Constructing CP-nets from Users Past Behaviors

Reza Khoshkangini
University of Padova, Padova, Italy
khosh@math.unipd.it

Maria Silvia Pini
Department of Information Engineering, University of Padova, Padova, Italy
pini@dei.unipd.it

Francesca Rossi
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
(on leave from University of Padova)

Dinh Van Tran
University of Padova, Padova, Italy
dinh@math.unipd.it

ABSTRACT

While recommender systems over time have significantly improved the task of finding and providing the best service for users in various domains, there are still some limitations regarding the extraction of users' preferences from their behaviors when they are dealing with a specific service provider. In this paper we propose a framework to automatically extract and learn users' conditional and qualitative preferences by considering past behavior without asking any information from the users. To do that, we construct a CP-net modeling users' preferences via a procedure that employs multiple Information Criterion score functions within an heuristic algorithm to learn a Bayesian network. The approach has been validated experimentally on a dataset of real users and the results are promising.

KEYWORDS

Bayesian Network, CP-net, Recommender System

1 INTRODUCTION

Over the past decades significant efforts have been undertaken among researchers, practitioners and companies to develop various types of recommender systems to meet users' requests [3, 23]. The aim of these systems is to personalize service recommendations for individual users, as well as aggregating users' preferences to recommend a service for a group of users in various domains from movies [26] to restaurants [14, 18], from hotels to recommending items or products, etc.

Advancement in recommender systems have been performed by considering the context, which is basically defined as any information that could be used to characterize the situation of an entity in a particular domain [11]. On the one hand, considering the context can improve the performance of the recommender systems, which leads to enhance the satisfaction degree of users by properly fulfilling their demands [17]. On the other hand, it can make the recommendation task more complex, since changes in the context may cause changes in users' preferences over time. Self-adaptive recommender systems have been developed to overcome such problems in the application domains, where the context of users and/or of services can influence the recommendation [6, 18, 24].

Users' preferences are often qualitative and conditional. For example, *if it is sunny, I prefer to go to a restaurant that has a garden with tables outside.* CP-nets [24] are a graphical model to represent in a compact and intuitive way this kind of preferences. Briefly, a CP-net is a graph in which each node is labeled with a table describing the user's preference over alternative values of this node given different values of the parent nodes. An example of CP-net is shown in Figure 1.a. CP-nets has been already used to model users' preferences in automated decision making and in modeling human preferences in real-world applications.

In this paper, we propose a framework to automatically construct CP-nets from users' past behaviors without demanding any information from the users. Users' past behavior (or preferences) is characterized, by the set of domain features, which are logged in their profiles, through the their interactions with the system. For example, in the restaurant recommendation domain, the users' past behavior may be defined by the restaurants that have been selected previously by the user and each restaurant can be seen as a combination of values to a set of features (such as, price range, location, type of restaurant, etc.).

To build a CP-net from agent's past behavior we perform the following steps.

- Given the user's history, for example previous selected restaurants, we first identify some of the most important attributes that have influenced the final choices. Then the selected attributes become the features/nodes of the CP-net.
- Then, we divide these features in three layers: root layer, intermediate layer and layer. The root layer contains only the root node, which is the most important feature according to both user's preferences and the procedure used to extract features. The root node may be for example the price of the restaurant if the possible options for a user in selecting a restaurant depend mainly on his budget.
  The target layer contains only the target node which is the most dependent feature. The target node will be selected according to the domain knowledge. For instance, in the restaurant recommendation domain, the target not may be

the type of the restaurant (or food), while in the movie recommendation domain, the target node may be the name of the movie. The intermediate layer contains all the other nodes which correspond to all other selected features.
- Now that we have divided the features in three groups, we need to express the dependencies among them. We assume that there are only outgoing arcs from the root node, and only in-going arcs in the target node, while to identify the dependency relations between features in the intermediate layer we learn a Bayesian Network (BN) from the user's preferences by exploiting some scoring functions (such as BIC, and AIC [8]) implemented by an Hill climbing algorithm. Finally, from the conditional probabilities annotated with each node of the BN we define the corresponding conditional preferences associated to every node of the CP-net.

Experimental results show that the presented approach for building CP-nets from users' past behavior is promising in the restaurant and movie recommendation domains. In the experiments performed on a real data-set the scoring function BIC is the best one.

The proposed approach to construct CP-nets from users' histories is important for two reasons: the former one is that users' preferences are essentially qualitative and constructing these qualitative tendencies in the system can increase the knowledge of the system from users' preferences. The latter is inspired from the former advantage, such that this construction is significantly important when the service system needs to deal with the context that brings up the dynamicity challenge in a real-time recommender system [18]. The change in context leads to change on players preferences over the time. Hence constructing users' preferences in a qualitative form enables self-adaptive recommender systems [18] to increase users' satisfaction degree.

For example, assume that we know from the context that there is blocked traffic on the road for going to the recommended restaurant, the system should recommend another restaurant starting from the built CP-nets modeling the users' preferences.

Modeling and learning the users' preferences expressed via CP-nets is a task that has been studied extensively by adopting various techniques, such as observing/asking multiple questions to the users [4]. In some studies, researchers start by assuming a dependency structure (the user's CP-nets) and then they try to learn the users' conditional preferences [9]. Bigot et al. [4] discussed the possibility of learning Probabilistic CP-nets (PCP-nets), which have been introduced in [10] in two settings (online and off-line). In this paper, Bayesian networks are used to learn PCP-nets. In both settings they assume to have the dependency graph and then they ask multiple queries to the users to build up and learn the structure of the network. Similarly, Guerin et al. [15] present an algorithm for learning CP-net preferences by interacting with users rather than using users' histories. Learning conditional preferences may be a tedious and costly task, even with acyclic CP-nets. However, the complexity of the problem can be reduced by interacting with the users to simplify the learning procedure. For instance, Koriche et al. [19] propose an approach to identify a preference ordering with binary domains, which uses membership queries. Similar work has been done in [12], where pairwise comparisons are used e.g., $O_i > O_j$ between outcomes, under certain assumptions/constraints on the inputs for learning the structure of the users' CP-nets.

The rest of the paper is organized as follows. Section 2 provides some basic notions of CP-nets and Bayesian nets. Section 3 presents our CP-net constructor and Section 4 describes the experimental evaluation and the results. Finally, Section 5 summarizes the paper and provides some hints for future work.

2 CP-NET AND BAYESIAN NETWORK

In this section we present the key notions of CP-nets and Bayesian Networks.

CP-net: CP-net [5] is a graphical model to represent conditional and qualitative preference relations between variables (aka features). Lets assume, there is a set of variables $V = \{X_1, ..., X_n\}$ with finite domains $D(X_1), ..., D(X_n)$. For each variable X_i, each user specifies a set of parents $P(X_i)$ that can affect her preferences over the value of X_i. So this defines a dependency graph such that every variable X_i may have $P(X_i)$ as its immediate predecessors. For each node X_i, there is a *conditional preference table* that shows for each possible combination of parents values the preference over values of X_i.

An example of CP-net is shown in Figure 1 (a). It contains three features (aka variables) $X1$, $X2$ and $X3$, standing for the Price, Quality and the Type of restaurant, respectively. $X1$ is an independent variable, while $X2$ depends on X_1 and $X3$ depends on both $X1$ and $X2$. This CP-net models the fact that the users strictly prefers a restaurant with a medium price (x_1^1) to an expensive one (x_2^1), while his preference between "medium" quality (x_1^2) and "high" quality (x_2^2) is conditioned on the price to be selected. In addition, the preference on the type of restaurant $Chinese = x_1^3$ and $Mexican = x_2^3$ depends on both the quality of the restaurant and the price.

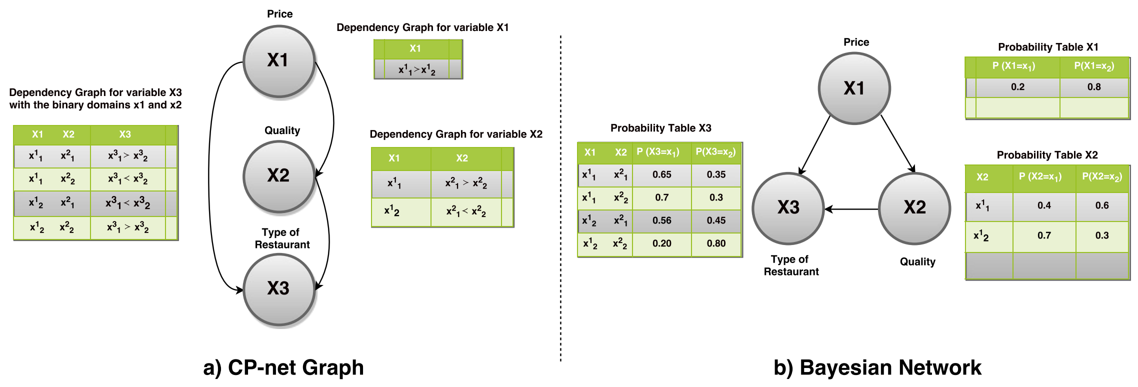
Bayesian Network (BN): A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG) $G = (V, E_G)$ [7, 16], where V is the set of features, and E_G represents the set of direct arcs (dependency) between the features, e.g., $X_i \rightarrow X_j$, and there is a constraint in BN that avoids any directed cycles (similarly to the concept of acyclic CP-net).

The strength of the correlation between features is defined in the second component of the BN, where there is a set of parameters in the network that show the conditional probability distribution between variables.

Considering n variables $V = \{X_1, ..., X_n\}$ in a BN, the joint distribution, that is shown by $P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$ can be factorized as follows:

$$\begin{aligned} P(X_1, ..., X_n) &= P(X_1) \times P(X_2|X_1) ... \times P(X_n|X_1, ..., X_{n-1}) \\ &= \prod_i P(X_i|X_1, ..., X_{i-1}) \end{aligned} \quad (1)$$

In a BN, the value of each node $X_i \in V$ is dependent on the values of its parents $P(X_1, X_2, ..., X_n) = \prod_i P(X_i|Pa(X_i))$, otherwise the node is defined as an independent node. For each node X_i, there



Dependency Graph for variable X3 with the binary domains x1 and x2

X1	X2	X3
x^1_1	x^2_1	$x^3_1 > x^3_2$
x^1_1	x^2_2	$x^3_1 < x^3_2$
x^1_2	x^2_1	$x^3_1 < x^3_2$
x^1_2	x^2_2	$x^3_1 > x^3_2$

Dependency Graph for variable X1

X1
$x^1_1 > x^1_2$

Dependency Graph for variable X2

X1	X2
x^1_1	$x^2_1 > x^2_2$
x^1_2	$x^2_1 < x^2_2$

Probability Table X3

X1	X2	P (X3=x_1)	P(X3=x_2)
x^1_1	x^2_1	0.65	0.35
x^1_1	x^2_2	0.7	0.3
x^1_2	x^2_1	0.56	0.45
x^1_2	x^2_2	0.20	0.80

Probability Table X1

P (X1=x_1)	P(X1=x_2)
0.2	0.8

Probability Table X2

X2	P (X2=x_1)	P(X2=x_2)
x^1_1	0.4	0.6
x^1_2	0.7	0.3

Figure 1: A CP-net and a Bayesian Network.

is a *conditional probability table* that shows for each possible combination of parents values the probability distribution over values of X_i.

An example of BN is shown in Figure 1 (b), where the probability that I target a Chinese x^3_1 or Mexican x^3_2 restaurant for a dinner depends on the values of his parents, that are Price (X1) and Quality of the restaurant (X1). It can be seen that the value of Quality of the restaurant depends also on the Price (X1). The conditional probability tables associated at each variable are shown in Figure 1 (b). For example, the conditional probability table associated to variable $X1$ shows that the first value in the domain of $X1$ has probability 0.2. Other conditional probability tables list the probability that the child variable/takes one of the values in its domain for each combinations of values in the domains of its parents.

3 TECHNICAL APPROACH

This section shows our approach to build a CP-net representing the conditional and qualitative preferences of a user starting from the history of the user. For the sake of clarification, we will explain how the constructor works in different sections by using examples of preferences, which are taken from the domain of the restaurant recommender system.

3.1 Feature Selection

Within hundred numbers of attributes logged in the data-set sourced by users' interactions at searching and selecting appropriate services (e.g., selecting the desired restaurant in this context), the features that are more important on influencing users' preferences should be selected. Thus, given a set of variables/features (from the data set) $V = \{X1, X2, \ldots, X_n\}$, without the loss of generality we assume that X_n is the variable corresponding to the target node of the CP-net, which is the most constrained variable.

In this context, target node points to the Type of restaurant/food, that may have as values the types of restaurants that the user has selected before and logged in his profile. We refer $X_n \equiv X_t$ as a target variable. The values of the target variables are based on the set of remaining variables $V \setminus \{X_n\}$.

Since every variable (among the other features that a user may consider to select a service/restaurant e.g., price, location, etc.) may influence users on targeting a service/restaurant, feature selection process is required to enhance the effectiveness of the framework. In other words, in feature selection, we aim to nominate a subset $V_s \subset V$ containing features that leverage the values of the target feature such that $|V_s| = m$ with a given m. Thus, we use *Information Gain*[1], which is widely used in the state of art for feature selection [21], to select the most important features to be considered in the constructor space. This method selects which feature/attribute represents the greatest information gain (more details about the function can be found in [21]). Eventually, the list of features in V_s is sorted in decreasing order to build the following list $\{X_0, X_{1s}, \ldots, X_{ms}\}$. Then, the system attaches X_t (as the target node) at the end of the list as follows $V = \{X_0, X_{1s}, \ldots, X_{ms}, X_t\}$ to achieve the desired list of features in feature selection process.

Once the process is done, the system breaks down the sorted features into three layers that are detailed in the following section.

3.2 Layers Extraction

Given the above list $V = \{X_0, X_{1s}, \ldots, X_{ms}, X_t\}$ we aim to build an acyclic and directed graph that consists of three layers: *Root layer*, *Intermediate layer*, and the *Target layer*. Hereafter, we describe in detail each layer and links connecting the nodes inside the graph. Notice that the terms "node" and "feature" refer to the same concept from now on.

- *Root Layer:* this layer contains only the root node, which is the most important feature among the others in the list. In other words, given the list "V", the first feature X_0 will be considered as the root node. Since, this node X_0 is an independent feature, it does not have any *income link* from the other nodes. As an example, considering the restaurant selection domain, where the possible options for a user in selecting a restaurant depend mainly on his budget. In this case, price of the restaurant could be the feature that corresponds to the root node.
- *Intermediate Layer:* the main procedure of extracting users' conditional preferences on the basis of the strength of relations between features under certain conditions or threshold,

[1]To use *Information Gain* function, we took the advantage of *FSelector* [20] library in R.

will be executed in this layer. This layer contains all the nodes except Root and the Target nodes as follows $\{X_{1s}, \ldots, X_{ms}\}$. To set the internal links between intermediate nodes, we need to measure the dependence between any pair of nodes (X_{is}, X_{js}). The algorithm adds a link between the these nodes that have dependence values higher or equal to a given threshold. This threshold value could be determined automatically or manually. In this paper we decide to fix this threshold manually as described in Section 4. Assume that the determined dependencies between all the features in the Intermediate layer are in [0.0 - 1], thus we can set the threshold Tr=0.7.

- *Target Layer:* as the name implies, this layer indicates the target node X_t that shows the user's preference (last attribute in the sorted list) toward the specific domain. Thus, this layer has only incoming links from the nodes, which are located in the Intermediate layer, or in the Root layer.

The action flow is shown in Figure 2, where the selected features in the list V (Fig 2 (a)) are segmented into three layers (Fig 2 (b)). Then, the proposed constructor integrates the layers (Fig 2 (c)) based upon the dependencies between the features in the second layer obtained by exploiting the score functions and the algorithm explained in the next Section 3.3.

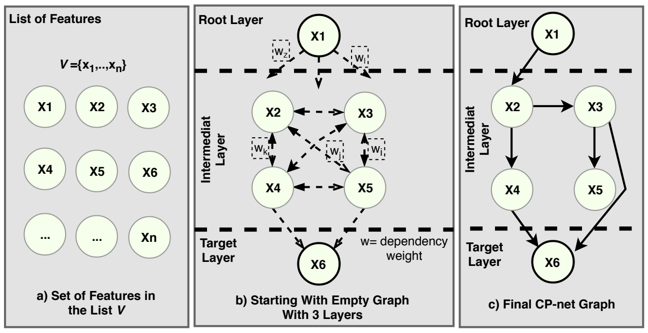


Figure 2: The CP-net Constructor Frontend.

3.3 Feature Dependency Measurement and Constructing CP-net

Due to the similarity between the concepts described above, we exploit Bayesian network's score functions to construct the main shape of users' CP-nets in the second layer.

Many algorithms and techniques have been developed to tackle the problem of building a Bayesian Network, whose performance vary according to the used score functions from data/domain to data/domain. Hence, we implement the framework by various kinds of score functions to decrease the miss classification results which lead to the suitable technique that accommodate with our data and provide the highest performance.

Score functions in Bayesian network lies into two main categories of *Information Theory Score* such as Mutual information test (MIT), Bayesian Information criterion (BIC), Akaike Information Criterion (AIC), Log Likelihood (LL), and Normalized Maximum likelihood (NMC)), and *Bayesian Score Function* such as; Bayesian Dirichlet (BD), and K2 [8].

In general, Bayesian Score Functions try to find a network B with a maximum value S_G with given data $S_G(B, D)$ [28]. In these functions, the procedure of constructing the network begins from a prior distribution and then computes the posterior probability distribution. The network with maximum number is the best network among the other possible networks. While Information Theory Functions act based on compression [2]. They attempt to recognize the most compact model over the data D with less score value, therefore, a model with minimum score is the best model among the others.

To show the constructor and how the score functions work, we use *Bayesian Information Criterion (BIC)* [22] throughout this paper, but we implement the constructor with all the possible functions (in the second layer) to evaluate the performance of the algorithm in the specific domain.

Bayesian Information Criterion [22] is the most popular score function among the others which is constructed based on likelihood function. *BIC* that is known "Schwarz criterion" model is a type of criterion model selection among the other models. However, the performance of the function highly differ from data-set to data-set, it is shown that BIC score function outperforms consistently the other score functions in constructing Bayesian network structure in different data-sets [22].

In this score function the lower value points the better fit the model, or the lower value represents the minimum information loss related to the candidate model which is shown in Equation 2.

$$BIC = -2 * ln(\theta) + ln(N) * k \tag{2}$$

where k refers to the degree of freedom to be estimated and N points the number of observations or sample size. The set of model parameters that maximize the likelihood function is shown by θ and $\ln(\theta)$ refers to the likelihood of the truth model. A more lower BIC value indicates the obtained model is more likely to be considered as the best and true network model among the others. More details about how these functions work are presented in [25]. Hence, to perform the above functions to construct the desired network in the second layer, we borrow the structures shown in [1], where the various algorithms have been discussed such as *Simulated Annealing algorithm, Heuristic algorithm, Genetic algorithms.*

Taking into account the advantage of the greedy search and heuristic algorithm [13], we use *Hill Climbing (HC)* algorithm to execute BIC, AIC, MIT, LL, BD, NMC, etc. functions to obtain the structure of the users' CP-nets in the second layer following the proposed structure shown in Algorithm 1.

Recalling the feature selection and breaking the list "V" into three layers, *HC* starts with an empty graph (G) in the second layer and attempts to find a model with high score (or minimum score) by incrementally searching among the other possible models from its local neighbors:

In other words, this is an optimization model that begins with an arbitrary structure of the network, then try to find a better network by incrementally tuning the scores. Hence, if a new model with high/minimum score is found, it will be substituted with the old model. Algorithm 1:

These steps are repeated until no further model with high/minimum score can be found. Although the algorithm has the problem of getting stuck in the local region that depends on the starting point, we took the privilege of its high performance and accuracy to build the network.

In the following section we show how connecting the layers to define the CP-net.

3.4 Converting Probability to Preferences

This section show how to interpret the strength correlation between nodes to user's preferences under the concept of CP-net. To this end, the system determines the *"maximum"* probability of a feature's value as the user's preference. It means if a node from the list "V" e.g., Price contains two values in the domain such as *cheap and expensive*, the system from the conditional probability table (in Bayesian Network) that is computed for each node in the list, picks the value/domain (e.g., cheap) that has highest probability as the user's preference (see the table associated to $X2$ in Figure 3). Once the structure of the network is obtained in the second layer, the system converts the probability into preference as shown below.

The procedure starts from the independent nodes. The highest probable value for a feature will be considered as the preferred value among the other possible values. The independent nodes (as parents) influence the value of the remaining nodes (as children) on basis of the probability table as shown in Figure 3. To better understand the conversion let us consider $v = \{X2,\ X3,\ X4\}$ $v \subset V$ with 3 features and their binary domains $D(X2) = \{x_1^2, x_2^2\}, D(X3) = \{x_1^3, x_2^3\}$ and $D(X4) = \{x_1^4, x_2^4\}$. As it is shown in Figure 3, $X2$ influences $X3$ and $X4$ in the second layer, and similarly $X4$ influences $X3$. Hence, $X3$ is identified as the most dependent node among the others with its probability table. The conditional probability table associated to $X3$ can be used to derive preferences over values in $D(X2)$ in the CP-net. In the conditional preference table of the CP-net associated to $X2$ we have x_2^2 more preferred than x_1^2 since the probability of x_2^2 is 0.8 while the probability of x_1^2 is 0.2. Thus, if x_2^2 is preferred for $X2$, then x_1^4 with probability 0.7 is the more preferred value for $X4$. Similarly, the value of $X3$ depends on the value $X2$ and $X4$ according to the probabilities. Here, we just gave a simple example of features with binary domains but this could be extended with n numbers of values in the domains for each feature in the sorted list V.

Algorithm 1: CPnet Constructor Algorithm

```
Rnode          ▷ root node
Inodes         ▷ intermediate nodes
Tnode          ▷ target node
Score G        ▷ score G
ScoreNG        ▷ new score G
Function start  with empty graph G
    $ScoreNG \leftarrow Score(G)$
    $Initialize \quad ScoreNG \quad with \quad G$
    while $Score(G) \quad not \quad minimize$ do
        $provide \quad acyclic \quad operation \quad G:$
        $add, delete, reverse$
        if $ScoreNG > Score(G)$ then
            $ScoreNG \leftarrow ScoreNG$
    $DepMatrix \leftarrow dependency(Rnode, Tnode, Inodes)$
    if $Depmatrix(Rnode, Tnode, Inodes) \geq Tr$ then
        $connectRnode \rightarrow InodesandTnode$
        if $DepMtx(Inodes, Tnode)$ then
            $connectInodes \rightarrow Tnode$
```

```
Function start  with empty graph G
    $ScoreNG \leftarrow Score(G)$
    $Initialize \quad ScoreNG \quad with \quad G$
    while $Score(G) \quad not \quad minimize$ do
        ⋮
```

```
if $ScoreNG > Score(G)$ then
    $ScoreNG \leftarrow ScoreNG$
```

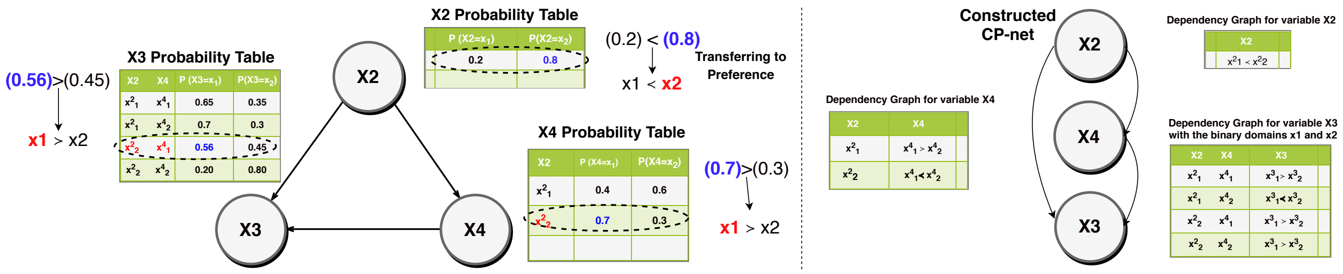
3.5 Connecting the Layers

This section is in charge of integrating the Root and the Target nodes to the nodes that are located in the Intermediate layer. Since, the Root node dominates the other nodes in the two layers, the system generates a matrix of dependency between them. Practically, the aim is to find the strength relations between the Root and the rest, hence we use *Chi-square*, *Information gain*, *Gain ratio* and *Symmetrical uncertainty* functions [21] to obtained these dependencies. This process of generating the dependency matrix is broken down into two sections. First, is applied between the Root layer and "Intermediate and Target" layers (both together, as the Root node dominates the rest of the nodes which are located in these two layers). Secondly, it will be employed between the Intermediate layer and the Target layer. Having at hand these two dependency matrix, we set a threshold $CV = [0-1]$ as a Confidence Value to eliminate the links between these layers which can not meet the threshold value. This helps to find out the most important dependencies within the user CP-net that can characterize the user preferences. The described work-flow produces the final user's CP-net that characterizes the user' preferences.

To validate the correctness of the obtained CP-net, we carried out the cross validation method on a real data-set that is detailed in the next section.

4 EXPERIMENTAL EVALUATION AND RESULTS

The main task of recommender systems is to provide the best personalized service for users in various domains. In this paper, we validate our procedure for constructing users' CP-nets from their past behaviors in the restaurant recommendation domain. In the following, we describe in detail the experimental setup.


(0.56)>(0.45)
x1 > x2
X3 Probability Table
X2 Probability Table
(0.2) < (0.8)
x1 < x2
Transferring to Preference
X4 Probability Table
(0.7)>(0.3)
x1 > x2
X2
X3
X4
Constructed CP-net
Dependency Graph for variable X4
Dependency Graph for variable X2
Dependency Graph for variable X3 with the binary domains x1 and x2

X3 Probability Table

X2	X4	P (X3=x_1)	P(X3=x_2)
x^2_1	x^4_1	0.65	0.35
x^2_1	x^4_2	0.7	0.3
x^2_2	x^4_1	0.56	0.45
x^2_2	x^4_2	0.20	0.80

X2 Probability Table

P (X2=x_1)	P(X2=x_2)
0.2	0.8

X4 Probability Table

X2	P (X4=x_1)	P(X4=x_2)
x^2_1	0.4	0.6
x^2_2	0.7	0.3

Dependency Graph for variable X4

X2	X4
x^2_1	$x^4_1 > x^4_2$
x^2_2	$x^4_1 < x^4_2$

Dependency Graph for variable X2

X2
$x^2 1 < x^2 2$

Dependency Graph for variable X3 with the binary domains x1 and x2

X2	X4	X3
x^2_1	x^4_1	$x^3_1 > x^3_2$
x^2_1	x^4_2	$x^3_1 < x^3_2$
x^2_2	x^4_1	$x^3_1 > x^3_2$
x^2_2	x^4_2	$x^3_1 > x^3_2$

Figure 3: From a Bayesian network to the corresponding CP-net.

4.1 Data Collection

We exploited two real datasets of users' preferences over the set of restaurants in Mexico City, and Movie in Denmark and UK, from UCL repository[2]. The former is statistics on users' restaurant selections. It contains 1116 samples where each sample carries 11 different features. Each sample indicates the selection of one restaurant performed by a user. The latter is statistics on users' movie ratings. Movie dataset includes 2297 instances with around 122 unique users where every sample holds 28 different features. Due to the page limit, we only present features in the restaurant selection dataset in Table 1, which shows the description of the features and their abbreviations.

Then we group the samples that belong to the same user. Next, we exclude the users who have recorded less than 10 selections or ratings. This is due to the fact that a small number of samples does not allow the system to efficiently learn and make inference. Thus, we select only 18 users with at least 10 recorded samples in the first dataset. While, the users whose ratings are logged in the movie dataset have enough instances, so all the unique players are included.

4.2 Evaluation & Model Selection

To achieve the best CP-net structure representing users' preferences, we execute the algorithm with a range of parameters, where each combination of parameter values specifies a specific structure of CP-net. For each round, to find the best model to make inference, the grid search technique is employed to find the optimal tuple of hyperparameter values.

Evaluation Setting for Dataset 1 (Restaurant): Since we have a limited number of samples for every user in Restaurant dataset, the iterative leave-one-out cross validation is the best option to use for our multi-class classification problem. In each round, one sample is left out to be considered as the test set and the remaining samples are used to learn a CP-net using our proposed method and to train the model using Bayesian Method. In the following, we have listed the parameters and the set of the values that we have used to construct multiple CP-nets.

- Number of selected features: We have run the algorithm with various number of features to find out the right number that can provide the best result. Thus we start from all features "11", and then we gradually reduce it to the top 4 important features in the list.
- Dependency: $\{0.5,\ 0.6,\ 0.7\}$: Since we have defined the dependencies between features in this interval [0-1], we tune the threshold similar to the number of features to find the best results.
- Direction: $\{0.5,\ 0.6,\ 0.7\}$: We also tuned the algorithm with three values to obtain the direction of the dependency between the features.
- Threshold (Tr) for connecting the first and the second layer: $\{0.03,\ 0.04,\ 0.05\}$. As mentioned above, we use different functions to calculate the dependencies between layers, thus we may have different values of threshold.
- Threshold (Tr) for connecting the second and the third layer: $\{0.03,\ 0.04,\ 0.05\}$

Evaluation Setting for Dataset 2 (Movie): In contrast to dataset 1, the unique users in Movie dataset contains fairly enough instances. Thus, 10-fold cross validation is implemented to validate the stability of the approach by feeding new data. In each round, 10% of data is left out to be considered as the test set and the rests −90%− are used to learn a CP-net using our proposed constructor and to train the model exploiting Bayesian Method. Similar settings and parameters have set for number of selected features, dependency, direction and Thresholds.

Once the model is trained, the algorithm computes a score e.g., for each restaurant and movie (rating) option. This score shows how probable is for the restaurant to be considered according the user's interest. Then the system selects a restaurant which has the highest probability value as the user's preference.

To evaluate the performance of the method we have used the following well-known metrics: (i) *Precision*, that is the fraction of retrieved instances that are relevant, (ii) *Recall*, that is the fraction of relevant instances that are retrieved, and (iii) *F-measure* that is a weighted average of precision and recall [27].

The performance of each class is computed (for every user), and then an average over all classes performance is considered as the performance of the method for the user.

4.3 Results

Table 2. presents our preliminary experimental results in the restaurant recommendation domain. In particular, it shows the performance of our approach by considering various score functions.

[2]https://archive.ics.uci.edu/ml/datasets.html

Table 1: Data-set Information

#	Features	Restaurant Selection: Description and domain
1	Ub	users budget (low, medium, high)
2	Q	quality of restaurant (low, medium and high)
3	Al	the restaurant serves alcohol or not (yes or not)
4	Sm	smoking area inside of the restaurant (yes or no)
5	Ac	accessibility of the restaurant (easy, medium or difficult)
6	Pr	restaurant price (low, medium and high)
7	Ar	outdoor restaurant or indoor restaurant
8	Os	other services, like internet, etc (yes or not)
9	Pk	the restaurant has parking area or not (yes or no)
10	Ur	restaurant rate which are given by user/ people (low, medium and high)
11	Tr	type of restaurant (Italian, Latin-American, Asian)

Such a performance is measured in terms of "Precision, Recall, and F-measure".

Result For Dataset 1: As it is shown in Table 2, within the score functions implemented by the hill-climbing algorithm, only BIC, AIC and MDL were applicable in our data-set. The parameters, which are automatically set to generate such results vary from each other, since the functions use different strategy to calculate their scores in constructing BN. BIC has set by 6 number of features, 0.7 arc dependency and 0.03 dependency between layers, while AIC provides its best result by 7 number of features, 0.6 for arc dependency and 0.05 for layer dependency. The values of precision and recall using BIC and AIC are very similar with ~0.70, however BIC performs slightly better both in terms of precision and recall. In contrast, MDL provides a pour results by 0.54, 0.57 and 0.555 in precision, recall and f-measure, respectively.

It is noticeable that BIC, AIC and MDL with a smaller and greater numbers of features (reported in Table 2.) result mostly incomplete or/and a very complex and cycle graph. In addition, we have not inserted results for K2 and Loglike since the approach with these score functions often construct CP-nets with cycles, specially after appending the nodes of the layers in the final step.

Table 2: The Performance of The Proposed Method on The Restaurant Data-set.

	Score Function	Precision	Recall	F-measure	Description	# of Features	Arc Tr	Dep TR
Dataset 1	BIC	0.71	0.708	0.706	applicable	6	0.7	0.03
	AIC	0.70	0.69	0.694	applicable	7	0.6	0.05
	Loglik	x	x	x	not-applicable	x	x	x
	K2	x	x	x	causing cycle	x	x	x
	MDL	0.54	0.57	0.555	applicable	4	0.5	0.05
Dataset 2	BIC	0.51	0.52	0.52	applicable	8	0.4	0.03
	AIC	x	x	x	causing cycle	x	x	x
	Loglik	0.54	0.43	0.485	applicable	8	0.5	0.03
	K2	0.55	0.42	0.485	applicable	7	0.6	0.03
	MDL	x	x	x	not-applicable	x	x	x

Result For Dataset 2: Analogous to the first evaluation, *BIC* function works slightly better in all metrics such as precision with 0.51, recall with 0.52, f-measure by 0.52 and ~0.50 accuracy, w.r.t *K2* and *Loglike* on this users' statistical data (Figure 4. shows a complex cp-net that is constructed by BIC with 8 features in r). However, the figures show a weak result for *BIC* function compared to the first assessment on Restaurant dataset.

In contrast to the first experiment, both *K2* and *Loglike* were correctly implemented by reporting around ~0.49 in F-measure. These low figures might have obtained due to existing 5 classes in the target layer (target node has 5 different domains including *Very Bad, Bad, Medium, Good* and *Very Good* which indicate the users' rating on different movies), which make classification and prediction arduous. Taking into account 20% accuracy as the baseline for random classification, the obtained figures also show around 30% above the baseline that is an admissible result. However further investigation is required both in terms of design and evaluation to increase and assess the performance of the constructor.

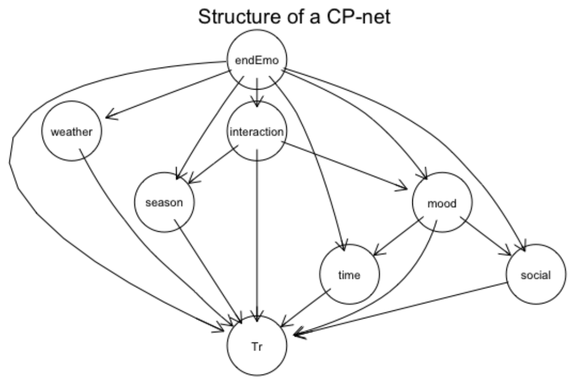


Figure 4: A complex CP-net with 8 Features.

5 CONCLUSION

We have presented a system for automatically constructing CP-nets modeling users' preferences from their past interaction with a service provider. To construct the user CP-net we have first constructed a Bayesian network. We have exploited an heuristic algorithm *Hill Climbing* to execute various score functions, such as BIC, AIC, Loglik, K2, MDL, in order to recognize the best graph model among all the possible models. Empirical results from restaurant selection ad Movie domains have shown that this constructor may have a significant impact to enhance users' satisfaction.

This construction may be very useful when the recommender system must deal with context and the users' preferences may change over time. We plan to integrate the proposed constructor in the Self-adaptive Context-Aware recommender system (SaCARS) presented in [18]. This integration allows SaCARS to completely learn and model the users' conditional preferences from their behavior without human interference.

REFERENCES

[1] [n. d.]. Artificial intelligence: A modern approach author. ([n. d.]).

[2] Hirotogu Akaike. 1998. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*. Springer, 199–213.

[3] Xavier Amatriain and Justin Basilico. 2016. Past, Present, and Future of Recommender Systems: An Industry Perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 211–214. https://doi.org/10.1145/2959100.2959144

[4] Damien Bigot, Jérôme Mengin, and Bruno Zanuttini. 2014. Learning Probabilistic CP-nets from Observations of Optimal Items.. In *STAIRS*. 81–90.

[5] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. 2004. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *J. Artif. Int. Res.* 21, 1 (Feb. 2004), 135–191. http://dl.acm.org/citation.cfm?id=1622467.1622473

[6] Yuriy Brun, Ron Desmarais, Kurt Geihs, Marin Litoiu, Antonia Lopes, Mary Shaw, and Michael Smit. 2013. A design space for self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*. Springer, 33–50.

[7] Luis M de Campos. 2006. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research* 7, Oct (2006), 2149–2187.

[8] Alexandra M Carvalho. 2009. Scoring functions for learning Bayesian networks. *Inesc-id Tec. Rep* (2009).

[9] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. 2010. Learning ordinal preferences on multiattribute domains: The case of CP-nets. In *Preference learning*. Springer, 273–296.

[10] Cristina Cornelio, Judy Goldsmith, Nicholas Mattei, Francesca Rossi, and K Brent Venable. 2013. Dynamic and probabilistic cp-nets. *mpref 2013* (2013).

[11] Anind K. Dey. 2001. Understanding and Using Context. *Personal Ubiquitous Comput.* 5, 1 (Jan. 2001), 4–7. https://doi.org/10.1007/s007790170019

[12] Yannis Dimopoulos, Loizos Michael, and Fani Athienitou. 2009. Ceteris Paribus Preference Elicitation with Predictive Guarantees.. In *IJCAI*, Vol. 9. 1–6.

[13] Thomas A Feo and Mauricio GC Resende. 1995. Greedy randomized adaptive search procedures. *Journal of global optimization* 6, 2 (1995), 109–133.

[14] Yanjie Fu, Bin Liu, Yong Ge, Zijun Yao, and Hui Xiong. 2014. User preference learning with multiple information fusion for restaurant recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 470–478.

[15] Joshua T Guerin, Thomas E Allen, and Judy Goldsmith. 2013. Learning CP-net preferences online from user queries. In *International Conference on Algorithmic DecisionTheory*. Springer, 208–220.

[16] Finn V Jensen. 1996. *An introduction to Bayesian networks*. Vol. 210. UCL press London.

[17] Reza Khoshkangini, Maria Silvia Pini, and Francesca Rossi. 2016. A design of context-aware framework for conditional preferences of group of users. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. Springer, 97–112.

[18] Reza Khoshkangini, Maria Silvia Pini, and Francesca Rossi. 2016. A self-adaptive context-aware group recommender system. In *Conference of the Italian Association for Artificial Intelligence*. Springer, 250–265.

[19] Frédéric Koriche and Bruno Zanuttini. 2010. Learning Conditional Preference Networks. *Artif. Intell.* 174, 11 (July 2010), 685–703. https://doi.org/10.1016/j.artint.2010.04.019

[20] Miron B Kursa, Witold R Rudnicki, et al. 2010. Feature selection with the Boruta package. *J Stat Softw* 36, 11 (2010), 1–13.

[21] Changki Lee and Gary Geunbae Lee. 2006. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information processing & management* 42, 1 (2006), 155–165.

[22] Zhifa Liu, Brandon Malone, and Changhe Yuan. 2012. Empirical evaluation of scoring functions for Bayesian network model selection. *BMC bioinformatics* 13, 15 (2012), S14.

[23] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender System Application Developments. *Decis. Support Syst.* 74, C (June 2015), 12–32. https://doi.org/10.1016/j.dss.2015.03.008

[24] Alberto Marchetti-Spaccamela, Andrea Vitaletti, Luca Becchetti, and Ugo Colesanti. 2008. Self-Adaptive Recommendation Systems: Models and Experimental Analysis. *2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* 00 (2008), 479–480. https://doi.org/doi.ieeecomputersociety.org/10.1109/SASO.2008.55

[25] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

[26] Chihiro Ono, Mori Kurokawa, Yoichi Motomura, and Hideki Asoh. 2007. A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion. In *Proceedings of the 11th International Conference on User Modeling (UM '07)*. Springer-Verlag, Berlin, Heidelberg, 247–257. https://doi.org/10.1007/978-3-540-73078-1_28

[27] David Martin Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. (2011).

[28] Marco Scutari. 2009. Learning Bayesian networks with the bnlearn R package. *arXiv preprint arXiv:0908.3817* (2009).