the different context relaxation model in our previous work [29,30]. DCR allows the context relaxation to be realized by looking at arbitrary subsets of the contextual variables instead of a single feature. More significantly, DCR applies this relaxation approach to multiple components of the recommendation algorithm using different relaxations for different components. In this paper, we propose DCW which weights contexts other than context selections or relaxations, and context weighting can be considered as a novel approach as another solution for the problem of context sparsity. To our best knowledge, contexts were not weighted in existing research on CARS.

Context-aware recommendation algorithms fall into three categories: contextual prefiltering, contextual post-filtering and contextual modeling [4] based on how and where context is applied in the recommendation process. As indicated by our previous work [29], DCR can be considered a hybrid context-aware modeling approach because contexts are applied in not only the neighborhood filtering process, but also the recommendation process; and DCW is similar. Our approach is unique in that contextual constraints are applied differentially across the algorithm components, rather than being fixed for the entire algorithm.

3 Differential Context Relaxation

As discussed above, the idea of DCR is that we treat a recommendation algorithm as a collection of functional components and apply context relaxation differently in each component. Furthermore, we treat the context as a set of constraints that the input to the component must satisfy. Those constraints can be relaxed to manage the balance between coverage and accuracy in recommendation.

For example, in movie recommendation as the example shown in Table 1, we might know where the user watched a film (in a theater vs at home), companion (solo, family, girlfriend, etc), and what day of the week. 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. This would amount to filtering the input to the neighborhood formation component using the context C. We might find that this produces too much sparsity – maybe there are no neighbors that meet this constraint – as shown in Table 1, no users rated *Titanic* in such contexts. It may be possible to form neighborhoods with a relaxed version of C, perhaps by dropping one of the variables to make the constraint less restrictive, 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*$\}$.

We take as our starting point the well-known Resnick's algorithm for kNN user-based collaborative filtering (uCF)recommendation as shown in Equation 1, where a is

Table 1. Example: users' contextual ratings for movie *Titanic*

User	Time	Location	Companion	Rating
U1	Weekend	Home	Girlfriend	4
U2	Weekday	Home	Girlfriend	5
U3	Weekday	Theater	Sister	4
U1	Weekday	Home	Sister	?

a user, i is an item, and N_a is a neighborhood of k users similar to a. The algorithm calculates $P_{a,i}$, which is the predicted rating that user a is expected to assign to item i.

$$P_{a,i} = \bar{r}_a + \frac{\sum\limits_{u \in N_a} (r_{u,i} - \bar{r}_u) \times sim(a,u)}{\sum\limits_{u \in N_a} sim(a,u)} \tag{1}$$

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.

Let c be the context for which a recommendation is sought, and let C_1, C_2, C_3, and C_4 be relaxations of the full set of contextual variables with corresponding selection vectors s_1, s_2, s_3, and s_4.[1] The algorithm components are as follows:

Neighborhood Selection. The original neighborhood selection component of Equation 1 selects the k nearest neighbors for user a with a rating on i, subject to a minimum similarity threshold. In DCR, we instead select only neighbors as N_{C_1} who have issued their ratings for i in a context matching c under relaxation C_1.

Neighbor Contribution. The computation of $\bar{r}_u$ in the original equation is replaced by the one in which this average baseline rating for user u is computed using a filtered set of ratings by the constraint C_2. We will denote this filtered version of the average with the notation $\bar{r}_{u,C_2}$, and subtract it from the user's rating r_{u,i,C_2} to calculate the contribution of this neighbor towards the prediction.

User Baseline. The computation of $\bar{r}_a$ in the original equation we replace with the one using the filtered ratings of the user a as above: $\bar{r}_{a,C_3}$.

User Similarity. The computation of neighbor similarity $sim(a, u)$ involves identifying ratings $r_{u,i}$ and $r_{a,i}$ where the users have rated items in common. For context-aware recommendation, we have ratings $r_{u,i,d}$ and $r_{a,i,e}$ instead. We will consider

[1] Note that our previous work [29,30] used only three components and did not examine contextual effects in the user similarity calculation. Our experiments show introducing context relaxation to this additional component can help improve prediction accuracy and also save coverage, it is because it helps discover better neighbors in CF.

only ratings where $\pi(d, C_4) = \pi(e, C_4)$. In other words, when comparing two rating profiles, we require that the ratings be issued in matching contexts relative to C_4. Unlike the other components, we do not require that d and e match the context c in which recommendations are being made. We will indicate this version of similarity with the function $sim_c(a, u, C_4)$ which is measured by Pearson correlation coefficient in our experiments.

Thus the four-component DCR model is described in Equation 2:

$$P_{a,i,c} = \bar{r}_{a,C_3} + \frac{\sum_{u \in N_{C_1}} (r_{u,i,C_2} - \bar{r}_{u,C_2}) \times sim_c(a, u, C_4)}{\sum_{u \in N_{C_1}} sim_c(a, u, C_4)} \quad (2)$$

DCR tries to find an optimal context relaxation for each component in recommendation algorithm. In this process, it aims to achieve a balance of alleviating the problem of sparse contexts and maximizing the contextual effect of each component. This approach has several drawbacks:

- Context relaxation is still a strict action – the algorithm must choose to include or exclude each variable relative to each component. It works well for dataset with dense contextual ratings, but in datasets with less contextual information, sparsity remains a problem.
- Algorithm components can be dependent. For example, the neighbor contribution component is dependent on neighbor selection: it is not guaranteed that neighbor u has ratings under C_2, because u is selected by a different constraint C_1. In this case, the model rolls back to the original component representation in uCF.

These considerations give rise to the idea of differential context weighting (DCW).

4 Differential Context Weighting

Binary selection can be considered a special case of weighting where only 0 and 1 values are permitted. DCW exploits this idea, using, instead of a selection vector s, a vector of weights σ, real values between 0 and 1. The weights are used to control the contribution of each contextual feature to the recommendation algorithm components in a manner similar to DCR. Feature weighted collaborative filtering is not novel, e.g. Ujjin *et al* [26] applied feature weighting to neighborhood-based collaborative filtering. This prior work did not include contextual features, however, and did not apply them differentially compared with our work.

In DCW, we introduce the weighting vectors and the notion of similarity of contexts to realize "differential" and "weighting". We are no longer concerned with filtering out certain ratings, but rather with assigning a score to all ratings based on context. DCW assumes that the more similar the contexts of two ratings were given, the more valuable those ratings will be in making predictions. Given a target context c, we need to assess how much to weight a rating $r_{u,i,d}$ issued in some different context d, subject

to a weighting vector σ. Our metric for the similarity of contexts is a simple one: the *weighted Jaccard* metric, J.

$$J(c, d, \sigma) = \frac{\sum_{f \in c \cap d} \sigma_f}{\sum_{f \in c \cup d} \sigma_f} \tag{3}$$

If we revisit the example shown in Table 1, we see that it is possible to use all three dimensions even if there are no users who have rated *Titanic* under the exact context $\{$*weekday, home, sister*$\}$. Because U3 rated *Titanic* under a partially-matching context $\{$*weekday, theater, sister*$\}$, it is possible to make use of this rating in making predictions for U1 based on the proposed Jaccard metric above – more similar the contexts of two ratings were given, these ratings are considered more reliable for further predictions. However, there is a limit to this effect: our experiments show that contexts with low similarity may add noise to the predictions. So, we use a set of similarity thresholds $\epsilon_1 ... \epsilon_4$ to filter ratings, for the each component. Context matches below the threshold are ignored. In addition, the weighting vectors assign different weights to each contextual dimension to indicate the power of influence instead of assuming all dimensions have equal effects.

As a result, DCW is supposed to be able to compensate the drawbacks of DCR – weighting is not as strict as relaxation because we can include more contextual ratings into calculations once it meets the minimal context similarity threshold, even if the algorithm components are dependent. With these preliminaries above, DCW is reformulated as Equation 4. The key parameters in this equation are the four σ vectors, one for each component, that weight the contribution of each contextual variable in that component, and the four ϵ values that set the threshold of context similarity in each component.

$$P_{a,i,\sigma} = \bar{\rho}(a, \sigma_3, \epsilon_3) + \frac{\sum_{u \in N_{a,\sigma_1,\epsilon_1}} (\rho(u, i, \sigma_2, \epsilon_2) - \bar{\rho}(u, \sigma_2, \epsilon_2)) \times sim_w(a, u, \sigma_4, \epsilon_4)}{\sum_{u \in N_{a,\sigma_1,\epsilon_1}} sim_w(a, u, \sigma_4, \epsilon_4)} \tag{4}$$

Neighborhood Selection. In DCR, we selected neighbors among those who had rated item i in a context matching the target context c. In the weighted version, we select neighbors by comparing their contexts for rating item i with the target context c and allowing them as neighbors if the context similarity is greater than a threshold ϵ_1. However, it is possible that a neighbor u has rated an item in multiple contexts, so we choose the maximally-similar context when applying the threshold. Alternatives are to choose minimally-similar context or averagely-similar context. In our experiments, we tried all three functions and the maximally-similar context is the best choice. We will denote this operation with N_{a,σ,ϵ_1} defined as follows, where c is the context of the current recommendation.

$$N_{a,\sigma,\epsilon_1} = \{u : \max_{r_{u,i,d}}(J(c, d, \sigma)) > \epsilon_1\}$$

Neighbor Contribution. In DCR, we chose a subset of the neighbor's ratings on which to compute that user's baseline and subtracted this average from the user's rating $r_{u,i,c}$. For DCW, it is possible that the user has multiple ratings for item i in different contexts that match the target context c to different degrees. These need to be combined via a weighted average and then from them subtracted an overall weighted average of all ratings issued in similar contexts. We will define our weighted average function for an item as follows:

$$\rho(u,i,\sigma,\epsilon_2) = \frac{\sum_{r_{u,i,d} \ni J(c,d,\sigma)>\epsilon_2} r_{u,i,d} \times J(c,d,\sigma)}{\sum_{r_{u,i,d} \ni J(c,d,\sigma)>\epsilon_2} J(c,d,\sigma)}$$

This function selects all ratings for i given by users u in contexts at least ϵ_2 similar to c, and applies a weighted average based on contextual similarity. Let I_u be the set of all items rated by user u. The overall average across all items rated in similar contexts is $\bar{\rho}$ defined here:

$$\bar{\rho}(u,\sigma,\epsilon_2) = \frac{\sum_{i \in I_u} \rho(u,i,\sigma,\epsilon_2)}{|I_u|}$$

User Baseline. The user baseline is the overall average of the target user's ratings of items in similar contexts, which is simply $\bar{\rho}(a,\sigma,\epsilon_3)$.

User Similarity. In DCR, we used Pearson correlation to compute the similarity between users, filtering out pairs of ratings with non-matching contexts. For sim_w, the weighted variant, we weight each comparison between ratings. We create the set T_{ϵ_4} by collecting all items i and pairs of contexts c and d for users a and u, respectively, such that each has rated i in that context and $J(c,d,\sigma) > \epsilon_4$.

$$T_{\epsilon_4} = \{\langle i,c,d\rangle \ni \exists r_{a,i,c}, r_{u,i,d} \wedge J(c,d,\sigma) > \epsilon_4\}$$

Once we have all of the relevant ratings and their contexts, we can compute a weighted version of the correlation function as shown in Equation 5.

$$sim_w(a,u,\sigma,\epsilon_4) = \frac{\sum_{\langle i,c,d\rangle \in T_{\epsilon_4}} (r_{a,i,c} - \bar{r}_a)(r_{u,i,d} - \bar{r}_u) J(c,d,\sigma)}{\sqrt{\sum (r_{a,i,c} - \bar{r}_a)^2 \sum (r_{u,i,c} - \bar{r}_u)^2 \sum_{\langle i,c,d\rangle \in T_{\epsilon_4}} J(c,d,\sigma)^2}} \quad (5)$$

5 Optimization

The remaining work for DCR and DCW is to find the optimal binary selection vectors or weighting vectors for algorithm components. In our DCR algorithm, we model relaxation as a process of binary selection – the vectors s_1, s_2, s_3 and s_4 are used to filter the contextual variables. In our prior work, the set of contextual variables was small and simple enough that we could use exhaustive search of all possible constraints [29]. However, this approach is not at all scalable. Moreover, the optimization space is highly non-linear and standard approaches such as gradient descent cannot be used. An efficient non-linear optimizer is required.

Particle swarm optimization (PSO) [17] is a form of swarm intelligence originally introduced by Eberhart and Kennedy in 1995. It is a population-based optimization approach inspired by social behaviors in swarming and flocking creatures like bees, birds or fish. It was introduced to the domain of information retrieval [12,11] and recommender systems [1,26,10] recently as a means of feature selection and feature weighting. Binary particle swarm optimization (BPSO) is a discrete binary version of the technique [18].

In PSO, optimization is produced by computations using a number of particles placed at different points in an optimization space. Each particle searches independently for the optimum, guided by the local value and the communication with the other particles. In our case, we use root mean square error (RMSE) as the value to be minimized and the position in the space corresponds to a set of weights for the σ vectors. If there are five contextual variables and four algorithm components, there will be a 20-dimensional search space.[2] Each particle records its personal best performance and corresponding best position. The algorithm also keeps track of the global best performance and corresponding position. These values are updated for the whole swarm in each iteration.

PSO and BPSO have been successfully demonstrated as efficient non-linear optimizers. They are to understand and implement, and there are several open source libraries online [3]. Our application of feature weighting for DCW is similar to the previous work by Ujjin *et al.* [26] on uCF, where they applied PSO and found it outperformed genetic algorithms. Furthermore, we realize DCR by feature selection and DCW by feature weighting – BPSO and PSO use binary vectors and real-value vectors as the position for particle respectively, a good match for the requirements for DCR and DCW.[4]

6 Experiment Setup

We evaluated our DCR and DCW models on four real-world datasets. For reasons of space, we discuss two of them in this paper.

Food Data is the "AIST context-aware food preference dataset" used and distributed by the author Hideki Asoh *et al* [24]. It is based on a survey of users' ratings on a menu under the context of different degrees of hunger: hungry/normal/full. The ratings were collected in two situations: real hunger is current degree of hunger, and virtual hunger is an imagined state of hunger. More information is shown in Table 2.

Movie Data is used by Adomavicius *et al* [3] and Karatzoglou *et al* [16], where the data was collected from a survey – subjects were requested to rate movies and report on the movie watching occasion. Three contextual variables were captured: Time (weekend, weekday), Companion (friends, parents, girlfriend or boyfriend, alone, siblings,etc) and Location (home, cinema). More information is shown in Table 2.

[2] It is 20-dimensional search space in DCR. Actually, there are additional dimensions due to the need for ϵ threshold values in DCW. In our experiments, we use the same threshold for all components in the algorithm and so only one additional dimension is needed in DCW.

[3] `http://www.particleswarm.info/Programs.html`

[4] As discussed in our prior work, we found that an improved version of BPSO [19] is was the best solution for feature selection in DCR [30]. For DCW, we use the variant constriction-factor PSO (CFPSO) [8], which promises quicker convergence than some other variants such as FIPSO [23] and CLPSO [21].

The rating scale for movie data is 1 to 13, and it is 1 to 5 for the food dataset. After filtering out subjects with less than 5 ratings and subjects with incomplete user profiles or item features, we got the final datasets shown in Table 2, where context-linked features [30] include both user profiles and item features. And the number of dimensions indicate the total amount of contexts and context-linked features.

Table 2. Description of Datasets

Dataset	# of users	# of items	# of ratings	# of dimensions	Density of Contexts
Food Data	212	20	6360	6	Dense
Movie Data	69	176	1010	5	Sparse

The predictive performance was measured by RMSE evaluated using 5-fold cross validation. We also measured coverage for each evaluation run, measured as the percentage of predictions for which we can find at least one neighbor. If we find no neighbors, the prediction is based on the user's baseline as shown in Equations 1, 2, and 4. We use a threshold 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 [24,3,16] on these two data sets used prediction errors, also.

The result below shows values from four algorithms. The first algorithm is uCF as described above, a context-free application of kNN collaborative filtering. We also implemented contextual prefiltering to provide a non-differential context-aware algorithm for comparison. This is just a variant of the DCR algorithm in which context is used only to pre-filter the neighbors.

We did a number of experiments using DCR on these datasets, the results of which are omitted for reasons of space. For the purposes of this paper, we compare against the top performer: DCR optimized using a 3-particle version of BPSO [5]. Finally, the fourth algorithm is DCW as described above.

There are several other DCW algorithm parameters worth mentioning. The minimal coverage thresholds are set as 0.7 for the food data and 0.5 for the movie data. These parameters were chosen based on performance on the standard collaborative recommendation algorithm and not changed for the context-aware algorithms. As mentioned above, we use the same ϵ threhold values for all components to reduce the number of dimensions in the optimization process. To reduce variability due to random particle positioning, we use the same initial particle positions for BPSO and PSO.

7 Experimental Results

We applied PSO to identify optimal weights for the DCW algorithm and compared the accuracy and coverage with those found with DCR. It is clear that feature weighting does offer improved accuracy over feature selection.

[5] We examined different numbers of particles for BPSO and PSO, where using 3 particles showed the best performances for these two datasets taking the running time into consideratoins. So we use 3-particle version of BPSO and PSO as comparisons.

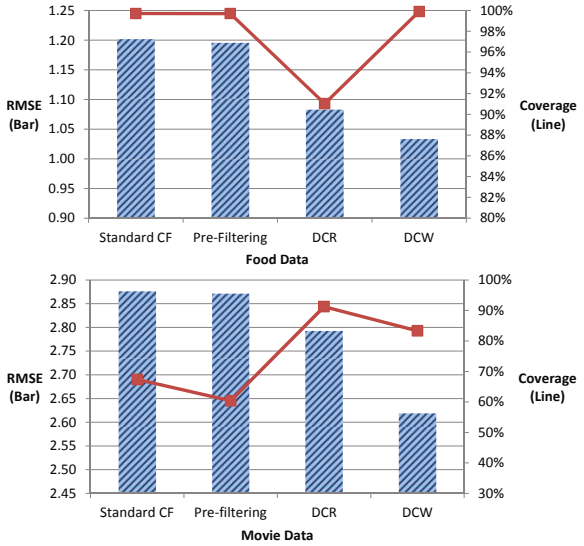

Fig. 1. Performance Comparisons

Consider the relative performance of the algorithms as shown in Figure 1. For food data and movie data, DCW shows strongly significant improvement comparing with standard CF, contextual pre-filtering and DCR. DCW outperforms DCR significantly in terms of RMSE, and it works well for the movie data where DCR does not offer significant improvements over the baselines [6].

DCW achieves higher coverage than DCR for food data. Surprisingly, in the movie data set, we see a small decline in coverage compared with DCR. We believe that the reason for this result is that the optimization criterion in PSO was RMSE, so the system may be minimizing RMSE at the expense of coverage in this dataset. One solution would be to optimize using a combination of RMSE and coverage. From other experimentation with this and other data sets, we believe that improvement over DCR in both coverage and RMSE is possible for DCW.

Table 3 provides running time for the algorithms, showing the number of iterations required to converge and the corresponding running time in seconds.

Table 3. Comparison of Running Performances Between BPSO and PSO

	Food Data		**Movie Data**	
	Iteration	Running Time	Iteration	Running Time
DCR via BPSO	11	66.9	18	4.9
DCW via PSO	13	248.4	66	23.2

[6] We use paired t-test to examine significance, where 0.05 is set as the threshold for p-value to evaluate the significance and p-value lower than 0.01 indicates strong significance.

The most significant difference between BPSO and PSO is that the value in the particle position – it is binary number for BPSO, switching between 0 and 1, and it is a decimal value ranging between [0, 1] for PSO. The search space for BPSO is limited because the combinations are countable. The unlimited search space of PSO results in much greater search times. In Table 3, it is not surprising that PSO required more iterations to converge and the average running time per iteration is increased due to the weighting process is more complex than selection. Our experiments find that the run time performance of DCW via PSO depends two factors: one is the number of contextual dimensions used – more dimensions, more parameters require to be learned; another one is the density of contextual ratings – more dense, the weighting calculations require more time. As shown by the food data which has more dimensions and more dense data, the running time goes up significantly.

We see that the finer-grained application of context in DCW is able to improve the predictions of DCR across multiple data sets. The results on coverage are inconsistent, due to the use of RMSE as the optimization criterion. Improved RMSE comes at the cost of greater computation time due to the learning complexity of PSO.

8 Conclusions and Future Work

This paper points out the drawbacks of DCR and introduces a generalization of DCR, differential context weighting (DCW), where contextual variables are weighted rather than selected, and it is demonstrated to compensate the drawbacks of DCR. DCW achieves better accuracy as compared to DCR with acceptable coverage and even better coverage than the baselines in our experiments. DCW also works when DCR does not work significantly. BPSO and PSO are shown as efficient optimizers for DCR and DCW respectively. However, DCW may require more costs due to the weighting process in DCW and learning complexity in PSO.

One direction in the future work is to explore other forms of similarity of contexts to DCW other than the simple Jaccard similarity, such as semantic similarities, which may help further ameliorate the sparsity problem. For example, in our Movie data set, the "companion" variable has values such as "friends", "family", "solo". It may make sense to treat "friends" and "family" as more similar than "friends" and "solo" because they are settings involving a group of individuals. Also, DCR and DCW are considered as general approaches to use context in recommendation and they can be integrated to other recommendation algorithms in the future. We have successfully applied them to item-based CF and slope one recommender, and we hope to further integrate them with latent factor models in the future, such as matrix factorization.

References

1. Abdelwahab, A., Sekiya, H., Matsuba, I., Horiuchi, Y., Kuroiwa, S.: Feature optimization approach for improving the collaborative filtering performance using particle swarm optimization. Journal of Computational Information Systems 8(1), 435–450 (2012)
2. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Magazine 32(3), 67–80 (2011)

3. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) 23(1), 103–145 (2005)
4. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender Systems Handbook, pp. 217–253 (2011)
5. Tasič, J.F., Košir, A., Odic, A., Tkalcic, M.: Relevant context in a movie recommender system: Users opinion vs. statistical detection. In: ACM RecSys 2012, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012). ACM (2012)
6. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Personal and Ubiquitous Computing, 1–20 (2011)
7. Bourke, S., McCarthy, K., Smyth, B.: The social camera: a case-study in contextual image recommendation. In: Proceedings of the 16th International Conference on Intelligent User Interfaces, pp. 13–22. ACM (2011)
8. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
9. De Pessemier, T., Deryckere, T., Martens, L.: Extending the bayesian classifier to a context-aware recommender system for mobile devices. In: 2010 Fifth International Conference on Internet and Web Applications and Services (ICIW), pp. 242–247. IEEE (2010)
10. Diaz-Aviles, E., Georgescu, M., Nejdl, W.: Swarming to rank for recommender systems (2012)
11. Diaz-Aviles, E., Nejdl, W., Schmidt-Thieme, L.: Swarming to rank for information retrieval. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 9–16. ACM (2009)
12. Drias, H.: Web information retrieval using particle swarm optimization based approaches. In: 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 1, pp. 36–39. IEEE (2011)
13. Gantner, Z., Rendle, S., Schmidt-Thieme, L.: Factorization models for context-/time-aware movie recommendations. In: Proceedings of the Workshop on Context-Aware Movie Recommendation, pp. 14–19. ACM (2010)
14. Hariri, N., Mobasher, B., Burke, R., Zheng, Y.: Context-aware recommendation based on review mining. In: Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011), p. 30 (2011)
15. Huang, Z., Lu, X., Duan, H.: Context-aware recommendation using rough set model and collaborative filtering. Artificial Intelligence Review, 1–15 (2011)
16. 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)
17. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
18. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108. IEEE (1997)
19. Khanesar, M., Teshnehlab, M., Shoorehdeli, M.: A novel binary particle swarm optimization. In: Mediterranean Conference on Control & Automation, MED 2007, pp. 1–6. IEEE (2007)
20. Lee, J.S., Lee, J.C.: Context awareness by case-based reasoning in a music recommendation system. In: Ichikawa, H., Cho, W.-D., Satoh, I., Youn, H.Y. (eds.) UCS 2007. LNCS, vol. 4836, pp. 45–58. Springer, Heidelberg (2007)
21. Liang, J., Qin, A., Suganthan, P., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation 10(3), 281–295 (2006)

22. Liu, L., Lecue, F., Mehandjiev, N., Xu, L.: Using context similarity for service recommendation. In: 2010 IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 277–284. IEEE (2010)
23. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Transactions on Evolutionary Computation 8(3), 204–210 (2004)
24. Ono, C., Takishima, Y., Motomura, Y., Asoh, H.: Context-aware preference model based on a study of difference between real and supposed situation data. In: Houben, G.-J., McCalla, G., Pianesi, F., Zancanaro, M. (eds.) UMAP 2009. LNCS, vol. 5535, pp. 102–113. Springer, Heidelberg (2009)
25. Park, H.-S., Yoo, J.-O., Cho, S.-B.: A context-aware music recommendation system using fuzzy bayesian networks with utility theory. In: Wang, L., Jiao, L., Shi, G., Li, X., Liu, J. (eds.) FSKD 2006. LNCS (LNAI), vol. 4223, pp. 970–979. Springer, Heidelberg (2006)
26. Ujjin, S., Bentley, P.: Particle swarm optimization recommender system. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS 2003, pp. 124–131. IEEE (2003)
27. Vargas-Govea, B., González-Serna, G., Ponce-Medellín, R.: Effects of relevant contextual features in the performance of a restaurant recommender system. In: ACM RecSys 2011, The 3rd Workshop on Context-Aware Recommender Systems, CARS-2011 (2011)
28. Woerndl, W., Huebner, J., Bader, R., Gallego-Vico, D.: A model for proactivity in mobile, context-aware recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 273–276. ACM (2011)
29. Zheng, Y., Burke, R., Mobasher, B.: Differential context relaxation for context-aware travel recommendation. In: Huemer, C., Lops, P. (eds.) EC-Web 2012. LNBIP, vol. 123, pp. 88–99. Springer, Heidelberg (2012)
30. Zheng, Y., Burke, R., Mobasher, B.: Optimal feature selection for context-aware recommendation using differential relaxation. In: ACM RecSys 2012, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012). ACM (2012)