Mining User Contextual Preferences

Sandra de Amo, Marcos L. P. Bueno, Guilherme Alves, Nádia F. Silva

Universidade Federal de Uberlândia, Brazil
deamo@ufu.br, marcos@facom.ufu.br, guilhermealves@comp.ufu.br, nadia.felix@gmail.com

Abstract. User preferences play an important role in database query personalization since they can be used for sorting and selecting the objects that most fulfill the user wishes. In most situations user preferences are not static and may vary according to a multitude of user contexts. Automatic tools for extracting contextual preferences without bothering the user are desirable. In this article, we propose CPrefMiner, a mining technique for mining user contextual preferences. We argue that contextual preferences can be naturally expressed by a Bayesian Preference Network (BPN). The method has been evaluated in a series of experiments executed on synthetic and real-world datasets and proved to be efficient to discover user contextual preferences.

Categories and Subject Descriptors: H.Information Systems [**H.m. Miscellaneous**]: Databases

Keywords: context-awareness, data mining, preference mining

1. INTRODUCTION

Elicitation of Preferences is an area of research that has been attracting a lot of interest within the database and AI communities in recent years. It consists basically in providing the user a way to inform his/her choice on pairs of objects belonging to a database table, with a minimal effort for the user. Preference elicitation can be formalized under either a *quantitative* [Burges et al. 2005; Crammer and Singer 2001; Joachims 2002] or a *qualitative* [Jiang et al. 2008; Koriche and Zanuttini 2010; de Amo et al. 2012b; Holland et al. 2003] framework. In order to illustrate the quantitative formulation, consider we are given a collection of movies and we wish to know which films are most preferred by a certain user. For this, we can ask the user to rate each movie and after that we simply select those films with the higher score. This method may be impractical when dealing with a large collection of movies. In order to accomplish the same task using a *qualitative* formulation of preferences, we can ask the user to inform some generic rules that reflect his/her preferences. For example, if the user says that he/she prefers romance movies to drama movies, then we can infer a class of favorite movies without asking the user to evaluate each film individually.

A qualitative framework for preference elicitation consists in a mathematical model able to express user preferences. In this article, we consider the *contextual preference rules* (cp-rules) introduced by [Wilson 2004]. This formalism is suitable for specifying preferences in situations where the choices on the values of an attribute depend on the values of some other attributes (context). For example in our movie database scenario, a user can specify his/her preference concerning the attribute *gender* depending on the value of the attribute *director*: For movies whose director is *Woody Allen* he/she prefers *comedy* to *suspense* and for movies from director *Steven Spielberg* he/she prefers *action* films to *drama*.

On both frameworks for expressing preferences (quantitative or qualitative), it is important to develop strategies to avoid the inconvenience for the user to report his/her preferences explicitly, a process that can be tedious and take a long time, causing the user not willing to provide such

We thank the Brazilian Research Agencies CNPq, CAPES (SticAmSud Project 016/09) and FAPEMIG for supporting this work.

information. In this context, the development of preference mining techniques allowing the automatic inference of user preferences becomes very relevant.

In this article we propose the algorithm *CPrefMiner*, a qualitative method for mining a set of *probabilistic* contextual preference rules modeled as a *preference bayesian network* (BPN). It extends the preliminary version presented in [de Amo et al. 2012a] with the following new features in Section 5: (1) Four new databases have been considered in the tests as well as two different validation protocols; (2) All the experiments have been executed over two baseline algorithms (classical classifiers) in order to show the superiority of CPrefMiner performance as well as the fact that classifiers are not suitable for preference mining tasks; (3) Two parameters aiming at calibrating the user's indecision and inconsistency have been introduced and a set of experiments have been carried out varying these parameters.

2. RELATED WORK

An extensive text presenting different theoretical approaches and techniques for preference learning can be found in [Fürnkranz and Hüllermeier 2011]. Roughly speaking, preference learning can be divided into two distinct problems: *label ranking* and *object ranking*. The problem of *Label ranking* consists in discovering rules relating user's personal information to the way they rank labels. The work of [Hüllermeier et al. 2008] discusses the differences underlying both problems and proposes a method for label ranking consisting in training a set of binary classifiers. On the other hand, *object ranking* aims at predicting which is the preferred object between two given objects. The present article focuses on this latter problem.

In [Holland et al. 2003] the authors propose a technique for mining user preferences, based on a qualitative approach, whose underlying model is the *pareto preference model*. The preference rules are obtained from log data generated by the server when the user is accessing a web site. Another approach to preference mining is presented in [Jiang et al. 2008]. In this work the authors propose using preference samples provided by the user to infer an order on any pair of tuples in the database. Such samples are classified into two categories, the *superior* and *inferior* samples and contain information about some preferred tuples and some non-preferred ones. From these rules, an order is inferred on the tuples. The underlying preference model is the *pareto preference model* as in [Holland et al. 2003]. In this model, preferences are not conditional or contextual, that is, preferences on values of attributes do not depend on the values of other attributes. Our contextual preference model is more expressive.

Concerning the topic of mining contextual preference rules, [Koriche and Zanuttini 2010] proposes a method for mining a CP-Net model [Boutilier et al. 2004] from a set of preferences supplied by the user. Like in our approach, preference samples are represented by ordered pairs of objects. The goal is to identify a target preference ordering with a binary-valued CP-net by interacting with the user through a small number of queries. In [de Amo et al. 2012b] some of the authors of the present article proposed a different method (ProfMiner), based on pattern mining techniques, to discover user *profiles* specified by a set of preference rules. The main advantage of CPrefMiner over ProfMiner is that it produces a compact preference model (Bayesian Preference Network), which induces a strict partial order over the set of tuples. Besides, the performance results for both algorithms do not differ significantly, although ProfMiner presents slightly better results than CPrefMiner.

3. PROBLEM FORMALIZATION

A *preference relation* on a finite set of objects $A = \{a_1, a_2, ..., a_n\}$ is a strict partial order over A, that is a binary relation $R \subseteq A \times A$ satisfying the irreflexivity and transitivity properties. Typically, a strict partial order is represented by the symbol $>$. Considering $>$ as a preference relation, we denote by $a_1 > a_2$ the fact that a_1 is preferred to a_2.

Definition 3.1 *Preference Database.* Let $R(A_1, A_2, ..., A_n)$ be a relational schema. Let $\text{Tup}(R)$ be the set of all tuples over R. A *preference database* over R is a finite set $\mathcal{P} \subseteq \text{Tup}(R) \times \text{Tup}(R)$ which is *consistent*, that is, if $(u, v) \in \mathcal{P}$ then $(v, u) \notin \mathcal{P}$. The pair (u, v), usually called a bituple, represents the fact that the user prefers *the tuple u to the tuple v*.

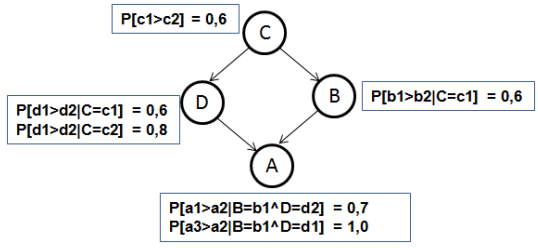
Example 3.2. Let $R(A, B, C, D)$ be a relational schema with attribute domains given by $\mathbf{dom}(A) = \{a_1, a_2, a_3\}$, $\mathbf{dom}(B) = \{b_1, b_2\}$, $\mathbf{dom}(C) = \{c_1, c_2\}$ and $\mathbf{dom}(D) = \{d_1, d_2\}$. Let I be an instance over R as shown in Figure 1(a). Figure 1(b) illustrates a preference database over R, representing a sample provided by the user about his/her preferences over tuples of I.

Id	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_2	b_1	c_1	d_2
t_4	a_1	b_2	c_1	d_2
t_5	a_2	b_1	c_2	d_1
t_6	a_3	b_1	c_1	d_1

(a)

(t_1,t_2)
(t_1,t_3)
(t_4,t_5)
(t_4,t_2)
(t_5,t_6)
(t_3,t_5)
(t_4,t_1)

(b)



(c)

Fig. 1: (a) An instance I, (b) A Preference Database $\mathcal{P}$, (c) Preference Network $\mathbf{PNet}_1$

The main objective of this article is to extract a *contextual preference model* from a preference database provided by the user. The contextual preference model is specified by a *Bayesian Preference Network* defined next.

Definition 3.3 *Bayesian Preference Network (BPN).* A *Bayesian Preference Network* (or BPN for short) over a relational schema $R(A_1, ..., A_n)$ is a pair (G, θ) where: (1) G is a directed acyclic graph whose nodes are attributes in $\{A_1, ..., A_n\}$ and the edges stand for attribute dependency; (2) θ is a mapping that associates to each node of G a *conditional probability table of preferences*, that is, a finite set of conditional probabilities of the form $P[E_2|E_1]$ where: (i) E_1 is an event of the form $(A_{i_1} = a_{i_1}) \wedge \ldots \wedge (A_{i_k} = a_{i_k})$ such that $\forall j \in \{1, ..., k\}$, $a_{i_j} \in \mathbf{dom}(A_{i_j})$, and (ii) E_2 is an event of the form "$(B = b_1)$ is preferred to $(B = b_2)$"[1], where B is an attribute of R, $B \neq A_{i_j}$ $\forall j \in \{1, ..., k\}$ and $b_1, b_2 \in \mathbf{dom}(B)$, $b_1 \neq b_2$.

Example 3.4 *BPN.* Let $R(A, B, C, D)$ be the relational schema of Example 3.2. Figure 1(c) illustrates a preference network $\mathbf{PNet}_1$ over R.

Each conditional probability $P[E2|E1]$ in a BPN table stands for a *probabilistic* contextual preference rule (cp-rule), where the condition event E_1 is the *context* and the event E_2 is the *preference*. A probabilistic contextual preference rule associated to a node X in the graph G represents a degree of belief of preferring some values for X to other ones, depending on the values assumed by its parents in the graph. For instance $P[D = d_1 > D = d_2|C = c_1] = 0.6$ means that the probability of $D = d_1$ be preferred to $D = d_2$ is 60% given that $C = c_1$.

The quality of a BPN as an ordering tool is measured by means of its *precision* and *recall*. In order to properly define the precision and recall of a preference network, we need to define the *strict partial order* inferred by the preference network. For lack of space, we do not provide the rigorous definition of the order here, but only describe it by means of an example. For more details see [de Amo and Pereira 2011].

[1]For the sake of simplifying the presentation we often say $b_1 > b_2$ instead of "$(B = b_1)$ is preferred to $(B = b_2)$".

Example 3.5 *Preference Order.* Let us consider the BPN $\mathbf{PNet}_1$ depicted in Figure 1(c). This BPN allows to infer a preference ordering on tuples over $R(A, B, C, D)$. According to this ordering, tuple $u_1 = (a_1, b_1, c_1, d_1)$ is preferred to tuple $u_2 = (a_2, b_2, c_1, d_2)$. In order to conclude that, we execute the following steps: (1) Let $\Delta(u_1, u_2)$ be the set of attributes where the u_1 and u_2 differ. In this example, $\Delta(u_1, u_2) = \{A, B, D\}$; (2) Let $\min(\Delta(u_1, u_2)) \subseteq \Delta$ such that the attributes in $\min(\Delta)$ have no ancestors in Δ (according to graph G underlying the BPN $\mathbf{PNet}_1$). In this example $\min(\Delta(u_1, u_2)) = \{D, B\}$. In order to u_1 be preferred to u_2 it is necessary and sufficient that $u_1[D] > u_2[D]$ and $u_1[B] > u_2[B]$; (3) Compute the following probabilities: p_1 = probability that $u_1 > u_2$ = $P[d_1 > d_2 | C = c_1] * P[b_1 > b_2 | C = c_1]$ = 0.6 * 0.6 = 0.36; p_3 = probability that $u_2 > u_1$ = $P[d_2 > d_1 | C = c_1] * P[b_2 > b_1 | C = c_1]$ = 0.4 * 0.4 = 0.16; p_2 = probability that u_1 and u_2 are incomparable = $P[d_1 > d_2 | C = c_1] * P[b_2 > b_1 | C = c_1] + P[d_2 > d_1 | C = c_1] * P[b_1 > b_2 | C = c_1]$ = 0.6*0.4 + 0.4*0.6 = 0.48. In order to compare u_1 and u_2 we focus only on p_1 and p_3 (ignoring the "degree of incomparability" p_2) and select the higher one. In this example, $p_1 > p_3$ and so, we infer that u_1 is preferred to u_2. If $p_1 = p_3$ we conclude that u_1 and u_2 are incomparable.

Definition 3.6 *Precision and Recall.* Let **PNet** be a BPN over a relational schema R. Let $\mathcal{P}$ be a preference database over R. The *recall* of **PNet** with respect to $\mathcal{P}$ is defined by Recall(**PNet**,$\mathcal{P}$)= $\frac{N}{M}$, where M is the cardinality of $\mathcal{P}$ and N is the amount of pairs of tuples $(t_1, t_2) \in \mathcal{P}$ compatible with the preference ordering inferred by **PNet** on the tuples t_1 and t_2. That is, the recall of **PNet** with respect to $\mathcal{P}$ is the percentage of elements in $\mathcal{P}$ which are correctly ordered by **PNet**. The *precision* of **PNet** is defined by Precision(**PNet**,$\mathcal{P}$)= $\frac{N}{K}$, where K is the set of elements of $\mathcal{P}$ which are comparable by **PNet**. That is, the precision of **PNet** with respect to $\mathcal{P}$ is the percentage of *comparable* elements of $\mathcal{P}$ (according to **PNet**) which are correctly ordered by **PNet**.

The Mining Problem we treat in this article is the following: Given a training preference database T_1 over a relational schema R and a testing preference database T_2 over R, find a BPN over R having good precision and recall with respect to T_2.

4. ALGORITHM CPREFMINER

The task of constructing a Bayesian Network from data has two phases: (1) the construction of a directed acyclic graph G (the network structure) and (2) the computation of a set of parameters θ representing the conditional probabilities of the model. This work adopts a score-based approach for structure learning.

4.1 Score Function

The main idea of the score function is to assign a real number in $[-1, 1]$ for a candidate structure G, aiming to estimate how good it captures the dependencies between attributes in a preference database $\mathcal{P}$. In this sense, each network arc is "punished" or "rewarded", according to the matching between each arc (X, Y) in G and the corresponding *degree of dependence* of the pair (X, Y) w.r.t. $\mathcal{P}$.

4.1.1 *The Degree of Dependence of a Pair of Attributes.* The *degree of dependence* of a pair of attributes (X, Y) with respect to a preference database $\mathcal{P}$ is a real number that estimates how preferences on values for the attribute Y are influenced by values for the attribute X. Its computation is carried out as described in Alg. 1. In order to facilitate the description of Alg. 1 we introduce some notations as follows: (1) For each $y, y' \in \mathbf{dom}(Y)$, $y \neq y'$ we denote by $T_{yy'}$ the subset of bituples $(t, t') \in \mathcal{P}$, such that $t[Y] = y \wedge t'[Y] = y'$ or $t[Y] = y' \wedge t'[Y] = y$; (2) We define support($(y, y')$,$\mathcal{P}$) = $\frac{|T_{yy'}|}{|\mathcal{P}|}$. We say that the pair $(y, y') \in \mathbf{dom}\ (Y) \times \mathbf{dom}\ (Y)$ is *comparable* if support((y, y'),$\mathcal{P}$) $\geq \alpha_1$, for a given threshold α_1, $0 \leq \alpha_1 \leq 1$; (3) For each $x \in \mathbf{dom}(X)$, we denote by $S_{x|(y,y')}$ the subset of $T_{yy'}$ containing the bituples (t, t') such that $t[X] = t'[X] = x$; (4) We define support($S_{x|(y,y')}$,$\mathcal{P}$) = $\frac{|S_{x|(y,y')}|}{|\bigcup_{x' \in \mathbf{dom}(X)} S_{x'|(y,y')}|}$; (5) We say that x is a *cause for* (y, y') *being comparable* if support($S_{x|(y,y')}$,$\mathcal{P}$) $\geq \alpha_2$, for a given threshold α_2, $0 \leq \alpha_2 \leq 1$.

Algorithm 1: The degree of dependence of a pair of attributes

Input: $\mathcal{P}$: a preference database; (X, Y): a pair of attributes; two thresholds $\alpha_1 > 0$ and $\alpha_2 > 0$.
Output: The Degree of Dependence of (X, Y) with respect to $\mathcal{P}$

for *each pair* $(y, y') \in$ ***dom***$(Y) \times$ ***dom***(Y), $y \neq y'$ *and* (y, y') *comparable* **do**
  for *each* $x \in$ ***dom***(X) *where* x *is a cause for* (y, y') *being comparable* **do**
    Let $f_1(S_{x|(y,y')}) = \max\{N, 1 - N\}$, where
    $N = \dfrac{|\{(t, t') \in S_{x|(y,y')} : t > t' \wedge (t[Y] = y \wedge t'[Y] = y')\}|}{|S_{x|(y,y')}|}$
  Let $f_2(T_{yy'}) = \max\{f_1(S_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$
Let $f_3((X, Y), \mathcal{P}) = \max\{f_2(T_{yy'}) : (y, y') \in \mathbf{dom}(Y) \times \mathbf{dom}(Y),\ y \neq y',\ (y, y')$ comparable$\}$
return $f_3((X, Y), \mathcal{P})$

4.1.2 *Score Function Calculus.* Given a structure G and a preference database $\mathcal{P}$ with n attributes, we define $score(G, \mathcal{P})$ as $score(G, \mathcal{P}) = \dfrac{\sum_{X,Y} g((X, Y), G)}{n(n-1)}$ (1)

where X and Y are attributes in a relational schema R. The function g is calculated by the following set of rules: (a) If $f_3((X, Y), G) \geq 0.5$ and edge $(X, Y) \in S$, then $g((X, Y), G) = f_3((X, Y), G)$; (b) If $f_3((X, Y), G) \geq 0.5$ and edge $(X, Y) \notin S$, then $g((X, Y), G) = -f_3((X, Y), G)$; (c) If $f_3((X, Y), G) < 0.5$ and edge $(X, Y) \notin S$, then $g((X, Y), G) = 1$; (d) If $f_3((X, Y), G) < 0.5$ and edge $(X, Y) \in S$, then $g((X, Y), G) = 0$.

Example 4.1. Let us consider the preference database $\mathbf{PrefDb}_1 = \{(t_1, t'_2), \ldots, (t_{13}, t'_{13})\}$, where $t > t'$ for every bituple (t, t') in $\mathbf{PrefDb}_1$ (table at the left side of Figure 2), constructed over a relational schema $R(A, B, C)$, where $\mathbf{dom}\ (A) = \{a_1, \ldots, a_4\}$, $\mathbf{dom}\ (B) = \{b_1, \ldots, b_5\}$ and $\mathbf{dom}\ (C) = \{c_1, \ldots, c_6\}$. In order to compute the degree of dependence of the pair (A, C) with respect to $\mathbf{PrefDb}_1$, we first identify the sets $T_{c_1,c_3} = \{(t_1, t'_1)\}$, $T_{c_2,c_4} = \{(t_2, t'_2)\}$, $T_{c_2,c_3} = \{(t_3, t'_3), \ldots, (t_9, t'_9)\}$ and $T_{c_5,c_6} = \{(t_{10}, t'_{10}), \ldots, (t_{13}, t'_{13})\}$. The thresholds we consider are $\alpha_1 = 0.1$ and $\alpha_2 = 0.2$. The support of T_{c_1,c_3}, T_{c_2,c_4}, T_{c_2,c_3} and T_{c_5,c_6} are 0.08, 0.08, 0.54 and 0.30, respectively. Therefore, T_{c_1,c_3} and T_{c_2,c_4} are discarded. Entering the inner loop for T_{c_2,c_3} we have only one set S, namely $S_{a_1|(c_2,c_3)} = T_{c_2,c_3} - \{(t_3, t'_3)\}$, since $t_3[A] \neq t'_3[A]$. The support of $S_{a_1|(c_2,c_3)}$ is $6/6 = 1.0$ and $N = 1/6$. Hence, $f_1(S_{a_1}) = 5/6$ and $f_2(T_3) = 5/6$. In the same way, for T_{c_5,c_6} we have $S_{a_2|(c_5,c_6)} = T_{c_5,c_6}$ with support $4/4 = 1.0$ and $N = 3/4$. Therefore, $f_1(S_{a_2|(c_5,c_6)}) = 3/4$ and $f_2(T_4) = 3/4$. Thus, the degree of dependence of (A, C) is $f_3((A, C), G) = \max\{3/4, 5/6\} = 5/6$. The degree of dependence for all pairs, with respect to G, are $f_3(A, B) = 4/6$, $f_3(B, A) = 0$, $f_3(A, C) = 5/6$, $f_3(C, A) = 0$, $f_3(B, C) = 1$, and $f_3(C, B) = 0$.

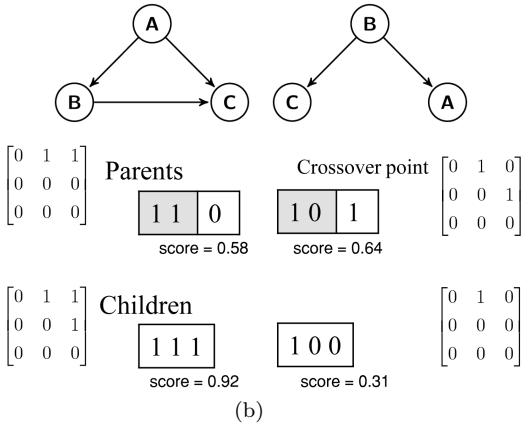
The score of the network G_1, the left one at Figure 2 (b) is given by $score(G_1, \mathbf{PrefDb}_1) = (4/6 + 1 + 5/6 + 1 + 1 + 1)/6 = 0.92$. Let us consider another candidate network G_2, the right one in Figure 2. Its score is given by $score(G_2, \mathbf{PrefDb}_1) = (-4/6 + 0 - 5/6 + 1 + 1 + 1)/6 = 0.25$. By analyzing these values, we conclude that G_1 captures more correctly the dependencies between pairs of attributes present in the preference database $\mathbf{PrefDb}_1$ than does G_2.

4.2 Mining the BPN Topology

The problem of finding a Bayesian Network from data is recurrent in literature and it is known to be not trivial. The search space of candidate structures grows more than exponentially on the number of attributes in the database [Jensen and Nielsen 2007]. Given such feature, we adopted a heuristic method - Genetic Algorithm (GA) [Goldberg 1989] - to perform the structure learning phase, along with the score function described above. Each genetic operator is detailed in the following.

Id	A	B	C	A	B	C	
1	a_2	b_1	c_1	a_3	b_4	c_3	$\mathbf{T}_{(c_1,c_3)}$
2	a_1	b_1	c_2	a_4	b_2	c_4	$\mathbf{T}_{(c_2,c_4)}$
3	a_2	b_2	c_2	a_3	b_3	c_3	$\mathbf{T}_{(c_2,c_3)}$
4	a_1	b_3	c_2	a_1	b_4	c_3	
5	a_1	b_3	c_3	a_1	b_4	c_2	
6	a_1	b_3	c_3	a_1	b_4	c_2	
7	a_1	b_3	c_3	a_1	b_4	c_2	
8	a_1	b_4	c_3	a_1	b_3	c_2	
9	a_1	b_4	c_3	a_1	b_3	c_2	
10	a_2	b_5	c_5	a_2	b_5	c_6	$\mathbf{T}_{(c_5,c_6)}$
11	a_2	b_5	c_5	a_2	b_5	c_6	
12	a_2	b_3	c_5	a_2	b_3	c_6	
13	a_2	b_3	c_6	a_2	b_3	c_5	

(a)



(b)

Fig. 2: (a) A preference database $\mathbf{PrefDb}_1$. (b) On top, Bayesian Network Structures G_1 and G_2 for the Preference Database $\mathbf{PrefDb}_1$. On bottom, crossing over two parents to produce two new individuals. The attribute ordering is ABC. Child with genotype $(1, 1, 1)$ corresponds to the upper structure shown on top of (b).

4.2.1 *Codification of Individuals.* To model every possible edge in a Bayesian Network with n attributes, it is possible to set a $n \times n$ square matrix m, with $m_{ij} = 1$ representing the existence of an edge from a node i to a node j, and $m_{ij} = 0$ otherwise. However, generating random structures and crossing them over in this fashion would potentially create loops. Since a Bayesian Network cannot have loops, this approach would require extra work to deal with this issue.

To avoid this situation, we adopted an upper triangular matrix m, in which every element of the main diagonal and below it are zero. For instance, suppose a database with attributes A,B and C. Considering an ordering ABC in m, it is possible for A to be a parent of B and C. Attribute B can be a parent of C, but not a parent of A, and so on. This may limit the search abilities of the GA, therefore we do γ runs of GA, each one with a different random ordering of attributes, to narrow this issue. In this work, we considered $\gamma = 50$ and the number of generations has been settled as 100.

In terms of chromosome genotype, an upper triangular matrix can be implemented as a binary array with $\frac{n(n-1)}{2}$ length. An example of this mapping can be seen on the bottom of Figure 2 (b). The initial population contains randomly generated individuals.

4.2.2 *Crossover and Mutation Operators.* To form a pair of parents to be crossed, initially each one is selected through a mating selection procedure, namely, tournament with size three. It works as follows: randomly select three individuals from the current population, then pick the best of them, i.e., the one with highest score. This procedure is done twice to form each pair of parents, since we need two parents to be crossed. Then, we are ready to apply crossover operator. Since at any generation a given attribute order is fixed, we can use directly two-point crossover to generate two new individuals from two parents. A position from 1 to the length of individual's size is randomly taken, mixing the genetic materials of two parents, obtaining two new individuals.

The mutation operator aims at adding diversity to population, applying small changes to new individuals, usually under a low probability of occurrence (in this work, 0.05). For each individual generated by crossover, a mutation toggles one position (randomly selected) of its binary array. When an individual is mutated, its score is recalculated, and the mutation is accepted only if it improves its score, otherwise the mutation is discarded.

4.2.3 *Fitness Assignment and Reinsertion Procedure.* Whenever new individuals are created, their score must be evaluated. Initially, since an individual genotype is represented by a binary array, it must be mapped to its phenotype representation, a Bayesian Network structure. Then, its score is

calculated using Eq. (1). At the end of j-th generation, we have two populations: parents (I_j) and offspring (I'_j), where $|I_j| = |I'_j| = \beta$. In order to attempting to achieve a faster convergence, we use an elitist procedure to select individuals to survive: pick the β fittest individuals from $I_j \cup I'_j$ to set the next population I_{j+1}.

4.3 Parameter Estimation

Once we have the topology G of the BPN, calculated at our previous step, we are now seeking for estimates of the conditional probabilities tables. Since we have a set of cases in our preference database $\mathcal{P}$, we can estimate such parameters using the Maximum Likelihood Principle [Jensen and Nielsen 2007], in which we calculate the maximum likelihood estimates for each conditional probability distribution of our model. The underlying intuition of this principle uses frequencies as estimates; for instance, if we want to estimate $P(A = a > A = a'|B = b, C = c)$ we need to calculate $\frac{N(A=a,B=b,C=c)}{N(A=a,B=b,C=c)+N(A=a',B=b,C=c)}$, where $N(A = a, B = b, C = c)$ is the number of cases where $(A = a, B = b, C = c)$ is preferred over $(A = a', B = b, C = c)$, and so on.

4.4 Complexity Analysis

The optimization problem of determining the Bayesian Network structure which maximizes the score function is intractable. In fact, as shown in [Robinson 1977] the number of possible structures which contains n nodes grows exponentially with n. Thus, heuristic methods having polynomial time complexity are normally adopted for accomplishing this task ([Cooper and Dietterich 1992]).

The complexity of creating the model (BPN): (i) the computation of the degree of dependency between each pair of attributes is $O(n^2.m)$, where n is the number of attributes and m is the number of training bituples; (ii) the computation of the topology (carried out by the genetic algorithm (GA)) is $O(n^2.q.r.\gamma)$, where q is the initial population size, r is the number of generations and γ is the number of attribute orderings considered; (iii) the computation of the conditional probability tables is $O(e.m)$, where e is the number of edges of the BPN produced by the GA. Therefore, the complexity for building the entire CrefMiner model is $O(n^2(m + q.r.\gamma))$.

The complexity of using the model (BPN) for ordering a bituple (t, t') is $O(n + e)$, which can be reduced to $O(n)$ when the BPN is a sparse graph. So, one can see that the computational cost for building the model (which is an offline task) is quadratic on the number of attributes n and the computational cost for using the model is linear on n.

5. EXPERIMENTAL RESULTS

In order to devise an evaluation of the proposed methods, in this article we designed experiments over synthetic and real data. A 10-fold cross validation protocol was considered, based on two different sampling strategies. In the following we describe the two sampling strategies, the data features and the results based on each sampling approach.

5.1 The Sampling Strategies for Cross-validation

Let D be a database of tuples of relation schema R($A_1, ..., A_n, Grade$), where each tuple over attributes $A_1, ..., A_n$ has an associated score (*grade*) given by the user, which specifies his/her preference on this tuple. From D one can build a preference database $\mathcal{P}(D)$ of bituples (u, v) over R($A_1, ..., A_n$): $(u, v) \in \mathcal{P}(D)$ iff $u \in D$ and $v \in D$, and $grade(u) > grade(v)$. CPrefMiner is trained and tested over a set of bituples (and not over a set of graded tuples as the classifiers used as baselines in our experiments). Next we describe the two strategies for preparing the training and test data.

Tuple-based K-cross-validation (strong protocol) The database D is partitioned into K (pairwise) disjoint subsets $D_1, ..., D_K$ approximately of the same size and stratified with respect to the grades (each subset D_i keeps the same proportion of tuples of grade i as in D). For $i = 1, ..., K$ let

Table I: Preference order assessment, where **PNet*** stands for the best BPN obtained by GA. Each $\mathcal{P}$ **Group** stands for 9 datasets. The number of attributes is indicated in the first column. The measures appearing in each row correspond to the average measure (recall or precision) between the 9 datasets.

$\mathcal{P}$ **Group**	Recall(**PNet***,$\mathcal{P}$)						Precision(**PNet***,$\mathcal{P}$)					
	5k	10k	20k	50k	100k	500k	5k	10k	20k	50k	100k	500k
4	0.947	0.951	0.950	0.951	0.950	0.951	0.948	0.951	0.950	0.951	0.950	0.951
6	0.949	0.949	0.948	0.948	0.949	0.949	0.949	0.949	0.948	0.948	0.949	0.949
8	0.950	0.947	0.949	0.949	0.950	0.949	0.951	0.948	0.949	0.949	0.950	0.949
10	0.951	0.952	0.950	0.952	0.951	0.952	0.954	0.953	0.950	0.953	0.952	0.952

Bt_i be a subset of $D_i \times D_i$, where $(u, v) \in Bt_i$ iff $grade(u) > grade(v)$. The tuple-based K-cross-validation protocol executes K rounds of training and testing, where at each round i, the training set is $B_1 \cup ... \cup Bt_{i-1} \cup Bt_{i+1} \cup ... \cup Bt_K$ and the testing set is Bt_i. Notice that in this protocol, at each round the bituples involved in the training set do not contain any tuple involved in the testing set.

Bituple-based K-cross-validation (weak protocol): The preference database $\mathcal{P}(D)$ is partitioned into K pairwise disjoint subsets approximately of the same size. The cross-validation in this approach follows the classical protocol, where each round i uses the subset i for testing and the remaining $K-1$ subsets of bituples for training. Notice that in this protocol, at each round the testing and training sets are disjoint but individual tuples involved in bituples of the testing set can appear in some training bituple.

5.2 Synthetic Data

Synthetic data[2] were generated by an algorithm based on Probabilistic Logic Sampling [Jensen and Nielsen 2007], which samples cases for a preference database $\mathcal{P}$ given a BPN with structure G and parameters θ. We have considered groups of networks with 4, 6, 8 and 10 nodes. For each group, we randomly generated nine networks with their respective parameters. Each attribute at a given setting has a domain with five elements. We also simulated the practical aspect of user's preference, in the sense that users usually do not elicit their preferences on every pair of objects, so we retained only a small percent (around $10-20\%$) from all possible preference rules that could be generated for a given network structure.

A set of experiments (using the weak protocol for 10-cross-validation) analyzing how a BPN infers the strict partial order described in Sec. 3 is depicted in Table I. We can see that the best BPN obtained by our method returned very good results, both for precision and recall measures. It is possible to note that the small differences between recall and precision, even in cases with large datasets (e.g. 500.000 bituples), indicates that the method leads to very few non-comparable bituples. Apart from that fact, the method infers a correct ordering by a percent around 95% in every scenario, having very small data fluctuations, an evidence of the method stability.

5.3 Real Data

In a second series of tests to evaluate CPrefMiner, we considered databases containing preferences related to movies, taken from APMD project[3] by PRISM Laboratory. Such project integrates data from MovieLens movie recommendation and IMDb forum. Six databases of graded films were considered (films evaluated by six different users), namely D_3, D_4, D_5, D_6 , D_7 and D_8. Each database has seven attributes. The experiments have been carried out following the strong and weak 10-cross-validation protocols. The sizes of the sets D_i and Bt_i are given in Table II[4].

We compare the performance of CPrefMiner with two baselines (the Bayesian classifier and J48), showing the superior quality of the CPrefMiner prediction capability. The poor results produced

[2]Available at `http://www.lsi.ufu.br/cprefminer/`
[3]Available at http://apmd.prism.uvsq.fr
[4]See http://guilhermealves.eti.br/research/cprefminer/stratified1.html for more details

Table II: Precision and recall following the weak protocol (bituple-based)

			Recall			Precision		
Dataset	Tuples	Bituples	J48	BayesNet	CPrefMiner	J48	BayesNet	CPrefMiner
$D3$	167	800	0,173	0,356	0,724	0,506	0,516	0,910
$D4$	305	2500	0,156	0,432	0,809	0,549	0,613	0,860
$D5$	607	12000	0,248	0,439	0,864	0,556	0,589	0,887
$D6$	823	23000	0,270	0,420	0,832	0,487	0,570	0,885
$D7$	1078	40000	0,345	0,492	0,874	0,556	0,655	0,892
$D8$	1416	70000	0,295	0,519	0,890	0,596	0,709	0,910

Table III: Precision and recall following the strong protocol (tuple-based)

	Recall			Precision			Time		
Dataset	J48	BayesNet	CprefMiner	J48	BayesNet	CprefMiner	J48	BayesNet	CprefMiner
$D3$	0,141	0,368	0,504	0,435	0,501	0,867	0,001s	0,003s	1s
$D4$	0,157	0,409	0,623	0,606	0,597	0,735	0,005s	0,005s	1s
$D5$	0,247	0,442	0,621	0,552	0,605	0,678	0,010s	0,008s	9s
$D6$	0,287	0,435	0,615	0,501	0,589	0,740	0,018s	0,013s	36s
$D7$	0,355	0,504	0,694	0,569	0,658	0,755	0,023s	0,012s	2m26s
$D8$	0,301	0,547	0,722	0,601	0,733	0,774	0,034s	0,016s	6m5s

by these classifiers show that classifiers are not suitable for preference mining tasks. The results concerning the weak protocol are presented in Table II and those concerning the strong protocol in Table III. The execution time for *building* the BPN is the same for both protocols and is showed only in Table III. In this table we also show the time needed for building the model of the classifiers J48 and BayesNet. The execution times for *using* the models are negligible: J48, BayesNet and CPrefMiner take respectively 0.4ms, 1.35ms and 327.53ms for ordering 100 bituples. As expected, the results of precision and recall obtained for CPrefMiner according to the weak protocol are better than those obtained according the strong protocol since for the weak protocol the films appearing in the testing subset may appear in the training set. Remind that a bituple appearing in the test do not appear in the training set. The results show that CPrefMiner, an algorithm specifically designed for a preference mining task, performs far better than classical classifiers.

Parameters for calibrating the inferred order by taking into account the user *undecidability* and *inconsistency*. Notice that in order to compare two tuples u_1 and u_2 according to the order described in Example 3.5, we simply ignore the probability p_2 which in some sort measures the user undecidability concerning his/her preferences over these tuples. Notice also that in order to infer a preference we simply consider the higher value between p_1 and p_3. The proximity of p_1 and p_3 measures in some sort the user *inconsistency* concerning his/her preferences over these tuples. We could be more precise when defining the preference ordering induced by a BPN by considering parameters k_1 and k_2 for calibrating respectively the user undecidability and inconsistency factors. That is: we say that u_1 is preferred to u_2 if the following conditions are verified: (1) $p_2 \leq k_1$ and (2) $p_1 \geq k_2(1-p_2)$. The first condition (resp. the second condition) controls how the user undecidability (resp. inconsistency) is taken into account in the inferred ordering. As k_1 increases (resp. as k_2 decreases) user undecidability (resp. inconsistency) importance decreases for deciding the preference ordering. The ordering considered in Example 3.5 corresponds to $k_1 = 1$ and $k_2 = 0.5$. We designed a set of experiments (over the movie database) intending to evaluate how would be the variation of precision and, specially, recall when we change the values of k_1 and k_2 factors. For lack of space we do not present the detailed results. Our conclusion is that the recall decreases very quickly as $k1$ decreases (from 1 to 0, $k2 = 0.5$), and the decreasing rate is higher for larger datasets. Precision decreases slower than recall as $k1$ decreases. Recall increases very quickly as $k2$ decreases (from 1 to 0.5 and $k1 = 1$). Precision also decreases very quickly as $k2$ decreases (from 1 to 0.5) and the decreasing rate is higher for larger datasets.

6. CONCLUSION AND FURTHER WORK

In this article we proposed an approach for specifying contextual preferences using Bayesian Networks and the algorithm *CPrefMiner* for extracting a *Bayesian Preference Network* from a set of user's past choices. As future work, we plan to compare the predictive quality of our (optimized) method with well-known ranking methods as RankNet, Rank SVM, Ada Rank and RankBoost [Joachims 2002; Freund et al. 2003; Burges et al. 2005; Xu and Li 2007], knowing that existing prototypes that implement these methods have to be adapted (in order to take directly as input pairwise preferences, and not only quantitative preferences). We also intend to deepen the study on the parameters k_1 (*indecision*) and k_2 (*inconsistency*) and design mining techniques which take into account such parameters. Finally, we plan to develop a recommender system using CPrefMiner as a tool for recommending new items for new users without bothering the user with a large amount of sampling evaluations beforehand.

REFERENCES

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., and Poole, D. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research.* vol. 21, pp. 135–191, 2004.

Burges, C. J. C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. N. Learning to Rank Using Gradient Descent. In *Proceedings of the International Conference on Machine Learning.* New York, NY, USA, pp. 89–96, 2005.

Crammer, K. and Singer, Y. Pranking with Ranking. In Proceedings of the Neural Information Processing Systems Conference, Vancouver, Canada, pp. 641–647, 2001.

de Amo, S., Bueno, M.L., Alves, G., Silva, N.F. Mining User Contextual Preferences. In *Proceedings of the Brazilian Symposium on Databases*, São Paulo, Brazil, pp. 177–184 ,2012a.

de Amo, S., Diallo, M., Diop, C., Giacometti, A., Li, H. D., and Soulet, A. Mining Contextual Preference Rules for Building User Profiles. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*, Viena, Austria, pp. 229–242, 2012b.

de Amo, S. and Pereira, F. A Context-Aware Preference Query Language: theory and implementation. Tech. Rep., Universidade Federal de Uberlândia, School of Computing, Brazil, 2011.

Cooper, G. F. and Dietterich, T. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning,* vol. 9(4), pp. 309–347, 1992.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research,* vol. 4, pp. 933–969, 2003.

Fürnkranz, J. and Hüllermeier, E. *Preference Learning.* Springer, 2011.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Massachusetts, 1989.

Holland, S., Ester, M., and Kiessling, W. Preference Mining: a novel approach on mining user preferences for personalized applications. In European Conference on Principles of Data Mining and Knowledge Discovery, Cavtat-Dubrovnik, Croatia, pp. 204–216, 2003.

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. Label Ranking by Learning Pairwise Preferences. *Artificial Intelligence,* vol. 172(16-17), pp. 1897–1916, 2008.

Jensen, F. V. and Nielsen, T. D. *Bayesian Networks and Decision Graphs.* Springer Publ. Company, Inc., 2007.

Jiang, B., Pei, J., Lin, X., Cheung, D. W., and Han, J. Mining Preferences from Superior and Inferior Examples. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, pp. 390–398, 2008.

Joachims, T. Optimizing Search Engines using Clickthrough Data. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 133–142, 2002.

Koriche, F. and Zanuttini, B. Learning Conditional Preference Networks. *Artificial Intelligence,* vol. 174(11), pp. 685–703, 2010.

Robinson, R. W. Counting Unlabeled Acyclic Digraphs. In C.H.C. Little (Ed.) *Combinatorial Mathematics.* Lecture Notes in Mathematics, vol. 622. New York, Springer-Verlag, 1977.

Wilson, N. Extending CP-Nets with Stronger Conditional Preference Statements. In *National Conference on Artificial Intelligence*, San Jose, California, USA, pp. 735–741, 2004.

Xu, J. and Li, H. AdaRank: a boosting algorithm for information retrieval. In *ACM SIGIR International Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, pp. 391–398, 2007.