

Boosting the Learning for Ranking Patterns

Nassim Belmecheri^{1,2}, Nouredine Aribi¹, Nadjib Lazaar², Yahia Lebbah¹ and Samir Loudni³

¹Lab. LITIO, Université Oran1, 31000, Oran, Algeria.

²LIRMM, Université de Montpellier, CNRS, 34000, Montpellier, France.

³ TASC (LS2N-CNRS), IMT Atlantique, 44307, Nantes, France.

Contributing authors: belmecheri.nassim@edu.univ-oran1.dz; ,aribi.nouredine@edu.univ-oran1.dz; Nadjib.Lazaar@lirmm.fr; lebbah.yahia@edu.univ-oran1.dz; samir.loudni@imt-atlantique.fr;

Abstract

Pattern mining is a valuable tool for exploratory data analysis, but identifying relevant patterns for a specific user is challenging. Various interestingness measures have been developed to evaluate patterns, but they may not efficiently estimate user-specific functions. Learning user-specific functions by ranking patterns has been proposed, but this requires significant time and training samples. In this paper, we present a solution that formulates the problem of learning pattern ranking functions as a multicriteria decision making problem. Our approach uses Analytic Hierarchy Process (AHP) to elicit weights for different interestingness measures based on user preference. We also propose an active learning mode with a sensitivity-based heuristic to minimize user ranking queries while still providing high-quality results. Experiments show that our approach significantly reduces running time and returns precise pattern ranking while being robust to user error, compared to state-of-the-art approaches.

Keywords: Interactive Data Mining, Machine Learning, Active Learning, Multicriteria decision making, Analytic Hierarchy Process.

1 Introduction

Data mining is a field that focuses on discovering relevant information from data and transforming it into useful knowledge. One of its significant subfields, pattern mining, involves searching and enumerating significant patterns in data. In recent years, there has been a notable shift in the pattern mining community from efficiency-based approaches to methods that can extract more meaningful patterns. Unfortunately, obtaining relevant results with traditional pattern mining methods can be challenging and time-consuming. There are two main issues: first, a vast number of patterns are discovered, many of which are redundant. Second, the preferences and domain experts' background knowledge is not considered. The significance of incorporating user preferences and knowledge was initially highlighted by Silberschatz and Tuzhilin [Silberschatz and Tuzhilin \(1995\)](#). The fundamental concept is to represent the user's preferences using objective interestingness measures. However, as noted in [Bie \(2011\)](#), using objective quality measures has limited practical utility since interestingness is subject to the particular user and task at hand. The interestingness of a pattern is determined solely based on the data.

In recent years, there has been a growing emphasis on user-centered, interactive, and anytime pattern mining [Dzyuba and van Leeuwen \(2013a\)](#); [Boley et al \(2013\)](#); [Dzyuba and van Leeuwen \(2017\)](#). This paradigm highlights the need to quickly present users with patterns that are likely to be interesting to them, and to obtain feedback that will influence subsequent iterations of the interactive mining process. An important aspect of this framework is the ability to learn user-specific functions for ranking patterns from feedback. Although the idea of learning user-specific ranking functions was first investigated in [Xin et al \(2006\)](#), it has been extended in recent years by Boley et al. [Boley et al \(2013\)](#) and Dzyuba et al. [Dzyuba and van Leeuwen \(2013a, 2017\)](#) in the context of interactive pattern mining. Additionally, Xuguang Bao et al. [Bao et al \(2021\)](#) and Liang Chang et al. [Chang et al \(2022\)](#) have adapted interactive data mining for extracting interesting spatial co-location patterns. These approaches leverage standard machine learning techniques to learn weighted vectors based on selected pattern features, such as items, transactions, and length.

In [Dzyuba and van Leeuwen \(2013a\)](#), a linear ranking function is learned using support vector machines (RankingSVM). In [Boley et al \(2013\)](#); [Dzyuba and van Leeuwen \(2017\)](#), the authors propose using Stochastic Coordinate Descent (SCD) [Shalev-Shwartz and Tewari \(2011\)](#) to learn a logistic function. While these methods enable the exploitation of user feedback to learn ranking functions, they also increase the computational cost of the learning task, particularly when the number of pattern-measure pairs used for ranking increases. Nonetheless, a crucial aspect of this framework is the ability to quickly present a set of patterns to the user and focus on what may be of interest using interestingness measures.

In this paper, we address the challenge of learning pattern ranking functions as a multicriteria decision making problem. To this end, we propose to use a *weighted linear function* to aggregate all the individual measures into a global ranking function. Linear functions have been widely adopted in learning to rank models because they are simple, interpretable, and efficient, yet can still effectively capture the underlying

relationships between input features and relevance labels [Borges et al \(2005\)](#); [Qin et al \(2010\)](#).

We propose a learning algorithm that is both fast and scalable, which enables us to maintain the expected weights of measures. Our approach leverages the Analytical Hierarchy Process (AHP) [Saaty \(1988\)](#) and a set of user-ranked patterns to learn the weighting vector of measures. Specifically, we seek to maximize the correlation between the unknown user’s ranking function and the learned AHP-based ranking function. We previously introduced this approach in our work [Belmecheri et al \(2022\)](#), where we presented the initial results of the passive version of the algorithm.

This study presents a new learning algorithm that can operate in both active and passive modes, specifically designed for pattern ranking. To improve performance in active mode, a sensitivity-based heuristic is proposed for selecting patterns. The algorithm is explained in detail using a running example. The approach was validated on large datasets using three different user simulations in both passive and active modes. The results indicate that the proposed approach is robust and can account for user mistakes and changes in ranking criteria in interactive learning scenarios.

We conducted experiments to evaluate the effectiveness of our approach using association rule mining as a case study [Agrawal et al \(1993\)](#). Our research questions were as follows: (1) Can user-specific preferences over all patterns be inferred from a sample of ranked patterns, and if so, what is the required input data size and learning time? (2) How does our new learning method based on AHP compare to the currently used method for learning user-specific ranking functions? (3) Is the sensitivity heuristic an appropriate choice for query selection in active learning mode? (4) How does the robustness of our approach compare to others in the face of user errors? Our experiments on various datasets demonstrated that our approach significantly reduces running time while accurately ranking patterns compared to state-of-the-art methods.

The paper is organized as follows: In Section 2, we review the related work in pattern mining and ranking functions. In Section 3, we provide some necessary preliminaries for our approach. Section 4 formulates the problem we tackle and describes our proposed method. A running example is presented in Section 5 to illustrate our approach. In Section 6, we report and discuss the empirical results of our experiments. Finally, we conclude the paper in Section 7.

2 Related work

Iterative/user-centric pattern mining has gained significant attention in recent years, where the user is involved in the mining process to improve the mining results by providing feedback. This approach is particularly useful in domains like medical diagnosis or fraud detection, where the user’s domain knowledge is essential. To make this process more efficient and effective, researchers have been exploring methods to learn ranking functions that can quickly present patterns to the user based on their interests. This problem, known as object ranking, has been studied extensively in machine learning and data mining communities.

One common approach to object ranking is to learn a ranking function based on pairwise comparisons. However, these methods can be time-consuming and expensive since they require a large number of pairwise comparisons to learn a reliable ranking function. To address this challenge, researchers have explored alternative methods such as multicriteria decision making. In our approach, we use the Analytical Hierarchy Process (AHP) to learn the measures' weighting vector, which maximizes the correlation between the user's ranking function and the learned AHP-based ranking function. In [Xin et al \(2006\)](#), the authors proposed a user-centric approach to mining interesting patterns based on the user's feedback. They developed a log-linear model for itemsets that considers the user's prior knowledge about the items and their relationships. The model was trained using RankingSVM formulations to learn a ranking function that assigns a score to each itemset. The authors also proposed a belief model that exploits belief probabilities assigned to transactions for more complex patterns. The authors demonstrated that their approach outperformed existing methods in terms of accuracy and efficiency for complex patterns where traditional methods may fail to capture the user's preferences.

Dzyuba et al. [Dzyuba and van Leeuwen \(2013b\)](#) extended the approach proposed by Xin et al. [Xin et al \(2006\)](#) and used RankingSVM for active preference learning to rank patterns. However, these methods suffer from computational inefficiencies, which limit their scalability on large datasets. In contrast, the AHP-based approach proposed in our work has a linear time complexity and can efficiently rank a large number of patterns.

In [Bhuiyan and Al Hasan \(2016\)](#), the authors proposed a user-centric generic framework (PRIIME) for learning ranking functions. In order to achieve this, regression techniques based on neural networks were adopted. They assigned a score to each pattern based on user feedback and trained a regression model on labeled patterns and their corresponding scores. The model predicts the score of new patterns based on their features and the user feedback. The proposed method outperformed existing methods in terms of accuracy and efficiency and can effectively mine interesting patterns and provide valuable insights to the user.

In [Bao et al \(2021\)](#), an efficient interactive approach is proposed for users to discover their preferred co-location patterns. The approach consists of a flexible interactive framework, an ontology-based similarity measure between co-location patterns, a pattern filtering model to express user preferences, and a pruning scheme to reduce the number of outputs.

In [Chang et al \(2022\)](#), the authors propose IDMBs (Interactive Data Mining based on Support Vector Machine), an interactive mining system that uses SVM to discover user-preferred co-location patterns. The system includes a filtering algorithm to select patterns, which are then annotated by the user, and an SVM model to train on these patterns and discover additional user-preferred co-location patterns.

Our proposed approach tackles the problem of learning a user-specific function for ranking patterns as a multicriteria decision making problem using AHP. Moreover, our method remains fast when the number of patterns used for ranking increases.

Most of the related work uses SVM-based algorithms to learn a user-specific function. According to [Bottou and Lin \(2007\)](#), the worst-case time complexity of the SVM algorithm varies between k^2 and k^3 on the number of samples k ; whereas our approach is linear on k (see section 4.4).

3 Background

In this section, we provide the necessary background information about pattern mining, interestingness measures commonly used to evaluate pattern quality, and multi-criteria decision making using the AHP method.

3.1 Pattern mining and interestingness measures

Data of all types can be represented as a triplet $(\mathcal{O}, \mathcal{A}, \mathcal{L})$, where $\mathcal{O}$ is a set of objects, $\mathcal{A}$ is a set of attributes, and $\mathcal{L}$ is the relational language used to express objects with respect to a set of attributes. This framework is inspired by the concepts of inductive databases [Raedt \(2002\)](#); [Imielinski and Mannila \(1996\)](#). The language $\mathcal{L}$ can be used to define various types of datasets. For instance, when $\mathcal{L}$ is a binary relation between attributes and objects (i.e., $\mathcal{L} \subseteq \mathcal{O} \times \mathcal{A}$), we get the classical itemset mining framework. When $\mathcal{L}$ expresses each object as a sequence of attributes, we get a sequential dataset. Finally, when $\mathcal{L}$ expresses each object as a graph where nodes and edges are labeled, we get a dataset of graphs.

The goal of data mining is to uncover meaningful patterns or regularities between objects within large datasets. To evaluate the quality of these patterns and determine the most relevant ones, interestingness measures are employed.

The field of data mining is actively researching how to measure the interestingness of discovered patterns with respect to user-specific preferences. Nine concepts have been introduced to assess the interestingness of patterns: *Conciseness*, *Generality*, *Reliability*, *Peculiarity*, *Diversity*, *Novelty*, *Surprisingness*, *Utility*, and *Applicability* [Geng and Hamilton \(2006a\)](#). Some of these concepts are correlated, such as *Surprisingness* and *Applicability*, while others are conflicting, like *Generality* and *Peculiarity*, and some are independent, like *Novelty* and *Utility*. Depending on the user's preferences, they may value certain concepts over others. To satisfy these preferences, a wide range of measures and aggregation techniques have been proposed [Geng and Hamilton \(2006b\)](#); [Kuznetsov and Makhalova \(2018\)](#).

3.2 Analytical Hierarchy Process (AHP)

AHP is a popular multicriteria decision-making method created by Saaty [Saaty \(1988\)](#). It involves breaking down the decision problem into a hierarchical structure and deriving criteria weights from the decision maker using pairwise comparisons, rather than relying solely on numerical values. After constructing a decision problem hierarchy using the AHP method, the importance weights of criteria can be determined through the following three main steps:

1. To estimate the relative importance of m criteria at a given level, the user is asked to make pairwise comparisons between them. These comparisons are used to construct a pairwise comparison matrix $\mathbf{A}$ of size $m \times m$, where a_{ij} represents the importance of criterion i relative to criterion j . If $a_{ij} > 1$, it means that criterion i is considered more important than criterion j . To ensure consistency, the matrix is filled such that $a_{ij} = 1/a_{ji}$ and $a_{ii} = 1$. AHP uses a scale of 1 to 9 to quantify the preference degree, where 1 indicates indifference and 9 indicates absolute preference.
2. After constructing the pairwise comparison matrix at a given level, various mathematical techniques proposed in the literature can be used to compute the weight vector $w = (w_1, \dots, w_m)^T$. These techniques include the eigenvector method (EVM) [Saaty and Vargas \(1984\)](#); [Takeda et al \(1987\)](#) and distance-based minimization methods, such as the Least Squares Method, Logarithmic Least Squares Method, Weighted Least Squares Method, Logarithmic Least Absolute Values Method, and Singular Value Decomposition [Gass and Rapcsák \(2004\)](#); [Blankmeyer \(1987\)](#); [Brunelli \(2014\)](#). The EVM method is the most commonly used and computes the weight vector w by solving the following characteristic equation:

$$\begin{cases} \mathbf{A} \cdot w = \lambda_{max} \cdot w \\ w^T \mathbf{1} = 1 \end{cases} \quad (1)$$

where $\mathbf{A}$ is the pairwise comparison matrix, λ_{max} is the highest eigenvalue of $\mathbf{A}$, and $\mathbf{1} = (1, \dots, 1)^T$. The constraint $\sum_{i=1}^m w_i = 1$ is added to help the solving process and avoid infinitely many solutions. Note that, if $\mathbf{A}$ is positive then the highest eigenvalue is real (i.e. $\lambda_{max} \in \mathbb{R}$) [Brunelli \(2014\)](#). Interestingly, Saaty [Saaty \(1977\)](#) mentioned that the perfect¹ normalized eigenvector w satisfies the following relation:

$$a_{ij} = \frac{w_i}{w_j}, \forall i, j = 1 \dots m. \quad (2)$$

In this case, $\lambda_{max} = m$ only when the comparison matrix is perfectly consistent; otherwise, λ_{max} is greater than m . This equation explains why some researchers prefer distance-based methods, which aim to minimize the distance between $(a_{ij})_{m \times m}$ and $(w_i/w_j)_{m \times m}$ directly. However, this leads to a non-linear and difficult-to-solve problem. Therefore, in this paper, we use the EVM method instead of distance-based methods.

3. Since AHP does not guarantee the transitivity of the user's preferences, the pairwise comparison may result in inconsistencies. To detect these inconsistencies, a consistency check is used by calculating the consistency ratio (CR), defined as $CR = \frac{CI}{RI}$, where RI is a constant obtained from the Random Consistency Index table of AHP. The consistency index of the EVM method is computed as follows:

$$CI = \frac{(\lambda_{max} - n)}{n - 1} \quad (3)$$

¹When the components of $\mathbf{A}$ are exactly obtained as the ratio between weights.

The reliability of the computed weighting vector w (solution of Equation (1)) is assessed by checking whether the CR value is less than 0.1.

It is common for hierarchies to contain multiple levels of criteria, especially when there are more than nine criteria. To handle this, it is beneficial to establish a hierarchy based on existing dependencies between the criteria. This can help optimize the decision-making process and reduce the number of required pairwise comparisons. For further information on this topic, we recommend referring to [Brunelli \(2014\)](#). However, in this paper, we will not delve into this aspect. In Example 1, we illustrate the AHP process.

Example 1 Consider a decision problem involving three criteria, where we aim to determine their relative importance in achieving an overall goal. To do so, we construct a pairwise comparison matrix $\mathbf{A}(a_{ij})_{3 \times 3}$, where each element a_{ij} represents the relative importance of criterion i compared to criterion j . The matrix is given as follows:

$$\mathbf{A} = \begin{pmatrix} & M_1 & M_2 & M_3 \\ M_1 & 1 & 1/2 & 1/4 \\ M_2 & 2 & 1 & 1/2 \\ M_3 & 4 & 2 & 1 \end{pmatrix}$$

To determine the weights of the criteria, we can employ the EVM method, which involves solving the equation $\mathbf{A}w = \lambda_{max}w$. The resulting weight vector is given by $w = (w_1, w_2, w_3) = (1/7, 2/7, 4/7)$, where each component represents the weight of the corresponding criterion.

4 AHP-based learning approach for ranking patterns

We aim to develop a ranking function that combines a set of interestingness measures while also taking into account user-specific preferences.

4.1 Problem statement: learning pattern rankings

In this section, we demonstrate that learning a ranking function that incorporates a set of interestingness measures to approximate user-specific preferences can be formulated as a multicriteria decision-making problem. In this context, the criteria correspond to the interestingness measures. The objective of this task is to learn a ranking function for patterns based on a sample of ordered patterns. To accomplish this, we adopt the approach of *ordered feedback*, where users are asked to provide a total order over a small set of patterns based on their subjective interestingness. This approach is inspired by previous research [Dzyuba et al \(2014\)](#).

Let $\triangleright$ be a binary preference relation between patterns. A ranking function based on $\triangleright$, denoted by $r_{\triangleright}(\cdot)$, takes as input a subset of patterns $\mathcal{P} \subseteq \mathcal{L}$ and returns as output a permutation $S = \langle P_{r_1}, P_{r_2}, \dots, P_{r_n} \rangle$ of $\mathcal{P}$, such that $|\mathcal{P}| = |S|$ and $\forall i, j \in [1, n], i < j, P_{r_i} \triangleright P_{r_j}$.

We aim to learn the user ranking function denoted by $r_{\triangleright_u}$, which is based on the user preference relation $\triangleright_u$. To achieve this, we use a set of interestingness measures $\mathcal{M} = \{M_1, \dots, M_m\}$ and their corresponding ranking functions $r_{\triangleright_{M_1}}, \dots, r_{\triangleright_{M_m}}$.

The rank of pattern $P_j \in \mathcal{P}$ w.r.t. the interestingness measure M_i is denoted by $\text{rank}_{M_i}(P_j)$. Here, $(P_k \triangleright_{M_i} P_l)$ means that $M_i(P_k) \geq M_i(P_l)$.

To frame the ranking problem as a learning problem, we need to find the weights w_i of a linear aggregation function over the measures from a set of ranked patterns that maximizes the correlation with the user ranking function $r_{\triangleright_u}$. Specifically, we aim to find the coefficients of the following function by solving a multi-criteria optimization problem:

$$f_w(P) = \sum_{i=1..m} w_i \cdot M_i(P) \quad (4)$$

Our approach leverages user feedback to perform pairwise comparisons between measures and estimate their relative importance based on individual measure scores. The pairwise comparison results are then aggregated into a matrix, which is used by the Analytic Hierarchy Process (AHP) to compute the weights w_i for each measure.

Comparing a single measure to a user ranking.

Let $S = \langle P_1, \dots, P_n \rangle$ be a user ranking of patterns. Let $\mathcal{M} = \{M_1, \dots, M_m\}$ be a set of interestingness measures for patterns. To evaluate how well a particular measure performs in terms of overall ranking accuracy compared to a user ranking, we use Kendall's W concordance coefficient [Kendall and Smith \(1939\)](#). For a measure M_i and a user ranking S , we define $K_i(S) \in [0, 1]$ as the Kendall's W coefficient between the ranking obtained using M_i and the user ranking S , where a value of 1 indicates perfect concordance and a value of 0 indicates no concordance.

$$K_i(S) = \frac{12\alpha}{|S|^3 - |S|}, \quad \alpha = \sum_{\ell=1}^{|S|} (R_\ell - \bar{R})^2, \quad (5)$$

where $R_\ell = \text{rank}_{M_i}(P_\ell) + \text{rank}_u(P_\ell)$ with $P_\ell \in S$, and $\bar{R} = \frac{1}{|S|} \sum_{\ell=1}^{|S|} R_\ell$.

Comparing two measures according to a user ranking.

Let M_i and M_j be two measures to be compared based on a given user ranking S . We can compute the gap $\Delta_{i,j}(S) = K_i(S) - K_j(S)$ to determine how much M_i is closer to the user ranking S than M_j . If $\Delta_{i,j}(S) \geq 0$, then M_i is closer, otherwise, the converse is true. When considering a set of user rankings $\mathcal{S} = \{S_1, \dots, S_n\}$, comparing two measures M_i and M_j on $\mathcal{S}$ involves estimating the measure closest to $\mathcal{S}$ by averaging the gaps over all the user rankings in $\mathcal{S}$: $\Delta_{i,j} = \frac{1}{n} \sum_{S_k \in \mathcal{S}} \Delta_{i,j}(S_k)$.

4.2 Learning weights process

The function [1](#) implements the learning process that computes a weight vector w (a weight for each measure) by exploiting the pairwise comparisons of measures. It takes as inputs a vector of pairwise comparisons Δ and its corresponding index l , a user ranking S , and a set of measures $\mathcal{M}$. The learning step is performed at line [2](#) where for each pair of measures, we first compute the Kendall W coefficient for each

measure i with respect to the user ranking S ($K_i(S)$). Then, we compute the gap between the pairs.

The function 1 implements the learning process that computes a weight vector w (a weight for each measure) by exploiting the pairwise comparisons of measures. It takes as inputs a vector of pairs comparisons Δ and the corresponding index l , a user ranking S , and a set of measures $\mathcal{M}$. The function incrementally updates Δ by taking into account the previous l user rankings at line 3. At line 5, the AHP matrix is constructed by scaling the $\Delta_{i,j}$ comparisons to the AHP values between 1/9 and 9. Once constructed, the criteria weight vector w is returned in line 12 by solving the minimization problem described in line 11, using the eigenvector method (EVM) Brunelli (2014).

Algorithm 1 learnWeights($\langle \Delta, l \rangle, S, \mathcal{M}$)

In : Set of measures $\mathcal{M} = \{M_1, \dots, M_m\}$; user ranking $S = \langle P_1, \dots, P_n \rangle$;

Out : Weight vector w ;

```

1  $\mathbf{A}[i, j] \leftarrow 1, \forall i, j \in \{1, \dots, m\}$ ;
2 foreach  $M_i, M_j \in \mathcal{M} : i < j$  do
3    $\Delta_{i,j} \leftarrow (K_i(S) - K_j(S))$ ;
4 end
5 foreach  $\Delta_{i,j} \in \Delta$  do
6   if  $\Delta_{i,j} < 0$  then
7      $\text{scale } \Delta_{i,j} \text{ to } (-9..-1)$ ;  $\mathbf{A}[j, i] \leftarrow |\Delta_{i,j}|$ ;  $\mathbf{A}[i, j] \leftarrow 1/|\Delta_{i,j}|$ ;
8   else
9      $\text{scale } \Delta_{i,j} \text{ to } (1..9)$ ;  $\mathbf{A}[i, j] \leftarrow \Delta_{i,j}$ ;  $\mathbf{A}[j, i] \leftarrow 1/\Delta_{i,j}$ ;
10 end
11  $w \leftarrow \text{solve} \left( \begin{array}{l} \mathbf{A} \cdot w = \lambda_{max} \cdot w \\ \text{subject to } \sum_{j=1}^m w_j = 1, \quad w_j > 0, \forall j = 1 \dots m. \end{array} \right)$ 
12 return  $w$ ;
```

After learning the weight vector w , we use it to compute the score of a given pattern P_i by applying a weighted aggregation function:

$$g_w(P_i) = \sum_{M_i \in \mathcal{M}} w_i \text{scale}(M_i(P)) \quad (6)$$

The *scale* function is used to normalize the interestingness values of the different measures to a $[0, 1]$ scale so that they can be combined using the weighted aggregation function.

4.3 AHPRank algorithm

We propose Algorithm 2 to implement the AHPRank function (Algorithm 1) for learning a weight vector in a passive/active mode. The algorithm takes as input a set of

measures $\mathcal{M}$, a set of user-ranked patterns $\mathcal{S}$, and a triplet of parameters $\langle \mathcal{P}, \theta, T \rangle$. Here, $\mathcal{P}$ is a collection of patterns, θ is a sample parameter used in the query generator for active learning, and T is the maximum number of iterations used as a stopping criterion in the algorithm.

AHPRank initializes the vector of pairs comparisons to zero in line 1. If AHPRank is called with a non-empty $\mathcal{S}$, it runs in passive mode by calling `learnWeights` function on the given user-ranked patterns $\mathcal{S}$ (lines 3-6). This produces a weight vector w . Otherwise, it runs in active mode by submitting T queries to the user (lines 7-15). In the active mode, the user is asked to rank a subset of patterns proposed by a query generator. Our query generator is based on a heuristic that exploits the quality of the weights learned from the previous iterations. The active learning mode follows the following iterative steps:

1. As a first step, a subset of patterns is sampled from the global set of patterns $\mathcal{P}$ using a random sampling approach. At line 10, the function `SamplePatterns` is called, which returns a sample $\mathcal{P}'$ of θ patterns.
2. Use the `SensitivityBasedHeuristic` function to select a pair of patterns. This heuristic is designed to select a high-quality pair of patterns to accelerate the convergence of the learning process (see function 3 in section 4.3.1 for details).
3. Interacting with the user to obtain feedback through a ranking query (line 12).
4. AHP-based learning of the user ranking function is performed by calling the `learnWeights` function on the user-provided ranking query (line 13).

The proposed method is designed to handle large sets of patterns, which can be computationally expensive to process. To address this issue, we introduce two steps in the active learning mode. First, we randomly sample a subset of patterns from the global set $\mathcal{P}$, which reduces the complexity of the second step as well as introduces a diversity factor. Second, we use a sensitivity-based heuristic to select pairs of patterns that are of high quality and diverse.

It is important to determine when to stop the learning process. Our method allows for a custom stopping criterion, but we also consider the case where a fixed number of iterations T is used. This simulates the scenario where the user can stop at any time based on their satisfaction with the learned ranking function.

4.3.1 A Pattern Selection Heuristic for AHPRank

Active learning is a challenging task since each learning step needs to enhance the learned function. The selection of appropriate patterns for the user to rank is the key to improving the learning process incrementally. However, generating the ideal set of patterns is a computationally demanding task and is known to be NP-hard [Ailon \(2012\)](#). To address this issue, we propose a heuristic (Function 3) based on sensitivity analysis for AHP models [ERKUT and TARIMCILAR \(1991\)](#). Our heuristic aims to select a pair of patterns that are close in terms of their overall score predicted by the learned function relative to the sum of the measure gaps:

$$\sigma(P_i, P_j) = \left| \frac{g_w(P_i) - g_w(P_j)}{\sum_{\ell=1}^m (M_\ell(P_i) - M_\ell(P_j))} \right|; i, j \in 1 \dots k. \quad (7)$$

Algorithm 2 AHPRank($\mathcal{M}, \mathcal{S}, \langle \mathcal{P}, \theta, T \rangle$)

In : Set of measures $\mathcal{M} = \{M_1, \dots, M_m\}$;
 Passive mode ($\mathcal{S} \neq \emptyset$): Set of user-ranked patterns $\mathcal{S} = \{S_1, \dots, S_n\}$;
 Active mode ($\mathcal{S} = \emptyset$): Collection of patterns $\mathcal{P}$; Sample size θ ;
 Number of iterations T ;

Output: w^t : weighting vector at the last iteration t ;

```

1 foreach  $\Delta_{i,j} \in \Delta$  do  $\Delta_{i,j} \leftarrow 0$ ;
2 if  $\mathcal{S} \neq \emptyset$  then
3   foreach  $S_k \in \mathcal{S}$  do
4      $w^t \leftarrow \text{LearnWeights}(\langle \Delta, k \rangle, S_k, \mathcal{M})$ ;           // Learn weights
5   end
6 end
7 else
8    $w^0 \leftarrow \mathbf{1}$ ;                                           // Weight vector
9   for  $t = 1, 2 \dots T$  do
10     $\mathcal{P}' \leftarrow \text{SamplePatterns}(\mathcal{P}, \theta)$ ;                 // Step 1
11     $Q_t \leftarrow \text{SensitivityBasedHeuristic}(\mathcal{P}', w^{t-1}, \mathcal{M})$ ; // Step 2
12     $S_t^* \leftarrow \text{AskRanking}(Q_t)$ ;                         // Step 3
13     $w^t \leftarrow \text{LearnWeights}(\langle \Delta, t \rangle, S_t^*, \mathcal{M})$ ; // Step4
14  end
15 end
16 return  $w^t$ 

```

A successful aggregation should have the property that if two patterns are close in overall predicted score, they should also be close in their measure values. Thus, the algorithm will be enforced into learning the correct preferences of the patterns when having the lowest σ value since the selected pair is surely close in terms of g_w values and very distant on the values of the measures. Example 2 illustrates the selection process of the function 3.

Example 2 Suppose we have two patterns P_1 and P_2 with four interestingness measures $\mathcal{M} = M_1, M_2, M_3, M_4$. Let $P_1 = \langle 0.1, 0.2, 0.3, 0.7 \rangle$, $P_2 = \langle 0.7, 0.3, 0.2, 0.1 \rangle$, and let the current learned weight vector be $w = \langle 0.25, 0.25, 0.25, 0.25 \rangle$. At this stage, $g_w(P_1) = 0.26$ and $g_w(P_2) = 0.26$. Although P_1 and P_2 have the same overall score, their measure values differ. This pair of patterns is an interesting candidate with $\sigma(P_1, P_2) = 0$, where getting the user's preference between them can improve the learning process and bring it closer to the user's ranking.

4.4 Complexity Analysis

Proposition 1 (Time complexity) *Algorithm 2 has a time complexity of $O(nk)$ in passive learning and $O(n)$ in active learning when the number of interestingness measures is bounded.*

Function 3 SensitivityBasedHeuristic($\mathcal{P}, w, \mathcal{M}$)

In : Collection of patterns $\mathcal{P}$; Weight vector w ; Set of measures $\mathcal{M}$;

Output: $Pair$: Selected pair of pattern

```

1  $\mathcal{P}' \leftarrow \text{Sort}(\mathcal{P}, w)$ ; // sort  $\mathcal{P}$  according to  $g_w$ 
2  $Pair \leftarrow \emptyset$ ;
3  $\sigma_{min} \leftarrow +\infty$ ;
4 for  $i \in 1..|\mathcal{P}'| : P_i, P_{i+1} \in \mathcal{P}'$  do
5    $score \leftarrow \sigma(P_i, P_{i+1})$ ;
6   if  $score < \sigma_{min}$  then
7      $\sigma_{min} \leftarrow score$ ;
8      $Pair \leftarrow \{P_i, P_{i+1}\}$ ;
9   end
10 end
11 return  $Pair$ ;

```

Proof Let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a set of n user rankings, where k is the size of the largest ranking S_i and m is a bounded number of measures in $\mathcal{M}$. Once the pairwise comparison matrix $\mathbf{A}$ is built, the preference vector of weights w can be computed using various mathematical techniques, including the eigenvector-based method (EVM) Brunelli (2014). The worst-case time complexity of the EVM approach is $O(m^3)$ Pan and Chen (1999), which is of constant time complexity $O(1)$ since m is constant. The time complexity of computing the matrix at line 3 of the `learnWeights` function is $O(m n k)$ since Kendall's W is in $O(k)$ (see Definition (5)). The complexity of lines 2, 5, and 11 in `learnWeights` are respectively $O(n m^2 + m n k)$, $O(m^2)$, and $O(m^3)$. Notice that in AHP, it is demonstrated in Saaty and Ozdemir (2003) that the number of criteria is recommended to be no more than seven ± 2 . Thus, assuming m is a small and fixed number of measures, we can set $m \leq 9$ and obtain an asymptotic quadratic complexity of $O(n k)$ when AHP is in passive mode. In other words, the approach takes $O(k)$ for each ranking in the given $\mathcal{S}$. When AHP is in active mode, where queries are pairs of patterns ($k = 2$), the time complexity is linear on the number of queries submitted to the user, i.e., $O(n)$. $\square$

It is important to emphasize that in practice, the set $\mathcal{S}$ is typically reduced to a single large ranking in passive learning mode ($n = 1$ and a large k), and to a set of n queries consisting of pairs of patterns in active learning mode (large n and $k = 2$). As a result, AHPRank can run in linear time, with a complexity that is linear in k for passive mode and linear in n for active mode. This low complexity makes AHPRank a fast approach for learning a user ranking function, as supported by the experimental evaluation in Section 6.

5 Running Example

To demonstrate our approach, we utilize an initial set of patterns $\mathcal{P} = \{P_1, \dots, P_{10}\}$, along with a user ranking $r_{\triangleright_u}$, and five interestingness measures $\mathcal{M} = \{M_1, \dots, M_5\}$. The user ranking for $\mathcal{P}$, rankings given by M_i on $\mathcal{P}$, and the

AHPRank results are presented in the columns of Table 1. It is important to note that none of the measures perfectly match the user ranking (column *user*).

<i>user</i>	<i>S</i>	<i>M</i> ₁ – rank		<i>M</i> ₂ – rank		<i>M</i> ₃ – rank		<i>M</i> ₄ – rank		<i>M</i> ₅ – rank		<i>g_w</i> – rank	
1	<i>P</i> ₇	0.95	1	0.48	8	0.79	2	0.30	9	0.80	1	0.72	1
2	<i>P</i> ₃	0.75	3	0.72	1	0.78	3	0.70	2	0.61	2	0.68	2
3	<i>P</i> ₆	0.80	2	0.49	7	0.50	9	0.65	4	0.60	3	0.61	3
4	<i>P</i> ₁	0.47	9	0.47	9	0.76	5	0.56	6	0.59	4	0.54	4
5	<i>P</i> ₈	0.56	6	0.65	4	0.63	8	0.69	3	0.40	5	0.53	5
6	<i>P</i> ₁₀	0.57	5	0.50	6	0.80	1	0.4	8	0.02	10	0.34	8
7	<i>P</i> ₅	0.62	4	0.62	5	0.66	6	0.57	5	0.27	7	0.48	7
8	<i>P</i> ₂	0.48	8	0.66	3	0.65	7	0.1	10	0.05	9	0.33	9
9	<i>P</i> ₄	0.50	7	0.68	2	0.77	4	0.50	7	0.35	6	0.50	6
10	<i>P</i> ₉	0.02	10	0.1	10	0.05	10	0.8	1	0.25	8	0.18	10

Table 1: Running example with 10 patterns and 5 measures.

The data presented in Table 1 is utilized by our algorithm to construct the AHP Matrix and learn a weight vector w over the set of interestingness measures $\mathcal{M} = \{M_1, \dots, M_5\}$. For simplicity, we refer to the AHP algorithm in the passive mode with $\mathcal{S} = \{S_a = \langle P_3, P_1, P_5, P_2, P_4 \rangle\}$. In this case, the user’s ranking order is $P_3 \triangleright_u P_1 \triangleright_u P_5 \triangleright_u P_2 \triangleright_u P_4$. AHP learns the weights by invoking `learnWeights` on S_a and computing the correlation between the user rankings of S_a and the rankings of the measures M_i . The matrix in (8) shows the Δ values for each measure pair. It is worth noting that no two measures perfectly match the user ranking (column *user*) for the pattern set $\mathcal{P}$.

To compute $\Delta_{1,2}(S_a)$, we need to calculate Kendall’s W $K_1(S_a)$ (resp., $K_2(S_a)$) between the ranking of M_1 (resp., M_2) and the user ranking S_a . The value of $\Delta_{1,2}(S_a)$ is given by $(K_1(S_a) - K_2(S_a))$, which in this case is $(0.25 - 0.25) = 0$. After scaling the values of Δ to $(-9.. - 1)$ for negatives and $(1..9)$ for positives, we can observe from Equation (8) that measure M_5 is better than M_4 with a degree of 6. The same degree of preference is observed between M_5 and M_3 , while M_3 and M_4 are indifferent.

$$\Delta = \begin{matrix} & \begin{matrix} (M1) & (M2) & (M3) & (M4) & (M5) \end{matrix} \\ \begin{matrix} (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{matrix} & \begin{pmatrix} & 0 & 0.42 & 0.42 & -0.43 \\ & & 0.20 & 0.28 & -0.25 \\ & & & -0.08 & -0.55 \\ & & & & -0.60 \end{pmatrix} \end{matrix} \xrightarrow{\text{Scaling}} \begin{matrix} & \begin{matrix} (M1) & (M2) & (M3) & (M4) & (M5) \end{matrix} \\ \begin{matrix} (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{matrix} & \begin{pmatrix} & 1 & 4 & 4 & -4 \\ & & 2 & 3 & -3 \\ & & & 1 & -6 \\ & & & & -6 \end{pmatrix} \end{matrix} \quad (8)$$

Afterward, the `learnWeights` function in AHPRank computes the AHP matrix A using the scaled Δ matrix (i.e., the average correlation gap) as input:

$$\mathbf{A} = \begin{matrix} & \begin{matrix} (M1) & (M2) & (M3) & (M4) & (M5) \end{matrix} \\ \begin{matrix} (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{matrix} & \begin{pmatrix} 1 & 1 & 4 & 4 & 1/4 \\ 1 & 1 & 2 & 3 & 1/3 \\ 1/4 & 1/2 & 1 & 1 & 1/6 \\ 1/4 & 1/3 & 1 & 1 & 1/6 \\ 4 & 3 & 6 & 6 & 1 \end{pmatrix} \end{matrix} \quad (9)$$

At the end, AHPRank computes the weighting vector w by solving a minimization problem. The learned weight vector $w = (w_{M_1}, w_{M_2}, w_{M_3}, w_{M_4}, w_{M_5})$ is $(0.24, 0.24, 0.065, 0.065, 0.39)$, reflecting the importance of each measure for achieving the goal of the user ranking function g_w .

Using the AHP interestingness measure g_w from equation 6, we can rank all patterns in $\mathcal{P}$ provided in Table 1. AHPRank achieves an overall ranking accuracy of 91% in this example, correctly ranking the first 5 patterns, the seventh, and the tenth ones.

6 Experiments

In this section, we empirically evaluate the effectiveness of our proposed pattern ranking framework AHPRank. We first introduce our case study on association rules mining and the different oracles we used to simulate user-specific rankings. Next, we present the research questions we aimed to answer, the experimental protocol we followed, and the results we obtained.

6.1 Mining associations rules (ARs)

Our approach, AHPRank, is experimentally evaluated on association rule mining, which is one of the most important and well-studied tasks in data mining. Association rules are implications of the form $X \rightarrow Y$, where X and Y are itemsets such that $X \cap Y = \emptyset$ and $Y \neq \emptyset$. Here, X represents the body of the rule and Y represents its head. The *frequency* of an itemset X in a dataset, denoted by $freq(X)$, is the number of transactions of the dataset containing X . The frequency of a rule $X \rightarrow Y$ is the frequency of the itemset $X \cup Y$, that is, $freq(X \rightarrow Y) = freq(X \cup Y)$.

Various interestingness measures for association rules have been proposed, including support, confidence, interest factor, correlation, and entropy. Tan et al. [Tan et al \(2004\)](#) conducted a study on the usefulness of existing measures in different application types and identified seven independent groups of consistent measures having similar properties, as shown in Table 2.

Groups	Measures
1	Yules Q , Yules Y, Odds Ratio
2	Cosine , Jaccard
3	Laplace , Support
4	ϕcoefficient , CollectiveStrength, piatetskyShapiro's
5	Goodman Kruskal's , Gini Index
6	Interest factor , added value, Klogsen K
7	certainty factor , Mutual Information, Cohen's κ

Table 2: Independent Subjective Measure Groups

To evaluate our approach, we selected one measure from each of the seven groups of consistent measures identified by Tan et al. [Tan et al \(2004\)](#). This gave us a set of seven measures that are independent and have similar properties. These measures are highlighted in bold in Table 2.

6.2 User feedback emulators

Since it can be challenging to evaluate an interactive approach with limited user feedback, we simulate user feedback by using three different objective target ranking functions:

- **RAND-EMU:** The user-specific ranking is equivalent to a random weighted aggregation function. For each measure M_i , we generate a random weight $w_i \in [0, 1]$ such that $\sum_{i=1}^m w_i = 1$.
- **LEX-EMU:** The user-specific ranking follows a lexicographic order on the measures. We define a lexicographic order lex on $\mathcal{M}$ as $lex(\mathcal{M}) = \langle l_1, \dots, l_m \rangle$ such that $\bigcup_{i=1}^m l_i = \mathcal{M}$. Given two patterns (P_1, P_2) , P_1 is preferred to P_2 if $(l_i(P_1) > l_i(P_2))$ or $(l_i(P_1) = l_i(P_2) \wedge l_{i+1}(P_1) > l_{i+1}(P_2))$ for $i = 1$ to $m - 1$.
- **CHI-EMU:** χ^2 is a statistical measure that is a good candidate to emulate user feedback [Ringuest \(1986\)](#). While χ^2 can be a complex function to approximate with non-trivial correlations, it is also a quality measure suggested in [Dzyuba and van Leeuwen \(2017\)](#). We use χ^2 as the target user-specific ranking function over ARs. For an association rule $X \rightarrow Y$, the χ^2 value is defined as:

$$\chi^2(X \rightarrow Y) = \frac{\left(\text{freq}(X \rightarrow Y) - \frac{\text{freq}(X)\text{freq}(Y)}{N} \right)^2}{\frac{\text{freq}(X)\text{freq}(Y)}{N}}, \quad (10)$$

where N is the number of transactions in the dataset.

6.3 Research questions

Our evaluation seeks to address the following research questions:

- **RQ1:** *Can we determine a user's preferences for a set of patterns based on a sample of ranked patterns? If so, how much data is needed and how long does the learning process take?*
- **RQ2:** *How does our proposed AHP-based learning method compare to the SVM-based baseline for automatically learning user-specific ranking functions?*
- **RQ3:** *How effective is AHPRank in an active learning context? Is the sensitivity heuristic a good choice for query selection?*
- **RQ4:** *How effective is AHPRank in an interactive data mining context?*

6.4 Experimental protocol

6.4.1 Implementation settings

We implemented our AHPRank approach in Java with two modes: the passive mode, denoted as AHPRank.0, and the active mode, denoted as AHPRank.1. The code is publicly available on GitHub at github.com/lirmm/AHPRank. We compared our approach to the state-of-the-art SVM-based approach RankingSVM [Dzyuba and van Leeuwen \(2017\)](#). We use RankingSVM.0 to denote the passive version and RankingSVM.1 to denote the active version, following [Dzyuba and van Leeuwen](#)

Dataset	$ \mathcal{T} $	$ \mathcal{I} $	Density(%)	Type of Data	#Rules
Hepatitis	137	68	50.00	Disease	0.5 M
Connect	67,557	129	33.33	Game steps	1 M
Mushroom	8,124	119	18.75	Species of mushrooms	1.5 M
T40	100,000	1,000	4.20	Synthetic dataset	2 M
Retail	88,162	16,470	0.06	Retail market basket data	2,5 M

T40:T40I10D100K

Table 3: Dataset Characteristics.

(2013a). All experiments were conducted on an Intel Core *i7* 2.4 GHz with 16 GB of RAM, with a maximum duration of one hour.

Metrics

We use three metrics to evaluate the performance of our approach:

1. The Spearman’s rank correlation coefficient ρ measures the accuracy of the learned ranking compared to the target ranking over n patterns. It is computed as follows:

$$\rho = 1 - \frac{6 \sum_{i=1}^n (\text{rank}_L(P_i) - \text{rank}_T(P_i))^2}{n(n^2 - 1)} \quad (11)$$

2. The recall metric $R@k$ evaluates the ability of our approach to identify the top k most interesting patterns. It is computed as follows:

$$R@k = \frac{|\{\text{rank}_L(P_i) \leq k : i \in 1..n \wedge \text{rank}_T(P_i) \leq k\}|}{k} \quad (12)$$

3. CPU time in seconds is measured for both the passive and active modes. The waiting time between two queries is also recorded for the active mode.

6.5 Benchmark Datasets

We selected several datasets of realistic sizes from the FIMI repository.² These datasets have different characteristics that represent various application domains. Table 3 reports the number of transactions ($|\mathcal{T}|$), the number of items ($|\mathcal{I}|$), the application domain, and the number of valid rules ($\#Rules$) corresponding to the initial rules mined using a standard association rules algorithm without any knowledge about the user, for each dataset. The datasets are presented in ascending order of $\#Rules$.

6.6 Passive learning results

In this section, we address the first two research questions (**RQ1** and **RQ2**). To that end, we perform a 5-fold cross-validation on each dataset, where we randomly select 20% of the rules as training data and use the remaining 80% for testing. This type of cross-validation allows us to evaluate the effectiveness of the approaches in learning from relatively small training sets. We report the results averaged over the 5 folds.

²fimi.uantwerpen.be/data/

Datasets	measures							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	VBM
RAND-EMU								
Hepatitis	0.78	0.92	0.29	0.97	0.79	0.85	0.45	0.97
Connect	0.68	0.62	0.95	0.70	0.60	0.45	0.57	0.95
Mushroom	0.84	0.97	0.28	0.83	0.55	0.73	0.36	0.97
T40	0.71	0.99	0	0.76	0.99	0.99	0	0.99
Retail	0	0.98	0.43	0.71	0.84	0.98	0.51	0.98
LEX-EMU								
Hepatitis	0.79	0.64	0	0.68	0.92	0.63	0	0.92
Connect	0.21	0.47	0.76	0.26	0	0.23	0.50	0.76
Mushroom	0.21	0.82	0.38	0.77	0.78	0.37	0	0.82
T40	0.76	0.98	0	0.77	0.99	0.97	0	0.99
Retail	0.10	0.78	0.54	0.49	0.84	0.80	0.58	0.84
CHI-EMU								
Hepatitis	0.20	0.92	0.36	0.96	0.73	0.85	0.43	0.96
Connect	0.09	0.15	0	0.02	0.29	0.01	0	0.29
Mushroom	0.51	0.64	0	0.46	0.28	0.88	0	0.88
T40	0.71	0.98	0.17	0.64	0.98	0.99	0.29	0.99
Retail	0	0.94	0.53	0.60	0.83	0.57	0.61	0.94
(1): YulesY (2): Cosine (3): Laplace (4): Leverage (5): Lambda (6): InterestFactor (7): Certainty								

Table 4: Correlation results with user ranking for using the emulators.

A) Analyzing the different user feedback emulators. Table 4 presents the results of the correlation analysis between the user-specific ranking functions (RAND-EMU, LEX-EMU, and CHI-EMU) and the seven interestingness measures, as well as the *Virtual Best Measure* (VBM), which returns the best rank correlation ρ provided by one of the seven measures. The analysis was performed on all datasets, and the results are reported in terms of Spearman’s rank correlation coefficient ρ .

The results show that the correlation between a given measure and a user-specific ranking function can vary significantly depending on the dataset. For instance, the Lambda measure is highly correlated with CHI-EMU on T40 dataset ($\rho = 98\%$), but it is weakly correlated on Connect dataset ($\rho = 29\%$). Similarly, the correlation between RAND-EMU or LEX-EMU and the measures also varies significantly across datasets. However, the VBM approach, which selects the best measure for each dataset, achieves a high level of accuracy (with a mean of 91%). These results suggest that a weighted aggregation of the selected measures could lead to a good trade-off between accuracy and robustness.

B) Comparing AHPRank.0 with RankingSVM.0. Table 5 presents the results of the k-folds cross-validation for RankingSVM.0 and AHPRank.0, where we report the averaged rank correlation ρ , recall values (R@10% and R@1%), and CPU time in seconds averaged over the folds. Since RankingSVM.0 can handle a training set of up to 100K rules within an hour, we compared it with our approach only on the Hepatitis dataset. We observed that RankingSVM.0 outperformed AHPRank.0 in terms of ranking accuracy for all user feedback emulators, although AHPRank.0 remained competitive with acceptable accuracy. The same trend was observed in terms of recall at the 10% and 1% top of the ranking (R@10% and R@1%). We noted

RAND-EMU								
	RankingSVM.0				AHPRank.0			
	ρ	R@10%	R@1%	t(s)	ρ	R@10%	R@1%	t(s)
Hepatitis	0.94	0.74	0.77	851	0.93	0.79	0.71	9
Connect	-	-	-	TO	0.95	0.72	0.63	26
Mushroom	-	-	-	TO	0.89	0.69	0.66	32
T40	-	-	-	TO	0.99	0.93	0.58	63
Retail	-	-	-	TO	0.93	0.95	0.96	66
mean	-	-	-	-	0.94	0.82	0.71	39

LEX-EMU								
	RankingSVM.0				AHPRank.0			
	ρ	R@10%	R@1%	t(s)	ρ	R@10%	R@1%	t(s)
Hepatitis	0.99	0.99	0.99	902	0.92	0.86	0.81	13
Connect	-	-	-	TO	0.60	0.46	0.50	21
Mushroom	-	-	-	TO	0.86	0.56	0.65	32
T40	-	-	-	TO	0.99	0.93	0.46	71
Retail	-	-	-	TO	0.78	0.68	0.65	68
mean	-	-	-	-	0.83	0.70	0.61	41

CHI-EMU								
	RankingSVM.0				AHPRank.0			
	ρ	R@10%	R@1%	t(s)	ρ	R@10%	R@1%	t(s)
Hepatitis	0.99	0.97	0.92	979	0.94	0.90	0.77	10
Connect	-	-	-	TO	0.15	0.28	0.67	17
Mushroom	-	-	-	TO	0.98	0.91	0.93	28
T40	-	-	-	TO	0.99	0.99	0.96	84
Retail	-	-	-	TO	0.91	0.96	0.95	50
mean	-	-	-	-	0.79	0.81	0.86	38

Table 5: 5-fold cross-validation results (passive learning).

that AHPRank.0 achieved a high correlation with the user ranking functions on most datasets. However, on Connect, we observed a weak correlation with CHI-EMU, while a high correlation was observed with RAND-EMU and LEX-EMU. This can be attributed to the fact that RAND-EMU and LEX-EMU are linear functions expressed with the given 7 interestingness measures, which explains their high accuracy, while CHI-EMU is a complex function that requires more extensive statistics to learn. Notably, the high accuracy of RankingSVM.0 came at the expense of longer running times. For instance, it took more than 15 minutes for RankingSVM.0 to learn from a training set of 100K rules, and exceeding 100K rules, RankingSVM.0 required more than one hour, while AHPRank.0 was able to handle 7.5M rules in less than 4 minutes.

In what follows, the observations and the conclusions drawn from CHI-EMU remain true for RAND-EMU and LEX-EMU. For the sake of simplicity, we only report the results on the complex function CHI-EMU.

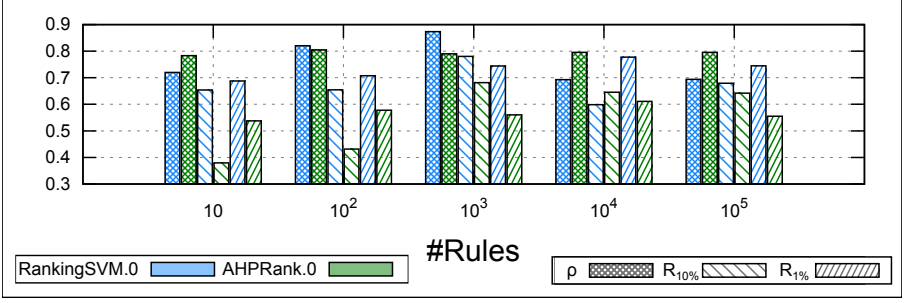


Fig. 1: Learning accuracy comparison between RankingSVM.0 and AHPRank.0 learning CHI-EMU on different training data size.

C) Impact of varying the size of the training data on the learning. It is crucial to emphasize that our proposed AHPRank approach aims to achieve a fast learning process while providing high accuracy in ranking patterns. To further support our observation on the scalability of AHPRank.0 compared to RankingSVM.0, we present in Figures 1 and 2 a performance comparison by varying the size of the training data when learning the CHI-EMU function, as similar results were obtained for RAND-EMU and LEX-EMU functions. For each dataset, we randomly select nb rules, and we compare the two approaches for $nb \in 10, 100, 1K, 10K, 100K$. The results are averaged over ten runs.

Regarding the Spearman correlation ρ , Figure 1 shows that the two approaches have a discrepancy of only 5% when the training data does not exceed $1K$. However, the gap becomes significantly more substantial (exceeding 10%) and in favor of AHPRank.0 when the training data contains $10K$ and $100K$ rules.

In terms of recall at 10% and 1%, RankingSVM.0 outperforms AHPRank.0 with a gap of 23% and 15% at $R@10\%$ and $R@1\%$, respectively, when the training data contains only 10 rules. However, as the size of the training data increases, the gap between the two approaches becomes narrower, with a difference of less than 5% for $R@10\%$, and remaining relatively constant for $R@1\%$.

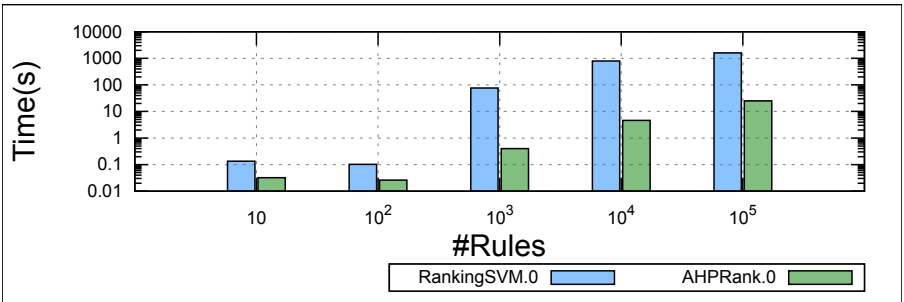


Fig. 2: CPU time comparison between RankingSVM.0 and AHPRank.0 learning using CHI-EMU emulator on different training data sizes averaged overall datasets.

In terms of CPU time, Figure 2 demonstrates that RankingSVM.0 can process training data of up to 100 rules in less than a second, whereas it takes more than one minute for 1K rules, 13minutes for 10K rules, and over 26minutes for 100K rules. In contrast, AHPRank.0 is capable of handling training data ranging from 100 to 100K rules in a time span ranging from 0.03 to 24.86 seconds, while still maintaining a ranking accuracy comparable to that of RankingSVM.

6.7 Active learning results

In this section, we aim to address research questions **RQ2** and **RQ3**, which involve comparing the active learning versions of RankingSVM and AHPRank. To achieve this, we employ a straightforward process that involves asking the user ranking queries on pairs of patterns (e.g., *do you prefer P_i to P_j ?*).

A) Evaluating the effectiveness of our sensitivity-based heuristic. In this section, we aim to evaluate the effectiveness of our Sensitivity-Based Generator (SBG) (see section 4) in improving the active learning process of AHPRank.1. To do so, we compare SBG to a Random Generator (RG), where RG selects a pair of patterns randomly from #Rules and submits them to both RankingSVM.1 and AHPRank.1. We conduct our experiments with human-in-the-loop, setting the number of queries (i.e., iterations of Algorithm 2) to a maximum of 20 queries. We repeat the experiment 10 times and take the average result to account for the sampling step of our SBG and the randomness of the RG approach. After a few tests, we set the sample size $\mathcal{X}$ picked at line 10 of algorithm 2 to $\theta = 10^3$, providing a good trade-off between time selection and the accuracy of the selected pair.

Figure 3 presents a scatter plot of 20 iterations comparing the performance of RankingSVM.1 and AHPRank.1 with both RG and SBG. We report the Spearman correlation ρ , recall at 10%, and recall at 1% for each iteration. The results show that the use of SBG outperforms RG, regardless of the learning algorithm, as can be observed from the scatter plot.

To support our observation on the effectiveness of SBG, we conducted a statistical test using the Wilcoxon Signed-Rank Test. We used a one-tailed alternative hypothesis with the null hypothesis that RG is more efficient than SBG: H_0 : The accuracy using RG $\geq$ The accuracy using SBG. The alternative hypothesis H_1 states that SBG outperforms RG. With this statistical test, we concluded that the use of SBG is more efficient than RG (H_1 accepted).

Table 6 reports the p-value, z-score, and confidence interval (CI) of each test. Except for the case of (RankingSVM.1, R@10%), we had strong evidence to reject the null hypothesis based on the p-value column. The CI column clearly shows that the use of SBG is better than RG. Consequently, we will use the SBG heuristic in RankingSVM.1 and AHPRank.1 in the following.

B) Comparing AHPRank.1 with RankingSVM.1. Table 7 presents a comparison between RankingSVM.1 and AHPRank.1 on five datasets, reporting the Spearman correlation ρ , recall at 10% and 1% stopping criterion T at 10, 50, and 100 queries, and the average latency time Time between two queries for RankingSVM.1 (never

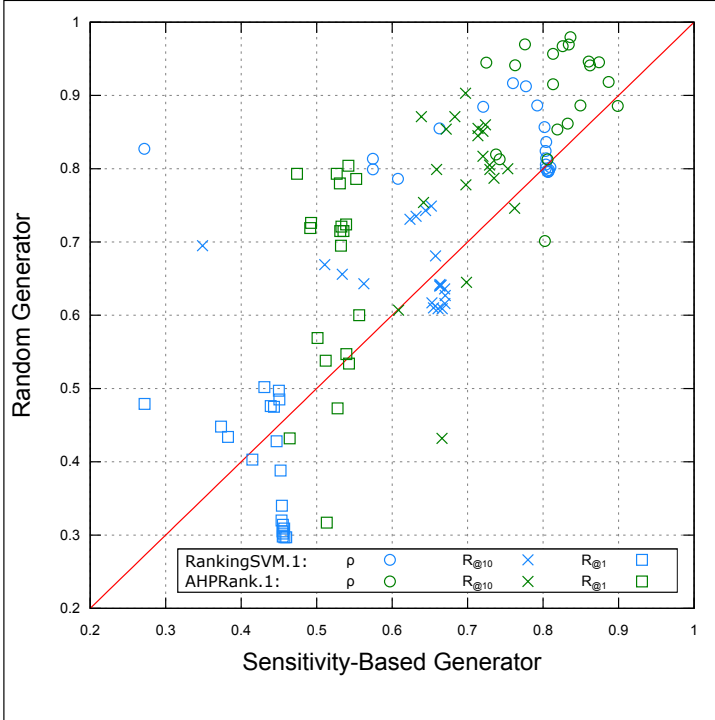


Fig. 3: Qualitative comparison of Random vs. Sensitivity-Based Query Generator using the Spearman correlation ρ and the recall at 10% and 1% of each iteration.

Approaches	Metrics	<i>p-value</i>	<i>z-score</i>	CI
RankingSVM.1	ρ	.00226	-2.8373	99%
	$R_{@10\%}$	.12302	-1.1573	70%
	$R_{@1\%}$	.06301	-1.5306	90%
AHPRank.1	ρ	.00034	-3.3973	99.9%
	$R_{@10\%}$	.00205	-2.8746	99%
	$R_{@1\%}$	.00139	-2.9866	99%

Table 6: Wilcoxon Signed-Rank Test (RG vs. SBG).

exceeding the latency bound of 0.1 seconds). The findings hold for the RAND-EMU, LEX-EMU, and CHI-EMU functions.

The main observation from Table 7 is that AHPRank.1 outperforms RankingSVM.1. Looking at the Hepatitis dataset, for example, AHPRank.1 achieves an accuracy of 78% with only 10 queries, while RankingSVM.1 achieves 73%. At 50 queries, AHPRank.1 achieves an accuracy of 95%, compared to 94% for RankingSVM.1. At 100 queries, AHPRank.1 reaches 98% accuracy, while RankingSVM.1 remains stable at 94%. In terms of recall, AHPRank.1 discovers the

		(a)	(b)	(c)	(d)	(e)
10 Queries						
(1)	ρ	0.73	0.04	0.69	0.99	0.97
	R@10%	0.61	0.28	0.38	0.92	0.91
	R@1%	0.44	0.49	0.36	0.45	0.67
	Time	0	0	0	0	0
(2)	ρ	0.78	0.10	0.93	0.99	0.99
	R@10%	0.66	0.36	0.80	0.97	0.91
	R@1%	0.51	0.50	0.71	0.81	0.93
50 Queries						
(1)	ρ	0.94	0.20	0.71	0.99	0.98
	R@10%	0.77	0.10	0.39	0.92	0.91
	R@1%	0.57	0	0.36	0.45	0.67
	Time	0	0	0	2	3
(2)	ρ	0.95	0.10	0.93	0.99	0.99
	R@10%	0.83	0.40	0.72	0.94	0.91
	R@1%	0.66	0.45	0.51	0.64	0.93
100 Queries						
(1)	ρ	0.94	0.29	0.72	0.99	0.98
	R@10%	0.75	0.20	0.41	0.92	0.91
	R@1%	0.53	0	0.36	0.45	0.67
	Time	0	2	3	3	10
(2)	ρ	0.98	0.10	0.81	0.99	0.99
	R@10%	0.89	0.35	0.49	0.95	0.91
	R@1%	0.78	0.42	0.37	0.74	0.93
(1): RankingSVM.1 (2): AHPRank.1 (a): Hepatitis (b): Connect (c): Mushroom (d): T40 (e):Retail						

Table 7: Qualitative evaluation of RankingSVM.1 vs AHPRank.1.

most relevant patterns in the first 50K and 5K patterns (out of 500K) with accuracies of 78% and 66% over 10 queries, respectively, compared to 61% and 44% for RankingSVM.1.

For the Retail dataset, AHPRank.1 achieves high accuracy over 10 queries, which remains stable over the following 90 queries. RankingSVM.1 results are less impressive on Retail and particularly on the recall metric. However, the main difference is in the waiting time Time, where RankingSVM.1 can keep the user waiting for more than 10 seconds between two queries. The same observation can be made on the other datasets, where RankingSVM.1 can be hampered by overall waiting time, even with queries of size 2. In contrast, AHPRank.1 shows an instantaneous behavior, taking less than 0.1 seconds between two queries. This represents a limitation in the use of RankingSVM.1, especially when the learning is integrated into an interactive data mining process, where a reasonable latency time for a human user is around a few seconds [Lallemand and Gronier \(2012\)](#).

6.8 Interactive learning results

In this section, we address the last research question **RQ4**, and conduct two experiments to evaluate the robustness and performance of our approach in the presence of human errors.

Our first experiment aims to evaluate the robustness of our approach in the face of human errors. To simulate situations where the user feedback may be incorrect, we randomly select a set of queries and swap the user preference with a probability Err . Specifically, a user who prefers pattern P_i over P_j will mistakenly prefer P_j over P_i with a probability Err .

Figure 4 compares the performance of RankingSVM.1 and AHPRank.1 under different levels of human error, ranging from 0% to 40%, and reports results averaged over 10 runs on the whole set of datasets using CHI-EMU and submitting 20 queries to the user. Our results show that AHPRank.1 is quite stable and robust even with high levels of human error, up to 40% (8 mistakes out of 20). The overall correlation between the learned function and the user one (ρ) remains stable at 77% without errors and drops only to 70% under $\text{Err} = 40\%$. However, RankingSVM.1 is stable only up to $\text{Err} = 20\%$, and then its accuracy drops significantly, with ρ decreasing from 71% to 54%. In terms of recall, the decline is less than 2% at $R@10\%$ and 8% at $R@1\%$ under AHPRank.1, while RankingSVM.1 experiences a decline exceeding 10% at both $R@10\%$ and $R@1\%$. In terms of CPU time, we observe waiting times ranging from 6 to 12 seconds under RankingSVM.1, whereas our approach never exceeds 0.02 seconds between two queries.

These results suggest that our approach is robust and can handle user errors, while RankingSVM.1 is less robust and can suffer from significant drops in accuracy when faced with human errors. Additionally, our approach provides faster query processing times, which is critical in interactive data mining scenarios where human users expect near-instantaneous feedback.

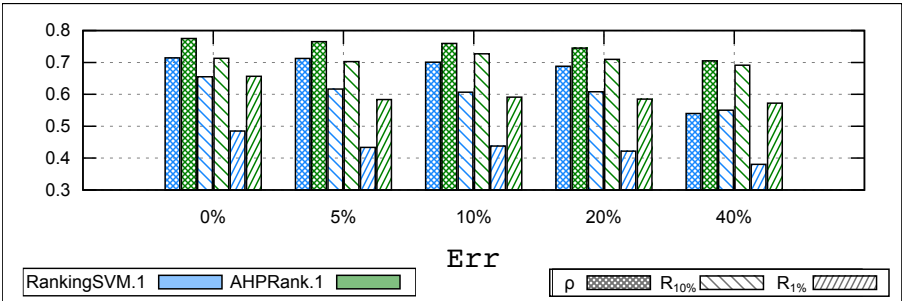


Fig. 4: Learning under mistakes (20 queries).

Our second experiment aims to evaluate the robustness of our approach when faced with an undecided user who begins with a set of preferences A but ends up with preferences B . Here, the initial user preferences A are biased towards the presented patterns until the user is more comfortable with preferences B .

To simulate such situations where the user's target function changes during the learning process, we conduct an experiment where we start with LEX-EMU as the target function to learn, and after x queries, we switch to RAND-EMU. This learning process takes $(20 - x)$ more queries, and we refer to it as the LEX2RAND target. We choose to switch from LEX-EMU to RAND-EMU because the two functions are

linear. We compare RankingSVM.1 and AHPRank.1 on LEX2RAND(x) target for a total of 20 queries, where $x \in 0, 5, 10$. Note that with $x = 0$, LEX2RAND(0) is equivalent to RAND-EMU.

Figure 5 shows the averaged results of 10 runs on all datasets over 20 queries. The main observation that we can draw is that AHPRank.1 is stable even when faced with a changing linear function to learn, even if the user’s preferences change halfway through the learning process (i.e., after 10 queries). However, changing the target function during the learning process can significantly impact the accuracy of RankingSVM.1 (a decline of 44% in terms of ρ). Regarding CPU time, our approach is 50 times faster than RankingSVM.1.

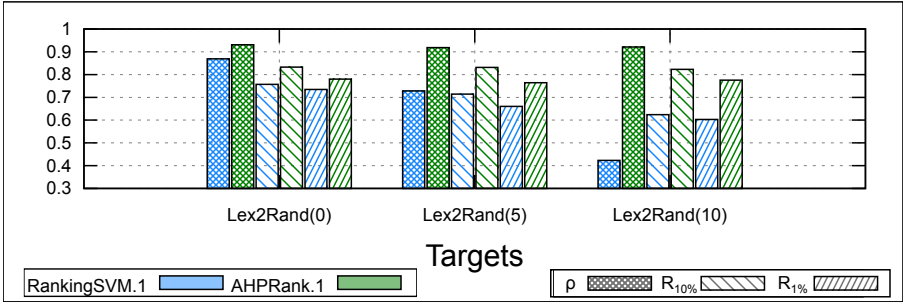


Fig. 5: Learning under swaps (20 queries)

To summarize, our research demonstrates that AHPRank provides a favorable balance between ranking accuracy and running time. With a time complexity linear to the size of the training data, it achieves a high ranking accuracy and takes only a few seconds to run, making it a practical solution for industrial use.

7 Conclusion

In this paper, we presented a novel framework for learning pattern ranking functions using the AHP multicriteria decision making method. Our proposed AHPRank algorithm can be applied in both passive and active learning modes, allowing the user to rank subsets of patterns according to their preferences. The learned weights are used to aggregate all the measures into a single ranking function using a weighted linear formulation, which was shown experimentally and statistically to be as close as possible to the user’s ranking. We applied our framework to the case study of Association Rules Mining and compared the results with state-of-the-art learning approaches. Our experiments showed that the AHPRank algorithm can efficiently learn the ranking function and outperform existing methods in terms of both ranking accuracy and running time. Overall, our proposed framework offers a practical and efficient approach for learning pattern ranking functions, with potential applications in a wide range of domains.

Acknowledgements. Funded by the European Union through the AI4CCAM project (Trustworthy AI for Connected, Cooperative Automated Mobility) under grant agreement No 10107691. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

References

- Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. *SIGMOD Rec* 22(2):207–216. <https://doi.org/10.1145/170036.170072>, URL <https://doi.org/10.1145/170036.170072>
- Ailon N (2012) An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *J Mach Learn Res* 13:137–164. URL <http://dl.acm.org/citation.cfm?id=2188390>
- Bao X, Gu T, Chang L, et al (2021) Knowledge-based interactive postmining of user-preferred co-location patterns using ontologies. *IEEE Transactions on Cybernetics* 52(9):9467–9480
- Belmecheri N, Aribi N, Lazaar N, et al (2022) Une méthode d'apprentissage par optimisation multicritère pour le rangement de motifs en fouille de données. *Revue des Nouvelles Technologies de l'Information Extraction et Gestion des Connaissances*, RNTI-E-38:289–296
- Bhuiyan MA, Al Hasan M (2016) Priime: A generic framework for interactive personalized interesting pattern discovery. In: 2016 IEEE International Conference on Big Data (Big Data), IEEE, pp 606–615
- Bie TD (2011) An information theoretic framework for data mining. In: *KDD*. ACM, pp 564–572
- Blankmeyer E (1987) Approaches to consistency adjustment. *Journal of Optimization Theory and Applications* 54:479–488
- Boley M, Mampaey M, Kang B, et al (2013) One click mining: interactive local pattern discovery through implicit preference and performance learning. In: *IDEA@KDD*. ACM, pp 27–35
- Bottou L, Lin CJ (2007) Support vector machine solvers. *Large scale kernel machines* 3(1):301–320
- Brunelli M (2014) *Introduction to the analytic hierarchy process*. Springer

- Burges C, Shaked T, Renshaw E, et al (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning, pp 89–96
- Chang L, Zhang Y, Bao X, et al (2022) Idmbs: An interactive system to find interesting co-location patterns using svm. In: Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part III, Springer, pp 518–521
- Dzyuba V, van Leeuwen M (2013a) Interactive discovery of interesting subgroup sets. In: International Symposium on Intelligent Data Analysis, Springer, pp 150–161
- Dzyuba V, van Leeuwen M (2013b) Interactive discovery of interesting subgroup sets. In: IDA, Lecture Notes in Computer Science, vol 8207. Springer, pp 150–161
- Dzyuba V, van Leeuwen M (2017) Learning what matters - sampling interesting patterns. In: PAKDD (1), pp 534–546
- Dzyuba V, van Leeuwen M, Nijssen S, et al (2014) Interactive learning of pattern rankings. *Int J Artif Intell Tools* 23(6). <https://doi.org/10.1142/S0218213014600264>, URL <https://doi.org/10.1142/S0218213014600264>
- ERKUT E, TARIMCILAR M (1991) On Sensitivity Analysis in the Analytic Hierarchy Process. *IMA Journal of Management Mathematics* 3(1):61–83. <https://doi.org/10.1093/imaman/3.1.61>, URL <https://doi.org/10.1093/imaman/3.1.61>, <https://arxiv.org/abs/https://academic.oup.com/imaman/article-pdf/3/1/61/6764589/3-1-61.pdf>
- Gass S, Rapcsák T (2004) Singular value decomposition in ahp. *European Journal of Operational Research* 154(3):573–584. [https://doi.org/https://doi.org/10.1016/S0377-2217\(02\)00755-5](https://doi.org/https://doi.org/10.1016/S0377-2217(02)00755-5), URL <https://www.sciencedirect.com/science/article/pii/S0377221702007555>
- Geng L, Hamilton HJ (2006a) Interestingness measures for data mining: A survey. *ACM Comput Surv* 38(3):9
- Geng L, Hamilton HJ (2006b) Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)* 38(3):9–es
- Imielinski T, Mannila H (1996) A database perspective on knowledge discovery. *Commun ACM* 39(11):58–64
- Kendall MG, Smith BB (1939) The problem of m rankings. *Ann Math Statist* 10(3):275–287. <https://doi.org/10.1214/aoms/1177732186>, URL <https://doi.org/10.1214/aoms/1177732186>

- Kuznetsov SO, Makhalova T (2018) On interestingness measures of formal concepts. *Information Sciences* 442:202–219
- Lallemand C, Gronier G (2012) Enhancing user experience during waiting time in hci: Contributions of cognitive psychology. In: *Proceedings of the Designing Interactive Systems Conference*. ACM, New York, NY, USA, DIS '12, pp 751–760, <https://doi.org/10.1145/2317956.2318069>, URL <http://doi.acm.org/10.1145/2317956.2318069>
- Pan VY, Chen ZQ (1999) The complexity of the matrix eigenproblem. In: Vitter JS, Larmore LL, Leighton FT (eds) *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, May 1-4, 1999, Atlanta, Georgia, USA. ACM, pp 507–516, <https://doi.org/10.1145/301250.301389>, URL <https://doi.org/10.1145/301250.301389>
- Qin T, Liu TY, Li H (2010) A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* 13:375–397
- Raedt LD (2002) A perspective on inductive databases. *SIGKDD Explor* 4(2):69–77
- Ringuest JL (1986) A chi-square statistic for validating simulation-generated responses. *Comput Oper Res* 13(4):379–385
- Saaty TL (1977) A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology* 15(3):234–281. [https://doi.org/https://doi.org/10.1016/0022-2496\(77\)90033-5](https://doi.org/https://doi.org/10.1016/0022-2496(77)90033-5), URL <https://www.sciencedirect.com/science/article/pii/0022249677900335>
- Saaty TL (1988) What is the analytic hierarchy process? In: *Mathematical models for decision support*. Springer, p 109–121
- Saaty TL, Ozdemir MS (2003) Why the magic number seven plus or minus two. *Mathematical and computer modelling* 38(3-4):233–244
- Saaty TL, Vargas LG (1984) Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios. *Mathematical Modelling* 5(5):309–324. [https://doi.org/https://doi.org/10.1016/0270-0255\(84\)90008-3](https://doi.org/https://doi.org/10.1016/0270-0255(84)90008-3), URL <https://www.sciencedirect.com/science/article/pii/0270025584900083>
- Shalev-Shwartz S, Tewari A (2011) Stochastic methods for l_1 -regularized loss minimization. *J Mach Learn Res* 12:1865–1892
- Silberschatz A, Tuzhilin A (1995) On subjective measures of interestingness in knowledge discovery. In: *KDD*. AAAI Press, pp 275–281
- Takeda E, Cogger K, Yu P (1987) Estimating criterion weights using eigenvectors: A comparative study. *European Journal of Operational Research* 29(3):360–369.

[https://doi.org/https://doi.org/10.1016/0377-2217\(87\)90249-9](https://doi.org/https://doi.org/10.1016/0377-2217(87)90249-9), URL <https://www.sciencedirect.com/science/article/pii/0377221787902499>

Tan PN, Kumar V, Srivastava J (2004) Selecting the right objective measure for association analysis. *Information Systems* 29(4):293–313

Xin D, Shen X, Mei Q, et al (2006) Discovering interesting patterns through user's interactive feedback. In: *KDD*. ACM, pp 773–778