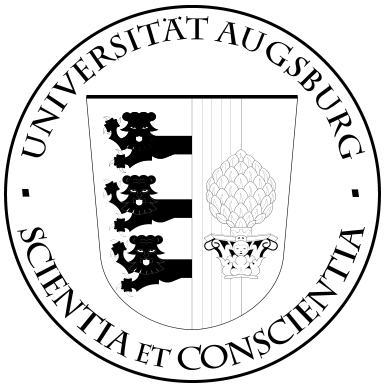
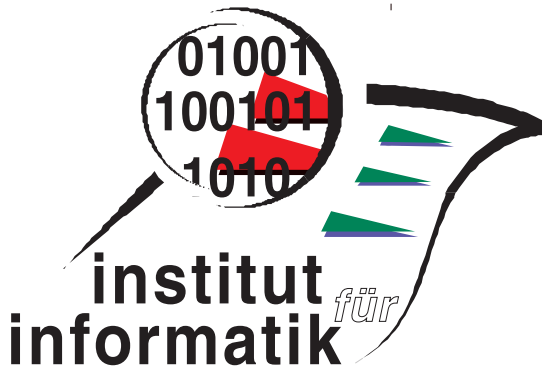
Universität Augsburg




# Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications

**S. Holland, M. Ester, W. Kießling**

Report 2003-5 Mai 2003




Institut für Informatik
D-86135 Augsburg

# Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications

Stefan Holland[1], Martin Ester[2], Werner Kießling[1]

[1] Institute of Computer Science, University of Augsburg, D-86159 Augsburg, Germany
{holland, kiessling}@informatik.uni-augsburg.de
[2] School of Computer Science, Simon Fraser University, Burnaby BC, Canada V5A 1S6
ester@cs.sfu.ca

**Abstract.** Advanced personalized e-applications require comprehensive knowledge about their user's likes and dislikes in order to provide individual product recommendations, personal customer advice and custom-tailored product offers. In our approach we model such preferences as strict partial orders with "A is better than B" semantics, which has been proven to be very suitable in various e-applications. In this paper we present novel Preference Mining techniques for detecting strict partial order preferences in user log data. The main advantage of our approach is the semantic expressiveness of the Preference Mining results. Experimental evaluations prove the effectiveness and efficiency of our algorithms. Since the Preference Mining implementation uses sophisticated SQL statements to execute all data-intensive operations on database layer, our algorithms scale well even for large log data sets. With our approach personalized e-applications can gain valuable knowledge about their customers' preferences, which is essential for a qualified customer service.

## 1 Introduction

The enormous growth of web content and web-based applications leads to an unsatisfactory behavior for users: search engines retrieve a huge number of results and they are left on their own to find interesting web sites or preferred products. Such a behavior leads not only to frustrated users but also to a reduction of turnover in commercial businesses because customers who are willing to buy cannot do it since they do not find the right product even if it is available. In recent years, several techniques have been developed to build user adaptive web sites and personalized web applications. For instance, E-commerce applications use link personalization to recommend items based on the customer's buying history or some categorization of customers based on ratings and opinions [13]. Another technique is content personalization: web pages present different information to different users based on their individual needs. Thereby, the user can indicate his preferences explicitly using the predefined tools of the underlying portal or the preferences may be inferred automatically from his profile.

State-of-the-art personalization techniques suffer from some drawbacks. Manually customizing web sites is not very feasible to the customer since it is a very time-consuming task to select relevant content from the huge repertoire provided by the

web portal. Personalizing products or web content automatically is a more promising approach. However, the current approaches of automatic personalization lack of preference models with limited expressiveness. State-of-the-art techniques either use scores to describe preferences [10] or just distinguish between liked and disliked values [2]. Thus, complex *"I like A more than B"*-relationships as well as *preferences for numeric attributes* cannot be expressed in a natural way. Furthermore, these approaches are not able to handle *dependencies among preferences*. For example, two preferences can be of equally importance to a customer or one preference can be preferred to another one.

A very expressive and mathematically well-founded framework for preferences has recently been introduced [7]. Customer preferences are modeled as strict partial orders with "A is better than B" semantics, where negative, numeric and complex preferences form special cases. This approach has been proven to be very suitable for modeling user preferences of almost any complexity. Standard query languages like SQL and XPATH were extended by such preferences [9] in order to deal carefully with user wishes. In this paper, we present algorithms for automatically mining such strict partial order preferences from user log data. Basic categorical and numerical preferences are discovered based on the frequencies of the different attribute values in the user log. These basic preferences are then combined to detect complex preferences.

The rest of the paper is organized as follows: After a survey of related work in section 2 we describe the underlying preference model and Preference Mining requirements in section 3. In section 4, we present algorithms for mining categorical, numerical and complex preferences. Section 5 summarizes the results of an extensive experimental evaluation of the accuracy and efficiency of the proposed algorithms. We conclude our paper with a summary and outlook in section 6.

## 2 Related Work

Several research groups have studied the usage of log data analysis for personalized applications. In particular, web log mining is a commonly used approach of analyzing web log data with data mining techniques for the discovery of sequential patterns, association rules or user clusters. Such mining techniques have been applied to provide personalized link recommendations to web users [12]. Thereby the user profile of the current user is matched against one or more previously discovered usage profiles.

Beeferman and Berger analyzed query log data of search engines [1]. They developed clustering algorithms in order to find groups of URLs that match various keywords given by the user. This approach is not only helpful for delivering better search results but also for the construction of web categories and the generation of ontologies. In [5], Joachims analyzed clickthrough data to improve the results of search engines. He uses the search results that are chosen by the user as additional information. He argues that selected items are better in the opinion of the user and applies this knowledge to find better rankings for future search results.

Our Preference Mining techniques can work either on web logs or query logs, whereby the latter not only occurs in search engines but also in state-of-the-art e-

commerce applications. The main advantage of our approach is the semantic expressiveness of the Preference Mining results. Our algorithms compute no scores to distinguish between liked and disliked values but detect intuitive preferences like positive or negative preferences, numerical preferences or even combinations of such preferences. Personalized web applications such as described in [13] can gain significant improvements by using such detailed knowledge about user preferences.

## 3 User Preferences in Log Data

In this section we revisit those aspects of the preference model of [7] that are relevant for the scope of this paper. We also define requirements on the user log data for mining such preferences.

### 3.1 Preferences as Strict Partial Orders

A preference P is defind as a strict partial order $P = (A, <_P)$, where $A = \{A_1, \ldots, A_k\}$ denotes a set of attributes with corresponding domains $dom(A_i)$. The domain of A is defined as Cartesian product of the $dom(A_i)$, $<_P \subseteq dom(A) \times dom(A)$ and $x <_P y$ is interpreted as "y is better than x". A set of intuitive preference constructors for base and complex preferences is defined.

The constructors for base preferences on categorical domains are POS(A, POS-set), NEG(A, NEG-set), POS/NEG(A, POS-set; NEG-set), POS/POS(A, POS1-set; POS2-set) and EXP(A, E-graph). The POS-set $\subseteq$ dom(A) of a POS preference defines a set of values that are better than all other values of dom(A). Analogously, the NEG-set of a NEG preference describes disliked values. The POS/NEG preference is a combination of the previous preferences and in a POS/POS preference optimal values (POS1-set) and alternative values (POS2-set) can be specified. In E-graph of an EXPLICIT preference a user can specify any better-than relationships.

The preference constructors for numerical domains include AROUND(A, z), BETWEEN(A, [low, up]), LOWEST(A) and HIGHEST(A). In an AROUND preference the desired value is z, but if this it not available values with nearest distance apart from z are best alternatives. For a BETWEEN preference the values within [low, up] are optimal. For LOWEST (HIGHEST) preferences lower (higher) values are better.

Preferences can inductively be combined with complex preference constructors. A Pareto preference $P = P_1 \otimes P_2$ treats the underlying preferences as equally important and a Prioritized preference $P = P_1 \,\&\, P_2$ treats $P_1$ as more important than $P_2$. For instance, P = POS(author, {Douglas Adams, Edgar Wallace}) & NEG(binder, {paperback}) denotes a POS preference for the authors Douglas Adams and Edgar Wallace, and a NEG preference for paperbacks, whereby the latter preference is less important.

This definition of preference constructors has been proven to be appropriate to describe complex user wishes. Preference engineering examples are shown in [7]. Our Preference Mining developments should be consistent to this preference model. Therefore, not only all base and complex preferences should be detectable by the

Preference Miner but also preference properties like preference hierarchies or preference algebra laws (see [7] for details) should be valid for the detected preferences.

### 3.2 Requirements on User Log Data in Web Applications

Data mining benefits from the availability of a huge amount of data since having many records ensures the statistical significance of patterns [11]. Log data of user transactions can have several sources like web server log-files or transaction logging on an application server.

*Web server logs* are generated by the web server when a user is visiting a web site. Such files can comply with standardized formats like the Common Logfile Format.[1] The log data includes the IP of the client host, the current timestamp and the URL (uniform resource locator) he is visiting. Valuable information about a user's wishes is stored in the URL, since it contains not only the address but also requested keywords or preferred product properties the user inserted into a web form. For example, if the user requests the book "The Raven" in the e-shop Barnes & Noble[2] the logged URL is *http://search.barnesandnoble.com/booksearch/re-sults.asp?WRD=The+Raven*. But web server logs also have some disadvantages, especially for e-commerce applications [11]. Events like "add to cart" or "change item" are not available in web logs. Furthermore, a user can deactivate cookies in his browser, so no session information or user identification is available. Preference Mining on web server logs requires some data preprocessing. User input like "The Raven" in the above example has to be extracted from the logged URL and has to be stored in a relational database since our Preference Mining algorithms work on database relations. Furthermore, user identification is required to detect preferences for each customer separately.

*Application server logs* can handle user transactions much better [11]. User and session identification can be accomplished with a login and logout mechanism. Another advantage is the capability to detect business events like "add to cart" or "buy items". For example, an e-commerce application server can record queries, search results, selected items and bought products for each customer separately. Furthermore, application server log data can be stored in databases and therefore huge amount of log data can be managed by using database technology. The Preference Mining algorithms can work directly on these log relations without any data preprocessing. For instance, analyzing the properties of bought products can lead to preferences about liked and disliked features, price preferences and dependencies between such preferences.

While browsing or shopping in an online environment, a customer has typically several different types of input fields for interacting with the underlying system. Text fields allow the input of keywords and choices allow the selection of static or dynamic predefined values of an attribute. To describe these different situations we define the closed world assumption and the difference between static and dynamic domains.

---

[1] http://www.w3.org/Daemon/User/Config/Logging.html

[2] http://www.barnesandnoble.com

**Definition 1 (Closed world assumption (CWA))**
The assumption that a customer knows all possible values of an attribute is called Closed World Assumption or CWA. If this assumption doesn't hold we abbreviate it with ¬CWA.

**Definition 2 (Static and dynamic domains)**
If a domain of attribute values is constant over time, we call it a static domain otherwise we call it a dynamic domain.

The CWA is required for the detection of negative preferences since only if the user knows all possible values we can assume dislike for values he never selected. Otherwise (¬CWA), we can't decide whether he doesn't know or doesn't like such values. For instance, in a book shop the customer knows all possible values for binder (paperback or hardcover) but doesn't know all available authors. After submitting a search query, a customer gets a set of results and chooses one or more of them as his preferred products. Such search results define dynamic domains and can lead to valuable clickthrough data, which can be used to get information about explicit user preferences since the clicked items of the query result are preferred by the user [5].

## 4 Preference Mining Algorithms

In this section we present algorithms for mining the strict partial order preferences introduced in section 3.1. Our methods work on log relations as described in section 3.2 and use appropriate data mining and statistical methodologies in order to detect the right preference and correct additional information like POS-sets. To detect basic preferences, we use the frequencies of the different values in the log relation.

**Definition 3 (Frequency of a value)**
Let A be an attribute of a log relation R and $x \in dom(A)$. The number of entries of x in R(A) is called frequency of x or $freq_A(x)$. If dom(A) is numerical, $freq_A([x_1, x_2])$ denotes the number of entries of all values between $x_1$ and $x_2$ ($x_1 \leq x_2$).

We have introduced the concept of user-defined preferences $P = (A, <_P)$. The actual user preferences shall be predicted from the implicit preferences hidden in the user log data. To that purpose, we introduce the concept of data-driven preferences denoted by $P_D = (A, <_{PD})$.

**Definition 4 (Data-driven preference)**
- For categorical domains dom(A) a data-driven preference $P_D = (A, <_{PD})$ is defined as: $x <_{PD} y$ iff $freq_A(x) < freq_A(y)$.
- For numerical domains dom(A) a data-driven preference $P_D = (A, <_{PD})$ is defined as: $x <_{PD} y$ iff $\exists \varepsilon > 0$: $freq_A([x-\varepsilon, x+\varepsilon]) < freq_A([y-\varepsilon, y+\varepsilon])$.

Depending on the design of the log data, values can be products (e.g. search results) or just product properties like color or price. If the frequency of a value x is zero, a customer has never selected the according value. If CWA holds, $freq_A(x) = 0$ means that a customer doesn't like the property x because he never selected it although he knows it.

Otherwise, if CWA doesn't hold, the customer may either not like the property x or may never have heard of it. The relation $freq_A(x) < freq_A(y)$ shows that the corresponding customer has selected y more often than x. In this sense the relation $x <_{PD} y$ denotes a preference.

Numeric domains need a slightly different approach to data-driven preferences. For instance, an attribute A may have the real numbers as domain ($dom(A) = \mathbb{R}$) and we want to test, if a user has a data-driven LOWEST(A) preference, i.e. lower values are better and should occur with higher frequencies. Since $\mathbb{R}$ consists of an infinity number of different values, the log relation only contains some of them and typically each value occurs only a few times in the log relation. Therefore, we use frequencies of intervals. E.g. for a data-driven LOWEST preference the relation $freq_A([x-\varepsilon, x+\varepsilon]) < freq_A([y-\varepsilon, y+\varepsilon])$ for $y < x$ must hold for some $\varepsilon$.

**Proposition 1:**
A data-driven preference defines a strict partial order.

**Proof:** see appendix.

### 4.1 Mining Categorical Preferences

Based on $P_D = (A, <_{PD})$ we can define data-driven preferences for categorical domains.

**Definition 5 (Data-driven preferences for categorical data)**
Let A be a categorical attribute of a log relation R and POS-set, NEG-set, POS1-set, POS2-set, $E \subseteq dom(A)$.

- There is a data-driven POS preference, iff $\forall x \in$ POS-set, $\forall y \notin$ POS-set: $y <_{PD} x$.
- There is a data-driven NEG preference, iff $\forall x \in$ NEG-set, $\forall y \notin$ NEG-set: $x <_{PD} y$.
- There is a data-driven POS/POS preference, iff $\forall x \in$ POS1-set, $\forall y \in$ POS2-set, $\forall z \notin$ (POS1-set $\cup$ POS2-set): $y <_{PD} x$ and $z <_{PD} y$.
- There is a data-driven POS/NEG preference, iff $\forall x \in$ POS-set, $\forall y \in$ NEG-set, $\forall z \notin$ (POS-set $\cup$ NEG-set): $z <_{PD} x$ and $y <_{PD} z$.
- Let $<_E$ be a strict partial order on E. A data-driven EXPLICIT preference holds, iff
  - $\forall x, y \in E$ with $x <_E y$: $x <_{PD} y$,
  - $\forall u \in E$, $\forall v \notin E$: $v <_{PD} u$.

For a data-driven POS preference the values in the POS-set must occur more often than the other values and in a data-driven NEG preference the other values must occur more often than the values in the NEG-set. POS/POS and POS/NEG run analogously. A data-driven EXPLICIT preference with underlying E-graph exists, if a value y occurs more often than any successor x in E-graph. Values outside the E-graph occur with lowest frequencies.

The main task for an algorithm for mining categorical preferences is the detection of proper POS-sets, NEG-sets, etc. Consider the following example of frequencies for an attribute author (CWA doesn't hold, the domain is static):

Table 1. Example of frequencies for an attribute “author”

| Douglas Adams | Edgar Wallace | Natalie Angier | Agatha Christie | John Grisham |
|---|---|---|---|---|
| 50 | 49 | 2 | 3 | 2 |

The set {Douglas Adams} is a correct POS-set for a data-driven POS preference. But intuitively, the set {Douglas Adams, Edgar Wallace} denotes are more reasonable POS-set since these two values occurred much more frequently than Natalie Angier, Agatha Christie and John Grisham. The following algorithm for mining categorical preferences uses cluster techniques in order to detect such proper sets.

**Algorithm 1: Miner for categorical preferences in static domains**
INPUT: log relation R, attribute A, dom(A)
(1) Compute for each value $x_i$ the frequency in the log relation $freq_A(x_i)$.
(2) Compute a clustering of the $x_i$ with $freq_A(x_i) \geq 1$ by using a clustering technique.
(3) Depending on the clustering results we have the following possibilities:
(a) There is only one cluster $C_1$ and CWA holds. Here we have a NEG(A, $\{x \in dom(A) | freq_A(x) = 0\}$) preference.
(b) There are two clusters $C_1$ and $C_2$, where $\forall c_1 \in C_1, \forall c_2 \in C_2$: $freq_A(c_2) < freq_A(c_1)$.
(b1) If ¬CWA, we have a POS(A, $C_1$) preference.
(b2) If CWA, there is a POS/NEG(A, $C_1$; $\{x \in dom(A) | freq_A(x) = 0\}$) preference.
(c) There are three clusters $C_1$, $C_2$ and $C_3$, where $\forall c_1 \in C_1$, $\forall c_2 \in C_2, \forall c_3 \in C_3$: $freq_A(c_3) < freq_A(c_2) < freq_A(c_1)$. Here we have a POS/POS(A, $C_1$; $C_2$) preference.
(d) There are more than three clusters $C_1$, …, $C_n$, where $\forall c_1 \in C_1, \forall c_2 \in C_2$, …, $\forall c_n \in C_n$: $freq_A(c_n) < \ldots < freq_A(c_2) < freq_A(c_1)$. Here we have an EXPLICIT preference EXP(A, $<_E$) with $c_n <_E \ldots <_E c_2 <_E c_1, \forall c_1 \in C_1, \forall c_2 \in C_2, \ldots, \forall c_n \in C_n$.
(e) In all other situations there is no data-driven preference.
OUTPUT: the detected preference or that no preference was found

By using a state-of-the-art clustering technique like k-means [4] – and silhouettes for getting the optimal number of clusters, see [14] – this algorithm detects two clusters $C_1$ = {Douglas Adams, Edgar Wallace} and $C_2$ = {Natalie Angier, Agatha Christie, John Grisham} leading to a POS(author, {Douglas Adams, Edgar Wallace}) preference in the above example. Data-driven NEG preferences can only be detected, if the user knows all possible values (CWA).

**Proposition 2:**
Preferences detected with Algorithm 1 have the following properties:

- POS preference: values within the POS-set occur more often in the log-relation than values outside the POS-set.
- NEG preference: Values within the NEG-set occur with lower frequencies than other values.
- POS/POS preference: Values within POS1-set occur more often than values within POS2-set and the latter values occur more frequent in the log relation than other values.

- POS/NEG preference: Values within POS-set occur more often than other values. The values within NEG-set occur with lowest frequencies.
- EXPLICIT preference: For each value within E-graph any successor occurs with lower frequency.

**Proof:** see appendix.

**Corollary 1:**
Algorithm 1 detects data-driven preferences.

**Corollary 2:**
Algorithm 1 detects strict partial order preferences.

**Proposition 3:**
Let n be the number of tuples in the log relation R and $k = |\pi_A(R)|$ the number of different values of an attribute A in R. By using hierarchical clustering algorithm 1 has the complexity $O(n + k^2)$.

**Proof:** see appendix.

In dynamic domains the CWA holds, because the user must know the varying values for his decisions. By selecting or clicking on one or more of the available values, the user provides preference knowledge since he prefers the selected items to the other available values. The following algorithm for mining such EXPLICIT preferences requires an advanced structure of the log relation. We assume we have the information (query_id, value, selected) within the log relation, whereby "value" contains a value available for the user, "selected" ($\in \{0,1\}$) denotes whether the according value was selected or not and "query_id" specifies which values belong to one search query. The ability of a low-cost construction of such log data has been shown in [5].

**Algorithm 2 (Miner for EXPLICIT preferences in dynamic domains)**
INPUT: log data in the format (query_id, value, selected)
(1) Compute the k occurring values ($x_1$, …, $x_k$) in the log relation. Initialize the better-than graph with E-graph = Ø.
(2) FOR(i = 1, …, k) and FOR(j = i + 1, …, k) DO:
  (a) Consider the query ids, whose according values contain $x_i$ and $x_j$.
  (b) Compute the number s of query ids, where $x_i$ was selected and $x_j$ wasn't.
  (c) Compute the number t of query ids, where $x_j$ was selected and $x_i$ wasn't.
  (d1) If s > t and there is no path from $x_j$ to $x_i$ in E-graph, set E-graph = E-graph $\cup$ ($x_j$, $x_i$). Otherwise, if a path from $x_j$ to $x_i$ exists, remove it.
  (d2) If s < t and there is no path from $x_i$ to $x_j$ in E-graph, set E-graph = E-graph $\cup$ ($x_i$, $x_j$). Otherwise, if a path from $x_i$ to $x_j$ exists, remove it.
  (d3) If s = t remove within E-graph all direct and transitive connections from $x_i$ to $x_j$ and vice versa.
OUTPUT: the detected EXPLICIT preference based on E-graph as better-than graph.

For two values $x_i$ and $x_j$ the algorithm computes the query ids that have both values in the result set. Now $x_i$ is better than $x_j$, if the user selected it more often. In step 4 cycles are removed. Therefore we check if there is a path from $x_j$ to $x_i$ in E-graph before inserting ($x_j$, $x_i$) and vice versa. Cycles can occur, if the browsing or shopping behav-

ior of the user has inconsistencies like blue $<_P$ red $<_P$ green $<_P$ blue. In such situations the preferences of the customer are not clear and therefore we leave out such relations. If s = t, the user is indifferent between $x_i$ and $x_j$ and therefore existing preference relations between $x_i$ and $x_j$ have to be removed.

**Proposition 4:**
EXPLICIT-preferences detected with algorithm 2 are strict partial orders.

**Proof:** see appendix.

**Proposition 5:**
Let n be the number of tuples in the log relation R and $k = |\pi_A(R)|$ the number of different values of an attribute A of R. Then the complexity of algorithm 2 is $O(n + k^2(n + k^2))$.

**Proof:** see appendix.

### 4.2 Mining Numerical Preferences

The distribution of numerical log data defines a statistical density function φ(x). Properties of this density function provide information about data-driven preferences. For instance, if φ(x) has an unique maximum at z and the gradient is positive for x < z and negative for x > z, there is an AROUND preference with around value z.

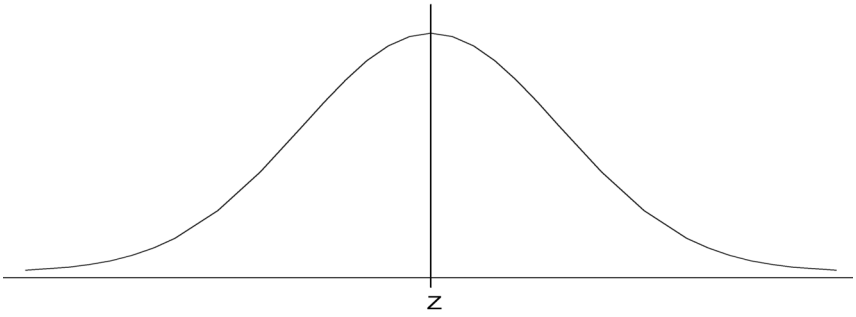



Fig. 1. Density function for a data-driven AROUND preference

**Definition 6 (Data-driven preferences for numerical data)**
Let A be a numerical Attribute of a log-relation R with density function φ(x).

- There is a data-driven LOWEST preference, iff φ(x) is monotonic decreasing.
- There is a data-driven HIGHEST preference, iff φ(x) is monotonic increasing.
- There is a data-driven AROUND preference with around value z, iff φ(x) is monotonic increasing for x < z and monotonic decreasing for x > z.
- There is a data-driven BETWEEN preference with [low, up]-interval, iff
  - φ(x) is monotonic increasing for x < low,
  - φ(x) is monotonic decreasing for x > up,
  - $\forall x \in [low, up], \forall y \notin [low, up]: \varphi(x) > \varphi(y)$.

- There is a data-driven SCORE preference with score function f, iff $\varphi(f(x))$ is monotonic increasing.

We show that these definitions are consistent to the data-driven preferences of def. 4.

**Proposition 6:**
The preferences of definition 6 are data-driven preferences, i.e. better values occur with higher frequencies.

**Proof:** see appendix.

For a data-driven LOWEST preference lower values must occur more often, i.e. $\forall x > y$: $freq_A([x-\varepsilon, x+\varepsilon]) < freq_A([y-\varepsilon, y+\varepsilon])$. This can be considered as $\forall x > y$: $P_A([x-\varepsilon, x+\varepsilon]) < P_A([y-\varepsilon, y+\varepsilon])$, whereas P denotes the probability. The latter formula holds, if the underlying density function is monotonic decreasing. Analogously, a monotonic increasing density functions denotes that higher values are better leading to a data-driven HIGHEST preference. For a data-driven AROUND preference, the density function must be increasing to the around-value and increasing for values greater than the around-value. The data-driven BETWEEN preference demands an increasing density function to the low-value, a decreasing density function beyond the up value and higher density values within [low, up] than outside since values in the [low, up]-interval must be better than the values outside. A data-driven SCORE preference occurs in an attribute A, if the density function of the score values f(x) is monotonic increasing because higher scores must be better.

Mining numerical preferences requires knowledge about the density function $\varphi(x)$ of the underlying data. Typically, such density functions are unknown and have to be estimated using the underlying numerical log data.

**Algorithm 3 (Miner for numerical preferences)**
INPUT: log-relation R, attribute A with numerical domain
(1) Compute a density estimation $\varphi'(x)$ of the considered numerical attribute A.
(2) Depending on $\varphi'(x)$ there are the following possibilities:
(a) If $\varphi'(x)$ is monotonic decreasing there is a LOWEST(A) preference.
(b) If $\varphi'(x)$ is monotonic increasing there is a HIGHEST(A) preference.
(c) If there is a value z whereby $\varphi'(x)$ is monotonic increasing for $x < z$ and $\varphi'(x)$ is monotonic decreasing for $x > z$, there is an AROUND(A, z) preference with around-value z.
(d) If there is a value low, whereby $\varphi'(x)$ is monotonic increasing for $x <$ low and there is a value up, whereas $\varphi'(x)$ is monotonic decreasing for $x >$ up and, additionally, $\varphi'(x) > \varphi'(y)$ holds for all $x \in$ [low, up] and $y \notin$ [low, up], then we have a BETWEEN(A, [low, up]) preference with lower boundary low and upper boundary up.
(e) In all other possibilities there is no numerical preference.
OUTPUT: the detected preference or that no preference was found

SCORE preferences with given score functions f can also be detected. Thereby the values f(x) are considered. If the density estimation $\varphi'(f(x))$ is monotonic increasing, higher values are better leading to a data-driven SCORE preference.

**Proposition 7:**
If the real density function is known, the preferences detected with algorithm 3 have the following properties:

- LOWEST preference: greater values occur with lower frequencies.
- HIGHEST preference: greater values occur with higher frequencies.
- AROUND preference: values with greater distance apart from the around-value occur with lower frequencies.
- BETWEEN preference: values within the [low, up] interval occur with highest frequencies. Additionally, values with greater distance apart from the interval borders occur with lower frequencies.

**Proof:** see appendix.

**Proposition 8:**
Algorithm 3 creates strict partial order preferences.

**Proof:** see appendix.

**Proposition 9:**
By using histograms as density estimation and Scott's rule for the bin-width computation [15], algorithm 3 has the complexity $O(kn + k^3)$, whereby n denotes the number of tuples in the log relation R and k is the number of bins of the histogram.

**Proof:** see appendix.

### 4.3 Mining Complex Preferences

**Definition 7 (Data-driven Prioritized preference)**
Let $P_D = (A, <_{PD})$ and $Q_D = (B, <_{QD})$ be two data-driven preferences and $x = (x_1, x_2)$, $y = (y_1, y_2) \in dom(A) \times dom(B)$. A data-driven Prioritized preference $P_D \,\&\, Q_D = (\{A, B\}, <_{PQ\text{-}D})$ is defined as: $x <_{PQ\text{-}D} y$ iff $x_1 <_{PD} y_1 \vee (x_1 = y_1 \wedge x_2 <_{QD} y_2)$.

In order to detect such complex data-driven preferences we need the definition of associate values.

**Definition 8 (Associate Values)**
Consider a log relation R(A, B, …). For $a \in \pi_A(R)$ the associate values in B are defined as $asv_{A,B}(a) = \pi^*_B(\sigma_{A=a}(R))$.

Thereby $\pi^*$ denotes the relational projection without removing duplicates.

**Algorithm 4 (Miner for Prioritized preferences)**
INPUT: log relation R(A, B, …) and a data-driven preference $P_D$ on A
(1) Compute the set M of maximal values of $P_D$ and for all $a_i \in M$ the set of associate values $asv_{A,B}(a_i)$.
(2) If there is the same preference $Q_D$ in all sets $asv_{A,B}(a_i)$ and $P_D$ does not occur in the associate values of the maxima of $Q_D$, there is a Prioritized preference $P = P_D \,\&\, Q_D$.
(3) Otherwise there is no Prioritized preference.
OUTPUT: the detected Prioritized preference or that no preference was found

A data-driven Prioritized preference $P = P_D \& Q_D$ exists, if, firstly, there is a data-driven preference $P_D$, and, secondly, in those tuples, which have equal values in A, there is a data-driven preference $Q_D$ in B. Thereby, we consider only the maximal values of P since users often don't care about a second-level preference, if the Prioritized preference isn't fulfilled optimal. If $P_D$ also occurs in the maximal values of $Q_D$, a Pareto preference has been found. Therefore, we have to eliminate this situation here.

In our previous example the preference $P_D$ = POS(author, {Douglas Adams, Edgar Wallace}) was detected. If above algorithm detects $Q_D$ = NEG(binder, {paperback}) (dom(binder) = {hardcover, paperback}) in the associate values of Douglas Adams and Edgar Wallace and, furthermore, the $P_D$ is not detected within the hardcover books, a Prioritized preference $P = P_D \& Q_D$ is found.

**Definition 9 (Data-driven Pareto preference)** Let $P_D = (A, <_{PD})$ and $Q_D = (B, <_{QD})$ be two data-driven preferences and $x = (x_1, x_2)$, $y = (y_1, y_2) \in dom(A) \times dom(B)$. A data-driven Pareto preference $P_D \otimes Q_D = (\{A, B\}, <_{PQ\text{-}D})$ is defined as: $x <_{PQ\text{-}D} y$ iff $(x_1 <_{PD} y_1 \wedge (x_2 <_{QD} y_2 \vee x_2 = y_2)) \vee (x_2 <_{QD} y_2 \wedge (x_1 <_{PD} y_1 \vee x_1 = y_1))$.

For a data-driven Pareto preference $P = P_D \otimes Q_D$ the tuple y is better than x, if $y_1$ is better than $x_1$ and $y_2$ is at least equally good to $x_2$ or vice versa.

**Algorithm 5 (Miner for Pareto preferences)**
INPUT: log relation R(A,B, …) and data-driven preferences $P_D$ on A and $Q_D$ on B.
(1) Compute the set $M_P$ of maximal values of $P_D$ and for all $a_i \in M_P$ the set of associate values $asv_{A, B}(a_i)$.
(2) Compute the set $M_D$ of maximal values of $Q_D$ and for all $a_j \in M_D$ the set of associate values $asv_{B, A}(a_j)$.
(3) If $Q_D$ can be detected in every set $asv_{A, B}(a_i)$ and, vice versa, $P_D$ can be detected in every set $asv_{B, A}(a_j)$, there is a Pareto preference $P = P_D \otimes Q_D$.
(4) Otherwise there is no Pareto preference.
OUTPUT: the detected Pareto preference or that no preference was found

A data-driven Pareto preference $P = P_D \otimes Q_D$ is detected, if $Q_D$ can be found in the associate values of $P_D$ and vice versa. This complies with the non-discrimination theorem stated in [7].

**Proposition 10:**
Algorithm 4 and 5 create strict partial order preferences.

**Proof:** see appendix.

**Proposition 11:**
If n denotes the number of tuples in the log-relation R, $k_1$ denotes the effort for mining $P_D = (A, <_{PD})$ and $k_2$ denotes the effort for mining $Q_D = (B, <_{QD})$, the algorithms 4 and 5 have the complexity $O(n^2+nk_1+nk_2)$.

**Proof:** see appendix.

Our results so far can be summarized into the following theorem.

**Theorem 1:**
All algorithms create strict partial order preferences.

## 5 Experimental Evaluation

In this section we present test results and performance measurements of an efficient database-driven implementation of a Preference Miner prototype.

### 5.1 Preference Mining Test Results

For our test environment we defined 35 preference profiles, where each profile contains between two and six preferences. In our simulation each user queries the product database between 25 to 50 times, whereby the exact number of requests is randomized. In each query a preference of the considered user is chosen and a product database is requested with it using Preference SQL [9]. The results are stored in a log database. Afterwards we use the Preference Mining algorithms to detect preferences within the log data. A comparison of the detected preference profiles with the predefined user preferences will show the effectiveness of the Preference Mining algorithms. To assess the quality of our results we define preference precision and preference recall.

**Definition 10 (Precision and recall for preferences)**
Preference precision and preference recall are defined as

$$\text{precision} = \frac{\text{number of correctly detected preferences of user i}}{\text{number of all detected preferences of user i}}$$

$$\text{recall} = \frac{\text{number of correctly detected preferences of user i}}{\text{number of all preferences of user i}}$$

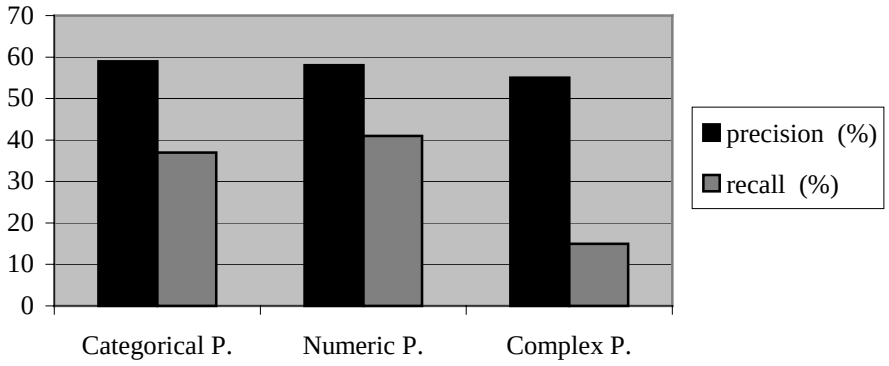



Fig. 2. Precision and recall for the different preference types

The test results in Fig. 2 show average precision and average recall over all test users. Mining categorical preferences leads to a 60 % precision and a 38 % recall, numerical preferences result in 58 % precision and over 40 % recall using histograms as density estimation, and combined preferences yield to 55 % precision and 15 % recall. An approximately 60 % precision denotes a very promising behavior of our Preference Min-

ing algorithms. Typical problems are user preferences for values that don't occur in the product database and dependencies between preferences: if a base preference of a complex preference is not detected, it is very difficult to detect the complex preference itself. Such problems also influence the preference recall. Note, that we filled the log relation with the search results. In real-life applications even better Preference Mining results can be achieved, if the selected results or query information is used.

### 5.2 Performance Measurements

In this section we analyze the efficiency of the Preference Miner prototype for large data sets. The underlying database system is an Oracle 8i database server on an AMD CPU with 1,3 ghz and 1,5 gigabyte main memory. For our tests, we created relations with 10,000, 20,000, 30,000, 40,000 and 50,000 tuples of synthetic data. Categorical attributes contain 20 different categories. Numerical attributes have a data range of 200 (maximal minus minimal value). For mining complex preferences we assume one categorical and one numerical attribute. Fig. 3 reports the average runtimes for detecting a single preference for the different preference types w.r.t. the number of tuples.

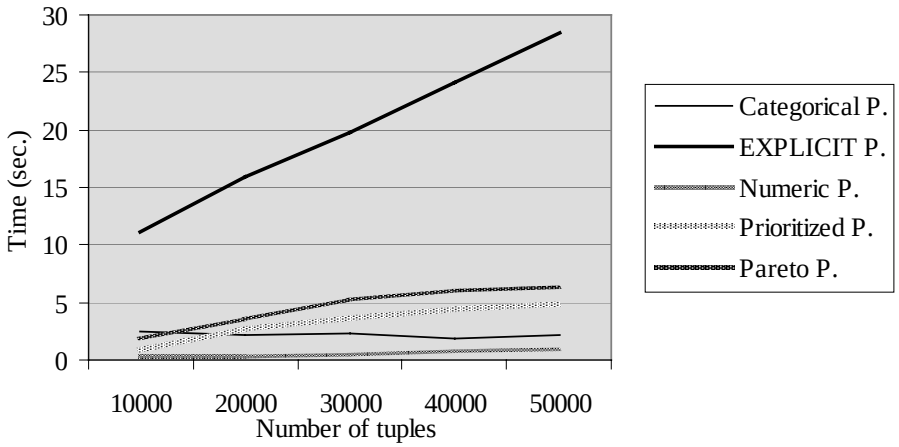



Fig. 3. Runtimes for detecting a single preference for the different preference types

Mining numerical preferences is the fastest task, since histograms can be computed very efficiently in the database layer. The miner for categorical data needs more effort since clustering is a more expensive iterative process. Mining Prioritized and Pareto preferences needs about 5 seconds in the average. The most expensive algorithm is the miner for EXPLICIT preferences (algorithm 2). The cost-intensive part is the cycle test and leads to a performance which depends linearly to the number of tuples. The efficiency of our Preference Mining algorithms allows their usage for *online Preference Mining*: while interacting with a customer an e-application can check online his preferences and react flexible to his wishes during the sales process.

### 5.3 Case Study

We also performed an analysis of the Preference Mining algorithms on the log data of the COSIMA application [8]. Over five hundred users queried the COSIMA comparison shop almost four thousand times. COSIMA offers shopping in the three categories books, cds and computer products. Our application server log records for each query the timestamp, the shop category, the preferred price interval and – depending on the shop category – title and author in the book shop, title and performer in the cd shop and product name and product group in the computer hardware category. Table 2 shows a short extract of the COSIMA log-data.

Table 2. COSIMA log data

| user | timestamp | category | title/name | author/perf/group | price1 | price2 |
|---|---|---|---|---|---|---|
| 1 | 05.11.00 12:19 | cd | Philadelphia | Mark Knopfler | null | null |
| 2 | 05.11.00 13:43 | hardware | Handy | Communication | null | 1000,00 |
| 3 | 05.11.00 14:37 | hardware | HP | Printer | 299,00 | 350,00 |

User 1 searched for the cd “Philadelphia” of “Mark Knopfler” and didn’t specify any price preferences, whereas user 2 had a price limit of 1000,00 German Marks for a mobile phone. User 3 queried for a HP printer with a preferred price range from 299,00 to 350,00 German Marks.

We applied the Preference Mining algorithms on the COSIMA log data to detect customer preferences. Thereby, we analyzed the log-data for each user separately. We got several interesting results. The Preference Miner detected lots of POS preferences and also one POS/POS preference for the shop category. Quite a few LOWEST preferences for price were detected by analyzing the lower price limit. NEG preferences and even combined preferences were also detected with the Preference Miner. Mining preferences for COSIMA users works very fast: On a computer with 1,3 ghz and 1,5 gigabyte main memory the Preference Miner needed less than one second to detect customer preferences.

Such preference knowledge can be very useful for personalized applications like COSIMA. Sales advice can be adapted to the customer’s individual preferences, e.g. if he likes minimal prices, COSIMA should emphasize cheap products or bargains. This approach also allows the selection of one designated product out of a query result set, which can be offered as first choice. Furthermore, preferences gained with Preference Mining are useful for personalized product recommendations and for the composition of individual product bundles.

## 6 Summary and Outlook

In this paper we have presented a novel approach for mining preferences from user log data based on the concept of strict partial order preferences. We presented several algorithms for the detection of categorical, numerical and complex preferences. Our prototype implementation executes all data-intensive operations on the database server and exhibits excellent efficiency. Our experimental results also demonstrate promising precision and recall of the detected user preferences.

Our next steps include the integration of user situations into preferences. Situations can be described with a set of parameters like current time and location, the user's role or physical and psychological condition of the user. Some preferences may only be relevant under specific situations; for example, in a bookshop a user may have different preferred categories whether he is at work or at home. A major task is the adaptation of our Preference Mining algorithms in order to detect *situated preferences*.

Another research task is the design of an appropriate storage structure for preferences. Such a *Preference Repository* should not only be able to record preferences detected with the Preference Miner but also preferences defined with Preference SQL or Preference XPATH. The integration of situations should also be possible as well as user identifiers to assign users and user groups. Finally, the Preference Repository shall also include a set of appropriate access operations for inserting, deleting and updating preferences. It can also be used to find users with similar preferences and with it product recommendations based on preferences can be offered. Therefore the Preference Repository is also a major step towards advanced personalized applications.

## References


1. D. Beeferman and A. Berger: *Agglomerative Clustering of a Search Engine Query Log*. In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 407-416, Boston, Massachusetts, USA, 2000.
2. J. Delgado and N. Ishii: *Online Learning of User Preferences in Recommender Systems*. In Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering, Stockholm, Schweden, 1999.
3. D. Eppstein: *Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs*. In 9th ACM-SIAM Symposium on Discrete Algorithms, p. 619-628, San Francisco, California, USA, 1998.
4. V. Estivill-Castro and M. E. Houle: *Robust Distance-Based Clustering with Applications to Spatial Data Mining*. In 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining , p. 327–337, Beijing, China, 1999.
5. T. Joachims: *Optimizing Search Engines using Clickthrough Data*. In Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002), Edmonton, Alberta, Canada, 2002.
6. D. Jungnickel: Graphs, Networks & Algorithms. Springer Verlag, January 1999.
7. W. Kießling: *Foundations of Preferences in Database Systems*. In Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002), p. 311-322, Hong Kong, China, 2002.

8. W. Kießling, S. Fischer, S. Holland and T. Ehm: *Design and Implementation of COSIMA - A Smart and Speaking E-Sales Assistant*. In 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2001), p. 21-30, San Jose, California USA, 2001.
9. W. Kießling and G. Köstler: *Preference SQL - Design, Implementation, Experiences*. In 28th International Conference on Very Large Data Bases (VLDB 2002), p. 990-1001, Hong Kong, China, 2002.
10. S.-J. Ko, J.-H. Lee: *User Preference Mining through Collaborative Filtering and Content Based Filtering in Recommender System*. In Proceedings of the 3rd International Conference on E-Commerce and Web Technologies (EC-Web 2002), p. 244-253, Aix-en-Provence, France, 2002.
11. R. Kohavi: *Mining E-Commerce Data: The Good, the Bad, and the Ugly*. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 8-13, San Francisco, California, USA, 2001.
12. B. Mobasher, R. Cooley and J. Srivastava: *Automatic Personalization Based on Web Usage Mining*. In Communications of the ACM, vol. 43 (8), p. 142-151, August, 2000.
13. G. Rossi, D. Schwabe and R. Guimaraes: Designing Personalized Web Applications. In Proceedings of the 10th World Wide Web Conference (WWW 2001), p. 275-284, HongKong, China, 2001.
14. P. J. Rousseeuw: *Silhouettes: A Graphical Aid to the Interpretations and Validation of Cluster Analysis*. Journal of Computational and Applied Mathematics, 20:53–65, 1987.
15. D. W. Scott. *On Optimal and Data-Based Histograms*. Biometrika, 66:605–610, 1979.

## Appendix

**Proposition 1:**
A data-driven preference defines a strict partial order.

**Proof:**
We have to show that the relation $<_{PD}$ of a data-driven preference $P = (A, <_{PD})$ is irreflexive and transitive.

**1. Attribute A has a categorical domain.**

*$<_{PD}$ is reflexive:*

Let $a \in \text{dom}(A)$ be a value of A.

$$a <_{PD} a$$
$$\Leftrightarrow \text{freq}_A(a) < \text{freq}_A(a)$$

leading to a contradiction.

*$<_{PD}$ is transitive:*

Let $a, b, c \in \text{dom}(A)$ be values of A.

$$a <_{PD} b \wedge b <_{PD} c \overset{!}{\Rightarrow} a <_{PD} c$$
$$\Leftrightarrow \text{freq}_A(a) < \text{freq}_A(b) \wedge \text{freq}_A(b) < \text{freq}_A(c) \Rightarrow \text{freq}_A(a) < \text{freq}_A(c)$$

The <-relation is transitive and therefore the last transformation holds.

**2. Attribute A has a numerical domain.**

*$<_{PD}$ is reflexive:*

Let $a \in \text{dom}(A)$ be a value of A.

$$a <_{PD} a$$
$$\Leftrightarrow \text{freq}_A([a-\varepsilon, a+\varepsilon]) < \text{freq}_A([a-\varepsilon, a+\varepsilon])$$

leading to a contradiction.

*$<_{PD}$ is transitive:*

Let $a, b, c \in \text{dom}(A)$ be values of A.

$$a <_{PD} b \wedge b <_{PD} c \overset{!}{\Rightarrow} a <_{PD} c$$
$$\Leftrightarrow \text{freq}_A([a-\varepsilon, a+\varepsilon]) < \text{freq}_A([b-\varepsilon, b+\varepsilon]) \wedge \text{freq}_A([b-\varepsilon, b+\varepsilon]) < \text{freq}_A([c-\varepsilon, c+\varepsilon])$$
$$\Rightarrow \text{freq}_A([a-\varepsilon, a+\varepsilon]) < \text{freq}_A([c-\varepsilon, c+\varepsilon])$$

As above the <-relation is transitive and therefore the last transformation holds.

□

**Proposition 2:**
Preferences detected with Algorithm 1 have the following properties:

- POS preference: values within the POS-set occur more often in the log-relation than values outside the POS-set.
- NEG preference: Values within the NEG-set occur with lower frequencies than other values.
- POS/POS preference: Values within POS1-set occur more often than values within POS2-set and the latter values occur more frequent in the log relation than other values.
- POS/NEG preference: Values within POS-set occur more often than other values. The values within NEG-set occur with lowest frequencies.
- EXPLICIT preference: For each value within E-graph any successor occurs with lower frequency.

**Proof:**
The case differentiation in step 3 of algorithm 1 checks for every preference the conformance of the according frequencies.

□

**Proposition 3:**
Let n be the number of tuples in the log relation R and $k = |\pi_A(R)|$ the number of different values of an attribute A in R. By using hierarchical clustering algorithm 1 has the complexity $O(n + k^2)$.

**Proof:**
Step (1): the computation of the frequencies needs a linear scan of the data (O(n)).
Step (2): hierarchical clustering computes a clustering of the k different values in $O(k^2)$ [3].
Step (3): the case differentiation can be done in constant time O(1).

Therefore the overall complexity of algorithm 1 is $O(n + k^2)$.

□

**Proposition 4:**
EXPLICIT-preferences detected with algorithm 2 are strict partial orders.

**Proof:**
The steps (d1), (d2) and (d3) of algorithm 2 create a directed graph (E-graph). Since cycles are avoided (steps (d1) and (d2)) this graph is acyclic. Since a directed acyclic graph represents a strict partial order [6], the constructed EXPLICIT preference EXP(A, E-graph) forms a strict partial order.

□

**Proposition 5:**
Let n be the number of tuples in the log relation R and $k = |\pi_A(R)|$ the number of different values of an attribute A of R. Then the complexity of algorithm 2 is $O(n + k^2(n + k^2))$.

**Proof:**
Step (1): the computation of the k occurring values needs a scan of the data (O(n)).
Step (2): the two nested for-loops have the effort $O(k^2)$.
Within the for-loops the following steps have to be executed:
Steps (2a), (2b) and (2c): the computation of the query ids and of the numbers s and t needs a scan of the data (O(n)).
Steps (2d1), (2d2) and (2d3) require the computation of a path in a graph with maximal k vertices. This task needs $O(k^2)$ by using Dijkstra's algorithm [6].
Therefore the overall complexity of algorithm 2 is $O(n + k^2(n + k^2))$.

□

**Lemma 1:**
Let A be a numerical attribute and φ(x) be a continuous density function of some values $x_1, \ldots, x_k \in$ dom(A). For a, b ∈ dom(A) we have

$$a <_{PD} b \Leftrightarrow \varphi(a) < \varphi(b)$$

**Proof:**
Since φ(x) is continuous and non-negative (as density function) there is an ε > 0 so that the following equivalence holds:

$$\begin{aligned}
\varphi(a) < \varphi(b) \quad &\Leftrightarrow \quad \int_{a-\varepsilon}^{a+\varepsilon} \varphi(x)dx < \int_{b-\varepsilon}^{b+\varepsilon} \varphi(x)dx \\
&\Leftrightarrow \quad P(a-\varepsilon \le X \le a+\varepsilon) < P(b-\varepsilon \le X \le b+\varepsilon) \\
&\Leftrightarrow \quad freq_A(a-\varepsilon, a+\varepsilon) < freq_A(b-\varepsilon, b+\varepsilon) \\
&\Leftrightarrow \quad a <_{PD} b
\end{aligned}$$

□

**Proposition 6:**
The preferences of definition 6 are data-driven preferences, i.e. better values occur with higher frequencies.

**Proof:**
- There is a data-driven LOWEST preference, iff φ(x) is monotonic decreasing. According to Lemma 1 greater values occur with lower frequencies.
- There is a data-driven HIGHEST preference, iff φ(x) is monotonic increasing. According to Lemma 1 greater values occur with higher frequencies.
- There is a data-driven AROUND preference with around value z, iff φ(x) is monotonic increasing for x < z and monotonic decreasing for x > z. According to Lemma 1, values with greater distance apart from z occur with lower frequencies.
- There is a data-driven BETWEEN preference with [low, up]-interval, iff
  - φ(x) is monotonic increasing for x < low,
  - φ(x) is monotonic decreasing for x > up,
  - ∀ x∈ [low, up], ∀ y∉ [low, up]: φ(x) > φ(y).

  According to Lemma 1 values within [low, up] occur with highest frequencies. Values with greater distance apart from the interval borders occur with lower frequencies.

- There is a data-driven SCORE preference with score function f, iff $\varphi(f(x))$ is monotonic increasing. According to Lemma 1 values with greater f(x) occur with higher frequencies.

□

**Proposition 7:**
If the real density function is known, the preferences detected with algorithm 3 have the following properties:

- LOWEST preference: greater values occur with lower frequencies.
- HIGHEST preference: greater values occur with higher frequencies.
- AROUND preference: values with greater distance apart from the around-value occur with lower frequencies.
- BETWEEN preference: values within the [low, up] interval occur with highest frequencies. Additionally, values with greater distance apart from the interval borders occur with lower frequencies.

**Proof:**
If the density function is known, algorithm 3 follows exactly the definitions for numerical preferences (def. 6). Therefore the above properties hold.

□

**Proposition 8:**
Algorithm 3 creates strict partial order preferences.

**Proof:**
The algorithm creates LOWEST, HIGHEST, AROUND and BETWEEN preferences. As shown in [7] these preferences are strict partial orders.

□

**Proposition 9:**
By using histograms as density estimation and Scott's rule for the bin-width computation [15], algorithm 3 has the complexity $O(kn + k^3)$, whereby n denotes the number of tuples in the log relation R and k is the number of bins of the histogram.

**Proof:**
Step (1): Scott's rule defines the bin width as $h = 3.49\,\hat{\sigma} n^{1/3}$ [15], which can be computed in linear time, since the estimated standard deviation needs O(n). For every value in the log-relation the according bin has to be detected O(kn). Therefore the computation of the whole histogram needs $O(n + kn) = O(kn)$.
Steps (2a), (2b) and (2c): the histogram consists of k bins. For LOWEST, HIGHEST and AROUND preferences maximal k bins have to be checked leading to O(k).
Step (2d): let $k_1, \ldots, k_m$ be the bins of the histogram. At first, the BETWEEN conditions have to be checked for the first ($k_1$) and the last ($k_m$) bin as upper and lower border, respectively (O(k)). This process has to be iterated for all $k_i$ and all $k_j$ with $k_i < k_j$. This leads to maximal $k^2/2$ iterations. Therefore the test for a BETWEEN preference has the complexity $O(k^3)$ leading to the overall complexity $O(kn + k^3)$.

□

**Proposition 10:**
Algorithm 4 and 5 create strict partial order preferences.

**Proof:**
The algorithms create Prioritized and Pareto preferences as output. Both Prioritized and Pareto preferences define strict partial orders [7].

□

**Proposition 11:**
If n denotes the number of tuples in the log-relation R, $k_1$ denotes the effort for mining $P_D = (A, <_{PD})$ and $k_2$ denotes the effort for mining $Q_D = (B, <_{QD})$ the algorithms 4 and 5 have the complexity $O(n^2+nk_1+nk_2)$.

**Proof:**
Algorithm 4:
Step (1): for each maximal value of $P_D$ (at most n) the associate values in B are computed with a linear scan of the log relation leading to $O(n^2)$.
Step (2): there are maximal n sets of associate values thus the miner for $Q_D$ has to be applied maximal n times leading to $O(nk_2)$. The miner for $P_D$ must also be applied at most n times ($O(nk_1)$).
Therefore the overall complexity is $O(n^2+nk_1+nk_2)$.
Algorithm 5:
The argumentation is the same as above.

□