Improving Pairwise Preference Mining Algorithms Using Preference Degrees

Juliete A. Ramos Costa, Sandra de Amo

Federal University of Uberlândia-MG, Brazil
juliete@mestrado.ufu.br, deamo@ufu.br

Abstract. Different preference mining techniques designed to predict a preference order on objects have been proposed in the literature, with very good accuracy results. In this article, we propose to consider not only the fact that the user prefer an item i_1 to an item i_2 but also the degree of his preference on the two items. We propose the algorithm FuzzyPrefMiner designed to predict fuzzy preferences and show through a series of experiments that it outperforms pairwise preference mining techniques whose training phase do not include information on preference degrees.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*concept learning*

Keywords: data mining, fuzzy preference, preference mining

1. INTRODUCTION

Most known recommendation systems, following a collaborative filtering or a content-based approach, rely on explicit user preference feedback data in order to provide recommendations on items not yet seen by the user. These explicit preference data are given by the *utility matrix* which contains at each position i, j the rating given by user i to item j. However, this explicit way to get user feedback lacks of effectiveness, since most users are unwilling to rate items while browsing the web. Moreover, the information obtained in that way may be biased, since it does not include preferences from users who are not keen on rating things. Another drawback of using explicit user feedback is related to the fact that the rating scale is fixed (normally going from 0 to N, where N is the maximum rating permitted). Let us suppose that a user u rated a movie f_1 as N since he/she loved it. Sometime later u saw the movie f_2 which he/she liked more than f_1. In this case, since past given ratings cannot be modified, u is forced to give equal ratings (namely N) to both movies, even if f_2 is largely preferred to f_1.

For those reasons, we argue that preference data obtained in an implicit manner (by inferring user preferences from his/her browsing behavior) and following a *pairwise* representation (the user preference is represented by a set of pairs of items (i_1, i_2) telling that i_1 is preferred to i_2) are more desirable. For instance, let us suppose that, while browsing the IMDB website[1], user u spent 30 seconds on the page of movie f_1 and 10 minutes on the page of f_2. One can infer that u is far more interested on film f_2 than on film f_1. One can go a step further and assume that u prefers f_2 to f_1 and represent this preference as a pair (f_2, f_1). Pairwise preference data obtained in this implicit way do not contain sufficient information for inferring *specific ratings* on items. However, they are sufficient for inferring a *ranking* of the top-k films the user would probably like, and consequently, recommend those films to him/her [Cohen et al. 1999].

[1]www.imdb.com

Copyright©2016 Permission to copy without fee all or part of the material printed in JIDM is granted provided that the copies are not made or distributed for commercial advantage, and that notice is given that copying is by permission of the Sociedade Brasileira de Computação.

Most preference mining methods proposed in the literature rely on explicit ratings provided by the user as input data represented by means of a *score function* [Cohen et al. 1999; Crammer and Singer 2001; Burges et al. 2005]. *Pairwise* representation [Jiang et al. 2008; Koriche and Zanuttini 2010; de Amo et al. 2013; de Amo et al. 2012] and *Ranking*[2] representation [Joachims 2002; Freund et al. 2003] are receiving more attention in recent research.

Preference mining techniques that use input data following a pairwise representation usually obtain this input data directly from the user, using a pre-processing step for transforming explicit user rating into pairs[3]. For instance, if the user evaluated item i_1 as 5 and i_2 as 1, this information is transformed into the pair (i_1, i_2). We claim that in this transformation, a lot of information has been lost. Indeed, if the ratings were 5 and 4, the same pair (i_1, i_2) would be obtained. In the first situation, the user prefers object i_1 to i_2 more *intensely* than in the second situation.

In previous work [de Amo et al. 2013] we proposed the algorithm CPrefMiner for mining a *crisp* contextual preference model from a set of preference samples represented as pairs of alternatives. The model is said to be *crisp* since the input data provided by the user is based on *yes/no* alternatives (either he/she prefers object i_1 to object i_2 or vice-versa) and the output model allows to predict if a new object i_3 is preferred to a new object i_4 or vice-versa. The mined preference model is said to be *contextual* since it is constituted by a set of *contextual preference rules* [Wilson 2004] of the form *IF <context> then I prefer 'this' to 'that'* . For instance: For films directed by Spielberg, I prefer Action to Comedy whereas for films directed by Woody Allen I prefer Comedy to Action. The preference model extracted by CPrefMiner is capable to infer an *order relation on objects* (a task that a classifier would not be able to achieve!), that is, it is capable to infer *transitivity* relationships (if $i_1 \succ i_2$ and $i_2 \succ i_3$ then $i_1 \succ i_3$ [4]) and also satisfying *irreflexibility* ($i_1 \not\succ i_1$).

The main hypothesis of this article is that *"the more information is enclosed in the input preference data, the more efficient is the mined preference model"*. We propose the method FuzzyPrefMiner to mine *fuzzy* contextual preference model from a set of preferences samples represented as triples (i_1, i_2, n) where i_1 and i_2 are items evaluated by a user u and n is the *degree of preference* $(0 \leq n \leq 1)$ of i_1 w.r.t. i_2 provided by user u. An extensive set of experiments comparing CPrefMiner and FuzzyPrefMiner validates this hypothesis: FuzzyPrefMiner is far more accurate then CPrefMiner to predict user preferences. Besides, the preference knowledge extracted by FuzzyPrefMiner is more refined then the one provided by CPrefMiner, since it is capable to predict not only if $i_1 \succ i_2$ but also the degree of preference of i_1 with respect to i_2. We present in this article a subset of the experiments carried out to validate our hypothesis.

Fuzzy preference modeling and reasoning has been extensively studied in the last decade [Chiclana et al. 1998; Herrera-Viedma et al. 2004; Ma et al. 2006; Xu et al. 2013]. However, to the best of our knowledge this kind of preference model has not yet been exploited in the preference mining research.

2. PROBLEM FORMALIZATION

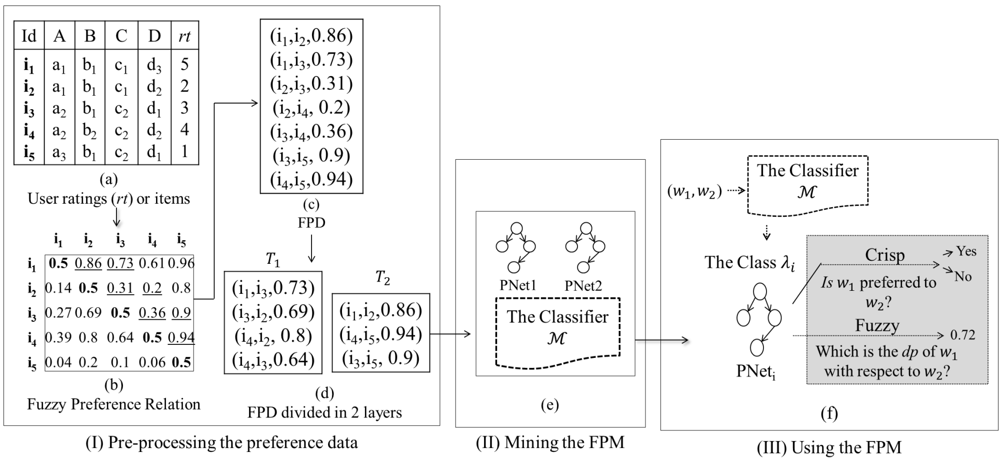
In this section, we formalize the problem of mining fuzzy contextual preferences, introducing the notion of *fuzzy preference model (FPM)*, how it is used to order things and some quality measures to evaluate its order predicting capability.

A *Fuzzy Preference Relation* over a set of objects $X = \{x_1, ..., x_n\}$ is a $n \times n$ matrix P such that $P_{ij} \in [0, 1]$ represents the *degree of preference (dp)* of u over objects x_i and x_j. Fuzzy preference relations have been introduced in Chiclana et al. [1998] and should satisfy certain conditions (see

[2]The user provides a sequence of objects in decreasing order of preference.

[3]Notice that, as argued before, pairwise preference data could also be obtained indirectly by examining the user's browsing behavior.

[4]the symbol $\succ$ standing for "is preferred to"



Id	A	B	C	D	*rt*
i_1	a_1	b_1	c_1	d_3	5
i_2	a_1	b_1	c_1	d_2	2
i_3	a_2	b_1	c_2	d_1	3
i_4	a_2	b_2	c_2	d_2	4
i_5	a_3	b_1	c_2	d_1	1

	i_1	**i_2**	**i_3**	**i_4**	**i_5**
i_1	**0.5**	0.86	0.73	0.61	0.96
i_2	0.14	**0.5**	0.31	0.2	0.8
i_3	0.27	0.69	**0.5**	0.36	0.9
i_4	0.39	0.8	0.64	**0.5**	0.94
i_5	0.04	0.2	0.1	0.06	**0.5**

$(i_1,i_2,0.86)$
$(i_1,i_3,0.73)$
$(i_2,i_3,0.31)$
$(i_2,i_4, 0.2)$
$(i_3,i_4,0.36)$
$(i_3,i_5, 0.9)$
$(i_4,i_5,0.94)$

T_1: $(i_1,i_3,0.73)$, $(i_3,i_2,0.69)$, $(i_4,i_2, 0.8)$, $(i_4,i_3,0.64)$

T_2: $(i_1,i_2,0.86)$, $(i_4,i_5,0.94)$, $(i_3,i_5, 0.9)$

Fig. 1. Fuzzy Contextual Preference Mining Process.

Chiclana et al. [1998] for details), the most important being the reciprocity property $P_{ij} = 1 - P_{ji}$ (which entails that $P_{ii} = 0.5$). One way of obtaining fuzzy preference relations from a set of item ratings provided by the user is to consider $P_{ij} = f(\frac{r_i}{r_j})$, where r_i is the rating u assigned to item x_i and f is a function belonging to the family $\{f_n : n \geq 1\}$, $f_n(x) = \frac{x^n}{x^n+1}$. This family of functions produces matrices satisfying all the properties required for fuzzy preference relations. Fig. 1(b) illustrates the fuzzy preference relation corresponding to the item-rating database of Fig. 1(a). Here, function f_2 has been considered for *fuzzification.*

A *Fuzzy Preference Database (FPD)* over a relation schema $R(A_1, ..., A_n)$ is a finite set $\mathcal{P} \subseteq \text{Tup}(R) \times \text{Tup}(R) \times [0,1]$ which is *consistent*, that is, if $(i, j, n) \in \mathcal{P}$ and $(j, i, m) \in \mathcal{P}$ then $m = 1 - n$. The triple (i, j, n) represents the fact that the user prefers *the tuple i to the tuple j* with a degree of preference n. Fig. 1(c) illustrates a fuzzy preference database corresponding to a *subset* of the information contained in the matrix depicted in Fig. 1(b) (the triples presented in (c) correspond to the underlined positions of (b)).

Pre-processing the FPD: Let $\mathcal{P}$ be a fuzzy preference database. First of all, triples with $dp < 0.5$ are transformed into triples with $dp \geq 0.5$. For instance, triple $(i_2, i_3, 0.31)$ in Fig. 1(c) becomes $(i_3, i_2, 0.69)$. After doing that, let dp_{min} (resp. dp_{max}) be the smallest (resp. the largest) dp appearing in (the transformed) $\mathcal{P}$.
Partition of $\mathcal{P}$ into k layers: First, let us partition the interval $I = [dp_{min}, dp_{max}]$ into k disjoints and contiguous subintervals $I_1, ..., I_k$, such that $\bigcup_{i=1}^{k} I_i = I$. Second, partition $\mathcal{P}$ into k disjoint subsets of triples $\mathcal{P}_1$,..., $\mathcal{P}_k$. Each $\mathcal{P}_i$ is obtained by inserting into $\mathcal{P}_i$ all triples (w_1, w_2, n) with $n \geq 0.5$ such that (w_1, w_2, n) or $(w_2, w_1, 1 - n) \in \mathcal{P}$. Fig. 1(d) illustrates this step.
Layers can be viewed as classes: Let $I_i = [a_i, b_i]$ and $\lambda_i = \frac{a_i+b_i}{2}$. Triples (w_1, w_2, n) in I_i are characterized by tuples w_1 and w_2 having an average "intensity" of preference λ_i between them. For instance, triples of layer T_1 in Fig. 1(d) are associated to class $\lambda_1 = 0.72$ (average of 0.73, 0.69, 0.8 and 0.64).

The Fuzzy Preference Model (FPM): The fuzzy preference model proposed in this article is a structure $\mathcal{F}_k = <\mathcal{M}, PNet_1, ..., PNet_k>$, where $\mathcal{M}$ is a classification model extracted from the set of layers $I_1, ..., I_k$ (considering elements in I_i as (w_1, w_2, λ_i) with the extra "class" dimension λ_i), and $PNet_i$ is a *Bayesian Preference Network* (BPN) extracted from layer I_i, for all $i = 1, ..., k$. BPNs have been introduced in de Amo et al. [2013] for mining *crisp* preference models from a set of pairwise preference data. In the crisp scenario, a *unique* BPN is extracted from the input data. This unique BPN is used to predict if (yes or no) a tuple w_1 is preferred to a tuple w_2. In the fuzzy scenario, we will

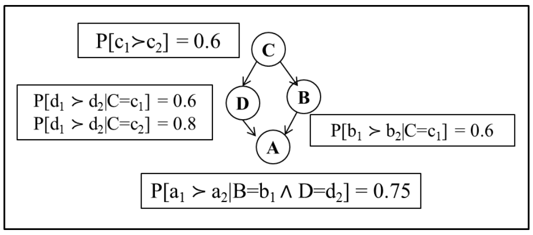


Fig. 2. A Bayesian Preference Network.

mine several BPNs, one for each layer. Fig. 1(e) illustrates a FPM extracted from the pre-processed preference input data of Fig. 1(d). BPNs are defined as follows:

Definition 2.1 *[de Amo et al. 2013].* A *Bayesian Preference Network* (BPN) over a relational schema R$(A_1, ..., A_n)$, consists in: (1) a directed acyclic graph G whose nodes are attributes in $\{A_1, ..., A_n\}$ and the edges stand for *attribute dependency* and (2) a mapping θ that associates to each node of G a finite set of conditional probabilities (see Fig. 2).

Fig. 2 illustrates a BPN $\mathbf{PNet}_1$ over the relational schema $R(A, B, C, D)$. Notice that, the preference on values for attribute B *depends* on the *context* C: if $C = c1$, the probability that value b_1 is preferred to value b_2 for the attribute B is 60%. The following example illustrates how BPNs are used to predict the order between two new tuples in the crisp scenario:

Example 1: Let us consider the BPN $\mathbf{PNet}_1$ depicted in Fig. 2. In order to compare two **new** tuples $w_1 = (a_1, b_1, c_1, d_1)$ and $w_2 = (a_2, b_2, c_1, d_2)$, we proceed as follows: (1) Let $\Delta(w_1, w_2)$ be the set of attributes for which w_1 and w_2 differ. In this example, $\Delta(w_1, w_2) = \{A, B, D\}$; (2) Let $\min(\Delta(w_1, w_2)) \subseteq \Delta(w_1, w_2)$ 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, $\min(\Delta(w_1, w_2)) = \{D, B\}$. The necessary and sufficient conditions for w_1 to be preferred to w_2 are: $w_1[D] \succ w_2[D]$ and $w_1[B] \succ w_2[B]$; (3) Compute the following probabilities: p_1 = probability that $w_1 \succ w_2 = P[d_1 \succ d_2|C = c_1] * P[b_1 \succ b_2|C = c_1]$ $= 0.6 * 0.6 = 0.36$; p_2 = probability that $w_2 > w_1 = P[d_2 \succ d_1|C = c_1] * P[b_2 \succ b_1|C = c_1] = 0.4 * 0.4 = 0.16$. In order to compare w_1 and w_2 we select the highest between p_1 and p_2. In this example, $p_1 > p_2$ and so, we infer that w_1 is preferred to w_2. If $p_1 = p_2$, a random choice decides if w_1 is preferred to w_2 or vice-versa. So, a BPN is capable to compare any pair of tuples: either it *effectively* infer the preference ordering without randomness or it makes a random choice.

How a FPM can be used to predict preference degree between tuples: Let $\mathcal{F}_k = <\mathcal{M}, PNet_1, ..., PNet_k>$ be a k-layer FPM and let w_1, w_2 be two new tuples. First of all, we have to infer the intensity of preference between these tuples. The classifier $\mathcal{M}$ is responsible for this task: it is executed on (w_1, w_2) and returns a class λ_i for this pair. Then the BPN $PNet_i$, ***specific*** for this layer, is executed on (w_1, w_2) to decide which one is preferred. This execution follows the same steps as illustrated in *Example 1.* If $PNet_i$ decides that w_1 is preferred to w_2 (resp. w_2 is preferred to w_1), the final prediction is: w_1 *is preferred to* w_2 *with degree of preference* λ_i (resp. $1 - \lambda_i$). Remark that, the classifier $\mathcal{M}$ only decides how intense the preference between the tuples is. The BPN decides the *preference ordering* between them. This prediction may be *effective* or *random* as remarked at the end of *Example 1.* Fig. 1(f) shows how a FPM can be used to predict both crisp and fuzzy preferences.

Measuring the quality of a fuzzy preference model: Since an exact match between the predicted *dp* (dp_p) and the *real* one (dp_r) is quite unlikely to occur, we consider a threshold $\sigma > 0$ to measure how good is the prediction. We consider that the predicted dp_p is *correct* if the following conditions are both verified: (1) $\mid dp_r - dp_p \mid \leq \sigma$ and (2) ($dp_r \geq 0.5$ and $dp_p \geq 0.5$) or ($dp_r < 0.5$ and $dp_p < 0.5$). We say that the inference is effective (resp. random) depending on the effectiveness or randomness of the BPN prediction. The quality of a FPM $\mathcal{F}_k$ is evaluated by calculating the following

measures on a *test* FPD $\mathcal{P}$:
The ***Accuracy****(acc)*: defined by $acc(\mathcal{F}_k,\mathcal{P},\sigma)=\frac{N_e+N_r}{M}$, where M is the number of triples in $\mathcal{P}$, N_e (resp. N_r) is the amount of triples $(t_1, t_2, dp_r) \in \mathcal{P}$ for which the dp_p inferred by $\mathcal{F}_k$ is effectively (resp. randomly) correct;
The ***Recall****(rec)*: defined by $rec(\mathcal{F}_k,\mathcal{P},\sigma)= \frac{N_e}{M}$;
The ***Randomness Rate****(rr)*: defined by $rr(\mathcal{F}_k,\mathcal{P},\sigma)= \frac{N_r}{M}$;
The ***Precision****(prec)*: defined by $prec(\mathcal{F}_k,\mathcal{P},\sigma)= \frac{N_e}{N_c}$, where N_c is the number of triples (t_1, t_2, dp_r) $\in \mathcal{P}$ that have been effectively compared by $\mathcal{F}_k$ (correctly or not);
The ***Comparability Rate****(cr)*: defined by $rec(\mathcal{F}_k,\mathcal{P},\sigma))= \frac{N_c}{M}$.

Example 2: Let us consider tuples w_1 and w_2 of *Example 1.* Let us also suppose the input FPD depicted in Fig. 1(d) with its 2-layer partitions T_1 and T_2. The corresponding classes are $\lambda_1 = 0.72$ (average *dp* of [0.64, 0.8]) and $\lambda_2 = 0.865$ (average *dp* of [0.83, 0.9]). Let us suppose that $dp_r(w_1, w_2) = 0.76$ and that the classifier $\mathcal{M}$ assigns the class λ_1 to (w_1, w_2). Notice that, the classifier only predicts that the intensity of preference between the two tuples is 0.72. It doesn't say anything about the *ordering* of preference between them. This task is achieved by the BPN_1 associated to partition T_1. Let us suppose that BPN_1 is the same as the one illustrated in Fig. 2. It compares w_1 and w_2 as described in *Example 1*, by calculating the probabilities $p_1 = 0.36$ and $p_2 = 0.16$. As $p_1 > p_2$ then the predicted *dp* is $dp_p = \lambda_1 = 0.72$. So, both dp_r and dp_p are ≥ 0.5 (condition (2) of the quality test is verified). Let us consider parameter $\sigma = 0.05$ as in the experiments of Section 5. Since $\mid 0.76 - 0.72 \mid = 0.04 < 0.05$, we conclude that the fuzzy model correctly and effectively predicted the *dp* of w_1 w.r.t. w_2.

The Problem of Mining Fuzzy Contextual Preferences: This problem consists in, given a fuzzy preference database and $k > 0$, return a k-layer fuzzy preference model $\mathcal{F}_k$ having good quality with respect to $acc, prec, rec, cr$ and rr measures and a given threshold σ.

3. ALGORITHM FUZZYPREFMINER

The FuzzyPrefMiner algorithm we propose to solve the Fuzzy Contextual Preference Mining Problem follows the same strategy of the algorithm CPrefMiner proposed in de Amo et al. [2013] to solve the crisp counterpart problem. The general framework of the algorithm is showed in Alg. 1. In this section, we present the details only of the parts of the method that have been modified in order to treat the fuzzy information. The specific procedures which have been modified are indicated inside a rectangle, in boldface.

3.1 Learning the Network Structure

Procedure *Extract*(Alg. 2)[de Amo et al. 2013] is responsible for learning the network topology G from the training preference database $\mathcal{P}$. G is a graph with vertices in $A_1, ..., A_n$. An edge (A_i, A_j) in G

Algorithm 1: The FuzzyPrefMiner Algorithm

```
Input: A Fuzzy Preference Database $\mathcal{P}_i$ over relational schema $R(A_1, ..., A_n)$ % corresponding to a layer in the
       entire training FPD $\mathcal{P}$;
Output: A BPN PNet
1 Extract($\mathcal{P}$, $G$);
2 for each $i = 1, \ldots, n$ do
3     $Parents(G, A_i) = [A_{i_1}, ..., A_{i_m}]$;
4     $\boxed{\textbf{CPTable}(A_i,\ \textbf{Parents}(A_i, G)) = [CProb_1, ..., CProb_l]}$;
5 return PNet
```

Algorithm 2: *Extract* Procedure

```
Input: $\mathcal{P}$: a fuzzy preference database over relational schema $R(A_1, ..., A_n)$; $\beta$: population size; $\vartheta$: number of
       generations; $\gamma$: number of attribute orderings
Output: A directed graph $G = (V, E)$, with vertices $V \subseteq \{A_1, ..., A_n\}$ that fits best to the fitness function
1  for each $i = 1, \ldots, \gamma$ do
2      Randomly generate an attribute ordering;
3      Generate an initial population $I_0$ of random individuals;
4      $\boxed{\textit{Evaluate individuals in } I_0\textit{, i.e., calculate their fitness}}$;
5      for each $j$-th generation, $j = 1, \ldots, \vartheta$ do
6          Select $\beta/2$ pairs of individuals from $I_j$;
7          Apply crossover operator for each pair, generating an offspring population $I'_j$;
8          Evaluate individuals of $I'_j$;
9          Apply mutation operator over individuals from $I'_j$, and evaluate mutated ones;
10         From $I_j \cup I'_j$, select the $\beta$ fittest individuals through an elitism procedure;
11     Pick up the best individual after the last generation;
12 return the best individual among all $\gamma$ GA executions
```

means that the preference on values for attribute A_j depends on values of attribute A_i. That is, A_i is part of the context for preferences over the attribute A_j. This learning task is performed by a genetic algorithm which generates an initial population $\mathbf{P}$ of graphs with vertices in $A_1, ..., A_n$ and for each graph $G \in \mathbf{P}$ evaluates a fitness function **score** on G. Full details on the codification of individuals and on the crossover and mutation operations are presented in de Amo et al. [2013].

The Fitness Function. This is the only part of the *Extract* procedure where the degrees of preference included in the input data are relevant. The degree of preference affects primarily the way one decides if a given structure is good or not.

The main idea of the fitness function is to assign a real number (a score) in $[-1, 1]$ for a candidate structure G, aiming to estimate how good it captures the dependencies between attributes in a fuzzy 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 $\mathcal{P}$ (see Alg. 3).

The *degree of dependence* of a pair of attributes (X, Y) with respect to a FPD $\mathcal{P}$ 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. 3. In order to facilitate the description of Alg. 3, 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 pairs $(t, t') \in \mathcal{P}$, such that $t[Y] = y \wedge t'[Y] = y'$ or $t[Y] = y' \wedge t'[Y] = y$; **(2)** If S is a set of triples in $\mathcal{P}$, we denote by $DP(S)$ the (multi)set constituted by the *dp* appearing in the triples of S (repetitions are considered as different elements); **(3)**

Algorithm 3: The degree of dependence between a pair of attributes

```
Input: $\mathcal{P}$: a fuzzy 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 $(y, y') \in \mathbf{dom}(Y) \times \mathbf{dom}(Y)$, $y \neq y'$ and $(y, y')$ comparable do
2      for each $x \in \mathbf{dom}(X)$ where $x$ is a cause for $(y, y')$ being comparable do
3          Let $f_1(S_{x|(y,y')}) = \max\{N, 1 - N\}$, where
4          $N = \frac{\{\sum dp \in DP(S_{x|(y,y')})\} : t \succ t' \wedge (t[Y]=y \wedge t'[Y]=y')}{\sum dp \in DP(S_{x|(y,y')})}$
5      Let $f_2(T_{yy'}) = \max\ \{f_1(S_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$
6  Let $f_3((X, Y), \mathcal{T}) = \max\{f_2(T_{yy'}) : (y, y') \in \mathbf{dom}(Y) \times \mathbf{dom}(Y),\ y \neq y',\ (y, y')$ comparable$\}$
7  return $f_3((X, Y), \mathcal{T})$
```

We define support((y, y'),$\mathcal{P}$) $= \frac{\sum_{dp \in DP(T_{y,y'})}}{\sum_{dp \in DP(\mathcal{P})}}$. 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, $0 \leq \alpha_1 \leq 1$; **(4)** For each $x \in \mathbf{dom}(X)$, we denote by $S_{x|(y,y')}$ the subset of $T_{yy'}$ containing the triples (t, t', n) such that $t[X] = t'[X] = x$; **(5)** We define support($(x|(y, y'))$,$\mathcal{P}$) $= \frac{\sum_{dp \in DP(S_{x|(y,y')})} dp}{\sum_{z \in \mathbf{dom}(X), dp \in DP(S_{z|(y,y')})} dp}$; **(6)** 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, $0 \leq \alpha_2 \leq 1$.

The fitness score associated to G is calculated as in de Amo et al. [2013] by the formula:

$$\frac{\sum\limits_{X,Y} g((X,Y),G)}{n(n-1)}$$

where function g is calculated as follows: **(1)** If $f_3((X, Y), \mathcal{P}) \geq 0.5$ and edge $(X, Y) \in G$, then $g((X, Y), G) = f_3((X, Y), \mathcal{P})$; **(2)** If $f_3((X, Y), \mathcal{P}) \geq 0.5$ and edge $(X, Y) \notin G$, then $g((X, Y), G) = -f_3((X, Y), \mathcal{P})$; **(3)** If $f_3((X, Y), \mathcal{P}) < 0.5$ and edge $(X, Y) \notin G$, then $g((X, Y), G) = 1$; **(4)** If $f_3((X, Y), \mathcal{P}) < 0.5$ and edge $(X, Y) \in G$, then $g((X, Y), G) = 0$. More details on the motivation behind this computation ***in the crisp scenario***, please see de Amo et al. [2013].

3.2 Calculating the Probability Tables Associated to each Vertex

Procedure *Parents(A_i, G)* returns the list of the parents of vertex A_i in the directed graph G. Procedure *CPTable*(A_i, Parents(A_i)) returns, for each vertex A_i of the graph returned by Procedure *Extract*, a list of conditional probabilities $[CProb_1, ..., CProb_l]$. Each conditional probability $CProb_i$ is of the form $Pr[E_1|E_2]$ where E_2 is an event of the form $A_{i_1} = a_1 \wedge ... \wedge A_{i_l} = a_l$ and E_1 is event of the form $(B = b_1) \succ (B = b_2)$.

The second point in the FuzzyPrefMining algorithm where the degree of preference is relevant is in procedure *CPTable*. 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 $P(A = a \succ A = a'|B = b, C = c)$ we need to calculate $\frac{\sum_{n \in DP(S(a,a'|b,c))}}{\sum_{m \in DP(S(a,a'|b,c))} + \sum_{m' \in DP(S(a',a|b,c))}}$, where $S(a, a'|b, c)$ = set of triples (t, t', dp) such that $t[B] = t'[B] = b$ and $t[C] = t'[C] = c$ and $t[A] = a$ and $t'[A] = a'$.

4. ASSESSMENT OF THE PREFERENCE MODEL CONSISTENCY

In order to assess the consistency of the preference model inferred by FuzzyPrefMiner, we used the weak transitivity property of a fuzzy preference relation [Herrera-Viedma et al. 2004]. A fuzzy preference relation is consistent if it meets the weak transitivity property, which states: Let M be a fuzzy preference relation. The relation is consistent if: $m_{ij} \geq 0.5, m_{jk} \geq 0.5 \Rightarrow m_{ik} \geq 0.5$ for every i, j, k of the relation.

This type of transitivity is the minimum property a fuzzy preference relation must meet for it to be considered consistent. It is interpreted as an "IF ... THEN" rule: If a tuple t_i is preferred to tuple t_j and the latter is preferred to tuple t_k, then the tuple t_i is also preferred to tuple t_k. Based on this property, Xu et al. [2013] proposed a method to find inconsistencies in fuzzy preference relations. Basically, finding all inconsistencies in a fuzzy preference relation is equivalent to finding all the size-3 cycles in the directed graph that represents this relation.

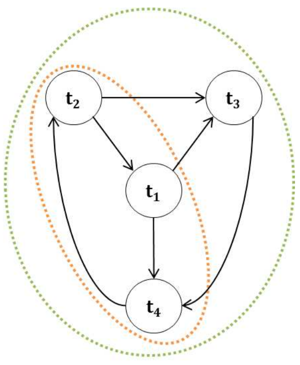
A size-3 cycle means a contradiction of the weak transitivity property since each cycle found represents an type $m_{ij} \geq 0.5$, $m_{jk} \geq 0.5$, and $m_{ik} < 0.5$ inconsistency. In order to find the size-3 cycles, Xu et al. [2013] also proposed an algorithm based only on the matrices and some definitions, as follows:

$$M = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ \hline t_1 & \mathbf{0.5} & 0.1 & 0.6 & 0.7 \\ t_2 & 0.9 & \mathbf{0.5} & 0.8 & 0.4 \\ t_3 & 0.4 & 0.2 & \mathbf{0.5} & 0.9 \\ t_4 & 0.3 & 0.6 & 0.1 & \mathbf{0.5} \end{array}$$

(a)

$$E = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ \hline t_1 & \mathbf{0} & 0 & 1 & 1 \\ t_2 & 1 & \mathbf{0} & 1 & 0 \\ t_3 & 0 & 0 & \mathbf{0} & 1 \\ t_4 & 0 & 1 & 0 & \mathbf{0} \end{array}$$

(b)



(c)

Fig. 3. (a) Fuzzy Preference Relation M, (b) Adjacency Matrix E and (c) Directed Graph.

Definition 4.1. (Adjacency Matrix): Let M be a fuzzy preference relation, an adjacency matrix E over M is defined by:

$$e_{ij} = \begin{cases} 1 & \text{if } m_{ij} \geq 0.5, i \neq j; \\ 0 & \text{if } m_{ij} = *; \\ 0 & \text{otherwise} \end{cases}$$

where $m_{ij} = *$ represents an empty position in the matrix, i.e., the user has specified no preference for this tuple pair. To illustrate, let us consider the following example:

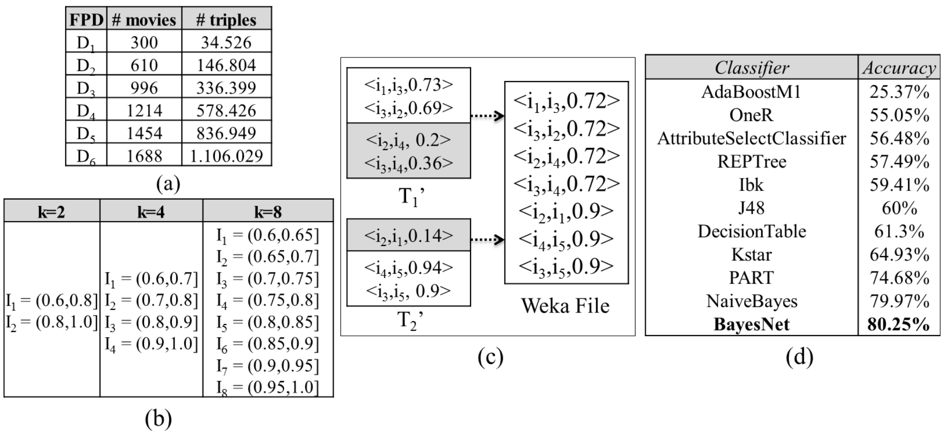
Example 3: Let the fuzzy preference relation be M, obtained from a set of tuples $Tup(R) = \{t_1, t_2, t_3, t_4\}$ illustrated in Fig. 3(a). Fig. 3(b) represents the adjacency matrix from M obtained by Definition 4.1, while Fig. 3(c) illustrates the directed graph that represents the preferences of the matrix E. The directed edge (t_1, t_4) represents the fact that tuple t_1 is preferred to tuple t_4, which is illustrated in matrix E by the value $e_{14} = 1$.

Two size-3 cycles can be observed in the graph (Fig. 3(c)): $c_1(t_2, t_1, t_4, t_2)$ and $c_2(t_2, t_3, t_4, t_2)$, and each cycle represents a contradiction of the weak transitivity property. In order to find all these cycles, Xu et al. [2013] defines the following theorem:

Theorem 4.2. *Let M be a fuzzy preference relation, E the adjacency matrix of M, and E^3 the third power of E. The number of cycles C of a graph that represents a preference matrix is given by:*

$$C = \frac{\sum e^3_{ij}}{3}, \text{ such that, } i = j \tag{1}$$

The sum of the elements in the main diagonal of the matrix E^3 divided by the value 3 corresponds to the number of size-3 cycles in a graph that represents the adjacency matrix of the fuzzy preference relation.



(a)

FPD	# movies	# triples
D_1	300	34.526
D_2	610	146.804
D_3	996	336.399
D_4	1214	578.426
D_5	1454	836.949
D_6	1688	1.106.029

(b)

k=2	k=4	k=8
I_1 = (0.6,0.8]	I_1 = (0.6,0.7]	I_1 = (0.6,0.65]
I_2 = (0.8,1.0]	I_2 = (0.7,0.8]	I_2 = (0.65,0.7]
	I_3 = (0.8,0.9]	I_3 = (0.7,0.75]
	I_4 = (0.9,1.0]	I_4 = (0.75,0.8]
		I_5 = (0.8,0.85]
		I_6 = (0.85,0.9]
		I_7 = (0.9,0.95]
		I_8 = (0.95,1.0]

(c)

$<i_1,i_3,0.73>$
$<i_3,i_2,0.69>$
$<i_2,i_4, 0.2>$
$<i_3,i_4,0.36>$
T_1'
$<i_2,i_1,0.14>$
$<i_4,i_5,0.94>$
$<i_3,i_5, 0.9>$
T_2'
$<i_1,i_3,0.72>$
$<i_3,i_2,0.72>$
$<i_2,i_4,0.72>$
$<i_3,i_4,0.72>$
$<i_2,i_1,0.9>$
$<i_4,i_5,0.9>$
$<i_3,i_5,0.9>$
Weka File

(d)

Classifier	*Accuracy*
AdaBoostM1	25.37%
OneR	55.05%
AttributeSelectClassifier	56.48%
REPTree	57.49%
Ibk	59.41%
J48	60%
DecisionTable	61.3%
Kstar	64.93%
PART	74.68%
NaiveBayes	79.97%
BayesNet	**80.25%**

Fig. 4. (a) The FPDs, (b) The Preference Layers, (c) Preparing the training data for classification and (d) Classifier's Accuracy.

5. EXPERIMENTAL RESULTS

5.1 Preparing the Experiments

The Fuzzy Preference Datasets (FPD). The datasets used in the experiments have been obtained from the GroupLens Project[5]. The original data are of the form (UserId, FilmId, Rating). More details about the movies, namely Genre, Actors, Director, Year and Language, have been obtained by means of a crawler which extracted this information from the IMDB website[6]. Six datasets of film evaluations, corresponding to six different users have been considered. Details about them are shown in Fig. 4(a). For instance, dataset D1 is a set of 34.526 triples (w_1, w_2, n) where w_1, w_2 are taken from a pool of 300 movies.

Training a Classifier. A classifier is trained on each dataset in order to predict the "intensity of preference" existing between two movies. A preliminary pre-processing is needed before the classifier training phase. Each preference layer should contain 50% of triples with $dp \geq 0.5$ and 50% with $dp < 0.5$ in order to contain equal amount of pairs (w_1, w_2) with "yes" and "no" answers to "*is w_1 preferred to w_2?*". This pre-processing is illustrated in Fig. 4(c), where the preference layers T_1 and T_2 of Fig. 1(d) have been transformed into layers T_1' and T_2'. After the calculation carried out by the *fuzzification* function f_2 (as explained in Section 3) we have $dp_{min} = 0.6$ and $dp_{max} = 1.0$ for each FPD considered in our experiments. The interval I $= [0.6, 1.0]$ is partitioned into k layers, and triples of each layer receives an extra class attribute with value λ_i, the average dp for the layer. The triples of the Weka File of Fig. 4(c) are depicted with their respective "classes" $\lambda_1 = 0.72$ and $\lambda_2 = 0.9$. The Weka classifiers have been trained on a FPD of 31713 triples partitioned into 4 layers: $T_1 = (0.6, 0.7]$, $T_2 = (0.7, 0.8]$, T_3= $(0.8, 0.9]$ and $T_4 = (0.9, 1.0]$. The accuracy results are shown in Fig. 4(d). The Bayesian classifiers *Naive Bayes* and *BayesNet* presented the best results. Besides, they execute faster than the other ones. Thus, the *BayesNet* has been chosen for the remaining experiments concerning FuzzyPrefMiner evaluation.

5.2 Performance and Scalability Results

A *30-cross-validation* protocol has been considered in the experiments with FuzzyPrefMiner. The FuzzyPrefMiner was implemented in Java and all the experiments were performed on a core i7 3.20GHZ

[5]www.grouplens.org
[6]www.imdb.com

FPD	***AdaBoostM1***					***DecisionTable***					***BayesNet***				
	acc	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*
D1	75.84%	74.06%	76.73%	95.52%	1.78%	77.29%	75.40%	78.39%	96.18%	1.90%	**82.48%**	**80.76%**	**83.67%**	**96.53%**	**1.72%**
D2	73.49%	71.86%	74.26%	96.89%	1.53%	77.53%	75.39%	78.73%	95.76%	2.13%	**83.29%**	**81.99%**	**84.20%**	**97.38%**	**1.30%**
D3	78.48%	73.48%	81.65%	89.99%	5.01%	78.25%	76.65%	79.14%	**98.85%**	1.60%	**87.49%**	**86.39%**	**88.33%**	97.80%	**1.09%**
D4	72.82%	72.14%	73.14%	**98.63%**	0.69%	76.47%	77.07%	77.23%	97.21%	1.40%	**86.39%**	**85.71%**	**86.91%**	98.62%	**0.68%**
D5	79.09%	75.81%	81.15%	93.43%	3.28%	75.29%	73.02%	76.50%	95.45%	2.26%	**86.10%**	**85.34%**	**86.66%**	**98.48%**	**0.76%**
D6	73.77%	72.95%	74.17%	98.35%	0.83%	79.36%	78.41%	79.93%	98.10%	0.95%	**83.80%**	**82.98%**	**84.35%**	**98.38%**	**0.81%**

Fig. 5. Influence of Classifiers.

FPD	**k=2**					**k=4**					**k=8**				
	acc	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*
D1	75.59%	73.23%	76.72%	95.46%	2.35%	82.48%	80.76%	83.67%	**96.53%**	**1.72%**	**87.41%**	**85.64%**	**88.80%**	96.45%	1.76%
D2	74.96%	73.28%	75.84%	96.62%	1.68%	83.29%	81.98%	84.20%	97.38%	1.30%	**88.42%**	**87.17%**	**89.45%**	**97.46%**	**1.25%**
D3	72.27%	70.71%	72.97%	96.91%	1.56%	87.49%	86.39%	88.33%	97.80%	1.09%	**90.09%**	**89.21%**	**90.83%**	**98.21%**	**0.89%**
D4	72.84%	72.14%	73.16%	98.61%	0.69%	86.39%	85.71%	86.91%	**98.62%**	**0.68%**	**89.96%**	**89.23%**	**90.56%**	98.53%	0.73%
D5	73.98%	73.10%	74.40%	98.26%	0.87%	86.11%	85.34%	86.66%	98.48%	0.76%	**89.25%**	**88.50%**	**89.85%**	98.49%	0.75%
D6	73.04%	72.31%	73.37%	98.56%	0.73%	83.80%	82.98%	84.35%	**98.37%**	**0.81%**	**89.48%**	**88.70%**	**90.07%**	98.48%	0.77%

Fig. 6. Influence of Preference Layers.

processor, with 32GB RAM, running on operating system Linux Ubuntu 12.10. The following default values have been set for the parameters involved in the algorithm: (1) *Extract* Procedure: $\beta = 50$, $\vartheta = 100$ and $\gamma = 20$; (2) Algorithm 3: $\alpha_1 = 0.2$ e $\alpha_2 = 0.2$. These parameters were tested and verified in our previous work [de Amo et al. 2013]. We prefer to keep the same parameter's values used in the CPrefMiner to make some comparisons and also the statistical significance test between these two algorithms (CPrefMiner and FuzzyPrefMiner). For evaluating the *acc*, *prec*, *rec*, *cr*, *rr* measures, a threshold $\sigma = 0.05$ has been considered.

Experiments have been carried out in order to test our original claim: *the more information is contained in the input data the better are the performance results.* We conducted a series of experiments to validate this hypothesis and then we compare the FuzzyPrefMiner with CPrefMiner.

Influence of Classifiers. In the first test (see Fig. 4(d)), the *Bayesnet* classifier stands as the most performatic. However, the stratification protocol used by *Weka* is not the same protocol used by FuzzyPrefMiner algorithm. Therefore, we performed a second test to support us in the choice of the classifier. The main goal of this test is to analyze the influence of different classifiers when applied with FuzzyPrefMiner algorithm in crisp scenario. We chose three classifiers by analyzing the results of the first test, they are: *AdaBoostM1*, *DecisionTable* and *BayesNet*.

We figured out that even with a different stratification protocol the *BayesNet* classifier got better accuracy, recall and precision in all data sets (see Fig. 5). Regarding comparability and randomness rates, the results are quite similar in the three classifiers.

Influence of the preference layers. Intending to evaluate how the algorithm behaves according to changes in the preference layers, we varied the number of preference layers on three levels. We divided the FPD on 2, 4 and 8 layers to analyze the performance in crisp scenario. Fig. 6 shows the results to accuracy, recall and precision. We realized that when the preference layer becomes more refined (i.e., $k = 8$) better results come up.

Regarding comparability and randomness rates (see Fig. 6), we note that the results do not have significant differences. The results of all k-values are satisfactory, since we have a high comparability rate and low values of randomness rate. In this way, the FuzzyPrefMiner has good prediction because most hits are obtained by the BPN preference order.

Influence of *Fuzzification* Functions. In order to evaluate how the algorithm behaves with the

FPD	n=1					n=2					n=3				
	acc	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*	*acc*	*rec*	*prec*	*cr*	*rr*
D1	77,29%	75,51%	78,51%	96,18%	1,79%	81,60%	79,93%	82,77%	96,47%	1,67%	**89,51%**	**88,15%**	**90,82%**	**97,06%**	**1,36%**
D2	81,07%	79,69%	82,09%	97,07%	1,38%	85,90%	84,75%	86,96%	97,43%	1,17%	**90,55%**	**89,50%**	**91,52%**	**97,80%**	**1,04%**
D3	85,27%	84,08%	86,25%	97,48%	1,19%	88,33%	87,41%	89,13%	98,07%	0,92%	**91,05%**	**90,23%**	**91,77%**	**98,31%**	**0,82%**
D4	84,56%	83,81%	85,14%	98,43%	0,75%	88,16%	87,44%	88,76%	98,51%	0,72%	**92,03%**	**91,32%**	**92,66%**	**98,55%**	**0,71%**
D5	84,13%	83,29%	84,77%	98,25%	0,84%	86,97%	86,18%	87,65%	98,31%	0,79%	**89,54%**	**88,79%**	**90,17%**	**98,47%**	**0,74%**
D6	84,98%	83,24%	84,75%	98,21%	0,85%	86,96%	86,18%	87,58%	98,40%	0,77%	**88,78%**	**88,08%**	**89,37%**	**98,55%**	**0,7%**

Fig. 7. Influence of *Fuzzification* Functions.

FPD	FuzzyPrefMiner: Crisp					CPrefMiner									
	acc	*rec*	*prec*	*cr*	*rr*	*acc*	γ	*rec*	γ	*prec*	γ	*cr*	γ	*rr*	γ
D1	**87.41%**	**85.64%**	**88.80%**	96.45%	**1.76%**	81.24%	0,001	79.46%	0,0012	82.34%	0,001	**96.51%**	0,0012	1.78%	0,0007
D2	**88.42%**	**87.17%**	**89.45%**	97.46%	1.25%	80.48%	0,0012	79.46%	0,001	81.18%	0,0014	**97.91%**	0,0008	**1.02%**	0,0005
D3	**90.09%**	**89.21%**	**90.83%**	98.21%	0.89%	83.86%	0,0018	83.23%	0,0016	84.33%	0,0019	**98.72%**	0,0005	**0.63%**	0,0005
D4	**89.96%**	**89.23%**	**90.56%**	**98.53%**	**0.73%**	81.80%	0,0023	80.45%	0,0028	82.72%	0,0023	97.27%	0,0021	1.35%	0,0015
D5	**89.25%**	**88.50%**	**89.86%**	**98.49%**	**0.75%**	81.37%	0,0027	80.01%	0,0031	82.25%	0,0028	97.29%	0,002	1.36%	0,0014
D6	**89.48%**	**88.70%**	**90.07%**	**98.48%**	0.77%	81.30%	0,0014	80.86%	0,0013	81.57%	0,0014	99.14%	0,0003	**0.44%**	0,0006

Fig. 8. Comparison with CPrefMiner algorithm - Crisp Scenario.

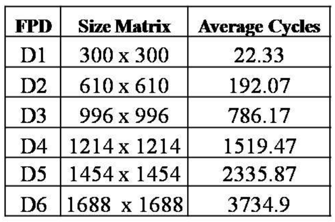
FPD	FuzzyPrefMiner: Fuzzy					CPrefMiner									
	acc	*rec*	*prec*	*cr*	*rr*	*acc*	γ	*rec*	γ	*prec*	γ	*cr*	γ	*rr*	γ
D1	**81.60%**	**79.93%**	**82.77%**	**96.57%**	**1.67%**	81.24%	0.0011	79.46%	0.0013	82.34%	0.0011	96.51%	0.0012	1.78%	0.0006
D2	**85.90%**	**84.73%**	**86.96%**	97.43%	1.17%	80.48%	0.0012	79.46%	0.001	81.18%	0.0014	**97.91%**	0.0008	**1.02%**	0.0005
D3	**88.33%**	**87.41%**	**89.13%**	98.07%	0.92%	83.86%	0.0018	83.23%	0.0016	84.33%	0.002	**98.72%**	0.0005	**0.63%**	0.0006
D4	**88.16%**	**87.44%**	**88.76%**	**98.51%**	**0.72%**	81.80%	0.0023	80.45%	0.0028	82.72%	0.0022	97.27%	0.0021	1.35%	0.0015
D5	**86.97%**	**86.18%**	**87.65%**	**98.32%**	**0.79%**	81.37%	0.0028	80.01%	0.0031	82.25%	0.0028	97.29%	0.002	1.36%	0.0013
D6	**86.96%**	**86.18%**	**87.58%**	**98.40%**	0.78%	81.30%	0.0014	80.86%	0.0013	81.57%	0.0014	99.14%	0.0003	**0.44%**	0.0006

Fig. 9. Comparison with CPrefMiner algorithm - Fuzzy Scenario.

change of *fuzzification* function, we tested three levels of the function $f_n(x) = \frac{x^n}{x^n+1}$ ($n = 1$, $n = 2$ and $n = 3$). We consider the FuzzyPrefMiner in the fuzzy scenario and the preference layers varying according to the function. For f_1, f_2 and f_3 we have the intervals $I_1 = [0.55, 0.85]$ with 6 layers, $I_2 = [0.6, 1.0]$ with 8 layers and $I_3 = [0.65, 1.0]$ with 7 layers, respectively. We have noticed that, the higher the value of n ($n = 3$), better the results obtained for accuracy, recall and precision (see Fig. 7). For results of comparability and randomness rates, we have noticed that these results do not have significant differences. Nevertheless, for the next tests, we will fix $n = 2$ for the *fuzzification* function, because this value is an intermediate level between the functions, it has a bigger preference interval and divides the degrees of preferences better in 8 layers of the interval.

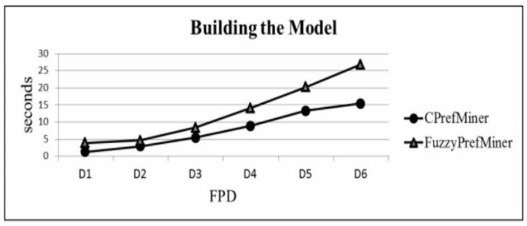
Comparison with *Baseline*. We consider a paired difference one tailed t-test to measure the statistical significance of the differences between the accuracy (resp. recall, precision, comparability rate and randomness rate) of the FuzzyPrefMiner (acting in the *crisp* and *fuzzy* scenarios) with the algorithm CPrefMiner [de Amo et al. 2013].

Both algorithms return BPNs, but CPrefMiner learns its BPN from a *crisp* training set and FuzzyPrefMiner from a *fuzzy* one. We adapted the technique described in Urdan [2010] (originally designed for comparing classifiers) for our fuzzy preference mining approach. For each one of the quality measures d' (where $d' \in \{acc, rec, prec, cr, rr\}$) let $\bar{d'}$ be the difference between the d'-measure of the FuzzyPrefMiner (crisp and fuzzy scenarios) and the respective d'-measure the CprefMiner. The main objective of the t-test is to compute the confidence interval γ for d' and test if the *Null* Hypothesis ($H0$) is rejected. The confidence interval γ is calculated as follows: $\gamma = t_{(1-\alpha,k-1)} \times \sigma_{d'}$, where $t_{(1-\alpha,k-1)}$ is a coefficient obtained in the distribution table with $(1 - \alpha)$ being the level of t-test and $k - 1$ is the degree of freedom. In all experiments we considered $\alpha = 0.95$ and $k = 30$. We evaluated

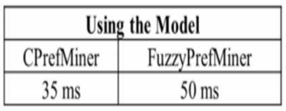


FPD	Size Matrix	Average Cycles
D1	300 x 300	22.33
D2	610 x 610	192.07
D3	996 x 996	786.17
D4	1214 x 1214	1519.47
D5	1454 x 1454	2335.87
D6	1688 x 1688	3734.9

(a)



(b)



Using the Model	
CPrefMiner	FuzzyPrefMiner
35 ms	50 ms

(c)

Fig. 10. (a) consistency analysis of the preference model, (b) time spent for building the models and (c) time spent for using the model on 1000 pairs of tuples.

the statistical significance for all the quality measures, because all of them were evaluated in the same size samples.

For these experiments, the main question is: *With* $\alpha = 0.95$ *of certainty, can we conclude that FuzzyPrefMiner (crisp and fuzzy scenarios) outperforms the CPrefMiner?* The null and alternative hypothesis for *acc*, *rec*, *prec* and *cr* are $H0$: FuzzyPrefMiner $\leq$ CPrefMiner and HA: FuzzyPrefMiner $>$ CPrefMiner. For the *rr* measure, the null and alternative hypothesis are $H0$: FuzzyPrefMiner $\geq$ CPrefMiner and HA: FuzzyPrefMiner $<$ CPrefMiner. The Fig. 8 and Fig. 9 show the results and the statistical significance test to compare the FuzzyPrefMiner with CPrefMiner. The results (crisp and fuzzy scenarios) show that for accuracy, recall and precision measures the $H0$ is rejected (since all the values contained in the confidence interval are positive), and, for these measures HA is substantiated. As for the comparability and randomness rate measures, the $H0$ is rejected in some FPD such as $D4$ and $D5$, and thus, HA is substantiated for these cases and for others FPD ($D1$, $D2$, $D3$ and $D6$) the observed differences has not statistical significance.

These results show the superiority of FuzzyPrefMiner over CPrefMiner, even when validated in the *fuzzy* scenario. As expected the performance of FuzzyPrefMiner in the *fuzzy* scenario decreases with respect to its performance in the *crisp* scenario, since the fuzzy validation protocol requires predicting the degree of preference between two tuples and not only their relative preference ordering.

Execution Time. Fig. 10(b) presents the time spent by CPrefMiner and FuzzyPrefMiner to build their respective models. In the case of FuzzyPrefMiner the total time includes the time spent for pre-processing the input data for the classifier (involving I/O operations), the time spent by the classifier *BayesNet* to build the classification model and the time spent to mine the 8 BPNs, one for each partition. As expected, FuzzyPrefMiner is more costly than CPrefMiner.

The Fig. 10(c) shows the time spent to predict the preference for 1000 pairs of tuples. Again, FuzzyPrefMiner is a little more costly than CPrefMiner taking nearly $1.4T_3$, where T_3 is the time spent by CPrefMiner to accomplish the same task.

Consistency Analysis. We analyzed the consistency of preference model, based on the amount cycles of size-3 of fuzzy preference relations inferred by the FuzzyPrefMiner. Fig. 10(a) shows the matrix size of each test FPD and the average cycles of 30 rounds of algorithm. The fuzzy preference relations obtained from the *fuzzification* function are consistent, because, the function ensures the weak transitivity property. Moreover, the fuzzy preference relation obtained by the FuzzyPrefMiner algorithm are not completely consistent, the algorithm does not hit all user preferences. However, we consider that there is a weak inconsistency in preference model, once the algorithm has good results for accuracy, recall and precision.

6. CONCLUSION

We studied the influence of the *degree of preference* in the performance of Preference Mining techniques specific for learning contextual preference models and propose the algorithm FuzzyPrefMiner for this task, with very promising results. We have fixed the parameter $\sigma = 0.05$ to the experiments in this paper. However, it is intended to change this value in futures works, as well as apply others fuzzification functions found in the literature.

Other future research concentrates on the design of techniques for *repairing inconsistency* in the fuzzy relation produced by FuzzyPrefMiner. Although the original BPN introduced in our previous works is capable to infer a strict partial order on the set of tuples, the fuzzy relation produced by FuzzyPrefMiner does not verify some transitivity properties of fuzzy relations such as weak transitivity or additive transitivity [Herrera-Viedma et al. 2004].

REFERENCES

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.

Chiclana, F., Herrera, F., and Herrera-Viedma, E. Integrating three representation models in fuzzy multipurpose decision-making based on fuzzy preference relations. *Fuzzy Sets and Systems* 97 (1): 33–48, 1998.

Cohen, W. W., Schapire, R. E., and Singer, Y. Learning to Order Things. *Journal of Artificial Intelligence Research* 10 (1): 243–270, 1999.

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

de Amo, S., Bueno, M. L. P., Alves, G., and da Silva, N. F. F. Mining User Contextual Preferences. *Journal of Information and Data Management* 4 (1): 37–46, 2013.

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

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. An Efficient Boosting Algorithm for Combining Preferences. *The Journal of Machine Learning Research* vol. 4, pp. 933–969, 2003.

Herrera-Viedma, E., Herrera, F., Chiclana, F., and Luque, M. Some Issues on Consistency of Fuzzy Preference Relations. *European Journal of Operational Research* 154 (1): 98–109, 2004.

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

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

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

Ma, J., Fan, Z.-P., Jiang, Y.-P., Mao, J.-Y., and Ma, L. A Method for Repairing the Inconsistency of Fuzzy Preference Relations. *Fuzzy Sets and Systems* 157 (1): 20–33, 2006.

Urdan, T. C. *Statistics in Plain English.* Taylor & Francis, 2010.

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

Xu, Y., Patnayakuni, R., and Wang, H. The Ordinal Consistency of a Fuzzy Preference Relation. *Information Sciences* vol. 224, pp. 152–164, 2013.