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. The traditional preference mining setting, referred to here as the *batch setting*, has been widely studied in the literature in recent years. However, the dynamic nature of the problem of mining preferences increasingly requires solutions that quickly adapt to change. The main reason for this is that frequently user's preferences are not static and can evolve over time. In this paper, 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 preferences 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. As a first attempt to evaluate these strategies, we implemented one of them, the *greedy* one, and showed 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

The huge increase in the volume of digital data seen in recent years was partly caused by applications in which the data is generated at very high rates, in the form of data streams. In general 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: the financial market, credit card transaction flow, web applications and sensor data. Traditional approaches 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; Oza and Russell 2001; Bifet and Kirkby 2009; Bifet et al. 2010; Gama 2010; Vivekanandan and Nedunchezhian 2011; Yong et al. 2012].

Most research work in data mining addresses the batch setting. In particular, 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 [Holland et al. 2003; Jiang et al. 2008; de Amo et al. 2012]. However, it is somewhat intuitive to think of user preferences as something dynamic that evolves over time. Besides, there is a huge loss of information when one tries to treat something dynamic as if it were static. The most crucial issues which makes 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; each tuple must be accepted as it arrives and once inspected or ignored it is discarded with no possibility to be inspected again; (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 on these challenges see [Rajaraman and Ullman 2011].

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

As a motivating example on the dynamic character of 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, the place where it occurred, its category, author and keywords. Note 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 in times of elections. Thus, in these times, a user can be more interest in *politics* than in *sports*. However, in times of Olympic Games, this same user might consider *sports* as his or her favorite category. Still in this scenario, it may be the case that a particular keyword is in vogue at the moment. After a certain time, it can be that new keywords appear and begin to be the focus of attention. Keywords that previously attracted much attention gradually disappear with time. This reflects the fact that the domain of an attribute may not be finite and may evolve with time. Thus, it is clear that a batch mining algorithm is not best suited for applications that operate in dynamic environments.

This work focus 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. Those films with the higher score are the preferred ones. A preference model here is a *score function*. In the qualitative formulation, the preference model consists in a set of rules specified in a given 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 some other attributes. For example in our movie database scenario, a user can specify his/her preference concerning the attribute *director* depending on the value of the attribute *genre*: For movies whose director is *Woody Allen* he/she prefers *comedy* to *suspense* and for movies from director *Steven Spielberg* he/she prefers *action* films to *drama*.

The main goal of this article is to propose two strategies for mining contextual preferences from a stream of preference samples. We also present the results of preliminary tests for one of these strategies executed on real data. We aim at showing that it is possible to obtain satisfactory results as fast as possible in order to meet the time constraints of data stream environments using minimal machine resources.

2. 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. In the next section, all the concepts introduced here are naturally generalized to the stream setting, by simply aggregating the time dimension.

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

Example 2.2. Let $R(A, B, C, D)$ be a relational schema with attribute domains given by $\mathbf{dom}(A) = \{a_1, a_2, a_3\}$, $\mathbf{dom}(B) = \{b_1, b_2\}$, $\mathbf{dom}(C) = \{c_1, c_2\}$ and $\mathbf{dom}(D) = \{d_1, d_2\}$. Let I be an instance

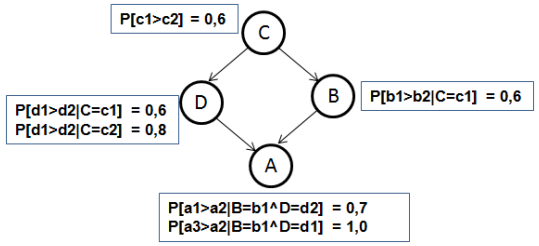
over R as shown in Figure 1(a). Figure 1 (b) illustrates a preference database over R, representing a sample provided by the user about his/her preferences over tuples of I.

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

(a)

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

(b)



(c)

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

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

Definition 2.3 *Bayesian Preference Network (BPN).* A *Bayesian Preference Network* (or BPN for short) over a relational schema $R(A_1, ..., A_n)$ is a pair (G, θ) where: (1) G is a directed acyclic graph whose nodes are attributes in $\{A_1, ..., A_n\}$ and the edges stand for attribute dependency; (2) θ is a mapping that associates to each node of G a *conditional probability table of preferences*, that is, a finite set of conditional probabilities of the form $P[E_2|E_1]$ where: (1) E_1 is an event of the form $(A_{i_1} = a_{i_1}) \wedge \ldots \wedge (A_{i_k} = a_{i_k})$ such that $\forall j \in \{1, ..., k\}$, $a_{i_j} \in \mathbf{dom}(A_{i_j})$, and (2) E_2 is an event of the form "$(B = b_1)$ is preferred to $(B = b_2)$", 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 2.4 *BPN.* Let $R(A, B, C, D)$ be the relational schema of Example 2.2. Figure 1(c) illustrates a preference network $\mathbf{PNet}_1$ over R.

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

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

Example 2.5 *Preference Order.* Let us consider the BPN $\mathbf{PNet}_1$ depicted in Figure 1(c). This BPN allows to infer a preference ordering on tuples over $R(A, B, C, D)$. According to this ordering, tuple $u_1 = (a_1, b_1, c_1, d_1)$ is preferred to tuple $u_2 = (a_2, b_2, c_1, d_2)$. In order to conclude that, we execute the following steps: (1) Let $\Delta(u_1, u_2)$ be the set of attributes where the u_1 and u_2 differ. In this example, $\Delta(u_1, u_2) = \{A, B, D\}$; (2) Let $\min(\Delta(u_1, u_2)) \subseteq \Delta$ such that the attributes in $\min(\Delta)$ have no ancestors in Δ (according to graph G underlying the BPN $\mathbf{PNet}_1$). In this example $\min(\Delta(u_1, u_2)) = \{D, B\}$. In order to u_1 be preferred to u_2 it is necessary and sufficient that $u_1[D] > u_2[D]$ and $u_1[B] > u_2[B]$; (3) Compute the following probabilities: p_1 = probability that $u_1 > u_2$ = $P[d_1 > d_2|C = c_1] * P[b_1 > b_2|C = c_1] = 0.6 * 0.6 = 0.36$; p_3 = probability that $u_2 > u_1$ =

$P[d_2 > d_1|C = c_1] * P[b_2 > b_1|C = c_1] = 0.4 * 0.4 = 0.16$; p_2 = probability that u_1 and u_2 are incomparable = $P[d_1 > d_2|C = c_1] * P[b_2 > b_1|C = c_1] + P[d_2 > d_1|C = c_1] * P[b_1 > b_2|C = c_1]$ = 0.6*0.4 + 0.4*0.6 = 0.48. In order to compare u_1 and u_2 we focus only on p_1 and p_3 (ignoring the "degree of incomparability" p_2) and select the higher one. In this example, $p_1 > p_3$ and so, we infer that u_1 is preferred to u_2. If $p_1 = p_3$ we conclude that u_1 and u_2 are incomparable.

Definition 2.6 *Precision and Recall.* Let **PNet** be a BPN over a relational schema R. Let $\mathcal{P}$ be a preference database over R. The *recall* of **PNet** with respect to $\mathcal{P}$ is defined by Recall(**PNet**,$\mathcal{P}$)= $\frac{N}{M}$, where M is the cardinality of $\mathcal{P}$ and N is the amount of pairs of tuples $(t_1, t_2) \in \mathcal{P}$ compatible with the preference ordering inferred by **PNet** on the tuples t_1 and t_2. That is, the recall of **PNet** with respect to $\mathcal{P}$ is the percentage of elements in $\mathcal{P}$ which are correctly ordered by **PNet**. The *precision* of **PNet** is defined by Precision(**PNet**,$\mathcal{P}$)= $\frac{N}{K}$, where K is the set of elements of $\mathcal{P}$ which are comparable by **PNet**. That is, the precision of **PNet** with respect to $\mathcal{P}$ is the percentage of *comparable* elements of $\mathcal{P}$ (according to **PNet**) which are correctly ordered by **PNet**.

The problem of mining contextual preferences in the batch setting is the following: Given a training preference database T_1 over a relational schema R and a testing preference database T_2 over R, find a BPN over R having good precision and recall with respect to T_2.

3. PROBLEM FORMALIZATION IN THE STREAM SETTING

The main differences between the batch and the stream settings concerning the preference mining problem we address in this article may be summarized as follows:

—*Input data*: to each sample bituple (u, v) provided by the user, representing his preference on tuple u over tuple v, is associated a timestamp t standing for the time the user made this choice. So, the input data from which a preference model will be extracted is a *preference stream database* 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.

—*Output*: the preference model to be extracted from the preference stream database is a *temporal BNP*, that is, a sequence of BNPs $< PNet_t : t \in \mathbb{N} >$. 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 BNP at each instant t*: At each instant t we are able to compare tuples u and u by employing the Preference Model $PNet_t$ produced by the mining algorithm at this instant. The preference order between u and v is denoted by $>_t$ and is obtained as illustrated in example 2.5.

—*Precision and Recall instant t*: The quality of the preference model $PNet_t$ produced 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 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 precision and recall of the preference model $PNet_t$ obtained at instant t is evaluated according to the formulae given in definition 2.6 applied to the (static) BNP $PNet_t$ and the non temporal preference database $\mathcal{P}$.

The Figure 2 illustrates the dynamic process of mining and testing the preference models from the preference stream database.

In fact, as we will see in the following section, the mining algorithm is able to produce the preference model by using only a reduced set of *statistic* extracted from the training sets at the instant t it is requested to act. The entire sets T_i of the preference samples stream are not needed in the mining

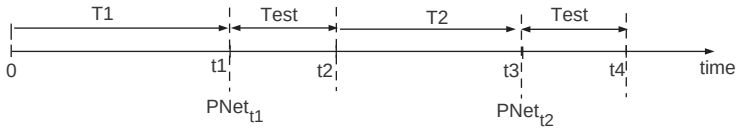


Fig. 2. The dynamics of the mining and testing processes through time

process. The statistics used at instant t_3 in order to infer the preference model at this instant is obtained by incrementing the statistics at instant t_1 with the statistics extracted from the preference samples arrived between t_2 and t_3. We suppose that the preference samples arrived between t_1 and t_2 are used only for testing the preference model $PNet_1$ and not for training the preference model $PNet_2$.

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

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

Output: whenever requested, return a BPN_t over R having good precision and recall, where t is the time instant of request.

REFERENCES

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. Moa: Massive online analysis. *J. Mach. Learn. Res.* vol. 11, pp. 1601–1604, Aug., 2010.

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

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

de Amo, S., 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 - Volume 01*. ICTAI '12. IEEE Computer Society, Washington, DC, USA, pp. 114–121, 2012.

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. ACM, New York, NY, USA, pp. 71–80, 2000.

Gama, J. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 2010.

Holland, S., Ester, M., and KieSSling, W. Preference mining: A novel approach on mining user preferences for personalized applications. Tech. rep., Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, 2003.

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. ACM, New York, NY, USA, pp. 390–398, 2008.

Oza, N. C. and Russell, S. Online bagging and boosting. In *In Artificial Intelligence and Statistics 2001*. Morgan Kaufmann, pp. 105–112, 2001.

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

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

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

Yong, Z., Jun, H., and He, G. *Bio-Inspired Computational Algorithms and Their Applications*. InTech, 2012.