Strategies for Mining User Preferences in a Data Stream Setting

Jaqueline A. J. Papini, Sandra de Amo, Allan Kardec S. Soares

Federal University of Uberlândia, Brazil
jaque@comp.ufu.br, deamo@ufu.br, allankardec@gmail.com

Abstract. In this article, we formally introduce the problem of mining *contextual* preferences in a data stream setting. *Contextual Preferences* have been recently treated in the literature and some methods for mining this special kind of preference have been proposed in the batch setting. Besides the formalization of the contextual preference mining problem in the stream setting, we propose two strategies for solving this problem. In order to evaluate our proposals, we implemented one of these strategies, the *greedy* one, and we compared its performance with a well known baseline in the literature, showing its efficiency through a set of experiments over real data.

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

Keywords: context-awareness, data mining, data streams, incremental learning, preference mining

1. INTRODUCTION

A data stream may be seen as a sequence of relational tuples that arrive continuously in variable time. Some typical fields of application for data streams are: financial market, web applications and sensor data. Traditional approaches for data mining cannot successfully process the data streams mainly due to the potentially infinite volume of data and its evolution over time. Consequently, several data stream mining techniques have emerged to deal properly with this new data format [Domingos and Hulten 2000; Bifet and Kirkby 2009; Vivekanandan and Nedunchezhian 2011].

Most of the research on preference mining has focused on scenarios in which the mining algorithm has a set of static information on user preferences at its disposal [Jiang et al. 2008; Amo et al. 2012]. The most crucial issues which make the process of mining in a stream setting much more challenging than in batch setting are the followings: (1) data are not stored and are not available whenever needed; (2) the mining process has to cope with both limited workspace and limited amount of time; (3) the mining algorithm should be able to produce the best model at any moment it is asked to and using only the training data it has observed so far. For more details see [Rajaraman and Ullman 2011].

As a motivating example on the dynamic character of the user preferences, consider an *online news site* that wants to discover the preferences of its users regarding news and make recommendations based on that. User's preferences on news depend on many factors, such as the media in which the news is available, its category, author and keywords. Notice that due to the dynamic nature of news, it is plausible that user's preferences would evolve rapidly with time. It may be the case that a news category could attract much attention at a particular time of the year, for example, *political* news at election time. Thus, at this time a user can be more interested in *politics* than in *sports*. However, at Olympic Games time, this same user might consider *sports* as his or her favorite news category.

This work focuses on a particular kind of preferences, the *contextual preferences*. Preference Models

We thank the Brazilian Research Agencies CNPq, CAPES and FAPEMIG for supporting this work.

can be specified under either a *quantitative* [Crammer and Singer 2001] or a *qualitative* [Amo et al. 2012] framework. In the quantitative formulation, preferences about movies for instance can be elicited by asking the user to rate each movie. In the qualitative formulation, the preference model consists in a set of rules specified in a mathematical formalism, able to express user preferences. In this article, we consider the *contextual preference rules* (cp-rules) introduced in [Wilson 2004]. A cp-rule allows to inform the preference on the values of an attribute depending on the values of other attributes.

Proposals for suitable algorithms to solve the problem of mining user preferences in streams have been little explored in the literature. [Jembere et al. 2007] presents an approach to mine preferences in an environment with multiple context-aware services, but uses incremental learning only for the context, and not for the user's preferences. [Somefun and La Poutré 2007] presents an online method that aims at using aggregated knowledge on the preferences of many customers to make recommendations to individual clients. Finally, [Shivaswamy and Joachims 2011] proposes an online learning model that learns through preference feedback, and is especially suitable for web search and recommendation systems. None of them specifically addresses the problem we tackle in the present article.

In this article we propose two strategies for mining contextual preferences from preference data samples coming in a stream format. An extensive set of experiments on real data has been designed to compare one of these strategies with two baseline algorithms consisting in adapting well known classification techniques in streams in order to extract a *quantitative* preference model from a preference stream. The results show that our strategy is efficient to mining preferences in streams and outperforms the accuracy and comparability rate of the baseline algorithms.

2. BACKGROUND

In this section we briefly introduce the problem of mining contextual preferences in a batch setting. Please see [Amo et al. 2012] for more details on this problem.

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 2.1 *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 *bituple*, represents the fact that the user prefers *the tuple u to the tuple v*.

Example 2.1. 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 Fig. 1(a). Fig. 1 (b) illustrates a preference database over R, representing a sample provided by the user about his/her preferences over tuples of I.

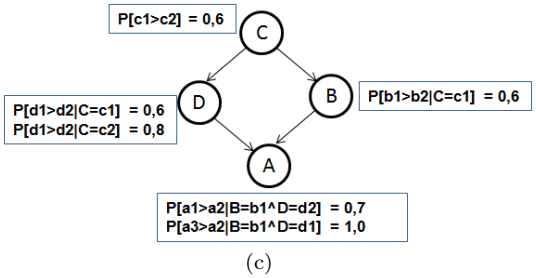
The problem of mining contextual preferences in the batch setting consists in extracting a *preference model* from a preference database provided by the user. The preference model is specified by a *Bayesian*

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 $\mathbf{PNet}_1$

Preference Network defined next.

Definition 2.2 *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 expressed by the formula $(A_{i_1} = a_{i_1}) \wedge \ldots \wedge (A_{i_k} = a_{i_k})$ 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 $(B = b_2)$", 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 2.2 *BPN.* Let $R(A, B, C, D)$ be the relational schema of the Example 2.1. Fig. 1(c) illustrates a preference network $\mathbf{PNet}_1$ over R.

Each conditional probability $P[E_2|E_1]$ in a BPN table stands for a *probabilistic* cp-rule, where the condition event E_1 is the *context* and the event E_2 is the *preference*. 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 $C = c_1$.

Before defining the accuracy, comparability rate and precision of a preference network, we need to define the *strict partial order* inferred by the preference network.

Example 2.3 *Preference Order.* Let us consider the BPN $\mathbf{PNet}_1$ depicted in Fig. 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 $\min(\Delta(u_1, u_2)) \subseteq \Delta(u_1, u_2)$ 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 $\min(\Delta(u_1, u_2)) = \{D, B\}$. 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_2 = probability that $u_2 > u_1 = P[d_2 > d_1|C = c_1] * P[b_2 > b_1|C = c_1] = 0.4 * 0.4 = 0.16$. In order to compare u_1 and u_2 we select the higher between p_1 and p_2. In this example, $p_1 > p_2$ and so, we infer that u_1 is preferred to u_2. If $p_1 = p_2$ we conclude that u_1 and u_2 are incomparable.

Definition 2.3 *Accuracy, Comparability Rate and Precision.* Let $\mathbf{PNet}$ be a BPN over a relational schema R. Let $\mathcal{P}$ be a *test* preference database over R. The *accuracy (acc)* of the $\mathbf{PNet}$ with respect to $\mathcal{P}$ is defined by $\text{acc}(\mathbf{PNet}, \mathcal{P}) = \frac{N}{M}$, where M is the number of bituples in $\mathcal{P}$ and N is the amount of bituples $(t_1, t_2) \in \mathcal{P}$ compatible with the preference ordering inferred by the $\mathbf{PNet}$ on the tuples t_1 and t_2. The *comparability rate (CR)* is defined by $\text{CR}(\mathbf{PNet}, \mathcal{P}) = \frac{F}{M}$ where F is the number of elements of $\mathcal{P}$ which are comparable by the $\mathbf{PNet}$. The *precision (prec)* is defined by $\text{prec}(\mathbf{PNet}, \mathcal{P}) = \frac{acc}{CR} = \frac{N}{F}$. Notice that each one of the three measures can be derived from the two others.

3. PROBLEM FORMALIZATION IN THE STREAM SETTING

The main differences between the batch and the stream settings as far as the contextual preference mining problem is concerned may be summarized as follows:

—*Input data*: to each sample bituple (u, v) collected from the stream of clicks from a user on a site, is associated a timestamp t standing for the time the user made this implicit choice. In this case, consider for example that the object represented by the tuple u was clicked first that the object represented by v (which may suggest that u is preferred to v), or even the object represented by v has not been clicked by the user. Let T be the infinite set of all timestamps. So, the input data from which a preference model will be extracted is a *preference stream* defined as a possibly infinite set $P \subseteq Tup(R) \times Tup(R) \times T$ which is *temporally consistent*, that is, if $(u, v, t) \in P$ then $(v, u, t) \notin P$. The triple (u, v, t) that we will call *temporal bituple*, represents the fact that the user prefers tuple u over tuple v at the time instant t.

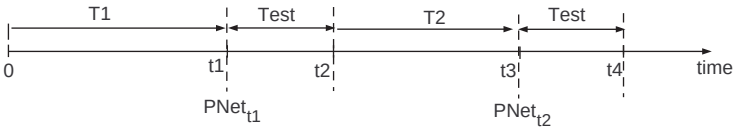


Fig. 2. The dynamics of the mining and testing processes through time

—*Output*: the preference model to be extracted from the preference stream is a *temporal BPN*, that is, a sequence of BPNs $< PNet_t : t \in \mathbb{N} >$. At each instant t the algorithm is ready to produce a preference model $PNet_t$ which will be used to predict the user preferences at this moment.

—*The preference order induced by a BPN at each instant t*: At each instant t we are able to compare tuples u and v by employing the Preference Model $PNet_t$ updated with the elements of the preference stream until the instant t. The preference order between u and v is denoted by $>_t$ and is obtained as illustrated in example 2.3.

—*The accuracy and comparability rate at instant t*: The quality of the preference model $PNet_t$ returned by the algorithm at instant t is measured by considering a finite set $Test$ of preference samples arriving at the system after instant t, that is, by considering a finite set $Test$ whose elements are of the form (u, v, t') with $t' \geq t$. Let $\mathcal{P}$ be the (non temporal) preference database obtained from $Test$ by removing the timestamp t' from its elements. The *acc* and *CR* measures of the preference model $PNet_t$ obtained at instant t are evaluated according to the formulae given in the previous section applied to the (static) BPN $PNet_t$ and the non temporal preference database $\mathcal{P}$.

Fig. 2 illustrates the dynamic process of mining and testing the preference models from the preference stream. In fact, as we will see in the following section, the mining algorithm is able to produce the preference model by using a reduced set of *statistics* extracted from the training sets T_i.

Now we are ready to state the problem of Mining Contextual Preferences in a Preference Stream:

Input: a relational schema $R(A_1, A_2, ..., A_n)$, and a preference stream over R.
Output: whenever requested, return a BPN_t over R having good accuracy and comparability rate, where t is the time instant of the request.

4. PROPOSED STRATEGIES

In this article we propose two strategies for mining user contextual preferences in the stream setting: the *Greedy Strategy* and the *Heuristic Strategy*.

In order to save processing time and memory, in both strategies we do not store the elements of the preference stream processed so far, we just collect *sufficient statistics* from it. Thus, the training of the model is made by online updating its sufficient statistics for every new element that comes in the stream. Example 4.1 illustrates the sufficient statistics collected from a preference stream S.

Example 4.1 *Sufficient Statistics.* Let $R(A, B, C)$ be a relational schema with a_1, $a_2 \in \text{dom}(A)$, b_3, b_5, $b_6 \in \text{dom}(B)$ and c_1, c_2, c_4, $c_5 \in \text{dom}(C)$. Let S be a preference stream over R as shown in Fig. 3(c), where T column shows the timestamp that the temporal bituple was generated, and $u_1 >_{t_i} u_2$ for every temporal bituple (u_1, u_2, t_i) in the preference stream, with $1 \leq i \leq 10$. Consider the sufficient statistics for attribute C shown in the Fig. 3(a) collected from the preference stream S until time instant t_9. Table on top of the Fig. 3(a) shows the *context counters* regarding the preferences over the attribute C, and table on the bottom shows the *general counters* over C. Context counters account for the possible causes for a particular preference over an attribute, and general counters stores the number of times that a particular preference over an attribute appeared in the stream. With the arrival of a temporal bituple at the time instant t_{10} - call it h, the sufficient statistics are updating as follows (see Fig. 3(b)): (1) Compute $\Delta(u_1, u_2)$, which is the set of attributes where u_1 and u_2 differ in h. In this example, $\Delta(u_1, u_2) = \{C\}$, then only the attribute C will have their statistics updated according to h; (2) Increment context counters a_1 and b_6 regarding the preference $c_2 > c_1$ (table on top

		$c_1 > c_2$	$c_2 > c_1$	$c_4 > c_5$
A	a_1	3	1	-
	a_2	-	-	2
B	b_3	1	-	1
	b_5	1	-	1

$c_1 > c_2$	4
$c_2 > c_1$	2
$c_4 > c_5$	3

(a)

		$c_1 > c_2$	$c_2 > c_1$	$c_4 > c_5$
A	a_1	3	2	-
	a_2	-	-	2
B	b_3	1	-	1
	b_5	1	-	1
	b_6	-	1	-

$c_1 > c_2$	4
$c_2 > c_1$	3
$c_4 > c_5$	3

(b)

	u_1			u_2		
T	A	B	C	A	B	C
t_1	a_1	b_3	c_1	a_1	b_3	c_2
t_2	a_1	b_3	c_2	a_1	b_5	c_1
t_3	a_2	b_5	c_2	a_1	b_3	c_1
t_4	a_2	b_3	c_4	a_2	b_6	c_5
t_5	a_1	b_5	c_1	a_1	b_5	c_2
t_6	a_2	b_3	c_4	a_2	b_3	c_5
t_7	a_1	b_3	c_1	a_1	b_5	c_2
t_8	a_2	b_5	c_1	a_1	b_6	c_2
t_9	a_1	b_5	c_4	a_2	b_5	c_5
t_{10}	a_1	b_6	c_2	a_1	b_6	c_1

(c)

Fig. 3. (a) Sufficient statistics for attribute C at the time instant t_9. (b) Sufficient statistics for attribute C at the time instant t_{10}. (c) Preference stream S until time instant t_{10}.

of the Fig. 3(b)). Notice that in the temporal bituple h the values a_1 and b_6 are possible contexts (causes) for the preference $c_2 > c_1$, just because they are equal in both tuples, i.e., $u_1[A] = u_2[A]$ and $u_1[B] = u_2[B]$. As until now we had no context b_6, it is inserted in the statistics; (3) Increment general counter of the preference $c_2 > c_1$ (table on the bottom of the Fig. 3(b)).

Memory Management. In order to limit the growth of the sufficient statistics, both strategies perform a simple memory management procedure described as follows. We can abstract our statistics as a tree, where the leaves are represented by general counters and context counters. For each leaf, we store the time of its last update. Periodically, in order to reduce runtime overhead, the strategies perform the memory management of their statistics, which in short consists in eliminating leaves that have not been visited since a long time (i.e., the time of its last update differs from the current time by an amount greater than a threshold). When a leaf is removed, we verify if the nodes traversed to reach this leaf recursively need to be removed as well, in the case of these nodes do not have other children. This means discarding information that is very old and is no longer relevant to the current scenario. According to tests carried out on real data this mechanism proved to be effective.

4.1 Greedy Strategy

The main idea of the Greedy Strategy is to create a preference relation from the most promising dependencies between attributes of a preference stream. In order to measure the dependence between a pair of attributes, we use the concept of *degree of dependence*. The degree of dependence of a pair of attributes (X, Y) is a real number that estimates how preferences on values for the attribute Y are influenced by values for the attribute X. We adapted the concept of *degree of dependence* defined in [Amo et al. 2012] to work with sufficient statistics. 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) We denote by $T^{time}_{yy'}$ the finite subset of temporal bituples $(u_1, u_2, t) \in S$, such that $t \leq time$, $(u_1[Y] = y \wedge u_2[Y] = y')$ or $(u_1[Y] = y' \wedge u_2[Y] = y)$; (2) We denote by $S^{time}_{x|(y,y')}$ the subset of $T^{time}_{yy'}$ containing the temporal bituples (u_1, u_2, t) such that $u_1[X] = u_2[X] = x$. Example 4.2 illustrates the computation of the degree of dependence on the statistics.

Example 4.2 *Degree of Dependence on the Statistics.* Let us consider the preference stream in the Fig. 3(c) until the time instant t_{10} and the snapshot (picture) Q of their sufficient statistics for attribute C shown in Fig. 3(b). In order to compute the degree of dependence of the pair (A, C) with respect to Q, we first identify the context counters related to A in the Fig. 3(b). The thresholds we consider are $\alpha_1 = 0.1$ and $\alpha_2 = 0.2$. The support of (c_1, c_2) and (c_4, c_5) are $(4 + 3)/10 = 0.70$ and $3/10 = 0.30$, respectively. Therefore, we do not discard any of them. Entering the inner loop for (c_1, c_2) we have only one set namely $S_{a_1|(c_1,c_2)}$. The support of $S_{a_1|(c_1,c_2)}$ is $5/5 = 1.0$ and $N = 3/5$. Hence, $f_1(S_{a_1|(c_1,c_2)}) = 3/5$ and $f_2(T_{c_1c_2}) = 3/5$. In the same way, for (c_4, c_5) we have $S_{a_2|(c_4,c_5)}$ with support $2/2 = 1.0$ and $N = 2/2 = 1.0$. Therefore, $f_1(S_{a_2|(c_4,c_5)}) = 1.0$ and $f_2(T_{c_4c_5}) = 1.0$. Thus, the degree of dependence of (A, C) is $f_3((A, C), Q) = \max\{3/5, 1.0\} = 1.0$. Besides, the degree of dependence of (B, C) is $f_3((B, C), Q) = 1.0$.

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

Input: Q: a snapshot of the sufficient statistics from the preference stream S at the time instant *time*; (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 Q at the time instant *time*

for *each pair* $(y, y') \in$ ***general counters*** *over* Y *from* Q, $y \neq y'$ *and* (y, y') *comparable* **do**

  for *each* $x \in$ ***dom***(X) *where* x *is a cause for* (y, y') *being comparable* **do**

    Let $f_1(S^{time}_{x|(y,y')}) = \max\{N, 1 - N\}$, where

$$N = \frac{|\{(u_1, u_2, t) \in S^{time}_{x|(y,y')} : u_1 >_t u_2 \wedge (u_1[Y] = y \wedge u_2[Y] = y')\}|}{|S^{time}_{x|(y,y')}|}$$

  Let $f_2(T^{time}_{yy'}) = \max\{f_1(S^{time}_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$

Let $f_3((X, Y), Q) = \max\{f_2(T^{time}_{yy'}) : (y, y') \in$ **general counters** over Y from Q, $y \neq y'$, (y, y') comparable$\}$

return $f_3((X, Y), Q)$

Given this, our strategy is straightforward and builds a BPN_t from sufficient statistics as follows: (1) Let S be a preference stream over the relational schema R. Take a snapshot Q of the sufficient statistics from S at the time instant t; (2) Calculate the degree of dependence (Alg. 1) between each possible pair of attributes of R according to Q. Let Ω be the resulting set of these calculations, with elements of the form (A_i, A_j, dd), where A_i and A_j are attributes of R and dd is the degree of dependence between (A_i, A_j); (3) Eliminate from Ω all elements whose $dd < 0.5$ (such dd indicates weak dependence); (4) Order the elements (A_i, A_j, dd) in Ω in decreasing order according to their dd; (5) Consider that the graph G_t of the BPN_t starts with a node for each attribute of R. Within a loop, take each $(A_i, A_j, dd) \in \Omega$ and insert the edge (A_i, A_j) in the graph G_t only if the insertion does not form cycles in G_t. Thus, graph G_t of the BPN_t is created; (6) Once we have G_t, we are now seeking for estimates of the conditional probabilities tables θ_t of the BPN_t. Since we have Q, we can estimate such parameters using the Maximum Likelihood Principle (see [Amo et al. 2012] for details).

This strategy is characterized as greedy, once in the attempt to construct a graph with no cycles it disposes edges that could be useful. However, this method is extremely efficient in terms of time and memory, and tends to produce good results, as shown in Section 5.

4.2 Heuristic Strategy

This strategy uses a heuristic method based on an incremental genetic algorithm (IGA) over a preference stream to perform the learning phase of the graph structure G_t of the BPN_t.

The main idea consists in evaluating the population of the IGA, not on a static database, but on the data coming from a stream. However, unlike the IGA proposed by [Vivekanandan and Nedunchezhian 2011], which evaluates the population over a window of the stream, we evaluated the population over the statistics collected from the preference stream S. This implies a low memory usage besides a higher processing speed. The fitness function used is the same proposed by [Amo et al. 2012] (*score function*), based in concept of *degree of dependence* between a pair of attributes (Alg. 1). However, in our strategy, the application of this function will be on a snapshot Q of the statistics. The snapshot is only for the evaluation process to be fair, so that individuals in a same population are not evaluated on different data. The score function assigns a real number in [-1, 1] for a candidate structure G_t, aiming to estimate how good it captures the dependencies between attributes in S. In this sense, each edge of the graph is "punished" or "rewarded", according to the matching between each edge (X, Y) in G_t and the corresponding degree of dependence of the pair (X, Y) with respect to Q.

In this strategy, the model learning process consists of two phases running in parallel: (1) Online updating of the sufficient statistics for every new temporal bituple that comes in the preference stream, just as the Greedy Strategy; (2) Running the IGA, where its population is always evaluated taking into account new data from the stream. In each generation of the IGA, the best individual (G_t) of

the population is kept in a variable called $BEST$ along with its conditional probability tables (θ_t), built in the same way as in the Greedy Strategy. Thus, whenever the model is tested over time, the BPN_t used will be the one stored in $BEST$.

This strategy builds a BPN_t as follows: (1) Let S be a preference stream over the relational schema R. Randomly generate an initial population P_0 of graphs without cycles over R; (2) Take a snapshot Q_0 of the statistics from S; (3) Evaluate the individuals in the population P_0 over Q_0 according to the *score function* [Amo et al. 2012]; (4) For each generation i, until there are data in the stream, do the following: (4.1) Select pairs of individuals for the crossover in P_i; (4.2) Apply crossover for each pair, generating an offspring population F_i; (4.3) Apply mutation over some individuals from F_i; (4.4) Take a snapshot Q_i of the statistics from S; (4.5) Evaluate the individuals of the population F_i over Q_i; (4.6) From $P_i \cup F_i$, select the $|P_i|$ fittest individuals according to their score in order to build the next population P_{i+1}; (4.7) Set in the variable $BEST$ the best individual of the population P_i, along with its set of conditional probability tables.

4.3 Complexity Analysis of Both Strategies

We divided the complexity analysis of both strategies in three parts, as explained below.
–*The complexity of processing each element of the stream and update the sufficient statistics*: (i) Scan a temporal bituple to save the Δ (set of attributes that have different values in the two tuples) and the B (set of attributes that have equal values in the two tuples) are $O(l)$, where l is the number of attributes; (ii) Update the statistics of an attribute is $O(|B|)$. Thus, update the statistics for all attribute in Δ is $O(|\Delta|.|B|)$. Therefore, the complexity for processing each element in both strategies in the worst case is $O(|\Delta|.|B|)$, where $|\Delta| + |B| = l$, and in the best case is $O(l)$.
–*The complexity of creating the preference model (BPN)*: (i) the computation of the degree of dependence between the pairs of attributes is $O(l^2)$; (ii) for the Greedy Strategy the computation of the topology is $\mathbf{O(l^2.e)}$, where e is the number of edges of the BPN. The sub-steps used in this calculation were: (a) the elimination of the pairs of attributes whose $dd < 0.5$ is $O(l^2)$, (b) the ordination of the pairs of attributes is $O(l^2.\log l)$ in the average case, and (c) for each pair of attributes the acyclicity test of the insertion of its edge in the graph is $O(e)$, so the total complexity of this sub-step is $O(l^2.e)$. For the Heuristic Strategy the computation of one iteration of the IGA is $\mathbf{O(l^2.q)}$, where q is the population size. The sub-steps used in this calculation were: (a) the selection for the crossover is $O(q)$, (b) the crossover operation is $O(l^2.q)$, (c) the mutation operation is $O(q)$, (d) the calculation of the fitness is $O(l^2.q)$, and (e) the select of the fittest individuals through an elitism procedure is $O(q.\log q)$; (iii) the computation of the conditional probability tables is $O(e.m)$, where m is the number of training instances processed. Therefore, the total complexity for building the model in the Greedy Strategy is $O(e.(l^2 + m))$ and in the Heuristic Strategy is $O(l^2.q + e.m)$. Notice that the difference of complexity between the two strategies is in the computation of the topology (in bold text), where the Greedy is dependent on the number of edges e (satisfying $e \leq l.(l-1)/2$) and the Heuristic is dependent on population size q of the IGA. Thus, for a small number of attributes ($e < q$), the Greedy Strategy tends to have better performance than the Heuristic Strategy. On the other hand, for large numbers of attributes and reasonable population sizes ($e > q$), the Heuristic Strategy tends to be more efficient.
–*The complexity of using the model (BPN)* for ordering a temporal bituple in both strategies is $O(l+e)$, which can be reduced to $O(l)$ when the BPN is a sparse graph.

5. EXPERIMENTAL RESULTS

We have implemented just the Greedy Strategy so far. In this section we describe the results concerning the execution of the Greedy Strategy over real datasets and the results comparing its performance with two well known baseline algorithms in the literature. The Greedy Strategy was implemented in Java and all the experiments performed on a Windows 7 machine with 3.40 GHz clocked processor and 12 GB RAM.

The Experiments Protocol. In order to evaluate the Greedy Strategy, we adapted the sampling

		Greedy Strategy			Hoeffding Tree					Naive Bayes				
User	Bituples	$\overline{acc}$	$\overline{CR}$	$\overline{prec}$	$\overline{acc}$	γ	$\overline{CR}$	γ	$\overline{prec}$	$\overline{acc}$	γ	$\overline{CR}$	γ	$\overline{prec}$
U_1	183,600	0.65	0.93	0.70	0.36	0.03	0.64	0.02	0.54	0.35	0.02	0.49	0.02	0.71
U_2	178,500	0.58	0.82	0.70	0.41	0.03	0.66	0.05	0.63	0.39	0.03	0.56	0.05	0.69
U_3	127,500	0.59	0.94	0.63	0.32	0.04	0.59	0.04	0.52	0.26	0.03	0.40	0.05	0.66
U_4	112,200	0.60	0.93	0.64	0.37	0.05	0.66	0.04	0.54	0.28	0.05	0.47	0.05	0.59
U_5	112,200	0.60	0.90	0.66	0.33	0.03	0.63	0.03	0.52	0.24	0.04	0.40	0.06	0.61
U_6	102,000	0.59	0.94	0.62	0.31	0.05	0.55	0.08	0.48	0.24	0.05	0.36	0.05	0.69
U_7	91,800	0.68	0.91	0.74	0.39	0.03	0.66	0.03	0.55	0.39	0.02	0.54	0.04	0.72
U_8	86,700	0.61	0.90	0.69	0.36	0.06	0.60	0.08	0.52	0.35	0.02	0.54	0.03	0.65
U_9	76,500	0.58	0.90	0.64	0.32	0.07	0.64	0.08	0.47	0.30	0.05	0.51	0.08	0.59
U_{10}	76,500	0.61	0.91	0.66	0.39	0.07	0.62	0.07	0.58	0.30	0.05	0.48	0.04	0.61

(a)

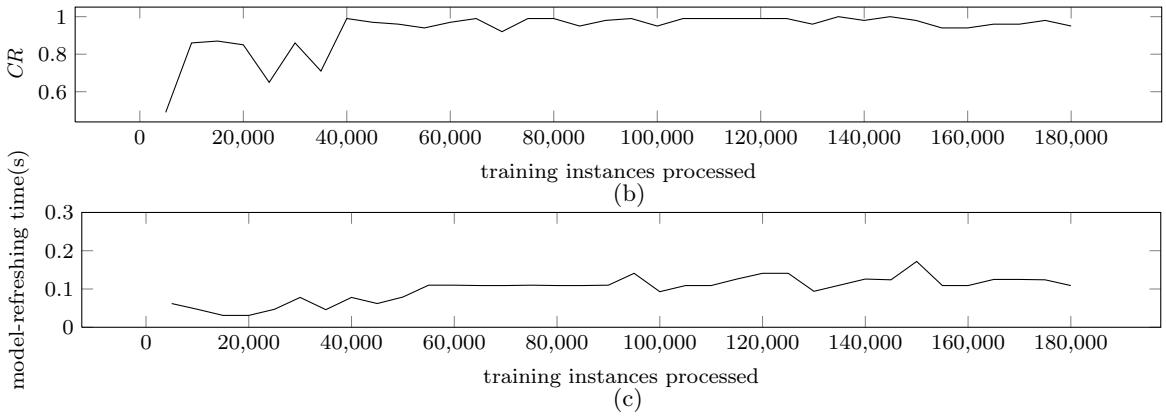


Fig. 4. Experimental Testing Set

technique proposed by [Bifet and Kirkby 2009] (holdout for data stream) to the preference mining scenario. This technique takes three parameters as input : n_{train}, n_{test} and n_{eval}. The n_{train} and n_{test} variables represent, respectively, the number of elements in the stream to be used to train and test the model at each evaluation. The variable n_{eval} represents the number of evaluations desired along the stream. For example, let us consider the values $n_{train} = 5{,}000$, $n_{test} = 100$ and $n_{eval} = 36$. Let $S = \{e_1, e_2, e_3, ...\}$ be a preference stream. The dynamic of evaluation for this example is as follows: (1) elements e_1 to e_{5000} from S are used to train the model; (2) elements e_{5001} to e_{5100} are used to test the quality of the model. The *acc*, *CR* and *prec* of the model are calculated according to this period; (3) elements e_{5101} to e_{10100} are used to train the model, and so on for 36 cycles.

Real Data. In order to evaluate the Greedy Strategy over real-world datasets, we considered data containing preferences related to movies collected by the GroupLens Research[1] from the MovieLens web site[2] concerning ten different users (named U_i, for $i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$). We simulated preference streams from these data, as follows: we stipulated a time interval λ, and each tuple in the dataset D_i of movies evaluated by the user U_i was compared to all others movies of D_i in a radius λ relative to its timestamp, thus generating temporal bituples for each user U_i. We calculated the λ (in days) for each user U_i according to the sparsity related to the time of the tuples in the dataset D_i, and the values obtained were: $U_1 = 77$, $U_2 = 37$, $U_3 = 6$, $U_4 = 82$, $U_5 = 140$, $U_6 = 50$, $U_7 = 60$, $U_8 = 69$, $U_9 = 7$ and $U_{10} = 135$. The resulting preference stream S_i has five attributes (director, genre, language, star and year), and its elements correspond to preferences on movies concerning user U_i. For example, the dataset D_1 is constituted by movies evaluated by the user U_1 from 5th Aug 2001 to 3rd Jan 2009 and the period comprising the ratings given by the ten users is Aug 2000 to Jan 2009.

The Baseline Algorithms. We did not find any algorithm in the literature that addresses the exact

[1]Available at `http://www.grouplens.org/taxonomy/term/14`
[2]Available at `http://movielens.umn.edu`

same problem we address in this article. Nevertheless, the classification task is the closest to the mining preference task among the numerous existing data mining tasks. So, we decided to design a baseline by adapting a classifier to compare the performance of the Greedy Strategy. In this approach, the classes correspond to the scores given by the users to the movies and can take the values: 1, 2, 3, 4 or 5 (the best score). This baseline has been designed in such a way that at each cycle of the holdout, both the training sample set and the test sample set of the classifier contain the same movies involved in temporal bituples used by the Greedy Strategy. This ensures a fair evaluation process. Besides, for a classifier to get a hit it is not necessary that it hits exactly the score given by the user to a particular movie in the test sample set. For example, suppose that the classifier has predicted the class c_1 to the movie f_1 and the class c_2 to the movie f_2, where f_1 and f_2 are in the test sample set. In this case, if f_1 was better rated than f_2 by the user in reality, so it is enough that $c_1 > c_2$ for the classifier to get a hit. On the other hand, if f_1 was better rated than f_2 by the user in reality and $c_1 = c_2$, then the baseline infers that the pair (f_1, f_2) is incomparable.

The Experiment Results. We consider a paired difference one tailed t-test to measure the statistical significance of the differences between the accuracy (resp. the comparability rate (CR), the precision (prec)) of the Greedy Strategy and the respective measure for each of the two baseline algorithms. We adapted the technique described in [Tan et al. 2005] (originally designed for comparing classifiers in a batch setting) for our preference mining task on data stream. For each one of the three quality measures d (where $d \in \{CR, acc, prec\}$) let $\overline{d}$ be the difference between the d-measure of the Greedy Algorithm and the respective d-measure of a given baseline algorithm. The main objetive of the t-test is to compute the confidence interval γ for d and test if the Null Hipothesis (H0) is rejected. The confidence interval γ is calculated as follows: $\gamma = t_{(1-\alpha),n_{eval}-1} \times \hat{\sigma}_{\overline{d}}$, where $t_{(1-\alpha),n_{eval}-1}$ is a coeficient obtained in the t-distribution table with $(1-\alpha)$ being the level of t-test and $n_{eval}-1$ is the degree of freedom. In all the experiments we considered $\alpha = 0.95$ and n_{eval} varied according the size of the stream. We evaluated the statistical significance of $\overline{d}$ where $d = CR$ and $d = acc$, since these measures are evaluated on samples of equal sizes for both algorithms.

Fig. 4(a) compares the performance of the Greedy Strategy with two baseline algorithms obtained by instantiating the baseline construction methodology described earlier with two classifiers widely used in the literature: Hoeffding Tree (HT) and Naive Bayes (NB). We used the MOA [Bifet et al. 2010] implementation of Naive Bayes and Hoeffding Tree in the default parameter setting[3]. The values for the holdout parameters were $n_{train} = 5,000$ and $n_{test} = 100$. For each user U_i, we calculated the value of n_{eval} in order to cover the whole stream S_i, and the following values were obtained: $U_1 = 36$, $U_2 = 35$, $U_3 = 25$, $U_4 = 22$, $U_5 = 22$, $U_6 = 20$, $U_7 = 18$, $U_8 = 17$, $U_9 = 15$ and $U_{10} = 15$. For the Greedy Strategy we used $\alpha_1 = 0.2$ and $\alpha_2 = 0.1$ as default values for the parameters of the Alg. 1 (referring to the degree of dependence). The choice of the parameters α_1 and α_2 depends on the data used in the tests and in this article they were chosen through an experimental study.

For these experiments, the main question is: With $\alpha = 95\%$ of certainty, can we conclude that the Greedy Strategy outperforms the other methods? The null and alternate hypotheses for acc and CR are $H0 : Greedy\ Strategy \leq HT$, NB and $HA : Greedy\ Strategy > HT$, NB. The results show that $H0$ is rejected (since all the values contained in the confidence interval are positive), and thus, HA is substantiated. This shows that even with limited data streams, the Greedy Strategy performs well, and outperforms both algorithms used as baselines. The poor results produced by these classifiers suggest that classifiers are not quite suitable for preference mining tasks. Thus, we argue that the Greedy Strategy, an algorithm specifically designed for preference mining, performs better than classical classifiers adapted for the preference mining task.

Performance Analysis. Fig. 4(b) shows the evolution of the comparability rate of the Greedy Strategy over the number of training instances processed. In this test, we used the stream S_1 (concerning

[3]Hoeffding Tree: gracePeriod $g = 200$, splitConfidence $c = 10^{-7}$, tieThreshold $t = 0.05$, numericEstimator $n = GAUSS10$

the user U_1) and we ploted all CR values used to compute the $\overline{CR}$ of 93% shown in the Fig. 4(a). Notice that with few training instances processed the Greedy Strategy can not compare many pairs of movies due the limited information that it has received so far. With the increasing in the number of processed training instances, the Greedy Strategy can learning more about the user's preferences and therefore it is able to compare more pairs of movies. Notice also that from 40,000 training instances processed the Greedy Strategy is able to compare almost all pairs of movies which are presented to it.

Execution Time. Fig. 4(c) shows the time measured in seconds taken by the Greedy Strategy to generate the model at each refreshing point. The same stream used in the experiment reported in Fig. 4(b) has been considered here. Notice that *model-refreshing time* (the time to build a new model) is very small, in the order of milliseconds. Though the Greedy Strategy builds the entire BPN at every holdout cycle, the sufficient statistics are updated incrementally, i.e., they are always ready to be used, which makes this process fast. Notice also that the time remains almost constant with the increasing in the number of training instances processed, mainly due to the efficiency of the memory management used.

6. CONCLUSION AND FURTHER WORK

In this article we proposed two strategies for extracting a Bayesian Preference Network from a preference stream and discussed their advantages and disadvantages. We implemented the Greedy Strategy and showed that it is fast and produces satisfactory results using real data. We also showed that the Greedy Strategy outperforms the accuracy and comparability rate of two baselines well known in the literature. As a promising future work, we plan to implement the Heuristic Strategy since the complexity analysis showed that it is feasible and tends to be more efficient than the Greedy Strategy in streams with larger numbers of attributes. We also plan to implement a synthetic preference stream generator, which will enable testing a data set as large as desired.

REFERENCES

Amo, S. d., Bueno, M. L. P., Alves, G., and Silva, N. F. CPrefMiner: An algorithm for mining user contextual preferences based on bayesian networks. In *Proceedings of the 2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. ICTAI '12. Washington, DC, USA, pp. 114–121, 2012.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* vol. 11, pp. 1601–1604, 2010.

Bifet, A. and Kirkby, R. Data stream mining: a practical approach. Tech. rep., The University of Waikato, 2009.

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

Domingos, P. and Hulten, G. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '00. NY, USA, pp. 71–80, 2000.

Jembere, E., Adigun, M. O., and Xulu, S. S. Mining context-based user preferences for m-services applications. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. WI '07. Washington, DC, USA, pp. 757–763, 2007.

Jiang, B., Pei, J., Lin, X., Cheung, D. W., and Han, J. Mining preferences from superior and inferior examples. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '08. New York, NY, USA, pp. 390–398, 2008.

Rajaraman, A. and Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, NY, USA, 2011.

Shivaswamy, P. K. and Joachims, T. Online learning with preference feedback. *CoRR* vol. abs/1111.0712, pp. 1–5, 2011.

Somefun, D. J. A. and La Poutré, J. A. A fast method for learning non-linear preferences online using anonymous negotiation data. In *Proceedings of the 2006 AAMAS workshop and TADA/AMEC 2006 conference on Agent-mediated electronic commerce*. TADA/AMEC'06. Springer, Berlin, Heidelberg, pp. 118–131, 2007.

Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2005.

Vivekanandan, P. and Nedunchezhian, R. Mining data streams with concept drifts using genetic algorithm. *Artif. Intell. Rev.* 36 (3): 163–178, 2011.

Wilson, N. Extending cp-nets with stronger conditional preference statements. In *Proceedings of the 19th national conference on Artifical intelligence*. AAAI'04. pp. 735–741, 2004.