# Mining Contextual Preferences in Data Streams

**Jaqueline A. J. Papini, Allan Kardec S. Soares** and **Sandra de Amo**
School of Computer Science
Federal University of Uberlândia, Brazil
jaque@comp.ufu.br, allankardec@gmail.com, deamo@ufu.br

## Abstract

The dynamic nature of the problem of mining preferences increasingly requires solutions that quickly adapt to change. The main reason for this is that most of the time the user's preferences are not static and can evolve over time. In this article we formalize the problem of mining contextual preferences in the stream setting and propose two algorithms to solve this problem – an incremental and a non-incremental one. We implemented these algorithms and showed their efficiency through a set of experiments over real and synthetic data.

## Introduction

Mining data streams is one of the most interesting subjects of data mining in recent years. A data stream may be seen as a sequence of relational tuples that arrive continuously at high speed and variable time. 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 stream mining techniques have emerged to deal properly with this new data format (Bifet et al. 2011).

Nevertheless, 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; de Amo et al. 2012). However it is natural to think on user preferences as something dynamic that evolves over time. As a motivating example, consider an *online news site* that wants to discover the preferences of its users 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. 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 in times of elections. Thus, in these times, a user can be more interested in *politics* than in *sports*. However, in times of Olympic Games, this same user might consider *sports* as his or her favorite category.

This work focuses on a particular kind of preferences – the *contextual preferences*. Preference Models can be specified under either a *quantitative* (Crammer and Singer 2001) or a *qualitative* (de 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 specifying that some values of a particular attribute are preferable to others in a given context.

In this article we propose two algorithms (FPSMining and IncFPSMining) for mining contextual preferences from preference data samples coming in a stream format. The experiments carried out show that our algorithms are efficient to mining user preferences in a stream setting and outperform the accuracy and comparability rate of the baseline.



## Background

In this section we briefly introduce the problem of mining contextual preferences in a batch setting. Please see (de 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. We denote by $a_1 > a_2$ the fact that $a_1$ is preferred to $a_2$. A *Preference Database* over a relation $R$ 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 $v$*. Fig. 1 (b) illustrates a preference database over $R$, representing a sample provided by the user about his/her preferences over tuples of $I$ (Fig. 1 (a)).

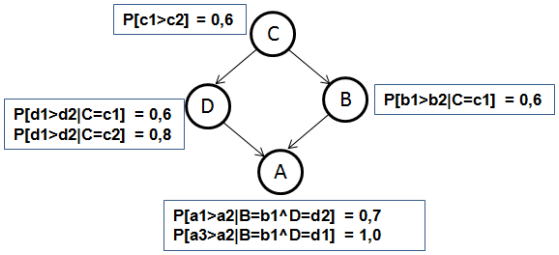
The problem of mining contextual preferences in the batch setting consists in extracting a *preference model* from a preference database provided by the user. The preference model is specified by a *Bayesian Preference Network* (BPN), defined by (1) a directed acyclic graph $G$ whose nodes are attributes and the edges stand for attribute dependency and (2) a mapping $\theta$ that associates to each node of $G$ a finite set of conditional probabilities. Fig. 1(c) 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%. A BPN allows inferring

| 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)

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

a *preference ordering* on tuples. The following example illustrates how this ordering is obtained.

***Example* 1** *Preference Order*. Let us consider the BPN $\mathbf{PNet}_1$ depicted in Fig. 1(c). In order to compare the tuples $u_1 = (a_1, b_1, c_1, d_1)$ and $u_2 = (a_2, b_2, c_1, d_2)$, we proceed as follows: (1) Let $\Delta(u_1, u_2)$ be the set of attributes where $u_1$ 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 $\Delta$ (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.

A BPN is evaluated by considering its *accuracy (acc)* and *comparability rate (CR)* with respect to a *test* preference database $\mathcal{P}$. The *accuracy* is defined by acc(**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 **PNet** on the tuples $t_1$ and $t_2$. The *comparability rate* is defined by CR(**PNet**,$\mathcal{P}$)= $\frac{F}{M}$ where $F$ is the number of elements of $\mathcal{P}$ which are comparable by **PNet**. The *precision (prec)* is defined by prec(**PNet**,$\mathcal{P}$)= $\frac{acc}{CR} = \frac{N}{F}$.

## Problem Formalization in the Stream Setting

The main differences between the batch and the stream settings concerning the preference mining problem that we address in this article 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. 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 $(u, v, t)$ (called *temporal bituple*), represents the fact that the user prefers tuple $u$ over tuple $v$ at $t$.

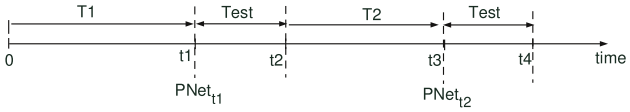



Figure 2: The mining and testing processes through time.

- *Output*: the preference model to be extracted from the preference stream is a *temporal BPN*, that is, a $PNet_t$ representing the model state at instant $t$. At each instant $t$ the algorithm is ready to return a model $PNet_t$ updated with the stream elements until instant $t$, 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 instant $t$. The preference order between $u$ and $v$ is denoted by $>_t$ and is obtained as illustrated in example 1.
- *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 process of mining and testing the preference model over time from the preference stream.

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

– *Input*: a relational schema $R(A_1, A_2, ..., A_n)$, and a preference stream $S$ over $R$.

– *Output*: whenever requested, return a $BPN_t$ over $R$, where $t$ is the time instant of the request.

## Algorithms for Mining Preferences in Streams

In this article we propose two algorithms for mining user contextual preferences in the stream setting: *FPSMining Algorithm* and *IncFPSMining Algorithm*. In order to save processing time and memory, in both algorithms we do not store the elements of the preference stream processed so

far, we just collect *sufficient statistics* from it. In both algorithms, the sufficient statistics are *incrementally* updated online for every new element that comes in the preference stream and the training process is carried out by extracting a preference model (a BPN) from these statistics. In order to limit the growth of the statistics, both algorithms perform the memory management occupied by them. The main difference between the two algorithms is in the way they build the preference model (the BPN): the first algorithm builds it from scratch whenever required, and the second one builds it incrementally. Example 2 illustrates the sufficient statistics collected from a preference stream.

***Example* 2** *Sufficient Statistics*. Let $S$ be a preference stream over $R(A, B, C)$ as shown in Fig. 3(c), where the $T$ column stands for the time when the temporal bituple was generated, and $u_1 >_{t_i} u_2$ ($u_1$ is preferred to $u_2$ at $t_i$) for every temporal bituple $(u_1, u_2, t_i)$ in the preference stream, for $1 \leq i \leq 10$. Consider the statistics for attribute $C$ shown in Fig. 3(a) collected from $S$ until the instant $t_9$. The tables on the top and on the bottom of Fig. 3(a) show respectively the *context counters* and the *general counters* over $C$. Context counters account for the possible causes for a particular preference over values of 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 $l = (u_1, u_2, t_{10})$ the statistics are updated 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 $l$. In this example, $\Delta(u_1, u_2) = \{C\}$, and so only the attribute $C$ will have its statistics updated with the arrival of $l$; (2) Increment context counters $a_1$ and $b_6$ regarding the preference $c_2 > c_1$ (table on the top of Fig. 3(b)). Notice that in the temporal bituple $l$ 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 ($u_1$ and $u_2$). Since we had no context $b_6$ so far, it is inserted in the statistics; (3) Increment general counter of the preference $c_2 > c_1$ (table on the bottom of Fig. 3(b)).

## The FPSMining Algorithm

The main idea of FPSMining (**F**ast **P**reference **S**tream **Mining**) is to create a preference relation from the most promising dependencies between attributes of a preference stream. 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 of the attribute $X$. Its computation is carried out as described in Alg. 1. In order to facilitate the description of Alg. 1 we introduce some notations as follows: (1) 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 3 illustrates the computation of the degree of dependence on the statistics.

***Example* 3** *Degree of Dependence on the Statistics*. Let us consider the preference stream in Fig. 3(c) until instant $t_{10}$ and the snapshot $Q$ of its 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 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 named $S_{a_1|(c_1,c_2)}$. The support of $S_{a_1|(c_1,c_2)}$ is $5/5 = 1.0$ and $N = 3/5$. Hence, $f_1(S_{a_1}) = 3/5$ 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 $f_3((A, C), Q) = \max\{3/5, 1.0\} = 1.0$.

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

**Input**: $Q$: a snapshot of the 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 $time$.

1 **for** *each pair* $(y, y') \in$ ***general counters*** *over* $Y$ *from* $Q$, $y \neq y'$ *and* $(y, y')$ *comparable* **do**
2  **for** *each* $x \in$ ***dom***$(X)$ *where* $x$ *is a cause for* $(y, y')$ *being comparable* **do**
3  Let $f_1(S^{time}_{x|(y,y')}) = \max\{N, 1 - N\}$, where $N =$
4  $\frac{|\{(u_1, u_2, t) \in S^{time}_{x|(y,y')} : u_1 >_t u_2 \wedge (u_1[Y] = y \wedge u_2[Y] = y')\}|}{|S^{time}_{x|(y,y')}|}$
5  Let $f_2(T^{time}_{yy'}) = \max\{f_1(S^{time}_{x|(y,y')}) : x \in \textbf{dom}(X)\}$
6 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$\}$
7 **return** $f_3((X, Y), Q)$

Given this, our algorithm is straightforward and builds a $BPN_t$ from the preference stream using the Alg. 2.

## The IncFPSMining Algorithm

The main idea of IncFPSMining is the following: for each *chunk* of $b$ temporal bituples arrived (parameter of the algorithm called "*grace period*") the current preference model $M$ built so far is *updated*. This model $M$ consists of a graph with some edges $e_1$,$e_2$,...,$e_n$, each one with degree of dependence $dd$ measured at the time that $M$ has been constructed. The *gap* of an edge $e_i$ measures how close $dd$ is from the minimum limit 0.5, that is, $gap = dd - 0.5$. Only the edges having *gap sufficiently high* are admitted at each update. The threshold is given by the *Hoeffding Bound* $\epsilon$ (Hoeffding 1963) associated to the random variable *gap*. It is computed as follows:

$$\epsilon = \sqrt{\frac{R^2 ln(\frac{1}{\delta})}{2n}}, \text{ where:}$$

-$R$ is the size of the range of values of the random variable $X$ associated to the problem considered. In our case, $X = gap$. Therefore, the higher value $R$ of $gap$ is 0.5. So, $R = 0.5$.

-$\delta$ is the probability that $X_{current} - X_{future} > \epsilon$.

-$n$ is the number of temporal bituples seen so far.

The Hoeffding Bound ensures (with an error probability $\delta$) that, if the degree of dependence $dd_t$ of an edge $e$

| | | $c_1 > c_2$ | $c_2 > c_1$ | $c_4 > c_5$ |
|---|---|---|---|---|
| $A$ | $a_1$ | 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)

Figure 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 the time instant $t_{10}$.

**Algorithm 2:** The FPSMining Algorithm

**Input**: $R(A_1, A_2, ..., A_n)$: a relational schema; $S$: a preference stream over $R$.
**Output**: whenever requested, return a $BPN_t$ over $R$.

Take a snapshot $Q$ of the statistics from $S$ at $t$.
**for** *each pair of attributes* $(A_i, A_j)$, *with* $1 \leq i, j \leq n, i \neq j$ **do**
    Use Alg. 1 for calculate the degree of dependence $dd$ between the pair $(A_i, A_j)$ according to $Q$
Let $\Omega$ be the resulting set of these calculations, with elements of the form $(A_i, A_j, dd)$
Eliminate from $\Omega$ all elements whose $dd < 0.5$ (indicates a weak dependence between a pair of attributes)
Order the elements $(A_i, A_j, dd)$ in $\Omega$ in decreasing order according to their $dd$
Start the graph $G_t$ of the $BPN_t$ with a node for each attribute
**for** *each element* $(A_i, A_j, dd) \in$ *ordered set* $\Omega$ **do**
    Insert the edge $(A_i, A_j)$ in the graph $G_t$ only if the insertion does not form cycles in $G_t$
Once $G_t$ was created, estimate the tables $\theta_t$ of the $BPN_t$, using the Maximum Likelihood Principle (see (de Amo et al. 2012) for details) over $Q$.
**return** $BPN_t$

at instant $t$ satisfies $dd_t - 0.5 \geq \epsilon$, when the number of temporal bituples seen so far was $n$, then in any future instant $t_{fut}$ its degree of dependence $dd_{fut}$ must satisfy $(dd_t - 0.5) - (dd_{fut} - 0.5) \leq \epsilon$. That is, $dd_t - dd_{fut} \leq \epsilon$ and so $dd_{fut}$ is not very far from the acceptable degree of dependence at the current instant $t$. So, the edges that were introduced in an earlier phase will not have their $dd$ worsened too much in the future, that is, they will not get closer to the limit $0.5$ than before.

This algorithm only considers the statistics related to edges that have not been inserted in the graph so far. Thus, we first select edges not belonging to the current graph, whose $dd$ verifies the Hoeffding Bound condition ($gap = dd - 0.5 \geq \epsilon$). For each of these edges we test if its inclusion produces cycles in the graph. If so, we evaluate the $dd$ of all edges in the cycle, and eliminate the edge with the worst $dd$. In case of tie in choosing edges to be eliminated from the cycle, we always choose to eliminate older edges. The IncFPSMining algorithm is described in Alg. 3.

**Algorithm 3:** The IncFPSMining Algorithm

**Input**: $R(A_1, A_2, ..., A_n)$: a relational schema; $S$: a preference stream over $R$.
**Output**: whenever requested, return a $BPN_t$ over $R$.

Let $G_t$ be a graph with a node for each attribute of $R$
**for** *each temporal bituple* $l$ *of* $S$ **do**
    Increment $n$, the number of elements seen until $t$
    **if** $n$ *mod grace period = 0* **then**
        Compute Hoeffding bound $\epsilon$
        Take a snapshot $Q$ of the statistics from $S$ at $t$
        **for** *each possible edge* $e_i$ *outside* $G_t$ **do**
            Use Alg. 1 for calculate the degree of dependence $dd$ of $e_i$ according to $Q$
        Let $\Omega$ be the resulting set of these calculations, with elements of the form $(e_i, dd)$
        Order the elements $(e_i, dd)$ in $\Omega$ in decreasing order according to their $dd$
        **for** *each pair* $(e_i, dd) \in$ *ordered set* $\Omega$ **do**
            **if** $dd - 0.5 \geq \epsilon$ **then**
                Insert the edge $e_i$ in $G_t$
                **if** $e_i$ *has created cycle in* $G_t$ **then**
                    Remove from $G_t$ the edge with lower $dd$ in the cycle
        Once $G_t$ was created, estimate the tables $\theta_t$ of the $BPN_t$, using the Maximum Likelihood Principle (see (de Amo et al. 2012) for details) over $Q$.

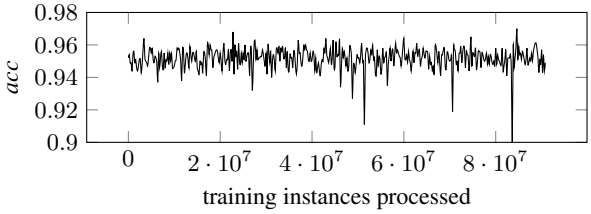
## Experimental Results

In this section we describe the results concerning the performance of our algorithms over synthetic and real datasets. The algorithms were implemented in Java and all the experiments performed on a Windows 7 machine with 3.40 GHz clocked processor and 12 GB RAM.

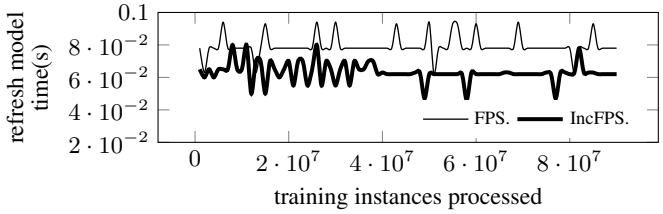
We adapted the statistical significance technique described in (Tan, Steinbach, and Kumar 2005) for our preference mining task on data stream. For this approach, we need to use a *t*-distribution to compute the confidence interval for the true difference between the algorithms: $d_t^{cv} = \overline{d} \pm \gamma$, where $\overline{d}$ is the average difference of the experiment, $\gamma = t_{(1-\alpha),n_{eval}-1} \times \hat{\sigma}_{d^{cv}}$ and $\hat{\sigma}_{d^{cv}}$ is the standard deviation. The coefficient $t_{(1-\alpha),n_{eval}-1}$ is obtained from a probability table with two input parameters, its confidence level $(1-\alpha)$ and the number of degrees of freedom $(n_{eval} - 1)$.

| Stream | IncFPS. | | | FPS. | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\overline{acc}$ | $\overline{CR}$ | $\overline{prec}$ | $\overline{acc}$ | $\gamma$ | $\overline{CR}$ | $\gamma$ | $\overline{prec}$ |
| 10m and 10att | 0.95236 | 1.000 | 0.95236 | 0.95151 | $3\times10^{-4}$ | 0.999 | $3\times10^{-4}$ | 0.95238 |
| 10m and 15att | 0.94762 | 1.000 | 0.94762 | 0.94673 | $3\times10^{-4}$ | 0.999 | $3\times10^{-4}$ | 0.94759 |
| 50m and 10att | 0.95239 | 1.000 | 0.95239 | 0.95162 | $1\times10^{-4}$ | 0.999 | $1\times10^{-4}$ | 0.95239 |
| 50m and 15att | 0.94765 | 1.000 | 0.94765 | 0.94678 | $1\times10^{-4}$ | 0.999 | $1\times10^{-4}$ | 0.94763 |
| 100m and 10att | 0.95240 | 1.000 | 0.95240 | 0.95160 | $9\times10^{-5}$ | 0.999 | $9\times10^{-5}$ | 0.95240 |
| 100m and 15att | 0.94771 | 1.000 | 0.94771 | 0.94684 | $1\times10^{-4}$ | 0.999 | $1\times10^{-4}$ | 0.94768 |

(a)




(b)




(c)

Figure 4: Experimental Results on Synthetic Data.

**The Experiments Protocol.** We adapted the sampling technique proposed by (Bifet et al. 2011) (based on holdout for data stream) to the preference mining scenario. This sampling technique takes three parameters as input : $n_{train}$, $n_{test}$ and $n_{eval}$. The $n_{train}$ and $n_{test}$ parameters represent, respectively, the number of elements in the stream to be used to train and test the model at each evaluation. The parameter $n_{eval}$ represents the number of evaluations desired along the stream. For example, let us consider the values[1] $n_{train}$ = 10k, $n_{test}$ = 1k and $n_{eval}$ = 9090. Consider $S = \{e_1, e_2, e_3, ...\}$ the preference stream used in the tests. The dynamic of the evaluation for this example is as follows: (1) elements $e_1$ to $e_{10k}$ from $S$ are used to train the model; (2) elements $e_{10001}$ to $e_{11k}$ are used to test the quality of the model. The *acc*, *CR* and *prec* of the model are calculated according to this test period; (3) elements $e_{11001}$ to $e_{21k}$ are used to train the model, and so on for 9090 cycles.

**Synthetic Data.** The synthetic data were generated by an algorithm based on Probabilistic Logic Sampling (Jensen and Nielsen 2007), which samples bituples for a preference stream $S$ given a BPN with structure $G$ and parameters $\theta$. We have considered different BPNs varying the number of attributes (10, 15) and the stream length[2] (10m, 50m, 100m); the size of the domain for each attribute is 10.

The values for the holdout parameters are $n_{train} = 10k$ and $n_{test} = 1k$. For tests with 10m, 50m, 100m of elements, we used $n_{eval_{10m}} = 909$, $n_{eval_{50m}} = 4545$ and $n_{eval_{100m}} = 9090$. For all tests, we used $\alpha_1 = 0.2$ and $\alpha_2 = 0.1$ (thresholds used in the *degree of dependence*). The choice of the parameters $\alpha_1$ and $\alpha_2$ depends on the data used in the tests and in this article they were chosen through an experimental study. The default values for IncFPSMining are $\delta = 10^{-7}$ (value usually used by the research community of data stream mining) and *grace period* = 1k. The value of *grace period* (1k) was defined according to $n_{train}$ (10k), so that would be performed ten updates in the model built so far along each period of training of the holdout protocol.

*1. Performance Analysis.* Fig. 4(a) shows the average values of *acc*, *CR* and *prec* obtained with FPSMining and IncFPSMining for streams with different sizes and numbers of attributes. The quality of both algorithms remained stable over the six different streams. In this test we also calculated the statistical significance regarding the slight improvement presented by IncFPSMining compared to FPSMining. Our question is: At $\alpha = 95\%$, can we conclude that IncFPSMining outperforms FPSMining? The null and alternate hypotheses for *acc* and *CR* are $H0 : IncFPS \leq FPS$ and $HA : IncFPS > FPS$. The results show that $H0$ is rejected, and thus, $HA$ is substantiated. Thus, although the difference between the algorithms is very small, it is statistically significant. Fig. 4(b) illustrates how *acc* of FPSMining evolves over time. Here we consider the stream with 100m of elements and 10 attributes of Fig. 4(a). This curve shows that the *acc* values of FPSMining were stable over time and also were very close to the average values (Fig. 4(a)).

*2. Execution Time.* Fig. 4(c) shows the time measured in seconds taken by FPSMining and IncFPSMining to generate the model at each refresh point. The same data used in the experiment (b) have been considered here. Notice that the time to refresh the model is very small for both algorithms, on the order of milliseconds. Notice also that for both algorithms the time remains almost constant with increasing number of training instances processed, mainly due to the efficiency of the memory management used. Finally, notice that IncFPSMining needs less time to generate the model, due to the incremental building of the graph of the BPN.

**Real Data.** In these tests we considered data containing preferences related to movies collected by GroupLens Research from the MovieLens web site[3] 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

[1] 1k = 1000.

[2] We often abbreviate *million* by *m* in the text.

[3] Available at http://movielens.umn.edu

| User | Tuples | FPS. | | | Hoeffding Tree | | | | | Naive Bayes | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\overline{acc}$ | $\overline{CR}$ | $\overline{prec}$ | $\overline{acc}$ | $\gamma$ | $\overline{CR}$ | $\gamma$ | $\overline{prec}$ | $\overline{acc}$ | $\gamma$ | $\overline{CR}$ | $\gamma$ | $\overline{prec}$ |
| $U_1$ | 7359 | 0.658 | 0.930 | 0.707 | 0.360 | 0.033 | 0.642 | 0.027 | 0.546 | 0.353 | 0.021 | 0.492 | 0.027 | 0.713 |
| $U_2$ | 7046 | 0.583 | 0.823 | 0.708 | 0.418 | 0.039 | 0.662 | 0.055 | 0.634 | 0.391 | 0.036 | 0.561 | 0.052 | 0.694 |
| $U_3$ | 5167 | 0.597 | 0.944 | 0.634 | 0.325 | 0.041 | 0.596 | 0.041 | 0.521 | 0.267 | 0.032 | 0.407 | 0.058 | 0.662 |
| $U_4$ | 4483 | 0.600 | 0.930 | 0.648 | 0.376 | 0.051 | 0.664 | 0.043 | 0.541 | 0.281 | 0.050 | 0.477 | 0.056 | 0.591 |
| $U_5$ | 4449 | 0.605 | 0.906 | 0.667 | 0.333 | 0.037 | 0.638 | 0.035 | 0.523 | 0.242 | 0.044 | 0.402 | 0.063 | 0.611 |
| $U_6$ | 4165 | 0.591 | 0.943 | 0.627 | 0.312 | 0.056 | 0.551 | 0.085 | 0.484 | 0.248 | 0.050 | 0.362 | 0.059 | 0.690 |
| $U_7$ | 3755 | 0.680 | 0.918 | 0.743 | 0.390 | 0.033 | 0.661 | 0.037 | 0.558 | 0.399 | 0.027 | 0.546 | 0.044 | 0.728 |
| $U_8$ | 3414 | 0.617 | 0.900 | 0.691 | 0.362 | 0.067 | 0.609 | 0.084 | 0.525 | 0.355 | 0.025 | 0.543 | 0.037 | 0.650 |
| $U_9$ | 3187 | 0.584 | 0.906 | 0.646 | 0.324 | 0.072 | 0.649 | 0.085 | 0.476 | 0.306 | 0.059 | 0.516 | 0.087 | 0.597 |
| $U_{10}$ | 3164 | 0.610 | 0.912 | 0.668 | 0.394 | 0.075 | 0.622 | 0.075 | 0.589 | 0.302 | 0.052 | 0.481 | 0.049 | 0.619 |

Table 1: Baseline with Real Data.

stipulated a time interval $\lambda$, 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 $\lambda$ relative to its timestamp, thus generating temporal bituples for each user $U_i$. 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.

*Baseline.* We did not find any published algorithm that addresses the exact same problem we address. As in data mining the classification task is the closest to mining preferences, we adapted a baseline with classifiers to compare the performance of our algorithms. In this approach, the classes are the ratings given by the users to the movies and can take the values: 1, 2, 3, 4 or 5. We designed this baseline so that in each cycle of the holdout, the sets of training and testing samples of our algorithms contain the same movies used by the classifiers. This ensures a fair evaluation process.

Table 1 compares the performance of FPSMining with two baselines widely used in the literature: Hoeffding Tree and Naive Bayes. We ran IncFPSMining on these same data, performing tests with both hypotheses: 1) assuming that FPSMining is strictly better than IncFPSMining and 2) vice versa. Both hypotheses were rejected, which leads us to conclude that our two algorithms have tied in these data. Therefore, we showed only the results for FPSMining in Table 1. For these experiments, we used the MOA (Bifet et al. 2010) implementation of Naive Bayes and Hoeffding Tree in the default configuration parameters[4]. The values for the holdout parameters were $n_{train} = 150$ and $n_{test} = 100$. For FPSMining we used $\alpha_1 = 0.2$ and $\alpha_2 = 0.1$. For these experiments, the main question is: At $\alpha = 95\%$, can we conclude that FPSMining outperforms the other methods? The null and alternate hypotheses for *acc* and *CR* are $H0 : FPS \leq HT, NB$ and $HA : FPS > HT, NB$. The results show that $H0$ is rejected, and thus, $HA$ is substantiated. This shows that FPSMining outperforms both algorithms used as baseline. We also calculated the statistical significance of the difference between IncFPSMining and the two baselines, and we have concluded that IncFPSMining outperforms both. Thus, we can conclude that our algorithms, which were specifically designed for preference mining, perform better than classical classifiers.

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

## Conclusion and Further Work

In this article we introduced the problem of mining user preferences in the stream setting and proposed the algorithms IncFPSMining (incremental) and FPSMining (non-incremental) to solve it. Both algorithms were implemented and a varied set of experiments showed that both are fast and produce satisfactory results. As a future work, we intend to study the behavior of both algorithms under stream data affected by concept drift.

## Acknowledgments

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

## References

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

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2011. Data stream mining: A practical approach. Technical report, The University of Waikato.

Crammer, K., and Singer, Y. 2001. Pranking with ranking. In *Proceedings of the NIPS 2001*, 641–647.

de Amo, S.; Bueno, M. L. P.; Alves, G.; and Silva, N. F. 2012. CPrefMiner: An algorithm for mining user contextual preferences based on bayesian networks. In *Proceedings of the ICTAI '12*, 114–121.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58:13–30.

Jensen, F. V., and Nielsen, T. D. 2007. *Bayesian Networks and Decision Graphs*. Springer Publishing Company, 2nd edition.

Jiang, B.; Pei, J.; Lin, X.; Cheung, D. W.; and Han, J. 2008. Mining preferences from superior and inferior examples. In *Proceedings of the KDD '08*, 390–398.

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

Wilson, N. 2004. Extending cp-nets with stronger conditional preference statements. In *Proceedings of the National Conference on Artifical Intelligence*, AAAI'04, 735–741.