Mining User Contextual Preferences

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

Universidade Federal de Uberlândia
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 which 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 paper, 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: preference mining, context-awareness, bayesian networks

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. 2012; 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 paper, 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 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 paper we propose the algorithm *CPrefMiner*, a qualitative method for mining a set of *probabilistic* contextual preference rules modeled as a *preference bayesian network* (BPN).

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 paper 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. 2012] some of the authors of the present paper proposed a different method (ProfMiner) to discover user *profiles* specified by a set of preference rules. In Section 5 we briefly compare the performance of CPrefMiner and ProfMiner. 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 it produces more accurate results than ProfMiner. However, concerning model interpretability, ProfMiner is more advantageous 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 $>$. So if $>$ is 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)

Fig. 1: (a) An instance I and (b) a Preference Database $\mathcal{P}$

The main objective of this paper 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 (1) 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 (2) 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 2 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 $C = c_1$.

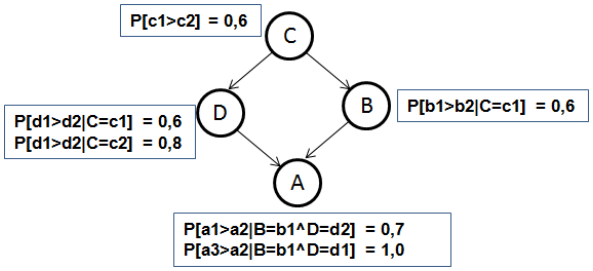


Fig. 2: Preference Network $\mathbf{PNet}_1$

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].

Example 3.5 *Preference Order.* Let us consider the BPN $\mathbf{PNet}_1$ depicted in Figure 2. This BPN allows to infer a preference ordering on tuples over $R(A, B, C, D)$. According to this ordering, tuple

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

$u_1 = (a_1, b_1, c_1, d_1)$ 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 paper 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) with respect to $\mathcal{P}$.

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$((y, y'),\mathcal{P}) = \frac{|T_{y,y'}|}{|\mathcal{P}|}$. 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$((x|(y, y'),\mathcal{P}) = \frac{|S_{x|(y,y')}|}{|\bigcup_{x' \in \mathbf{dom}(X)} S_{x'|(y,y')}|}$; (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, $0 \leq \alpha_2 \leq 1$.

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

$$score(G, \mathcal{P}) = \frac{\sum_{X,Y} g((X, Y), G)}{n(n-1)} \quad (1)$$

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 $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.
Output: The Degree of Dependence of (X, Y) with respect to $\mathcal{P}$

1 **for** *each pair* $(y, y') \in \boldsymbol{dom}(Y) \times \boldsymbol{dom}(Y)$, $y \neq y'$ *and* (y, y') *comparable* **do**
2 **for** *each* $x \in \boldsymbol{dom}(X)$ *where* x *is a cause for* (y, y') *being comparable* **do**
3 Let $f_1(S_{x|(y,y')}) = \max\{N, 1 - N\}$, where
 $N = \dfrac{|\{(t, t') \in S_{x|(y,y')} : t > t' \wedge (t[Y] = y \wedge t'[Y] = y')\}|}{|S_{x|(y,y')}|}$
4 Let $f_2(T_{yy'}) = \max\{f_1(S_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$
5 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$\}$
6 **return** $f_3((X, Y), \mathcal{P})$

where X and Y are attributes in a relational schema R. The function g is calculated by the following set of rules: (a) If $f_3((X, Y), \mathcal{P}) \geq 0.5$ and edge $(X, Y) \in G$, then $g((X, Y), G) = f_3((X, Y), \mathcal{P})$; (b) If $f_3((X, Y), \mathcal{P}) \geq 0.5$ and edge $(X, Y) \notin G$, then $g((X, Y), G) = -f_3((X, Y), \mathcal{P})$; (c) If $f_3((X, Y), \mathcal{P}) < 0.5$ and edge $(X, Y) \notin G$, then $g((X, Y), G) = 1$; (d) If $f_3((X, Y), \mathcal{P}) < 0.5$ and edge $(X, Y) \in G$, 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 Fig. 3), 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 upper one at middle side of Fig. 3 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 lower one in Fig. 3. 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

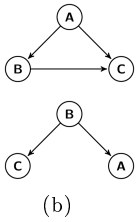
The problem of finding a Bayesian Network from data is recurrent in literature and it is known to be not trivial. The search space of candidate structures grows more than exponentially on the number of attributes in the database [Jensen and Nielsen 2007]. Given such feature, we adopted a heuristic method - 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.

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.

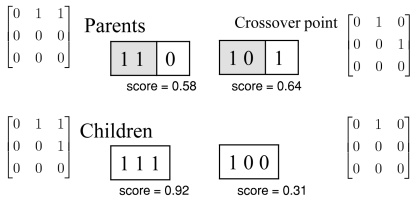
To avoid this situation, we adopted an upper triangular matrix m, in which every element of the

	t			t'			
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)



(c)

Fig. 3: (a) A preference database $\mathbf{PrefDb}_1$. (b) Bayesian Network Structures G_1 and G_2 for the Preference Database $\mathbf{PrefDb}_1$. (c) Crossing over two parents to produce two new individuals. The attribute ordering is ABC. Child with genotype $(1, 1, 1)$ corresponds to structure shown in the upper side of (b).

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 (in the work, $\gamma = 20$), each one with a different random ordering of attributes, to narrow this issue.

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 at the right side of Fig. 3. The initial population contains individuals randomly generated.

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 indivual'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 the its binary array. When an individual is mutated, its score is recalculated; attempting to provide more selective pressure, the mutation is accepted only if it improve an individual's 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 fenotype representation, a Bayesian Network structure. Then, its score is calculated using equation shown before. At the end of j-th generation, we have two populations: parents (I_j) and offspring (I'_j), where $|I_j| = |I'_j|$. To attempt a faster convergence, we use an elitist procedure to select individuals to survive: pick the population's size 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 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.

5. EXPERIMENTAL RESULTS

In order to devise an evaluation of the proposed methods in this paper, we designed experiments over synthetic and real data. A 10-fold cross validation protocol was considered. We established as GA parameters a population with 50 individuals, evolved for 100 generations. In the following, we describe the data features and the results for both categories of experiments.

5.1 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 fact that users usually do not elicit their preferences on every bituple, 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 analyzing how a BPN infers the strict partial order described in Sec. 3 is depicted in Tab. 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. Examples of mean runtime required to run GA for all datasets with 4 attributes over 5.000 and 500.000 bituples are around 1s and 1min, respectively; the time needed for all datasets with 10 attributes in such setting were around 1s and 4min.

Table I: Preference order assessment, where **PNet*** stands for the best BPN obtained by GA. Each $\mathcal{P}$ **Group** stands for 9 datasets with the number of attributes indicated in the first column.

$\mathcal{P}$ **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

5.2 Real Data

Four databases, namely, D_1 (1.000 bituples), D_2 (3.000 bituples), D_3 (10.000 bituples) and D_4 (30.000 bituples) were considered for this second set of experiments. Each database has seven attributes, and its bituples corresponds to a user's preferences over films [3]. This setting also allows us to compare

[2] Available at `http://www.lsi.ufu.br/cprefminer/`
[3] More details about the datasets and its contents are available at http://www.lsi.ufu.br/cprefminer/downloads/index.php

CPrefMiner with the ProfMiner method of [de Amo et al. 2012] that adapts techniques for association rule mining to the preference mining scenario. The results are presented in Tab. II. We can notice that for the smaller datasets (D_1 and D_2), the two methods are very competitive, obtaining very similar results of precision and recall. However, the main gain of CPrefMiner over ProfMiner can be observed in the larger datasets, namely, the datasets D_3 and D_4, where the high precision and recall values obtained by CPrefMiner are sustained, while the results by ProfMiner are around 10% worse.

Table II: Evaluation of CPrefMiner in preference order inference, compared with ProfMiner algorithm.

	CPrefMiner		**ProfMiner**	
Dataset	Recall	Precision	Recall	Precision
D_1	0.989	**1.000**	**0.994**	0.994
D_2	0.982	0.986	**0.991**	**0.991**
D_3	**0.988**	**0.990**	0.903	0.905
D_4	**0.994**	**0.995**	0.905	0.905

6. CONCLUSION AND FURTHER WORK

In this paper 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 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 propose metrics to evaluate user's *indecision* and *inconsistency* and design mining techniques which take into account such parameters.

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. *J. Artif. Intell. Res.* 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 *ICML*. Vol. 119. ACM, pp. 89–96, 2005.

Crammer, K. and Singer, Y. Pranking with ranking. In *NIPS*. MIT Press, pp. 641–647, 2001.

de Amo, S., Diallo, M., Diop, C., Giacometti, A., Li, H. D., and Soulet, A. Mining contextual preference rules for building user profiles. In *14th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), Viena*, 2012.

de Amo, S. and Pereira, F. A context-aware preference query language: Theory and implementation. Tech. rep., Universidade Federal de Uberlândia, School of Computing., 2011.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.* vol. 4, pp. 933–969, December, 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 *PKDD*. LNCS, vol. 2838. Springer, pp. 204–216, 2003.

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. Label ranking by learning pairwise preferences. *Artif. Intell.* 172 (16-17): 1897–1916, 2008.

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

Jiang, B., Pei, J., Lin, X., Cheung, D. W., and Han, J. Mining preferences from superior and inferior examples. In *KDD*. ACM, pp. 390–398, 2008.

Joachims, T. Optimizing search engines using clickthrough data. In *KDD*. ACM, pp. 133–142, 2002.

Koriche, F. and Zanuttini, B. Learning conditional preference networks. *Artif. Intell.* 174 (11): 685–703, 2010.

Wilson, N. Extending cp-nets with stronger conditional preference statements. In *AAAI*. pp. 735–741, 2004.

Xu, J. and Li, H. AdaRank: a boosting algorithm for information retrieval. In *SIGIR*. ACM, pp. 391–398, 2007.