Learning a Region of User's Preference for Product Recommendation

Anbarasu Sekar and Sutanu Chakraborti

Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai - 600036
{anbu,sutanuc}@cse.iitm.ac.in

Abstract. A *Conversational Recommender System*(CRS) aims to simulate the real world setting where a shopkeeper tries to learn the preferences of a user through conversation and the user, in turn, learns about the product base. Often the preference of the user is captured in the form of feature weights and they are updated assuming each feature is independent. These feature weights are used in product retrieval from the product base. The independence assumption of the features is not always a good choice, and we will address this concern in our work. Each product in the product base has its own prospective buyers characterized by a certain combination of preference choices. The goal of this work is to discover knowledge about the preference choices of prospective buyers for each product offline and, later use it to recommend products to the users based on their preference choices.

Keywords: Case-based Recommendation; Conversational Recommender Systems; Compromise; Tradeoff; Preference feedback; Preference region; Dominance region; Constraint Programming

1 Introduction

In Case-Based Reasoning(CBR) based approaches to product recommendation, the product domain is treated as case base and the products are treated as cases. Customers often do not have sufficient ideas about the case base. They tend to learn more about the cases that may be of interest to them as they get involved in the process of shopping. This means that the preference of the customers evolves as they keep learning about the domain. This renders single shot recommender systems ineffective. A Conversational Recommender System(CRS) tries to overcome this drawback by constantly updating the preference of the user based on his/her interaction with the system. Learning user preferences is crucial in a recommender system. Modelling user preferences is a well-known task in machine learning community. In Case-based recommender systems, the user preferences are modelled as similarity function so that the products in the case base can be ordered in a way that is reflective of the user's order of priority. Most similarity functions associate each feature with a weight, which can be used to capture the preference of the user on that feature. Often each feature is assumed

In Proceedings of the ICCBR 2016 Workshops. Atlanta, Georgia, United States of America

to be independent of the rest of the features, and hence each feature weight is updated independently. Few works in literature relax this assumption([3],[4],[5]).

In this work, we propose a novel way of capturing user preferences. In the work by Mouli et al. [5] the authors model user preferences as soft constraints that overcome the drawback of assuming the features to be independent. We employ a similar way of learning user preferences. However, our work markedly differs from their work, in the way the preference knowledge is represented. Most of the works capture user's preference in the form of a feature weight vector. To the best of our knowledge, our work is unique in representing user's preference in the form of a region in preference weight space defined by a set of constraints. *The main contribution of this work is to discover under what combinations of preference weights each case in the case base is preferred over a set of cases that may probably be its competitors and use this knowledge in the recommendation process.* The knowledge mined from the case base, specific to a particular case, is represented as a region in preference weight space, which we term it as *dominance region* of that case. The *dominance region* of a case can be assumed as a way of indexing the cases in the case base, which can be used for predicting the prospective buyers of that particular case. We propose a model, Predicted Preference Region based Model(PPRM) based on the *dominance region* discovered for each case in the case base. In Section 2 we will present the background and an overview of previous works that are related to our work. In Section 3 we will go into the details of our work followed by Evaluation and Discussion of the results.

2 Background and Related Work

As discussed before, a CRS involves the user in a conversation which may take several iterations before the user could settle for a case of his desire. A CRS requires a mechanism where the preferences of the user may be learnt. A response from the user is essential in the task of learning user preferences. There are several ways in which a response could be obtained from the user and different CRSs process the response in different ways to learn user preferences. The user preferences modeled in the form of feature weights are used for case retrieval.

2.1 Preference-based feedback

In each iteration of a CRS, the system displays a set of recommended cases and the user gives a feedback to the system based on the kind of cases recommended to him/her. In preference based feedback systems, the buyer selects a case of his preference from the recommended set of cases which acts as a feedback for the system. In our work using CRS, the system needs preference feedback from the user.

2.2 More-Like-This(MLT) and weighted More-Like-This(wMLT)

In MLT, query(a vector of feature values) is the only way in which the preference of a user is captured. The preference feedbacks given by the user in a given iteration becomes the query for the next iteration and hence the cases in the

next iteration are the cases that are more like the case chosen by the user in the previous iteration[1]. There is a catch in this approach. The user may not prefer all features of the selected case and the information on why the user rejected the other cases in the recommended set is lost. In wMLT[1] this drawback is overcome by assigning weights to the features. The feature weights along with the user's query capture the preference of the user. If the feature value present in the preference feedback is also present in the cases rejected by the user then the weight of the corresponding feature is reduced proportionally to the number of rejected cases in which that feature value is present.

2.3 Critique based Recommender Systems

In critique based recommender systems, feedback is collected on specific features of the case. In some domains, case features can be classified as "More is Better"(MIB) and "Less is Better"(LIB). An example, in camera domain, Zoom is a MIB feature and Price is a LIB feature. Critiques could be something like "more of a MIB feature" or "less of a LIB feature" depending on the cases available the case base[2]. The feature weight is updated proportionally to the difference between the feature value of the query and the feature value of the case selected from the recommended set.

2.4 Recommender Systems without feature independence assumption

The drawback with all the systems discussed previously is that the feature weights are updated in each cycle independent of the other features.

Compromise Driven Retrieval(CDR)
CDR proposed by McSherry[3] is one of the approaches that looks at features as dependent on each other. Consider a hypothetical Camera data set given in in Table 1, with only three features say "Price", "Resolution", "Optical Zoom". Given the query and the cases, say a user chooses Case 1. We can see that the user is ready to pay extra 500 rupees for the gain of 5 MP in resolution and 5 X in Zoom with respect to the query. Such a choice is termed as, compromise on Price for the gain in Resolution and Optical Zoom. The customer is ready to compromise on Price provided the customer gains in other two features. When given a choice between two cases that are equally similar to the user's query, CDR recommends a product that makes a compromise on a subset of the features on which the other case compromises.

	Price(Rs.)	Resolution(MP)	Optical Zoom(X)
Case 1	1000	10	10
Case 2	600	4	5
Query	500	5	5

Table 1: Toy Camera DataSet and Query

The aim of CDR is to make sure that all possible compromises that are available in the case base are suggested to the user in the recommendation process as we do not know in advance the compromises the user is ready to make.

Preference Constraints
In an earlier work by Pu et al. [4] the authors try to learn the preferences of the user by learning the various *tradeoffs* the user is ready to make. We regard tradeoffs as related to compromise. For example: how much is a user willing to pay/tradeoff for added feature like 'Bluetooth' in a mobile phone? The buyer is made to enter his choice of compromise explicitly. Each compromise choice is converted to a soft constraint, which is used to model buyer's preference. The cognitive load on the user is considerably high. In the work by Mouli et al.[5] the authors elicit the constraints automatically based on the preference feedback of the buyer. The user's choice of a case out of the recommended cases is reasoned to arrive at the feature preference of the user. For the Toy dataset in Table 1, if customer's choice is Case 1 then it is reasoned that the utility of Case 1 is greater than the utility of Case 2. On the basis of Multi Attribute Utility theory(MAUT)[6] and classification of features(MIB or LIB), we can arrive at a constraint $6 \times w_{resolution} + 5 \times w_{zoom} \geq 400 \times w_{price}$ where $w_{resolution}$, w_{zoom} and w_{price} indicates the preferences given to resolution(MIB feature), zoom(MIB feature) and price(LIB feature) respectively. The feature values are normalized to lie between 0 and 1. The normalisation procedure takes the type of feature(MIB or LIB) into consideration. For a LIB feature, the highest value gets normalised to 0 and the lowest value gets normalised to 1 and vice versa for a MIB feature. In general, a constraint could look something like $w_{resolution} \times \Delta resolution + w_{zoom} \times \Delta zoom \geq w_{price} \times \Delta price$. Here the Δ values are the differences in the feature values. In this process, we reason why the user prefers a case over the other case. The knowledge is captured in the form of constraints on the preference weights of the user. The user is not required to explicitly mention his tradeoffs as in the former work. If there are k cases in the recommended set, we reason why a user has selected the preferred case over the *k-1* cases. Similar to the aforementioned example the preferred case can be compared with every other case in the recommended set to get a set of *k-1* constraints(tradeoff constraints). In addition to the tradeoff constraints, we have a constraint on preference weights, that the sum of the preference weights in a preference weights vector is equal to 1(preference constraint). These constraints are solved for the preference weights. The solution set is a region in the preference weight space, which we call it as *preference region*. The current preference weights are updated by considering a point that lies in this *preference region* and is at a minimum distance from the current preference weights. The assumption is that the user's preferences don't change much in each cycle. This approach is termed by the authors as Compromise Driven Preference Model(CDPM).

3 Our Approach

We draw our inspiration from the idea of capturing feature dependent user's preferences from the work by Mouli et.al in [5]. We present our idea in Section

3.1. The other contribution of our work is to address the assumption of CDPM that the user's preferences don't change much in each cycle. The truth of this assumption is questioned and a new way of choosing a preference weights vector is proposed in Section 3.2.

3.1 Recommender System based on Preference and Dominance Region

Dominance Region

A product is designed to cater to preferences of a set of users, and the set of tradeoffs they are willing to make. For example in camera domain, people who desire professional cameras tend to trade off 'price' for a feature like 'zoom'. We can safely claim that each case offers a combination of tradeoffs that is acceptable to its target users, which is mapped to a region in preference weight space, we term this region as *dominance region*. This region is the *preference region* of the prospective users of a particular case. *Dominance region* of a case is the predicted *preference region* of the prospective users of a case in the case base. For any given case in the case base, the prediction of the *dominance region* of that case is done in two steps. In the first step, we assume a hypothetical buyer who prefers the given case over all its n nearest cases(cases in the small neighbourhood of the case). In the next step, we determine the *preference region* of the hypothetical user. We call the *preference region* of the hypothetical user as the *dominance region* of the given case. Our hypothesis is

If a case is desirable to the user then his preference region overlaps with the dominance region of the case.

Case Recommendation

Unlike CDPM where a single weight vector is maintained as the preference of the user, we maintain a region of weights that we keep updating in each cycle. In our system, the utility of a case is approximated by a combination of overlap score and similarity score, that we call CScore. The overlap score is based on the amount of overlap between the case's *dominance region* and user's *preference region*, the similarity score is based on the distance between the case and user's query. In Equations 1 and 2 P, Q, PR, DR_P denotes case, query, *preference region* and *dominance region* of case P respectively. Equation 1 has the constraint that $\alpha + \beta = 1$. In Equation 2 euclideanDistance is the Euclidean distance between cases P and Q positioned in the product space.

$$CScore(P, Q, PR) = \alpha \times overlapScore(DR_P, PR) + \beta \times similarity(P, Q) \quad (1)$$

$$similarity(P, Q) = \frac{1}{1 + euclideanDistance(P, Q)} \quad (2)$$

Computing the amount of overlap between the *dominance region* and *preference region* is not a straight forward task. We used Monte Carlo based technique to find the approximate amount of overlap between the two regions. We have a constraint that the sum of the weights of the preference vector needs to be equal to 1. Any given *preference region* or *dominance region* should lie within the bounding region defined by the constraint that the sum of the feature weights is equal to 1. From this bounding region, we uniformly sampled points and recorded if they belong to at least one of the regions or to both. The amount of intersection or overlap is based on Jaccard measure and is given in Equation 3. The approximate centroid of the *preference region* is computed by taking the average of the points that are uniformly sampled from the region where the sum of the feature weights is equal to 1 and which also lies in the *preference region*.

$$overlapScore(DR_P, PR) = \frac{no.\ of\ points\ in\ both\ regions}{no.\ of\ points\ in\ at\ least\ one\ region} \tag{3}$$

Overlap and Similarity

The α and β in equation 1 control the weightage given to the Overlap score and Similarity score. It is possible for a case that is less similar to the query to have its *dominance region* overlapping to a greater extent with the *dominance region* of the case of interest, so we want to search in the neighbourhood of the query. The β parameter controls the amount of diversity in the recommended cases and it is updated dynamically based on the cases chosen by the buyer at each recommendation cycle. Initially, both α and β are set to 0.5. The rank of the preferred case in the ordered list based on overlap score and similarity score is used to update the parameters. If the preferred case ranks higher in the similarity score based list than in the overlap score based list, then we assume that the buyer is interested in looking for very similar items and hence the β value is boosted. If this is not the case, then α value is boosted. Since $\alpha+\beta$ should be equal to 1, we normalize the values to satisfy the constraint. Hence boosting one of the parameters will result in the reduction of the other. A formulation for boosting the parameter α is given in the Equation 4. A similar formula is used to boost β too. We wanted to boost the values by a small amount proportional to the value of the parameter. Let P be the preferred case. $\rho(OverlapList, P)$ is the rank of P in the ordered list of the recommended cases based on Overlap Score. $\rho(SimilarityList, P)$ is the rank of P in the the ordered list of the recommended cases based on Similarity Score.

$$\alpha_{new} = \alpha_{old} + (\frac{\rho(SimilarityList, P) - \rho(OverlapList, P)}{|recommendedList|} * \alpha_{old}) \tag{4}$$

Methodology

Step 0: Computing *dominance region*: For each case in the case base, assume n nearest neighbours as the competitors of the case. It is assumed that the case

Variables
DB: Case base;
P: Preferred case;
RS: Recommended Set;
Q: Query;
PR: *Preference region*;
DR_X: *Dominance region* of case X;
α: overlapScore parameter; β: similarity parameter;
Result: Desired case
initializeVariables();
setDominanceRegions();
setAlphaBetaParameters();
Q ← getInitialUserQuery();
RS ← getKNearestCases(Q,DB,k);
while *Desired case* $\notin$ *RS* **do**
    P← getUsersChoice(RS);
    updateAlphaBetaParameters(RS,P);
    PR← getPreferenceRegion(RS,P);
    Q←P;
    RS← getRecommendationSet(PR,Q,α,β);
end

Algorithm 1: Overall Algorithm

under consideration is better than all its competitors. For n competitors we get n constraints, by comparing each of the n competitors with the given case. These constraints are solved for the feature weights. We get a region of weights as solution set. This region is the *dominance region* of the case (DR_{case}). This is done offline and is used in the recommendation process.

Step 1: The user gives his initial preference, usually on a subset of the features of the case. This is the query Q.

Step 2: Initially we assume uniform weights for all features and get k nearest cases to the query from the case base as the recommended set of cases.

Step 3: The user picks up a case as his preferred case. We compare the preferred case with each of the $k-1$ recommended cases and get $k-1$ tradeoff constraints. These tradeoff constraints along with the preference constraint are solved for the feature weights. We get a region of weights as the solution set. This region is the *preference region* of the user(PR).

Step 4: The preferred case becomes the new Query. We select m nearest neighbours based on unweighted distance measure to the Query. We remove all cases that are in the k cases in the previously recommended set. We order the remaining cases based on the CScore, the top k cases are shown to the user as

the recommendations.

Step 5: We go to step 3 and continue the cycle till user is satisfied with one of the cases from the recommended set of cases.

Variables
NN: Nearest Neighbours to Query;
Result: RS
NN← getKNearestCases(Q,DB,k);
NN←NN-RS;
RS← ∅;
for *each* $X \in NN$ **do**
  overlapScore← getOverlapScore(DR_X,PR);
  similarity← getSimilarity(X,Q);
  CScore← getCScore(overlapScore,similarity,α,β);
  insertInOrderBasedOnCScore(RS,CScore,X);
end

Algorithm 2: getRecommendationSet Algorithm

3.2 Centroid-based Preference weights

To pick a single preference weight vector from the *preference region*, authors in [5] have made use of an assumption that the current preference weight does not change much from the previous iteration, which may not be true always. The assumption may be true in the later stages of the conversation when the buyer does not explore the domain as much as in the initial stages of the conversation. As an attempt to work around this assumption instead of choosing a preference vector that is nearest to the current feature weights from the *preference region*, we decided to choose the centroid of the *preference region* as the vector of updated preference weights. The efficiency of the method is evaluated and the results are reported in the evaluation section. We call this method Centroid-based Preference Weights Model (CPWM).

4 Evaluation

We have used camera dataset[8] with 210 cameras and PC dataset[1] with 120 computers for evaluation. We evaluate the effectiveness of our method by a usual method conducted in preference based conversational recommender systems[7]. A case is randomly removed from the case base and the case that is most similar to the removed case is set as the target case. A subset of feature values from the removed case is used as the query. Three queries are formed from the removed case, of sizes 1(Hard), 3(moderate) and 5(easy). This subset is also randomly picked. For each of the three queries, we evaluate the system. The behaviour of the user is simulated in each iteration by picking a case that is most similar to the target(Similarity is used as a means of simulating a user). The selected case is treated as the preferred case and it becomes the query for the next iteration. We

stop the conversation cycle when the target case appears in the recommended set of cases. The number of cycles required to reach the target is recorded. We repeat the whole process 1000 times by randomly picking a case from the case base. The average number of cycles required for each category is used as a measure for comparison.

5 Results and Observations

The average number of cycles required is measured from 1000 queries for the method based on dynamic preference based method, centroid based method and our method and the results are as given in Table 2 and Table 3.

	Query size 1	Query size 3	Query size 5
CDPM	12.82	8.51	4.47
CPWM	13.26	7.65	4.05
PPRM	7.49	3.37	2.13

Table 2: Average cycle length in Camera dataset

	Query size 1	Query size 3	Query size 5
CDPM	6.99	4.55	2.27
CPWM	7.63	4.14	2.10
PPRM	3.20	1.76	1.35

Table 3: Average cycle length in PC dataset

We can see 41.5%, 60.3% and 52.3% reduction in number of cycles with respect to the query of sizes 1, 3 and 5 respectively with PPRM when compared to the CDPM in Camera dataset, and 54.2%, 61.3% and 40.5% reduction in number of cycles with respect to the query of sizes 1, 3 and 5 respectively with PPRM when compared to the CDPM in PC dataset. However, CPWM performs very similar to the CDPM. To study the effect of the α and β parameters we ran the experiments with different starting values of the parameters and the results are tabulated in Table 4.

	Query size 1	Query size 3	Query size 5
$\alpha = 0.75$, $\beta = 0.25$	7.17	3.53	2.25
$\alpha = 0.5$, $\beta = 0.5$	7.49	3.37	2.13
$\alpha = 0.25$, $\beta = 0.75$	7.07	3.70	2.27

Table 4: Average cycle length in Camera DataSet for different starting values of parameters

The average cycle length does not vary much for the different initial values of α and β. The weights get adjusted appropriately during the conversation, result-

ing in very similar results. To study the effect of the size of nearest neighbours from which the *dominance region* is predicted, we conducted the experiments for various Competitors sizes. The results are given in Table 5. There is not much variations in the results that could be captured from the experiments, suggesting robustness with respect to neighbourhood size.

No. of competitors	Query size 1	Query size 3	Query size 5
3	7.42	3.60	2.14
5	7.49	3.37	2.13
7	7.12	3.69	2.24
10	7.29	3.28	2.27

Table 5: Effect of Competitors size on Average Cycle length in Camera DataSet

6 Conclusion and Future Work

Our work exploits the knowledge mined from the case base and couples it with the learnt user preference in the process of recommendation. The considerable reduction in the number of cycles suggests the soundness of our hypothesis. In this work the neighbours of a case is considered as the competitors of that case in its *dominance region* computation, this is an assumption that could be far from reality. To get the actual competitors of a product, we would like to use appropriate datasets and evaluate our work by conducting live user study.

References

1. L. McGinty and B. Smyth. Comparison-based recommendation. In S. Craw and A. Preece, editors, Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002, pages 575-589, Aberdeen, Scotland. Springer Verlag, 2002.
2. L. Chen and P. Pu. Preference-based organization interfaces: aiding user critiques in recommender systems. In User Modeling 2007, pages 77-86. Springer, 2007.
3. McSherry, David. Similarity and compromise. International Conference on Case-Based Reasoning. Springer Berlin Heidelberg, 2003.
4. P. Pu and B. Faltings. Decision tradeoff using example-critiquing and constraint programming. Constraints, 9(4):289-310, 2004.
5. Mouli, S. Chandra, and Sutanu Chakraborti. Making the Most of Preference Feedback by Modeling Feature Dependencies. Proceedings of the 9th ACM Conference on Recommender Systems. ACM, 2015.
6. Keeney, Ralph L., and Howard Raiffa. Decisions with multiple objectives: preferences and value trade-offs. Cambridge university press, 1993.
7. L. Mc Ginty and B. Smyth. Evaluating preference-based feedback in recommender systems. In Artificial Intelligence and Cognitive Science, pages 209-214. Springer, 2002.
8. http://josquin.cs.depaul.edu/ rburke/research/downloads/camera.zip