FPSMining: A Fast Algorithm for Mining User Preferences in Data Streams

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 article, we address 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. As main contribution of this article, we formalize the contextual preference mining problem in the stream setting and propose an algorithm for solving this problem. We implemented this algorithm 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*

Keywords: context-awareness, data mining, data streams, 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: the financial market, credit card transaction flow, 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; Gama 2010].

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, in most situations, user preferences are dynamic. For instance, consider an *online news site* that wants to discover the preferences of its users regarding news and make recommendations based on that. Note that due to the dynamic nature of news, it is plausible that user's preferences would evolve rapidly with time. In times of elections, a user can be more interested in *politics* than in *sports*. In times of Olympic Games, the other way round.

The most crucial issues which makes the process of mining in a stream setting much more challenging than in a 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].

This work focus on a particular kind of preferences, the *contextual preferences* ([de Amo et al.

2012]). The preference model is a set of *contextual preference rules*.

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 user 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 to use aggregated knowledge on the preference 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 address.

The main goal of this article is to propose an algorithm for mining contextual preferences from a *preference stream* sample. We also present the results of a set of experiments for this algorithm executed on real data.

2. BACKGROUND ON CONTEXTUAL PREFERENCE MINING IN THE BATCH SETTING

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.

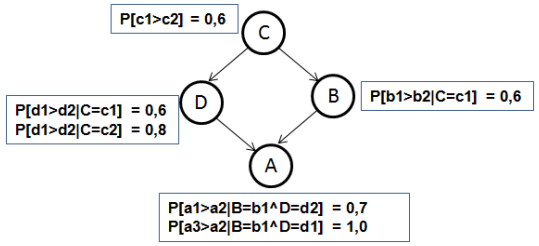
A *preference relation* on a finite set of objects $A = \{a_1, a_2, ..., a_n\}$ is a strict partial order over A, that is a binary relation $R \subseteq A \times A$ satisfying the irreflexivity and transitivity properties. We denote by $a_1 > a_2$ the fact that a_1 is preferred to a_2. A *Preference 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)).

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 *preference model* from a preference database provided by the user. The preference model is specified by a *Bayesian Preference Network* (BPN), specified by (1) a directed acyclic graph G whose nodes are attributes and the edges stands for attribute dependency and (2) a mapping θ 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 to infer a *preference ordering* on tuples. The following example illustrates how this ordering is obtained.

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

Example 2.1 *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 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

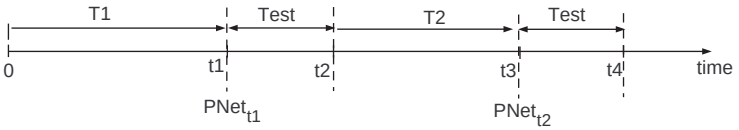


Fig. 2. The dynamics of the mining and testing processes through time

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 *Precision* and *Recall* with respect to a *test* preference database $\mathcal{P}$. The *recall* is defined by Recall(**PNet**,$\mathcal{P}$)= $\frac{N}{M}$, where M is number of bituples in $\mathcal{P}$ and N is the amount 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 *precision* is defined by Precision(**PNet**,$\mathcal{P}$)= $\frac{N}{K}$, where K is the number of elements of $\mathcal{P}$ which are comparable by **PNet**.

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) 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) 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 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 preference stream until the instant t. The preference order between u and v is denoted by $>_t$ and is obtained as illustrated in example 2.1.

—*Precision and Recall at 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 equations given in the previous section applied to the (static) BPN $PNet_t$ and the non temporal preference database $\mathcal{P}$. The precision and recall of the algorithm are measured repeatedly over time. Fig. 2 illustrates this 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 only a reduced set of *statistics* 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 process.

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 having good precision and recall, where t is the time instant of request.

		$c_1 > c_2$	$c_2 > c_1$	$c_4 > c_5$
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}.

4. THE FPSMINING ALGORITHM

In this article we propose an algorithm for mining user contextual preferences in the stream setting: the *FPSMining algorithm*.

In order to save processing time and memory, in this algorithm we do not store the elements of the preference stream processed so far, just collect *sufficient statistics* from it in an online way. 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 $u_1 >_{t_i} u_2$ (u_1 is preferred to u_2 at t_i) 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 Fig. 3(a) collected from the preference stream S until time instant t_9. The table on top of Fig. 3(a) shows the *context counters* regarding the preferences over the attribute C, and the 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 l, 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 l. In this example, $\Delta(u_1, u_2) = \{C\}$, then only the attribute C will have their statistics updated according to l; (2) Increment context counters a_1 and b_6 regarding the preference $c_2 > c_1$ (table top of the Fig. 3(b)). Note that in the temporal bituple l 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. As until now we had no context b_6, it is inserted in the statistics; (3) Increment general counter of preference $c_2 > c_1$ (table down of the Fig. 3(b)).

The main idea of the FPSMining algorithm 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) with respect to a snapshot Q of the sufficient statistics from the preference stream S at the time instant t (computed by Alg. 1) is a real number that estimates how preferences on values for the attribute Y are influenced by values for the attribute X. In Alg. 1, for each pair (y, y') appearing in the *general counters* over Y of the sufficient statistics Q (line 1), $T^{time}_{yy'}$ denotes the set 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 $(u_1[Y] = y' \wedge u_2[Y] = y)$; support$((y, y'),Q) = \frac{|T^{time}_{yy'}|}{n}$, where n is the number of elements of the preference stream processed until the time instant $time$ and $|T^{time}_{yy'}|$ is the sum of the values of two general counters of Y from Q. We say that a pair (y, y') in the general counters over Y is *comparable* (line 1) if support$((y, y'),Q) \geq \alpha_1$, for a given threshold α_1, $0 \leq \alpha_1 \leq 1$. For each $x \in \mathbf{dom}(X)$ (line 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 $u_1[X] = u_2[X] = x$; support$(S^{time}_{x|(y,y')},Q)$ $= \frac{|S^{time}_{x|(y,y')}|}{|\bigcup_{x' \in \mathbf{dom}(X)} S^{time}_{x'|(y,y')}|}$, where $|S^{time}_{x|(y,y')}|$ is obtained by the sum of the values of two context counters

of Y from Q: fixing the line for $X = x$, and sum the column values $y > y'$ and $y' > y$. We say that x is a *cause for* (y, y') *being comparable* (line 2) if support($S^{time}_{x|(y,y')}$,Q) $\geq \alpha_2$, for a threshold α_2, $0 \leq \alpha_2 \leq 1$.

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*.

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
$$N = \frac{|\{(u_1, u_2, t) \in S^{time}_{x|(y,y')} : u_1 >_t u_2 \wedge (u_1[Y] = y \wedge u_2[Y] = y')\}|}{|S^{time}_{x|(y,y')}|}$$
4 Let $f_2(T^{time}_{yy'}) = \max\{f_1(S^{time}_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$
5 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$\}$
6 **return** $f_3((X, Y), Q)$

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

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, where t is the time instant of request.

1 Take a snapshot Q of the sufficient statistics from S at the time instant t.
2 **for** *each pair of attributes* (A_i, A_j), *with* $1 \leq i, j \leq n, i \neq j$ **do**
3 Use Alg. 1 for calculate the degree of dependence between the pair (A_i, A_j) according to Q
4 Let Ω be the resulting set of these calculations, with elements of type (A_i, A_j, dd), where dd is the degree of dependency between the pair (A_i, A_j)
5 Eliminate from Ω all elements whose $dd < 0.5$ (indicates a weak dependence between a pair of attributes)
6 Order the elements (A_i, A_j, dd) in Ω in decreasing order according to their dd
7 Start the graph G_t of BPN_t with a node for each attribute of R
8 **for** *each element* $(A_i, A_j, dd) \in$ *ordered set* Ω **do**
9 Insert the edge (A_i, A_j) in the graph G_t only if the insertion does not form cycles in G_t
10 Once the graph G_t of BPN_t was created, estimate the conditional probabilities tables θ_t of BPN_t, using the Maximum Likelihood Principle (see [de Amo et al. 2012] for details) over Q.
11 **return** BPN_t

5. EXPERIMENTAL RESULTS

The data used in the experiments contains preferences related to films collected by GroupLens Research [1] from the MovieLens web site [2]. We simulated a preference stream from these data, as follows: we stipulated a time interval λ (measured in days or hours), and each tuple in the dataset was compared to all others in a radius λ relative to its timestamp, thus generating temporal bituples. The resulting preference stream has five attributes (director, genre, language, star and year), and its elements correspond to preferences about movies determined by a particular user.

In order to evaluate our algorithm, we adapted the sampling technique proposed by [Bifet and Kirkby 2009], based on holdout for data stream, for the preference scenario. The sampling technique used takes input from three parameters: 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.

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

In the tests, we vary all parameters of the algorithm. Fig. 4(a), (b), (c), (d) show the results for varying the parameters *user* (each *user* has a different stream), λ (each λ produces a different stream), n_{train} and n_{test}, α_1 and α_2, respectively. In each figure, except for the parameter that was varied, the default values used were: $user$=$U1$, λ=1 day, n_{train}=1,000 and n_{test}=100, α_1=0.2 and α_2=0.1. The n_{eval} values were calculated according to the total size of the stream.

Fig. 4(a) shows that the more movies the user evaluated (tuples), the better the recall and precision of our algorithm with respect to their preferences. Fig. 4(b) shows that until λ=1 day the quality values increase as the number of elements in the preference stream increases. Fig. 4(c),(d) show that our algorithm was stable to the variation of the parameters n_{train}, n_{test}, α_1 and α_2 in these data.

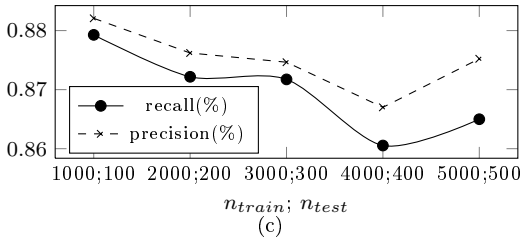
In the tests carried out the recall and precision values were satisfactory, showing that in most cases our algorithm compared correctly temporal bituples submitted to it, thereby proving to be efficient for mining user preferences in a stream setting. The execution time for the generation of the model was 16 ms and the time to complete one cycle of the holdout was 31 ms.

User	Tuples	Bituples	Recall	Precision
$U1$	7359	1580710	0.871	0.874
$U2$	6047	1658450	0.794	0.801
$U3$	4483	563419	0.778	0.784
$U4$	4449	198618	0.769	0.785

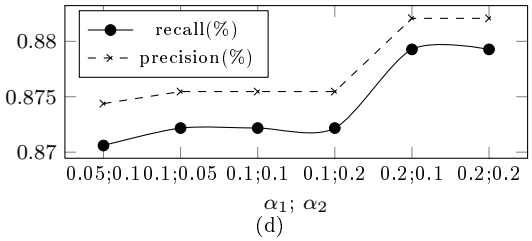
(a)

Stream	λ	Bituples	Recall	Precision
$S1$	1h	227100	0.800	0.814
$S2$	12h	645319	0.846	0.853
$S3$	1d	1048228	0.879	0.882
$S4$	7d	1580710	0.871	0.874
$S5$	15d	1759753	0.875	0.876

(b)



(c)



(d)

Fig. 4. Experimental Test Set

6. CONCLUSION AND FURTHER WORK

In this article we proposed the FPSMining algorithm to solve the problem of mining user contextual preferences in the stream setting. We show that it is fast and produces satisfactory results on real data. As immediate future work, we intent to implement two different learning technique based on FPSMining (an incremental and an ensemble algorithms) as well as a synthetic preference stream generator for tests with a huge amount of data.

REFERENCES

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

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 *IEEE 24th International Conference on Tools with Artificial Intelligence*. 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*, 2000.

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

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*, 2007.

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

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

Shivaswamy, P. K. and Joachims, T. Online learning with preference feedback. *CoRR* vol. abs/1111.0712, 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 AAMAS workshop and TADA/AMEC conference*. pp. 118–131, 2007.