Active Preference Learning for Ranking Patterns

Vladimir Dzyuba, Matthijs van Leeuwen, Siegfried Nijssen, and Luc De Raedt
Department of Computer Science, KU Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

***Abstract*—Pattern mining provides useful tools for exploratory data analysis. Numerous efficient algorithms exist that are able to discover various types of patterns in large datasets. However, the problem of identifying patterns that are genuinely interesting to a particular user remains challenging. Current approaches generally require considerable data mining expertise or effort and hence cannot be used by typical domain experts.**

We show that it is possible to resolve this issue by interactive learning of user-specific pattern ranking functions, where a user ranks small sets of patterns and a general ranking function is inferred from this feedback by *preference learning* techniques. We present a general framework for learning pattern ranking functions and propose a number of active learning heuristics that aim at minimizing the required user effort. In particular we focus on Subgroup Discovery, a specific pattern mining task.

We evaluate the capacity of the algorithm to learn a ranking of a subgroup set defined by a complex quality measure, given only reasonably small sample rankings. Experiments demonstrate that preference learning has the capacity to learn accurate rankings and that active learning heuristics help reduce the required user effort. Moreover, using learned ranking functions as search heuristics allows discovering subgroups of substantially higher quality than those in the given set. This shows that active preference learning is potentially an important building block of interactive pattern mining systems.

***Keywords*—preference learning; active learning; pattern mining**

I. Introduction

Pattern mining is an important concept in data mining. Informally, a pattern is a statement in a certain language that concisely describes an *interesting* subset of the entire dataset. Patterns essentially provide summaries of regions in the data in the form of comprehensible descriptions. Many variations of pattern mining have been proposed and many algorithms exist that are able to efficiently mine patterns from large datasets. For example, Subgroup Discovery concerns patterns in labelled data. In the context of a bank providing loans, the fact that 16% of loans with $purpose = used\ car$ are not repaid (whereas the proportion for the entire dataset is 5%) is an interesting pattern.

However, the adoption of pattern mining by domain experts is still limited. *Pattern explosion* is a common problem: large collections of patterns are returned, and a domain expert has to invest substantial effort to identify those patterns that are relevant to her specific interests or goals. Solutions such as manual filtering of results or tuning algorithm parameters are simply inaccessible to domain experts. Top-k pattern mining is commonly proposed as a solution, where a smaller set of top patterns with respect to a certain interestingness measure is mined. However, this approach has a number of inherent problems as well. *Objective* interestingness measures only concern the structure of the data and do not take into account knowledge and goals of a user. As a result, re-discovery of common knowledge is a typical issue. *Subjective* interestingness measures account for the user-specific context via a model of the dataset or of the entire domain. An expert has to be familiar with a particular model type, e.g. Bayesian networks. This shifts the problem of the user to specifying the right model, which is yet another non-trivial task.

In this paper we study the challenging problem of identifying a subjective interestingness measure with minimal effort from the user. We rely on the following assumptions: 1) a user has an implicit preference between any pair of patterns, which does not change over time; 2) the costs of eliciting the entire preference relation, i.e. expressing it in an analytical form, are prohibitively high; 3) however, for any pair of patterns, the user can accurately identify which of the two she prefers, i.e. considers subjectively more interesting than the other.

To this end, we propose a generic algorithm for the interactive learning of pattern ranking functions. The algorithm receives a set of patterns as input, elicits user feedback, and infers a ranking function from the feedback. Providing feedback requires a user to rank small sets of patterns according to their perceived interestingness. The main building blocks are a preference learning algorithm and an active learning component. Preference learning infers a pattern ranking function by generalizing from sample pattern rankings provided by the user. The goal of active learning is to select small sets of patterns (*queries*) that are shown to the user so that the amount of input required to learn an accurate approximation of the target ranking is minimized. We propose a number of query selection heuristics to achieve this. The learned ranking function is essentially a subjective interestingness measure.

We discuss the application of the proposed approach in the context of Subgroup Discovery. Existing algorithms require the a priori specification of an objective subgroup quality measure. However, even for pattern mining experts it is often unclear which one works best in a specific context. The proposed approach starts from a set of top subgroups according to an arbitrary objective measure and learns a subjective measure tailored to a specific context. The learned measure can then be used to mine novel interesting subgroups.

In order to perform an extensive evaluation, we emulate user preferences over subgroups using χ^2 (*Chi-squared*), a well-known subgroup quality measure. Experiments show that the algorithm is able to learn accurate pattern rankings and that query selection heuristics help reduce the amount of input required for learning. Moreover, learned ranking functions generalize well and allow discovering novel high-quality subgroups when plugged into a standard Subgroup Discovery algorithm.

II. Related work

Pattern mining is a well-known data mining task aimed at exploratory data analysis [1]. The importance of taking user knowledge and goals into account in order to discover genuinely interesting patterns was first emphasized by Tuzhilin [2]. However, most works considering user-specific pattern quality measures and interactive pattern mining are quite recent. One group of approaches relies on specifying a certain background knowledge model in advance, such as a Bayesian network [3] or a Maximum Entropy distribution [4]. The other group employs user-provided feedback to guide the pattern search, e.g. to update a pattern sampling distribution [5]; see Kontonasios et al. [6] for a recent overview. These approaches typically assume specific notions of interestingness and do not allow learning general subjective interestingness measures.

We focus on a specific pattern mining task, Subgroup Discovery [7], an instance of *supervised descriptive rule discovery* [8]. The proposed methods for interactive Subgroup Discovery include iterative refinement of constraints on subgroup descriptions [9] and interactive beam search based on reweighing subgroup quality measures [10]. An approach based on active learning of subjective quality measures has been recently proposed by Boley et al. [11].

Preference learning is an umbrella term that encompasses several loosely related tasks [12]. In this paper we deal with *object ranking*, i.e. acquiring ranking functions from sample orders [13]. The RANKING SVM [14] is a well-known object ranking algorithm that we employ.

Active object ranking is related to the problem of *learning to rank* in information retrieval. A number of general heuristics aimed at improving top results of search engines were developed [15][16]. Methods that specifically target object ranking algorithms exploit probabilistic models of a document collection [17] or relations between documents [18]. A theoretical analysis of query complexity of active object ranking has been presented recently [19].

Similar to our work, the RANKING SVM has been used to identify interesting patterns in an interactive manner. Xin et al. [20] investigate learning a user-specific ranking of frequent patterns (primarily itemsets and sequences). A clustering-based method similar to information retrieval approaches is used to select sample patterns for feedback. However, they only consider a specific learning target based on the discrepancy between the expected and observed support of a pattern. Furthermore, they do not consider using learned ranking functions to search for novel patterns.

The work of Rueping [21] demonstrated the feasibility of learning subgroup rankings and applying learned ranking functions to discover high-quality subgroups. However, Rueping does not discuss active learning aspects and uses a custom variant of the learner and data modifications that are specific to Subgroup Discovery. Therefore, it cannot be straightforwardly generalized to other pattern mining tasks.

III. Preliminaries

Pattern mining The pattern mining task is formally defined as follows [22]. Given a dataset $\mathcal{D}$, a language $\mathcal{L}$ defining subsets of $\mathcal{D}$ (for example, logical formulae over domains of attributes), and a selection predicate φ that determines whether an element $p \in \mathcal{L}$ describes an interesting subset of $\mathcal{D}$, the task is to find descriptions of all interesting subsets $\{p \in \mathcal{L} \mid \varphi(p, \mathcal{D}) \text{ is true}\}$. Therefore, a pattern consists of a description p and the subset of $\mathcal{D}$ defined by this description. In this paper we focus on one particular pattern mining setting, Subgroup Discovery.

Subgroup Discovery is a supervised pattern mining task concerned with finding subsets of a dataset that have a substantial deviation in a property of interest as compared to the entire dataset, with a strong emphasis on obtaining comprehensible descriptions. Formally it is defined as follows. Let $A = \{A_1, \ldots, A_{l-1}, A_l\}$ denote a set of attributes, where each attribute A_j has a domain of possible values $\mathrm{Dom}(A_j)$. Then a dataset $\mathcal{D} = \{t_1, \ldots, t_n\} \subseteq \mathrm{Dom}(A_1) \times \ldots \times \mathrm{Dom}(A_l)$ is a bag of tuples over A. The attribute A_l is a *binary target attribute*, i.e. the property of interest, while the other attributes are *description attributes*. A *subgroup description* p is a conjunction of boolean atoms over description attributes, e.g. $A_1 = a \wedge A_2 > 0$. A *subgroup cover* G is a bag of tuples that satisfy the predicate defined by p: $G_p = \{t \in \mathcal{D} \mid p(t) \text{ is true}\}$. The size of the cover $|G|$ is also called *subgroup coverage*. A *quality measure* φ is a function that quantifies the interestingness of a subgroup. Let G^c (resp. $\mathcal{D}^c$) denote the set of tuples from a class c in the subgroup cover (resp. in the entire dataset). Examples of subgroup quality measures include $Sensitivity(G) = \frac{|G^+|}{|\mathcal{D}^+|}$, $Specificity(G) = 1 - \frac{|G^-|}{|\mathcal{D}^-|}$, and

$$\chi^2(G) = \sum_{c \in \{-,+\}} \frac{(|G|(|G^c| - |\mathcal{D}^c|))^2}{|G||\mathcal{D}^c|} + \frac{(|G|(|G^c| - |\mathcal{D}^c|))^2}{(|\mathcal{D}| - |G|)|\mathcal{D}^c|}$$

The Subgroup Discovery problem is defined as follows: given a dataset $\mathcal{D}$, a quality measure φ, and an integer k, find the set of k highest-quality subgroups. For example, the dataset *credit-g* (see Section VI) contains information about 1000 loans. The target attribute indicates whether a loan has been repaid (positive class) or not. The entire dataset contains 700 positive tuples. The subgroup G with a description $checking\ status = no\ checking$ covers 394 instances, out of which 348 are positive. It has the highest value of χ^2 among all subgroups with one atom in the description: $\chi^2(G) = 103.96$.

Object ranking The object ranking task is formally defined as follows. Let $\mathcal{X}$ denote the universal set of all possible objects. Each object $\vec{x} \in \mathcal{X}$ is represented by a feature vector $[x_1, \ldots, x_m]$. A ranking f is a total strict order over a subset of $\mathcal{X}$: $f = \vec{x}_{f_1} \succ \ldots \succ \vec{x}_{f_n}$, where $\succ$ is a precedence relation. Object ranking is a structured prediction task that given a set of rankings $F = \{f_1, \ldots, f_N\}$ learns a function R that accurately ranks any subset of $\mathcal{X}$, including objects not seen in F. The learning bias is that objects with similar feature values are ranked close to each other.

We restrict ourselves to the RANKING SVM [14], an SVM-based object ranking algorithm, which was initially developed in the context of learning document ranking functions in information retrieval. It is based on *pairwise* preferences, i.e. rankings are treated as a set of corresponding ranked pairs. For example, $\vec{x}_1 \succ \vec{x}_2 \succ \vec{x}_3$ corresponds to $\{(\vec{x}_1 \succ \vec{x}_2), (\vec{x}_1 \succ \vec{x}_3), (\vec{x}_2 \succ \vec{x}_3)\}$.

A simplified problem statement is as follows: given a set of ranked queries F that is transformed into the set of equivalent pairwise preferences $\{(\vec{x}_{ik} \succ \vec{x}_{jk}) \in f_k \mid f_k \in F\}$, learn feature weights $\vec{w}$ that minimize the number of incorrectly ordered pairs:

$$\begin{aligned} \text{minimize: } V(\vec{w}, \xi) = \frac{1}{2}\ \vec{w} \cdot \vec{w} + C\ \sum \xi_{ijk} \\ \text{subject to: } \vec{w} \cdot (\vec{x}_{ik} - \vec{x}_{jk}) \geq 1 - \xi_{ijk} \\ \xi_{ijk} \geq 0 \end{aligned}$$

As is common to SVM-based methods, ξ_{ijk} are slack variables, and C is a trade-off parameter. The problem is essentially equivalent to a standard classification SVM formulation for pairwise difference vectors.

Learned weights $\vec{w}$ define a ranking function R that can be used to rank any subset of $\mathcal{X}$: $R(x) = \sum_{\vec{x}_i^* \in SV} \alpha_i^* \vec{x} \cdot \vec{x}_i^*$, where $\vec{x}_i^*$ are the support vectors and α_i^* are the Lagrange multipliers. Although it is possible to use kernels to learn non-linear ranking functions, linear ranking functions have several advantages, such as computational efficiency and interpretability (via studying absolute values of feature weights).

IV. Learning pattern rankings

A. Problem definition

We rely on the following assumption to define the problem: for a given dataset $\mathcal{D}$ and a pattern language $\mathcal{L}$, there exists a ranking $\mathcal{R}^*$ of all patterns in $\mathcal{L}$ according to their subjective interestingness for the current user, i.e. $p_i \succ p_j$ implies that the user considers the pattern p_i more interesting than p_j. We also assume that $\mathcal{R}^*$ is consistent with the structure in $\mathcal{D}$ and $\mathcal{L}$, i.e. that relative positions of patterns are determined by their observable properties.

Obviously, the user cannot generate the entire ranking $\mathcal{R}^*$ explicitly. However, she can rank reasonably small sets of patterns according to their relative interestingness. The problem is then defined as follows: given a dataset $\mathcal{D}$ and a set of sample pattern rankings F fully consistent with $\mathcal{R}^*$, learn a ranking function R over patterns in $\mathcal{L}$ so that the learned ranking is maximally consistent with $\mathcal{R}^*$. The number of sample rankings that the user has to explicitly provide, i.e. user effort, has to be minimized. We rely on preference learning to fulfill the former requirement and on active learning to fulfill the latter. We discuss ranking consistency measures in Section VI. User effort is measured by the number of elicited pairwise preference judgements $(p_i \succ p_j) \in F$.

B. Algorithm outline

In this section we present an algorithm for learning a pattern ranking function (Algorithm 1). It receives a collection of patterns $\mathcal{P} \subset \mathcal{L}$ as input. A standard pattern mining algorithm can be used to mine the input collection. It is initially ranked according to an objective measure. In practice, any measure can be used, e.g. *coverage*. This ranking is referred to as the *source ranking*.

At each iteration a subset of $\mathcal{P}$ is shown to the user and ranked by her according to the subjective interestingness of the patterns. Ranked sets of patterns are used as training data for learning a ranking function. The learned ranking function can

Algorithm 1 Active Preference Learning for Pattern Ranking

```
Input: Dataset $\mathcal{D}$, ranked collection of patterns $\mathcal{P}$
Output: Ranking function $R$ for patterns over $\mathcal{D}$
1: $F = \emptyset$, $R = SourceRanking(\mathcal{P})$
2: $PV = ConvertToVectors(\mathcal{P}, \mathcal{D})$
3: repeat
4:     $q = SelectQuery(PV, R)$
5:     $F = F \cup GetFeedback(q)$
6:     $R = LearnRankingFunction(PV, F)$
7: until Stopping criterion is met
8: return $R$
```

then be used to rank the input collection as well as to score unobserved patterns.

Pattern representation (Line 2) In order to apply preference learning, patterns are represented as vectors of numeric features. Pattern features have to capture properties that make patterns interesting to a user. Various feature sets for subgroups are shown in Figure 1; we discuss them in detail in Section V.

Active learning (Line 4) Query selection methods select sets of patterns that will be shown to the user. Assuming that the query size is fixed, the goal is to minimize the number of queries required to attain a certain ranking accuracy. The methods take into account such factors as the current estimated quality of a pattern, the estimation uncertainty, the diversity of the query, or the structure of the data.

Feedback format (Line 5) A user provides her feedback in the form of rankings. For example, if the query is $\{p_1, p_2, p_3\}$, the feedback $p_3 \succ p_1 \succ p_2$ implies that p_3 is the most interesting pattern, and p_2 is the least interesting pattern.

This feedback format is computationally more expensive for a user than *graded feedback*, i.e. assigning scores from a predefined scale. However, we argue that it has two advantages relevant to the iterative setting. First, it requires neither a deep understanding of the scale by a user, nor a thorough scale calibration. Second, graded feedback can be converted to the ordered format, albeit at a cost of reduced granularity.

Learning rankings (Line 6) Ranked queries, i.e. sets of patterns that are ranked by the user according to their subjective interestingness, are used as training data for an object ranking algorithm. It learns a ranking function R that returns a number for any possible feature vector. Any pattern can be scored using R, therefore the learned ranking function is essentially a general subjective interestingness measure.

Stopping criteria (Line 7) Stopping criteria can consider marginal effects of additional queries on the learned ranking or limit the maximal user effort. In the simplest case, the user manually stops the algorithm, as soon as she considers her information need satisfied.

C. Active learning

Active preference learning is a challenging problem. Selecting an optimal query is NP-hard [19], therefore in most cases exact query selection methods are computationally too expensive to be used in a truly interactive setting. Consequently, heuristic methods are commonly used.

	A_1	A_2	A_t
t_1	T	T	T
t_2	T	F	T
t_3	F	F	F

(a) Toy dataset

$A_1 = T$

Attributes		Cover			Coverage	Pos.cov.	Neg.cov.	Obj.quality (χ^2)	Length
A_1	A_2	t_1	t_2	t_3					
1.0	0.0	1.0	1.0	0.0	2.0	2.0	0.0	6.0	1.0

(b) Vector representation of the subgroup $A_1 = T$

Fig. 1: A toy example of representing a subgroup as a numerical vector.

Query selection methods balance exploration of the pattern space with exploitation of available preference feedback. In the context of pattern mining, the source ranking is a strong starting point. The common method to ensure sufficient exploration is to maintain diversity among queried objects. We consider a number of heuristics that can be categorized into three groups: quality-based greedy heuristics inspired by methods from information retrieval (IR), uncertainty-based heuristics specific to the RANKING SVM learner, and randomized methods. Several heuristics explicitly take objective quality measures into account.

IR-inspired heuristics IR-inspired heuristics were initially developed in the context of improving search engines, hence they inherently aim at identifying a small number of top-ranking objects (*documents*). These greedy heuristics rely on the availability of an objective quality measure (*relevance*). The query selection process always starts from a set including the currently top-ranked pattern and proceeds with greedily selecting patterns that maximize the heuristic. Let p denote a candidate pattern, and q the current (incomplete) query.

When applying these heuristics, we start from the raw values of the source pattern quality measure φ, and progressively interpolate the values of the learned ranking function in order to take into account the current estimation of the target ranking:

$$Quality(p) = \mu\ R(p) + (1-\mu)\ \varphi(p)$$

where μ is an interpolation parameter and R is the learned ranking function.

MMR (*Maximal Marginal Relevance*) [15] aims to select a high-quality pattern that is dissimilar from already selected patterns. Dissimilarity is defined as the minimal distance to an already selected pattern, e.g. Euclidean distance between pattern vectors. The parameter $\alpha \in [0;1]$ is a quality-diversity trade-off parameter.

$$MMR(p,q) = \alpha\ Quality(p) + (1-\alpha)\ Diversity(p,q)$$
$$\text{where } Diversity(p,q) = \min_{p' \in q} dist(p,p')$$

RDD (Active learning to achieve *Relevance*, *Diversity*, and *Density*) [16] exploits the structure in $\mathcal{P}$ by adding a density term. The intuition behind this approach is that querying patterns from dense regions provides more information about preferences. Density of a region around a pattern is quantified as the average distance to all other patterns.

$$RDD(p,q) = \alpha\ Quality(p) + \beta\ Density(p,\mathcal{P}) + $$
$$+(1-\alpha-\beta)\ Diversity(p,q)$$
$$\text{where } Density(p,\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{p' \in \mathcal{P}} dist(p,p')$$

MMR and RDD only maintain local diversity, i.e. diversity *within the current query*. We aim to exploit global diversity, i.e. diversity *between the queries*, by introducing a new heuristic GLOBALMMR. It is an extension of MMR, where the diversity term is redefined as $Diversity(p,Q) = \min_{p' \in Q} dist(p,p)$, where $Q = q \cup \bigcup_{f_i \in F} f_i$ is the union of all queries, including the current incomplete one.

In all computations, values of the quality measure, the learned ranking function, and the distance measure are normalized to the range $[0;1]$. For the quality measure and the learned ranking function, the minimal and the maximal values over $\mathcal{P}$ are used as range limits. For distance, the upper limit is estimated by the diameter of the object set, i.e. $\max_{p_i,p_j \in \mathcal{P}} dist(p_i,p_j)$.

Uncertainty-based heuristics SVMBATCH (Algorithm 2) is a straightforward extension of the batch query selection method for classification SVMs by Brinker [23]. This method aims at selecting a diverse set of examples with high prediction uncertainty. Uncertainty is quantified as the distance of a candidate example to the margin, whereas diversity is quantified by maximal cosine similarity between an example and already selected examples. This method only considers examples that lie on or within the margins.

In case of pairwise preferences, an individual example is a *pair* of patterns. Pairs are explicitly represented as differences between respective pattern vectors, similar to their representation in the RANKING SVM formulation. The total number of candidates is proportional to $|\mathcal{P}|^2$, therefore in order to reduce computational costs we introduce an additional pruning step. All pair vectors P_{ij} for which $||P_{ij}|| < minlen$ are removed from the candidate set. The intuition behind this pruning technique is that pair vectors with low norms correspond to highly similar patterns, and reducing uncertainty of predicting relative positions of similar patterns is less useful for learning a general ranking. Note that the distance between a pair vector P_{ij} and the hyperplane is proportional to the difference between values of the ranking function for p_i and p_j. However, the exact value has to be computed explicitly.

Our preliminary experiments confirmed the utility of batch querying, e.g. querying the union of the two most informative pairs yields a larger performance improvement than three consecutive queries of the single most informative pair (in both cases 6 pairs are queried). Pruning reduces the runtime and does not have any negative impact on learning performance.

Randomized heuristics Our preliminary experiments have shown that even non-biased uniform sampling of subgroups from $\mathcal{P}$ results in reasonably high learning performance, hence

Algorithm 2 Batch query selection for RANKING SVM

```
Input: Pattern vectors $PV$, weights $w$, query size $k$, trade-off $\lambda$, pruning parameter $minlen$
 1: $q \leftarrow \emptyset$, $Pairs \leftarrow \emptyset$
 2: for all $p_i, p_j \in PV$ do                ▷ Generate candidate pairs
 3:     $P_{ij} = p_i - p_j$
 4:     if $dist(P_{ij}, w) \leq 1 \wedge ||P_{ij}|| \geq minlen$ then
 5:         $Pairs \leftarrow Pairs \cup P_{ij}$
 6: repeat
 7:     $P^*_{ij} = \underset{P \in Pairs}{\operatorname{argmin}} \; \lambda \times dist(P,w) + (1-\lambda) \times \max_{q_i, q_j \in q} \cos(P, Q_{ij})$
 8:     $q \leftarrow q \cup \{p^*_i, p^*_j\}$
 9: until $|q| < k$
10: return $q$
```

we decided to further explore randomized query selection methods that sample subgroups proportional to values of IR-inspired heuristics. The overall procedure is as follows:

1) For each pattern a sampling weight w is computed, e.g. $w(p) = MMR(p, q)$.
2) If the minimal weight w_{min} is equal to 0, a Laplace-like correction is applied, i.e $\frac{1}{|\mathcal{P}|}$ is added to all weights.
3) A random number drawn uniformly from the range $[0; \sum_{p \in \mathcal{P}} w(p)]$ determines the sampled pattern.

Hence, the probability of sampling a pattern p is $\frac{w(p)}{\sum_{p' \in \mathcal{P}} w(p')}$.

V. Application to Subgroup Discovery

The learned ranking function generalizes beyond the training data F and the input pattern set $\mathcal{P}$. Hence, it is essentially a general subjective interestingness measure defined over $\mathcal{L}$. It can be used to discover novel patterns that are likely to be interesting to the user. We employ the proposed approach to learn a subjective subgroup quality measure.

First, a set of subgroups $\mathcal{P}$ is mined from the dataset $\mathcal{D}$ by a standard Subgroup Discovery algorithm, using any objective quality measure φ, e.g *Sensitivity* or *Specificity*. The ranking of $\mathcal{P}$ by φ is the *source ranking*.

Then Algorithm 1 is used to learn a user-specific ranking function R. We consider the following features for the representation of subgroups (see Figure 1). Binary feature sets *attributes* and *cover* contain features that indicate whether a description attribute is present or absent in the subgroup description and whether a tuple is covered by the subgroup. Numeric features include *coverage* $|G|$, *coverage on positives* $|G^+|$ and *negatives* $|G^-|$, objective *quality* (value of φ), and *length* of the description. In our experiments, the numeric features are discretized into a number of bins, based on the distribution of values in $\mathcal{P}$. Values of φ are used in query selection heuristics, when necessary.

In order to use the learned ranking function R in search, we extend a beam search–based Subgroup Discovery algorithm, DSSD [24]. Essentially, R is used as a search heuristic φ. Whenever the values of R have to be computed, e.g. during beam selection, subgroups are converted into vectors using the same representation that was used during learning.

VI. Experiments

In previous sections we described a framework for interactive learning of pattern ranking functions. Here we present a set of experiments that aim to answer the following questions:

Q1) Is it possible to learn preferences over patterns, given only sample rankings as input? If yes, how much training data is required?
Q2) Which pattern features are important for learning?
Q3) Does active learning reduce the user effort? Which query selection methods perform better with respect to various performance measures?
Q4) Do the learned ranking functions enable the discovery of novel interesting patterns when used as search heuristics?

A. Evaluation methodology

User feedback emulation Evaluating interactive data mining algorithms is hard, for experts are scarce, and it is virtually impossible to collect enough data for drawing reliable conclusions. In order to perform an extensive evaluation we use an objective ranking of subgroups as the target ranking. We emulate user feedback by ranking subgroups using an objective subgroup quality measure (*target measure*), which is not known to the learning algorithm.

We use χ^2 as the target measure. χ^2 balances subgroup coverage and difference between class frequencies in the subgroup cover and the entire dataset, hence learning the ranking defined by χ^2 is sufficiently complex, i.e. it does not amount to identifying trivial correlations between features.

Performance measures The goal of learning rankings is two-fold: 1) to identify interesting subgroups in $\mathcal{P}$ and 2) to learn an accurate overall ranking of $\mathcal{P}$. Therefore, we use several ranking distance measures to quantify learning performance. Let $\mathcal{R}^*_{\mathcal{P}}$ denote the target ranking of $\mathcal{P}$, $\hat{\mathcal{R}}_{\mathcal{P}}$ the learned ranking, and $\hat{\mathcal{R}}_{\mathcal{P}}(i)$ the learned rank of the i-th element in the target ranking:

1) In order to evaluate the capacity of the algorithm to identify the most interesting patterns in $\mathcal{P}$, we consider Recall at k:

$$Rec_k = \left| \left\{ i \in \{1, 2, \ldots, k\} \mid \hat{\mathcal{R}}_{\mathcal{P}}(i) \leq k \right\} \right|$$

2) In order to evaluate the overall ranking accuracy, we consider rank correlation and discounted error. Spearman's rank correlation coefficient ρ is based on the sum of squared differences between learned and target ranks for each element:

$$\rho = 1 - \frac{6 \; D_s(\mathcal{R}^*_{\mathcal{P}}, \hat{\mathcal{R}}_{\mathcal{P}})}{|\mathcal{P}|(|\mathcal{P}| - 1)}, \text{ where } D_s = \sum (i - \hat{\mathcal{R}}_{\mathcal{P}}(i))^2$$

Rank correlation essentially assigns equal weights to all elements, whereas Discounted Error DE assigns larger weights to higher-ranked elements:

$$DE = \sum \frac{|i - \hat{\mathcal{R}}_{\mathcal{P}}(i)|}{ln(i + 1)}$$

Performance measures calculated for the entire ranking, such as ρ or DE, are less relevant if the ultimate goal is to identify

TABLE I: Datasets and subgroup sets used in experiments. For each dataset, three sets of 1000 subgroups were mined using *Coverage*, *Sensitivity*, or *Specificity* as the subgroup quality measure. ρ_0 is the correlation between the source ranking and the target ranking.

Dataset	Size	Attr.	Source rankings, ρ_0		
			Coverage	Sensitivity	Specificity
breast-w [bw]	683	9	0.26	0.61	0.02
credit-a [ca]	653	15	−0.26	−0.06	0.51
credit-g [cg]	1000	20	0.11	0.33	0.86
diabetes [d]	768	8	−0.01	0.17	0.43
ionosphere [io]	351	34	0.19	0.21	0.29
vote [v]	232	16	0.33	0.84	0.51

top-ranking patterns. However, if the goal is to learn a search heuristic, the capacity to correctly identify low-quality patterns is important as well. Note that reported values of DE are normalized to the range of $[0, 1]$.

In order to estimate the convergence rate of the algorithm, for each performance measure we report values of the *area under performance curve* (AUC) in addition to absolute values. The performance curves are constructed as follows: for each iteration i, the value of a performance measure after i iterations is recorded. The larger the area, the fewer iterations are required to attain high values of the performance measure.

To quantify user effort, we consider two cost measures. The total number of distinct queried pairs C_F serves as an estimate of the required user effort: $C_F = |\{(p_{ik}, p_{jk}) \mid f_k \in F; p_{ik}, p_{jk} \in f_k\}|$. C_F is equal to the number of pairwise preferences that a user has to compute in order to provide the feedback. The second measure, the average target rank of queried subgroups $\mathcal{R}^*_{avg}$, indicates whether the query itself includes interesting subgroups.

Data For our empirical evaluation we used datasets from the UCI repository[1]. Tuples with missing attribute values were removed from all datasets.

Input subgroup sets were mined using DSSD with the following parameters (see [24] for details): minimal coverage $= 0.1\ |\mathcal{D}|$, beam width $= 100$, maximal depth $= 5$. Numeric attributes were discretized on-the-fly by local binning of occurring values into 6 equal-sized bins. The cover-based beam selection heuristic was applied with the default trade-off parameter settings. 10000 subgroups were mined initially, then 1000 subgroups were selected from this large set using the same selection heuristic.

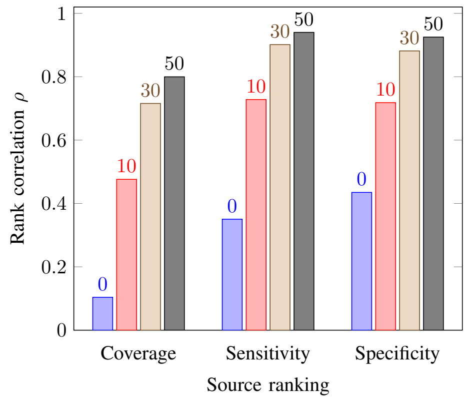
For each dataset three subgroup sets were mined using one of the following subgroup quality measures, *Sensitivity*, *Specificity*, or *Coverage* (essentially a non-supervised quality measure). We have intentionally chosen simple quality measures so that source and target rankings are substantially different. Table I presents the characteristics of the datasets and corresponding subgroup sets.

In the following experiments we use the standard implementation of the RANKING SVM[2]. The trade-off parameter is initially set to $C_0 = 0.005$. Per recommendations of the

[1]http://archive.ics.uci.edu/ml/

[2]http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Fig. 2: Rank correlation with the target ranking. Relatively small queries suffice to learn accurate rankings: querying 10 elements allows attaining $\rho \geq 0.7$ for *Sensitivity* and *Specificity* subgroup sets and $\rho \geq 0.5$ for *Coverage* subgroup sets.



authors, it is increased after each iteration, i.e. the effective value is $C_0 \times iteration$.

B. Experimental results

Q1) Learning pattern rankings In order to verify the feasibility of learning pattern ranking functions from sample rankings, we conducted experiments with uniformly random query selection. For each source ranking, a random subset of size S is generated, ordered by χ^2, and used as the training data. This procedure is repeated 10 times; the average obtained rank correlation ρ is reported.

Figure 2 shows the results for $S \in \{0, 10, 30, 50\}$, where $S = 0$ corresponds to the correlation between the source ranking and the target ranking. Subgroup sets are grouped by the source quality measure, and results are aggregated over all datasets. The results confirm that it is possible to learn accurate ranking functions using only sample rankings. Moreover, this requires a reasonable amount of training data: high values of ranking correlation, $\rho \geq 0.7$, are attained for $S \leq 30$.

Q2) Pattern features In order to evaluate the importance of various feature sets we performed the following procedure. Similar to the previous experiments, random subsets of $\mathcal{P}$ are used as the training data. For each selection of training data, we incrementally construct the subgroup representation. At each step the feature set that results in the largest ρ is added to the representation. The procedure continues as long as ρ increases.

We consider all feature sets described in Section V. All numeric features are discretized into 5 bins. The size of the training data is 30 subgroups. For each subgroup set, the training data selection procedure was performed 10 times.

Results are shown in Table II. Features that are related to χ^2 are more likely to be selected at the first iteration as well as included in the best feature set. In all following experiments we use feature sets that consist of *cover*, *attributes*, *positive* and *negative coverage*.

Q3) Query selection We now present the comparison of query selection strategies. We quantify performance by average ranks. For each source ranking, various query selectors

TABLE II: Subgroup features. Fractions show how often a feature set was chosen at the first iteration, average ρ attained with this feature set only, and how often this feature set was included in the best feature combination.

Feature set	**First feature set** Likelihood	ρ	**Best feature set** Likelihood
Pos.coverage	0.16	0.54	0.75
Cover	0.65	0.81	0.66
Neg.coverage	0.15	0.58	0.65
Quality	0.00	0.32	0.41
Support	0.00	0.59	0.37
Attributes	0.03	0.47	0.29
Length	0.00	0.23	0.25

were evaluated and ranked according to AUC for respective performance measures. Tied ranks are assigned the highest rank from the equivalent range. Finally, ranks for a specific query selector are averaged over all subgroup sets.

We compare the following strategies: MMR($\alpha = 0.3$), RDD($\alpha = 0.1$, $\beta = 0.2$), GLOBALMMR($\alpha = 0.1$), and their randomized counterparts; SVMBATCH($\lambda = 0.1$, $minlen = 0.1$), where $minlen = 0.1$ denotes pruning all candidate pairs with the norm less than 0.1 of the maximal norm of a binary vector of the same dimensionality; and a non-biased randomized strategy RAND(UNIFORM). Parameter values were selected based on preliminary experiments and are in general biased towards higher query diversity. For IR-inspired query selectors, the interpolation coefficient was set to $\mu = 0.7$, and the Euclidean distance measure was used. To compute the ranks of randomized selectors, 10 experiments were conducted, and median values of performance measures were used.

All experiments were conducted with $\text{number of iterations} = 10$ and $\text{query size} = 5$. The maximal effort is then $C_F = 10 \times \binom{5}{2} = 100$. A single query of 15 subgroups has approximately equivalent costs, $C_F = \binom{15}{2} = 105$, therefore we report the median performance over 10 experiments with RAND(UNIFORM) and query size $= 15$ as the baseline.

Table III presents the results regarding the performance of query selectors. In line with their original design goals, IR-inspired selectors ensure the highest Rec_{10}, whereas their randomized counterparts do not have this property. The differences between query selectors with respect to Rec_{100} are not as pronounced, however IR-inspired selectors converge to high values of recall faster. Moreover, the performance of all query selectors is substantially higher than the baseline.

On the other hand, global query diversity is required for learning accurate overall rankings. Randomized selectors that essentially ensure the largest degree of diversity result in the highest values of ρ; this holds for the non-biased selector as well. However, introducing bias into randomized selectors proves to be beneficial, as biased randomized selectors result in consistently high performance in terms of DE, unlike RAND(UNIFORM) that ranks low with respect to DE. SVMBATCH is the only deterministic query selector whose performance is comparable to randomized methods in terms of ρ and DE. The performance of IR-inspired selectors is lower than the baseline, albeit still relatively high.

TABLE III: Performance of query selectors. IR-inspired query selectors ensure the highest Rec_{10}, whereas methods that maintain global diversity result in accurate overall rankings.

Selector	Avg.AUC rank / Avg.value after 10 iterations			
	Rec_{10}	Rec_{100}	ρ	DE
RDD(0.1, 0.2)	2.2 / 0.47	3.2 / 0.63	4.5 / 0.70	4.7 / 0.29
MMR(0.3)	2.6 / 0.54	4.7 / 0.64	6.6 / 0.68	6.4 / 0.30
GLOBALMMR(0.1)	2.8 / 0.54	4.1 / 0.62	5.1 / 0.73	5.3 / 0.28
SVMBATCH(0.1, 0.1)	4.6 / 0.43	4.4 / 0.66	4.3 / 0.80	4.6 / 0.22
RAND(RDD(0.1, 0.2))	5.5 / 0.35	5.1 / 0.68	3.8 / 0.81	3.6 / 0.21
RAND(UNIFORM)	5.6 / 0.32	5.3 / 0.67	3.9 / 0.81	4.8 / 0.22
RAND(MMR(0.3))	5.6 / 0.34	4.7 / 0.69	4.1 / 0.82	3.3 / 0.21
RAND(GLOBALMMR(0.1))	6.0 / 0.38	4.6 / 0.68	3.7 / 0.83	3.3 / 0.21
RAND(UNIFORM), $S = 15$	0.20	0.56	0.78	0.26

TABLE IV: Costs of query selectors. IR-inspired query selectors only maintain local query diversity and consequently have the lowest costs. Introducing bias in query selection results in queries that contain subgroups of higher quality.

Selector	C_F Distinct queried pairs	$\mathcal{R}^*_{avg}$ Avg.target rank in query
RDD(0.1, 0.2)	1.50 / 58.22	4.4 / 227.6
MMR(0.3)	1.50 / 60.56	3.2 / 186.5
RAND(MMR(0.3))	3.17 / 99.89	3.8 / 226.1
SVMBATCH(0.1, 0.1)	3.17 / 99.89	4.6 / 228.4
RAND(RDD(0.1, 0.2))	3.22 / 99.94	4.8 / 231.8
GLOBALMMR(0.1)	3.28 / 100.00	3.9 / 225.0
RAND(GLOBALMMR(0.1))	3.28 / 100.00	5.5 / 240.2
RAND(UNIFORM)	3.28 / 100.00	5.7 / 239.4

Table IV shows the values of cost measures. MMR and RDD only maintain local query diversity, hence there is a substantial overlap between queries, which results in the substantially lower effort in terms of C_F. All selectors that aim to maintain global query diversity, especially randomized selectors, incur much higher costs. In addition, queries selected by IR-inspired methods (as well as most randomized counterparts) are characterized by high target ranks of selected subgroups. The non-biased randomized selector has the lowest average rank with respect to this measure.

Q4) Using learned functions in search Finally, we evaluate the capacity of learned ranking functions to generalize to unobserved subgroups. We use query size $= 5$ for learning. Search parameters were identical to the parameters used for mining the source rankings.

Figure 3 presents a detailed view of experimental results for the dataset *credit-g*, the source ranking *cg-Specificity*, and 10 iterations of learning. Boxplots show the distribution of χ^2 in the resulting subgroup sets. The results confirm that learned ranking functions have the capacity to identify subgroups with substantially higher quality. In all cases the maximal value of χ^2 is practically twice as large as in the source ranking.

Table V summarizes the results of experiments with all 18 source rankings and the SVMBATCH query selector. The results confirm the generalization capacity of learned ranking functions: median and maximal values of χ^2 increase substantially compared to source rankings. Furthermore, learning more accurate rankings increases the scale of improvement.

Fig. 3: Distribution of values of χ^2 in the subgroup set *credit-g-Specificity* and in top-1000 subgroups discovered in *credit-g* using learned ranking functions as well as using χ^2 directly.

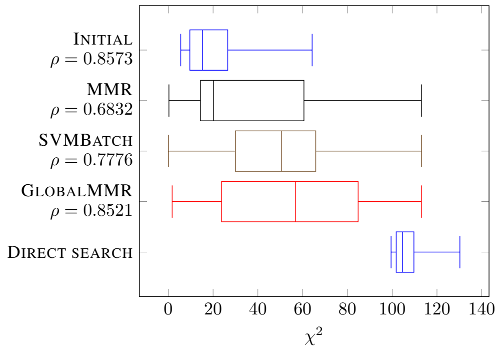


TABLE V: Using learned ranking functions in search allows discovering high-quality subgroups. ΔQ_k is a median ratio between the kth percentile of χ^2 values in top-1000 discovered subgroups and in the source subgroup set. Learning accurate overall rankings improves quality of discovered subgroups.

Selector	Iter.	ρ_{avg}	DE_{avg}	$\Delta_{Q_{25}}$	$\Delta_{Q_{50}}$	$\Delta_{Q_{100}}$
	1	0.34	0.52	0.45	0.79	0.99
SVMBATCH	5	0.68	0.33	2.10	1.68	1.05
	10	0.80	0.22	2.21	2.52	1.08

VII. Conclusions

We presented a general framework for interactive learning of pattern ranking functions. It requires a user to rank sets of patterns by their perceived interestingness and uses preference learning to infer a general ranking function from this input. An active learning component is used to minimize user effort.

We applied this framework to Subgroup Discovery, a supervised pattern mining task. Using a well-principled evaluation method, we demonstrated that it is possible to learn complex preferences over sets of subgroups using an off-the-shelf preference learning algorithm. Learned ranking functions generalize well and enable the discovery of novel high-quality subgroups when used as search heuristics. Experiments with active learning heuristics showed a trade-off between accuracy of learned rankings and user effort.

Directions for future work include investigating the effect of coarse-grained or noisy feedback on learning performance, learning preferences over sets of patterns instead of individual patterns, and shifting from the pool-based active learning to *query synthesis*, i.e. directly mining patterns for queries. A user study is required to evaluate the practical applicability of the proposed framework.

Acknowledgements This work was supported by the Research Foundation–Flanders by means of a Postdoc grant and the project "Instant Interactive Data Exploration" and by the European Commission under the project "Inductive Constraint Programming", contract number FP7-284715. Matthijs van Leeuwen is supported by a Rubicon grant of the Netherlands Organisation for Scientific Research (NWO).

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, *Advances in Knowledge Discovery and Data Mining*, 1996, ch. Fast Discovery of Association Rules, pp. 307–328.

[2] A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery," in *Proceedings of KDD*, 1995, pp. 275–281.

[3] S. Jaroszewicz and D. A. Simovici, "Interestingness of frequent itemsets using Bayesian networks as background knowledge," in *Proceedings of KDD*, 2004, pp. 178–186.

[4] T. De Bie, "An Information Theoretic Framework for Data Mining," in *Proceedings of KDD*, 2011, pp. 564–572.

[5] M. Bhuiyan, S. Mukhopadhyay, and M. A. Hasan, "Interactive Pattern Mining on Hidden Data: A Sampling-based Solution," in *Proceedings of CIKM*, 2012, pp. 95–104.

[6] K.-N. Kontonasios, E. Spyropoulou, and T. De Bie, "Knowledge discovery interestingness measures based on unexpectedness," *WIREs: Data Mining and Knowledge Discovery*, vol. 2, no. 5, pp. 386–399, 2012.

[7] W. Klösgen, *Advances in Knowledge Discovery and Data Mining*, 1996, ch. Explora: A Multipattern and Multistrategy Discovery Assistant, pp. 249–271.

[8] P. Kralj Novak, N. Lavrač, and G. Webb, "Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining," *Journal of Machine Learning Research*, vol. 10, pp. 377–403, 2009.

[9] M. Atzmueller, "Exploiting background knowledge for knowledge-intensive subgroup discovery," in *Proceedings of IJCAI*, 2005, pp. 647–652.

[10] V. Dzyuba and M. van Leeuwen, "Interactive Discovery of Interesting Subgroup Sets," in *Proceedings of IDA'2013*, in press.

[11] M. Boley, M. Mampaey, B. Kang, P. Tokmakov, and S. Wrobel, "One Click Mining — Interactive Local Pattern Discovery through Implicit Preference and Performance Learning," in *Interactive Data Exploration and Analytics Workshop at KDD*, 2013, pp. 28–36.

[12] E. Hüllermeier and J. Fürnkranz, Eds., *Preference learning*. Springer Berlin Heidelberg, 2011.

[13] T. Kamishima, H. Kazawa, and S. Akaho, "A Survey and Empirical Comparison of Object Ranking Methods," in *Preference Learning*, J. Fürnkranz and E. Hüllermeier, Eds. Springer Berlin Heidelberg, 2011, ch. III, pp. 181–202.

[14] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of KDD*, 2002, pp. 133–142.

[15] X. Shen and C. Zhai, "Active feedback in ad hoc information retrieval," in *Proceedings of SIGIR*, 2005, pp. 59–66.

[16] Z. Xu, R. Akella, and Y. Zhang, "Incorporating Diversity and Density in Active Learning for Relevance Feedback," in *Proceedings of ECIR*, 2007, pp. 246–257.

[17] F. Radlinski and T. Joachims, "Active exploration for learning rankings from clickthrough data," in *Proceedings of KDD*, 2007, pp. 570–579.

[18] Z. Xu, K. Kersting, and T. Joachims, "Fast Active Exploration for Link-Based Preference Learning Using Gaussian Processes," in *Proceedings of ECML/PKDD*, 2010, pp. 499–514.

[19] N. Ailon, "An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity," *Journal of Machine Learning Research*, vol. 13, pp. 137–164, 2012.

[20] D. Xin, X. Shen, Q. Mei, and J. Han, "Discovering Interesting Patterns Through Users Interactive Feedback," in *Proceedings of KDD*, 2006, pp. 773–778.

[21] S. Rueping, "Ranking interesting subgroups," in *Proceedings of ICML*, 2009, pp. 913–920.

[22] H. Mannila and H. Toivonen, "Multiple uses of frequent sets and condensed representations," in *Proceedings of the KDD*, 1996, pp. 189–194.

[23] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proceedings of ICML*, 2003, pp. 59–66.

[24] M. van Leeuwen and A. Knobbe, "Diverse subgroup set discovery," *Data Mining and Knowledge Discovery*, vol. 25, no. 2, pp. 208–242, Jun. 2012.