One Click Mining—Interactive Local Pattern Discovery through Implicit Preference and Performance Learning[*]

Mario Boley, Michael Mampaey, Bo Kang, Pavel Tokmakov, and Stefan Wrobel
Fraunhofer IAIS and University of Bonn
Schloss Birlinghoven, Sankt Augustin, Germany
{mario.boley,stefan.wrobel}@iais.fhg.de
{michael.mampaey,bo.kang,pavel.tokmakov}@uni-bonn.de

ABSTRACT

It is known that productive pattern discovery from data has to interactively involve the user as directly as possible. State-of-the-art toolboxes require the specification of sophisticated workflows with an explicit selection of a data mining method, all its required parameters, and a corresponding algorithm. This hinders the desired rapid interaction—especially with users that are experts of the data domain rather than data mining experts. In this paper, we present a fundamentally new approach towards user involvement that relies exclusively on the implicit feedback available from the natural analysis behavior of the user, and at the same time allows the user to work with a multitude of pattern classes and discovery algorithms simultaneously without even knowing the details of each algorithm. To achieve this goal, we are relying on a recently proposed co-active learning model and a special feature representation of patterns to arrive at an adaptively tuned user interestingness model. At the same time, we propose an adaptive time-allocation strategy to distribute computation time among a set of underlying mining algorithms. We describe the technical details of our approach, present the user interface for gathering implicit feedback, and provide preliminary evaluation results.

1. INTRODUCTION

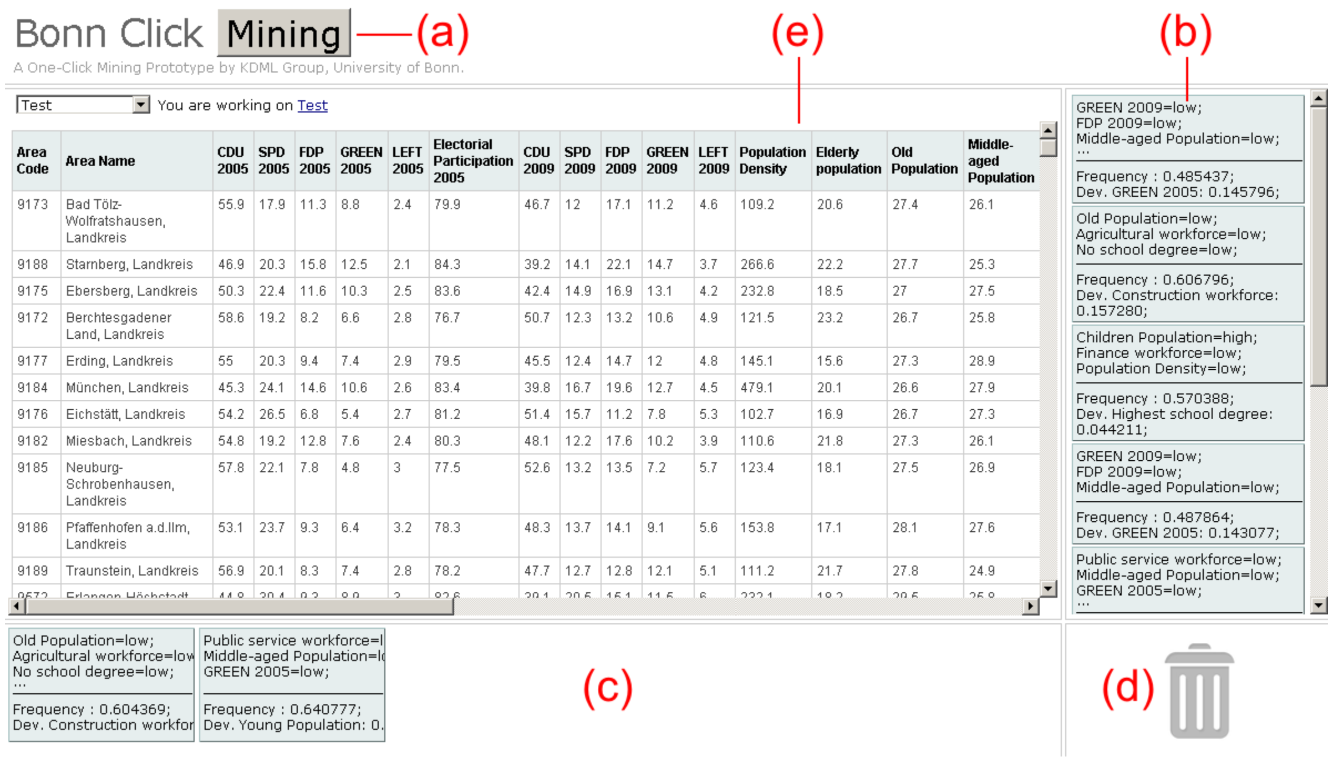
Productive pattern discovery from data is known to be an iterative process that ideally requires a tight interaction between a discovery system and a user who is an expert of the data domain [Fayyad et al., 1996]. State of the-art data analysis suites (e.g., Rapid Miner [Mierswa, 2009], WEKA [Hall et al., 2009], KNIME [Berthold et al., 2008]) rely on an explicit construction of a discovery workflow including selection of a discovery method along with its parameters, a corresponding mining algorithms, and post-processing of results. The resulting high number of alternative formalizations of a single analysis task poses a substantial burden on creating and iteratively refining these workflows—especially for users that lack deep technical understanding of data mining methods—and ultimately it hinders the desired rapid interaction [Cao, 2012]. Previous research tried to alleviate this problem by assisting the user in constructing or selecting a workflow (see Morik and Scholz [2004] or Bernstein et al. [2005]) or by providing direct active user-involvement for individual parts of the workflow such as for the post-processing [Xin et al., 2006] or even the mining itself [Goethals et al., 2011]. However, all these approaches still expose the user to the complexity of the discovery worklow and/or require technical knowledge about its components that goes way beyond the semantic knowledge about the data domain.

In this paper, we present a fundamentally new approach towards user involvement that for the first time requires neither an explicit formalization of analysis goals in terms of a workflow or another specification language nor any technical data mining knowledge that goes beyond the pure data semantics. Instead, the approach relies exclusively on the implicit feedback available from the natural analysis behavior of the user when he investigates the data and mining results. At the same time, the approach allows the user to work with a multitude of pattern classes and mining algorithms simultaneously without even knowing the details of each algorithm. In particular, the approach avoids all method selection and configuration steps from standard processes, and instead an individual mining step is started by pressing once a single dedicated mine button. Hence we refer to this process as *one-click mining.*

Naturally, the goal of this process is to produce patterns that are relevant to the latent user interest as fast as possible. We show how this goal can be achieved by the interplay of two appropriately designed online learning/optimization components. On the one side, there is model of the hidden user interest based an a suitably designed feature representation of all pattern types that are included in the range of the analysis system. The learning of the corresponding model parameters is based on the recently proposed co-active learning model [Shivaswamy and Joachims, 2012, Raman et al., 2012]. On the other side, there is a time-allocation strategy that distributes the computational time-budget available in each discovery round among a set of underlying mining algorithms. We model this task as a multi-armed bandit explo-

[*]This article is a short version containing only preliminary experimental results. Complete evaluation will be available in a future full version.

IDEA'13, August 11th, 2013, Chicago, IL, USA.
Copyright 2013 ACM 978-1-4503-2329-1 ...$15.00.



Bonn Click Mining

A One-Click Mining Prototype by KDML Group, University of Bonn.

Test You are working on Test

Area Code	Area Name	CDU 2005	SPD 2005	FDP 2005	GREEN 2005	LEFT 2005	Electorial Participation 2005	CDU 2009	SPD 2009	FDP 2009	GREEN 2009	LEFT 2009	Population Density	Elderly population	Old Population	Middle-aged Population
9173	Bad Tölz-Wolfratshausen, Landkreis	55.9	17.9	11.3	8.8	2.4	79.9	46.7	12	17.1	11.2	4.6	109.2	20.6	27.4	26.1
9188	Starnberg, Landkreis	46.9	20.3	15.8	12.5	2.1	84.3	39.2	14.1	22.1	14.7	3.7	266.6	22.2	27.7	25.3
9175	Ebersberg, Landkreis	50.3	22.4	11.6	10.3	2.5	83.6	42.4	14.9	16.9	13.1	4.2	232.8	18.5	27	27.5
9172	Berchtesgadener Land, Landkreis	58.6	19.2	8.2	6.6	2.8	76.7	50.7	12.3	13.2	10.6	4.9	121.5	23.2	26.7	25.8
9177	Erding, Landkreis	55	20.3	9.4	7.4	2.9	79.5	45.5	12.4	14.7	12	4.8	145.1	15.6	27.3	28.9
9184	München, Landkreis	45.3	24.1	14.6	10.6	2.6	83.4	39.8	16.7	19.6	12.7	4.5	479.1	20.1	26.6	27.9
9176	Eichstätt, Landkreis	54.2	26.5	6.8	5.4	2.7	81.2	51.4	15.7	11.2	7.8	5.3	102.7	16.9	26.7	27.3
9182	Miesbach, Landkreis	54.8	19.2	12.8	7.6	2.4	80.3	48.1	12.2	17.6	10.2	3.9	110.6	21.8	27.3	26.1
9185	Neuburg-Schrobenhausen, Landkreis	57.8	22.1	7.8	4.8	3	77.5	52.6	13.2	13.5	7.2	5.7	123.4	18.1	27.5	26.9
9186	Pfaffenhofen a.d.Ilm, Landkreis	53.1	23.7	9.3	6.4	3.2	78.3	48.3	13.7	14.1	9.1	5.6	153.8	17.1	28.1	27.6
9189	Traunstein, Landkreis	56.9	20.1	8.3	7.4	2.8	78.2	47.7	12.7	12.8	12.1	5.1	111.2	21.7	27.8	24.9

GREEN 2009=low;
FDP 2009=low;
Middle-aged Population=low;
...

Frequency : 0.485437;
Dev. GREEN 2005: 0.145796;

Old Population=low;
Agricultural workforce=low;
No school degree=low;

Frequency : 0.606796;
Dev. Construction workforce: 0.157280;

Children Population=high;
Finance workforce=low;
Population Density=low;

Frequency : 0.570388;
Dev. Highest school degree: 0.044211;

GREEN 2009=low;
FDP 2009=low;
Middle-aged Population=low;

Frequency : 0.487864;
Dev. GREEN 2005: 0.143077;

Public service workforce=low;
Middle-aged Population=low;
GREEN 2005=low;
...

Old Population=low;
Agricultural workforce=low
No school degree=low;
...

Frequency : 0.604369;
Dev. Construction workfor

Public service workforce=l
Middle-aged Population=l
GREEN 2005=low;

Frequency : 0.640777;
Dev. Young Population: 0.

Figure 1: Visual layout of one-click mining prototype that contains (a) mine-button, (b) result candidate area, (c) result analysis board, trash can (d), and data view (e).

ration/exploitation problem where the payoffs correspond to the utility of the mined patterns. Since this utility is measured by an evolving approximation of the user interest, we address this problem by a bandit algorithm suitable for shifting payoffs [Cesa-Bianchi and Lugosi, 2006]. Overall, we end up with a very general method that can aggregate any combination of data mining tasks, for which results can be mined by parameter-free anytime algorithms and be represented in a suitable joint feature space.

After recapping basic pattern discovery concepts from a unifying perspective (in Sec. 2), we present in detail the one click mining framework along with visual components that support its user/system dialog (Sec. 3). We then give a proof of concept based on an exemplary instantiation that combines subgroup discovery and association discovery (Sec. 4). The resulting prototypical one-click mining system, called *Bonn click mining* (see Fig. 1), is demonstrated in the context of a socio-economic data analysis task. We document that the system is able to quickly provide interesting patterns corresponding to latent analysis goals that have been learned implicitly.

2. PATTERN DISCOVERY

In this section we provide and repeat general formal definitions for pattern discovery starting from pattern classes (or languages), over scoring functions that assess pattern interestingness, up to mining algorithms that aim to find interesting patterns. As illustrative examples, we recall subgroup discovery [Klösgen, 1996] and association discovery [Webb, 2011], which are also used in our one-click mining prototype.

As notational convention, throughout this paper we denote by $[n]$ for a positive integer $n \in \mathbb{N}$ the set $\{1, \ldots, n\}$, and by $\mathbb{B}$ the set of truth values $\{\text{true}, \text{false}\}$.

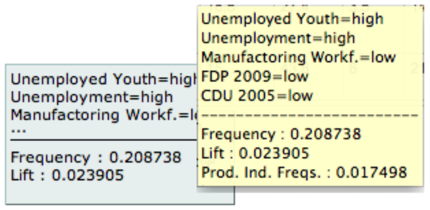
2.1 Pattern Languages

Standard approaches to local pattern discovery that are fully automatized usually rely on a specific pattern (descriptor) language along with a single choice of a measure for assessing the utility or interestingness of a descriptor. In contrast, one-click mining can aggregate a mixture of different pattern discovery methods and interestingness measures. Hence, we have to introduce an explicit notion of pattern that combines a descriptor with the information of why it is supposed to be interesting (i.e., wrt what measure).

We assume a given fixed **dataset** $D = \{d_1, \ldots, d_m\}$ of m **data records** $d \in D$, each of which is described by a set of n **attributes** $A = \{a_1, \ldots, a_n\}$. All attributes a_i assign to each data record a value from their **attribute domain** V_i, i.e., $a_i : D \to V_i$. Here, we assume that attributes are either **numerical**, i.e., $V_i \subseteq \mathbb{R}$ and we use $\leq$ to compare attribute values, or **categorical**, i.e., $|V_i|$ is finite and its values are incomparable. A **pattern language** $\mathcal{L}$ is a set of **pattern descriptors** $s \in \mathcal{L}$ to each of which we can associate a local **extension** $D(s) \subseteq D$ in the data.

For example, in **association discovery**, where one aims to find attribute/value combinations that show a high co-occurrence in the data, one usually considers the language $\mathcal{L}_{\text{cnj}}$ of **conjunctions** of constraints on individual attribute values. That is, $\mathcal{L}_{\text{cnj}}$ contains descriptors s of the form

$$s = c_{i_1}(\cdot) \wedge \cdots \wedge c_{i_s}(\cdot)$$



Unemployed Youth=high
Unemployment=high
Manufactoring Workf.=low
FDP 2009=low
CDU 2005=low

Frequency : 0.208738
Lift : 0.023905
Prod. Ind. Freqs. : 0.017498

Figure 2: Association pattern with a descriptor containing five attribute constraints and a rationale that contains two elementary interestingness functions and one function derived from them.

such that $c_{i_j} : V_{i_j} \to \mathbb{B}$ is a binary constraint on attribute a_{i_j} for all $j \in [i_s]$. Correspondingly, the extension of s is defined as

$$D(s) = \{d \in D \colon c_{i_1}(a_{i_1}(d)) \wedge \cdots \wedge c_{i_k}(a_{i_k}(d))\} \ .$$

In particular, for a categorical attribute a_i we consider **equality constraints** $c(v) \equiv v = b$ with $b \in V_i$, and for numerical attributes we consider **interval constraints** $c(v) \equiv v \in [l, u]$ with $l, u \in V_i$. We refer to $\mathrm{cnst}(s) = \{c_{i_1}, \ldots, c_{i_s}\}$ as the constraints and to $\mathrm{attr}(s) = \{a_{i_1}, \ldots, a_{i_s}\}$ as the attributes **contained** in s, respectively. We assume that each attribute is contained at most once and call $|\mathrm{cnst}(s)| = |\mathrm{attr}(s)| = i_s \in \mathbb{N}$ the **size** of s.

A further example of a pattern language is provided by **subgroup discovery** where one is interested in pattern descriptors that are at the same time fairly general (have a large extension) and that show an unusual distribution of one specific **target attribute** $a_t \in A$ in their extension. The corresponding pattern language is $\mathcal{L}_{\mathrm{sgd}} = \mathcal{L}_{cnj} \times [n]$, i.e., it contains descriptors $s = (c, t)$ with a conjunctive descriptor c annotated by the index t of a target attribute. The conjunction c also defines the extension of the subgroup descriptor s, i.e., $D(s) = D(c)$.

2.2 Interestingness and Patterns

In order to assess how interesting a pattern descriptor is as an observation in the data, there exists a wide range of **interestingness functions** $f : \mathcal{L} \to \mathbb{R}_+$ that have been developed across different pattern discovery methods (see Geng and Hamilton [2006]). A basic example is the **frequency** $f_{\mathrm{frq}}(s)$ of a pattern descriptor s, which is defined as its generality measured as the fraction of data records that are part of the pattern's extension, i.e.,$f_{\mathrm{frq}}(s) = |D(s)| \,/\, |D|$. Another general example is the **relative shortness** of a pattern defined by $f_{\mathrm{sho}}(s) = (n - |\mathrm{attr}(s)|)/n$.

Specifically for **subgroup discovery interestingness**, corresponding to its semantics, one typically uses functions of the form

$$f^b_{\mathrm{sgd}}(s, t) = f_{\mathrm{frq}}(s)^b f_{\mathrm{dv}}(s, t) \ , \qquad (1)$$

i.e., a multiplicative combinations of frequency (weighted by a real-valued parameter $b \in [0, 1]$) and a **target deviation function** f_{dv}. In this work, we choose f_{dv} to be the total variation distance between the distribution of the target attribute in the pattern extension $S = s(D)$ and the distribution in the complete data, i.e.,

$$f_{\mathrm{dv}}(s, t) = \sup_{X \subseteq V_t} |\mathbf{p}^t_S(X) - \mathbf{p}^t_D(X)| \ .$$

Here, $\mathbf{p}^t_S$ and $\mathbf{p}^t_D$ are a (fitted) distribution of attribute values of a_t in the pattern extension and the complete data, respectively (see Appendix A for computational details). This function provides a uniform interpretation of interestingness for categorical and numerical target variables.

Association interestingness is usually quantified as the difference between the frequency of the pattern and its expected frequency if we assume that some of its parts are satisfied independently. Here, we use a first order approximation to the leverage measure [Webb, 2010] that can be computed efficiently. That is, we consider the following **additive lift** measure defined by

$$f_{\mathrm{lft}}(s) = \left(f_{\mathrm{frq}}(s) - \prod_{c \in \mathrm{cnst}(s)} f_{\mathrm{frq}}(c) \right) / 2^{|\mathrm{cnst}(s)| - 2} \ .$$

Thus, conceptually this measure assumes as null hypothesis that all individual constraints of the descriptor are satisfied independently.

A **pattern** as a pair $(s, F) \in \mathcal{L} \times 2^{\mathcal{F}}$ where $s \in \mathcal{L}$ is a descriptor and $F \subseteq \mathcal{F}$ a rationale consisting of one *or more* interestingness measures with an appropriate domain. As we will see below, it is useful to potentially have more than one function in the interestingness rationale, because standard measures often are a function of several elementary functions—like in subgroup discovery interestingness, which is a function of frequency and target deviation. By giving feedback on a pattern annotated by elementary measures, a user implicitly provides insight into his preferences about all other measures that can be computed from these elementary measures. Fig. 2 shows an example of a pattern as displayed in our one-click mining prototype. By $\mathcal{P} = \mathcal{L} \times 2^{\mathcal{F}}$ we denote the **set of patterns** defined by $\mathcal{L}$ and $\mathcal{F}$.

2.3 Mining Algorithms

For the actual generation of patterns, let us denote by $\mathcal{M}$ a set of k **mining algorithms** that is at our disposal. For notational convenience we sometimes identify $\mathcal{M} = [k]$. From our perspective an individual mining algorithm $m \in \mathcal{M}$ can simply be treated as producing a random set of result patterns $m(t) \subseteq \mathcal{P}$ with an unknown distribution that depends on the time $t \in R_+$ that the algorithm is running. Of course, we usually know more about a given mining algorithm such as the pattern language it uses and the interestingness measure it optimizes. Yet from the perspective of one click mining it is sufficient to treat the algorithms as black boxes as long as they satisfy the following two requirements.

We assume that all algorithms are *parameter-free*. In practice this can mean that a single algorithm either uses a specific parameter assignment of a mining algorithm or it is in fact a meta-algorithm that includes a parameter-selection procedure. Moreover, all algorithms should be *anytime algorithms*, i.e., conceptually at every moment in time after they are started they maintain a current solution that can be retrieved when the algorithm is terminated preemptively. This is necessary, because in one click mining the time budget available for a given run of an algorithm is determined by the user ad hoc. These requirements can (or are automatically) satisfied by a wide range of modern pattern discovery algorithms that provide various pattern types. Examples are Slim [Smets and Vreeken, 2012], pattern sampling [Boley et al., 2012], or beam search approaches [van Leeuwen and Knobbe, 2011].

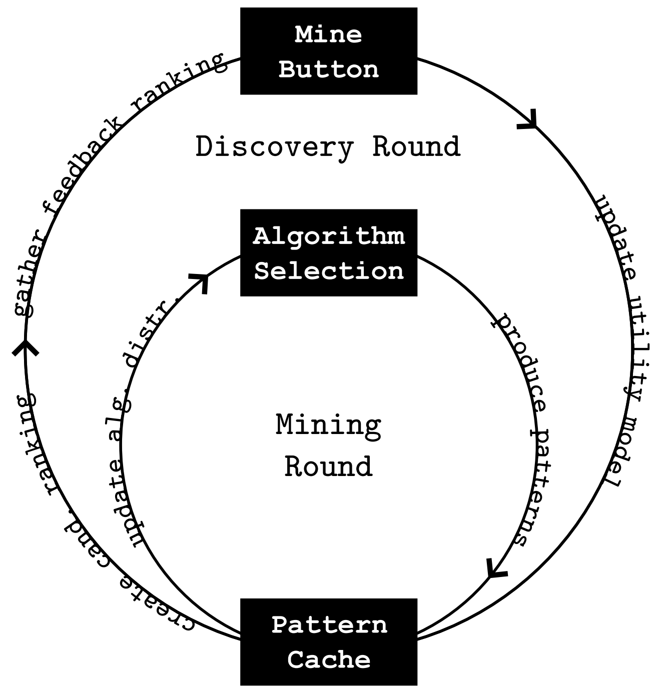


Figure 3: Temporal structure of discovery process: one discovery round can contain several mining rounds.

3. ONE-CLICK MINING

We now show how different pattern discovery methods and algorithms can be combined within a one-click mining system. After outlining the general process dynamics and the visual components that support the user/system interaction, we describe in detail the two online learning components involved in that process: the ranking mechanism of the mined results and the exploitation/exploration strategy for controlling the underlying mining algorithms.

3.1 Discovery Process and Visual Elements

The key idea of one-click mining is to not occupy the user with the technicalities of the mining workflow, but instead to let her focus on investigating the produced results on a semantic level. In this subsection we present visual elements that support this investigation and show how the interaction with these elements can be incorporated into a sensible discovery process that produces useful patterns.

The most characteristic visual element is the **mine button** (Fig. 1 (a)), which the user presses in order to see new mining results. Implicitly, pressing this button also indicates that the user is done with inspecting the results that were displayed before. Moreover, there is a result **candidate area** (Fig. 1 (b)), which is used to present ranked mining results. From here the user can investigate results, delete those that she considers useless by drawing them to the **trash can** (Fig. 1 (c)), and move those which she wishes to use for further analysis to the result **analysis board** (Fig. 1 (d)). These elements support an interactive *discovery process*, which is defined as follows (see Fig. 3).

The main temporal unit of the process are **discovery rounds** $t \in \mathbb{N}$ that correspond to the period between two consecutive activations of the mine button. As further, more refined, temporal structure, during each discovery round there is a number of **mining rounds** $l \in \mathbb{N}$, each of which corresponding to a background execution of a mining algorithm. Here, the last mining round within a given discovery round is terminated preemptively in order to end synchronously with this discovery round. Hence, every mining round can be associated to a unique discovery round. We count mining rounds consecutively and denote by $t(l)$ the discovery round in which mining round l occurs and conversely by $l(t)$ the first mining round within the discovery round t.

An individual mining round l consists of first selecting a mining algorithm m_l at random according to a **distribution for algorithm selection** $\boldsymbol{\pi}_l \colon \mathcal{M} \to [0,1]$, running m_l and fetching its result patterns P_l. All mining results are then stored in a **pattern cache**, for which we denote by $C_l \subseteq \mathcal{P}$ the state before the results of mining round l are added. The cache has a finite **cache capacity** $c \in N$ such that at all times l it is enforced that $|C_l| \leq c$. Finally, the performance of m_l is assessed by comparing C_l and C_{l+1} (using a current approximation of the pattern utility defined below). Based on this information the selection distribution for the next round $\boldsymbol{\pi}_{l+1}$ is determined and automatically started.

In the beginning of a discovery round t, a **candidate ranking** r_t of size c is constructed from the current state of the pattern candidate cache $C_{l(t)}$ and displayed in the result candidate area. Formally, a **ranking** of patterns is an ordered list $r = \langle r_1, \ldots, r_k \rangle \in \mathcal{P}^*$ such that $r_i \neq r_j$ for all $i, j \in [k]$. Let us denote by $\{r\} = \{r_1, \ldots, r_k\}$ the (unordered) set of patterns contained in the ranking, by $r^i = \langle r_1, \ldots, r_i \rangle$ the i-**prefix** of r for $i \leq k$, and by $|r| = |\{r\}| = k$ the size of r. The **set of rankings** is denoted by $\mathcal{R}$ and the set of all rankings of some fixed size c is denoted $\mathcal{R}^c$.

The ranking r_t is based on a current **utility approximation** $\hat{u}_t$ of the user-subjective pattern utility u. After the user indicates that she is done with investigating the candidate ranking (by clicking mine as described above), a feedback ranking $\overline{r}$ can be constructed from her actions. For the definition of this feedback ranking, let us denote by T_t and B_t all the patterns that have been deleted to the trash can and promoted to the result board until the end of round t, respectively. The feedback ranking consists of all patterns that were promoted in that round to the result board followed by all patterns in the candidate ranking that have been inspected by the user and were neither deleted nor promoted. Let

$$x = \max\{i \in [c] \mid x_i \notin (B_{t+1} \setminus B_t) \cup (T_{t+1} \setminus T_t)\}$$

be the maximal index of a pattern that has been either promoted or deleted during round t. Formally, the **feedback ranking** is defined by

$$\overline{r}_t = \langle b_1, \ldots, b_k, r_{i_1}, \ldots, r_{i_l} \rangle$$

where $\{b_1, \ldots, b_k\} = B_{t+1} \setminus B_t$ in the order of their promotion and $\{r_{i_1}, \ldots, r_{i_l}\} = \{r_t^x\} \setminus (B_{t+1} \cup T_{t+1})$ with $i_j < i_{j'}$ for $j < j' \leq x$. At the end of the discovery round, a new utility approximation $\hat{u}_{t+1}$ is inferred by comparing $\overline{r}_t$ with r and the next discovery round starts.

3.2 Learning and Construction of Rankings

We now turn to the precise details of the ranking mechanism that is used in the beginning of a discovery round in order to compute a candidate ranking from the content of the pattern cache. With this mechanism we aim for two goals: firstly, we want to allow the user to find new patterns that are maximally *relevant* in terms of her specific utility preferences, and, secondly, we want to provide her a sufficient amount of *diversity* in the displayed mining results. The latter goal is important in a user-based search process, because if the user is not able to express previously unspecified aspects of her preferences the whole discovery process can get stuck in a local maximum.

In order to achieve these goals, we adapt the co-active learning process proposed in Shivaswamy and Joachims [2012] and Raman et al. [2012] for maintaining the parameter vector $\mathbf{w}_t$ of a **ranking utility function**

$$\hat{u}_t(r) = \langle \mathbf{w}_t, \varphi(x_t, r)\rangle$$

over the discovery rounds $t \in \mathbb{N}$. This function is defined by a joint **feature map** $\varphi\colon X \times \mathcal{R} \to \mathbb{R}^{\mathcal{F}}$ that maps a pattern ranking together with the current system state to an $|\mathcal{F}|$-dimensional real-valued feature vector, i.e., the feature representation is determined by the set $\mathcal{F}$ of interestingness functions. The **system state** $x_t \in X$ in discovery round t is given by the contents of the pattern cache, the result board, and the trash can, respectively, i.e., $x_t = (C_{l(t)}, B_t, T_t)$. The component of φ corresponding to function $f \in \mathcal{F}$ is defined as a discounted aggregate of the individual patterns' contribution to feature f, i.e.,

$$\varphi_f(x_t, r) = \left\| \left(\frac{\delta(x_t, r_i)\varphi_f(r_i)}{\log(i+1)} \right)_{i=1}^{|r|} \right\|_d$$

where $\varphi_f(r_i)$ is given by the feature map for individual patterns (defined below) and $\delta(x_t, r_i)$ is an indicator function that is 1 if pattern r_i neither already present on the result board B_t or in the trash can T_t and 0 otherwise. The choice $d \in \mathbb{N} \cup \{\infty\}$ of the norm is a **diversity parameter** that determines the trade-off between relevance and diversity when evaluating a ranking.

The feature map for the individual patterns is designed to allow a maximal cross-pattern inference and at the same time to only require minimal attention of the user: while the actual pattern p can only contain the values for some base interestingness functions, the feature vector of p also contains values for all interestingness functions that the user can infer from these base functions. For example the rationale of the pattern in Fig. 2 contains only the interestingness functions f_{frq}, and f^t_{dv}. However, this is enough to infer also the values for the multiplicative combinations $f^b_{\text{frq}} f^t_{\text{dv}}$, and, hence, we can also use these in the feature reperesntation of the pattern. Formally, the individual components of the **pattern feature map** are defined by

$$\varphi_f(s, X) = \begin{cases} f(s) & \text{, if } f \in \hat{X} \\ 0, & \text{, otherwise} \end{cases}$$

where $\hat{X}$ denotes the set of all feature functions in $\mathcal{F}$ that can be computed based on functions in the rationale X, i.e.,

$$\hat{X} = \{ f \in \mathcal{F} \colon f_1, \ldots, f_k \in X, \\ f(s) = g(f_1(s), \ldots, f_k(s), s) \} \ .$$

This means the feature representation of a pattern consists of all function values of interestingness functions in the rationale X and those that can be inferred from these values. Other features—that are not in X or that cannot be inferred from X—are set to zero. Note that feature functions that only depend on the pattern descriptor (such as the relative shortness f_{sho}) are always part of $\hat{X}$. Hence, if $\mathcal{F}$ contains features such as descriptor length and indicator features for the presence of the specific constraints, then these features are relevant for all patterns.

Algorithm 1 Greedy ranking

```
Require: Patterns $P \subseteq \mathcal{P}$, size $c \in \mathbb{N}$, utility fct $u\colon \mathcal{R} \to \mathbb{R}$
Ensure: Ranking $r^c_{\text{grd}}(P)$ s.t. $u(r^c_{\text{grd}}(P))/u(r^c_{\text{opt}}(P)) \geq 1/3$
1. for $i = 1, \ldots, c$ do
2.     set $r_i \in \arg\max_{p \in P \setminus \{r_1, \ldots, r_{i-1}\}} u(\langle r_1, \ldots, r_{i-1}, p \rangle)$
3. return $\langle r_1, \ldots, r_c \rangle$
```

With the definition of the ranking utility we can now specify the candidate ranking r_t that is displayed to the user at the beginning of every discovery round t. Naturally, one would want this to be the **optimal ranking** of length c (cache capacity) with respect to the current model, i.e.,

$$r^c_{\text{opt}}(C_{l(t)}) \in \arg\max \{ \hat{u}_t(x_t, r) \colon r \in \mathcal{R}^c(C_{l(t)}) \} \ .$$

Unfortunately, using a reduction from the max-k-cover problem (see, e.g., [Feige et al., 2011]), one can show that it is **NP**-hard to compute this optimal ranking and even to approximate one within a ratio larger than $(1 - 1/e)$. This holds already for very simple feature spaces $\mathcal{F}$, in particular for the one used in our prototype (see Sec. 4). On the other hand, a **greedy ranking** $r^c_{\text{grd}}(P)$ can be constructed efficiently by Alg. 1, which iteratively grows a solution by adding in each step to the current partial ranking the pattern that maximizes the utility. For all pattern sets $P \subseteq \mathcal{P}$ this solution can be computed in time $O(c\,|P|)$ and satisfies the approximation guarantee

$$\hat{u}_t(r^c_{\text{grd}}(P))/\hat{u}_t(r^c_{\text{opt}}(P)) \geq 1/3 \ .$$

This result can be proven by observing that the space of partial rankings can be represented by the intersection of two Matroids and that $\hat{u}$ is *sub-modular* with respect to that set system. The approximation guarantee then follows from a general performance theorem for the greedy algorithm from Fisher et al. [1978].

Finally, we can specify how to update the parameter vector of the ranking utility at the end of each discovery round. Following Raman et al. [2012], we can update by the following multiplicative **utility update rule**

$$\mathbf{w}_{t+1,f} = \mathbf{w}_{t,f} \exp(\theta_t(\varphi_f(\overline{r}_t) - \varphi_f(r_t)))/Z \qquad (2)$$

where Z is a normalization factor that ensures that $\|\mathbf{w}_t\|_2 = 1$ and $\theta_t = 1/(2S\sqrt{2^{\lfloor \log t \rfloor}})$ is a decreasing **utility learning rate** depending also on a bound $S \geq \max_{r,B} \|\varphi(r)\|_\infty$ on the max-norm of all rankings (in our prototype we can, e.g., use $S = c^{1/d}$; see Sec. 4). The approximation guarantee of the greedy algorithm and a certain guarantee on the quality of the user feedback imply that this update mechanism has a controlled regret over an optimal weight vector.

3.3 Online Control of Mining Algorithms

It remains to specify the algorithm selection distribution $\boldsymbol{\pi}_l$ that is used in mining round l. As mentioned earlier, we consider the output of mining algorithms as random variables following a distribution that depends on the available running time. In order to asses the mining performance, the system can only observe the output of algorithms that it actually uses and initially the performance of all algorithms are unknown. Thus, the system is facing an exploitation/exploration problem of the multi-armed bandit style (see, e.g., Cesa-Bianchi and Lugosi [2006]). In order to apply known strategies for this kind of problem, we first have to model the reward that is generated by a mining algorithm when it is executed.

Let us P_l denote the **result pattern set** returned by the mining algorithm executed in round l (denoted by $m_l \in \mathcal{M}$) and by c_l the **computation time** it used to produce these results. Then the mining performance of round l can be quantified by the utility gain per time of the ranking that can be constructed from the old and the new patterns together, i.e., by

$$(u(r_{\mathrm{opt}}(P_l \cup \{r_{t(l)}\})) - u(r_{t(l)}))/c_l \ .$$

Of course, the system has no access to the true utility function and cannot compute an optimal ranking efficiently. Hence, it has to rely on its current approximation $\hat{u}_{t(l)}$ and the greedily approximated ranking to estimate the performance, i.e., it has to use the estimated relative **utility gain**

$$g_l = (\hat{u}_{t(l)}(r_{\mathrm{grd}}(P_l \cup \{r_{t(l)}\})) - \hat{u}_{t(l)}(r_{t(l)}))/c_l \ .$$

Thus, the observed reward generated by a mining algorithm depends not only the current system state but also on the current approximation of the user utility, both of which evolve over time.

This means that we need an exploitation/exploration strategy that is robust to *non-stationary rewards.* To this end, we employ an algorithm of Cesa-Bianchi and Lugosi [2006, p. 160] that has an optimally bounded regret. Throughout all mining rounds $l \in \mathbb{N}$, it maintains **performance potential weights** $\mathbf{v}_l \in \mathbb{R}_+^k$ starting with $\mathbf{v}_1 = (1, \dots, 1)$. The algorithm m_l to run in mining round l is then chosen at random according to the **algorithm selection distribution** $\boldsymbol{\pi}_l \in [0,1]^k$, which is a mixture of the distribution given by $\mathbf{v}$ and the uniform distribution, i.e., it is given by

$$\boldsymbol{\pi}_{l,i} = ((\gamma_l - 1)\mathbf{v}_i)/V + \gamma_l/k$$

where V normalizes the sum of the entries of $\mathbf{v}$ to one. The **bandit mixture coefficient** γ_l depends on the mining round and will be specified below. After the result of a mining round is observed the potentials are updated multiplicatively by the **bandit update rule**

$$\mathbf{v}_{l+1,i} = \mathbf{v}_{l,i} \exp(\eta_l \overline{g}_{l,i})$$

where η_l is the **bandit learning rate** $\eta_l = \gamma_l/(2k)$ and $\overline{g}_{l,i}$ an optimistic estimator of the performance of algorithm i in round l that is defined by

$$\overline{g}_{l,i} = \begin{cases} (g_l + \beta_l)/\boldsymbol{\pi}_{m_l}), & \text{if } i = m_l \\ \beta_l/\boldsymbol{\pi}_{m_l}), & \text{otherwise} \end{cases} \ .$$

By choosing

$$\beta_l = \sqrt{\ln(10k)/(k2^{\lfloor \log l \rfloor})}$$

one can make sure that the bias of the performance estimates is not too large while still being optimistic with high probability. Depending on β_l one can also chose the bandit mixture coefficient as $\gamma_l = 4k\beta_l/(3+\beta_l)$.

This concludes the formal description of the one-click mining framework. All algorithmic ideas are summarized again in Alg. 2. Note that this is a compressed listing that needs a slight addition in order to avoid concurrency issues: When the mine-click procedure terminates the currently running mining algorithm, this triggers a call of the algorithm-end procedure. In this case the algorithm-end procedure should only be carried out until step 6 and the new mining algorithm is only started after the remaining steps 5-7 of the mine-click procedure are finished.

Algorithm 2 One-click Mining

```
Initialization:
1. init utility weights $\mathbf{w}_1 \leftarrow (1,\dots,1)/|\mathcal{F}|$
2. init performance weights $\mathbf{v}_1 \leftarrow (1,\dots,1)$
3. init discovery and mining round $t, l \leftarrow 1$
4. draw algorithm $m \in \mathcal{M}$ uniformly at random
5. run $m$ blocking for time $c_{\mathrm{init}}$ (result patterns $P$)
6. init candidate buffer $C_1 = P$ and present $r_{\mathrm{grd}}(C_1)$

On Algorithm End:
1. update candidate buffer $C_{l+1} = C_l \cup P_l$
2. asses $g_l = (\hat{u}_{t(l)}(r^c_{\mathrm{grd}}(C_{l+1})) - \hat{u}_{t(l)}(r^c_{\mathrm{grd}}(C_l))/c_l$
3. for all $i \in \mathcal{M}$ do
4.     $\overline{g}_{l,i} \leftarrow \begin{cases} (g_l + \beta_l)/\boldsymbol{\pi}_{m_l}, & \text{if } i = m_l \\ \beta_l/\boldsymbol{\pi}_{m_l}, & \text{otherwise} \end{cases}$
5.     $\mathbf{v}_i \leftarrow \mathbf{v}_i \exp(\eta_l \overline{g}_{l,i})$
6. $l \leftarrow l + 1$
7. run algorithm $m_l \sim \boldsymbol{\pi}_l$ in background where
       $\boldsymbol{\pi}_{l,i} = (1-\gamma_l)\mathbf{v}_i/V + \gamma_l/k$

On Mine Click:
1. assess feedback ranking $\overline{r}_t$
2. for all $f \in \mathcal{F}$ do
3.     $\mathbf{w}_{t+1,f} = \mathbf{w}_{t,f} \exp(\theta_t(\varphi_f(\overline{r}_t) - \varphi_f(r_t)))$
4. terminate current algorithm $m_l$
5. construct and show greedy ranking $r_{t+1} = r_{\mathrm{grd}}(C_{l-1})$
6. reset $C_l = \{r_{t+1}\}$
7. $t \leftarrow t + 1$
```

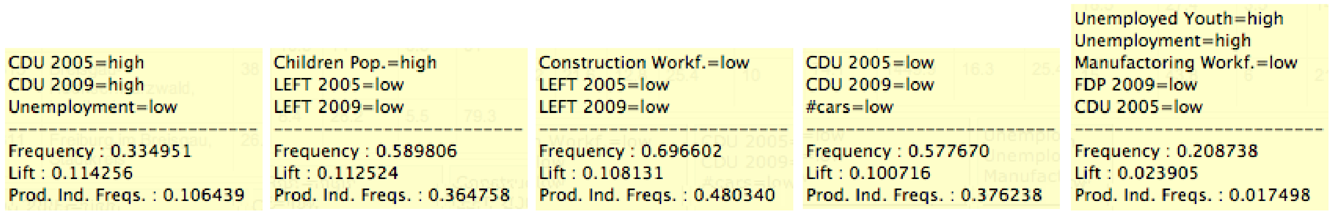
4. PROOF OF CONCEPT

In order to provide a proof of concept for the one-click-mining approach, we present an exemplary pattern discovery session performed by the prototypical one-click mining implementation called *Bonn click mining.* In this session, we deal with the pattern discovery use case of election analysis (see Grosskreutz et al. [2010]).

4.1 Prototype Configuration

The prototype is configured to combine association and subgroup discovery; both with a range of interestingness functions that is able to express different trade-offs between pattern frequency and lift or target deviation, respectively. In addition there are functions that express certain preferences on the form of the pattern only, i.e., its descriptor. On the algorithm side, there is a mixture of 8 deterministic beam-search and randomized pattern sampling algorithms.

More precisely, the pattern language $\mathcal{L}_{bcm}$ of the prototype is the combination of association and subgroup pat-



```
CDU 2005=high
CDU 2009=high
Unemployment=low
--------------------------
Frequency : 0.334951
Lift : 0.114256
Prod. Ind. Freqs. : 0.106439
```

```
Children Pop.=high
LEFT 2005=low
LEFT 2009=low
--------------------------
Frequency : 0.589806
Lift : 0.112524
Prod. Ind. Freqs. : 0.364758
```

```
Construction Workf.=low
LEFT 2005=low
LEFT 2009=low
--------------------------
Frequency : 0.696602
Lift : 0.108131
Prod. Ind. Freqs. : 0.480340
```

```
CDU 2005=low
CDU 2009=low
#cars=low
--------------------------
Frequency : 0.577670
Lift : 0.100716
Prod. Ind. Freqs. : 0.376238
```

```
Unemployed Youth=high
Unemployment=high
Manufactoring Workf.=low
FDP 2009=low
CDU 2005=low
--------------------------
Frequency : 0.208738
Lift : 0.023905
Prod. Ind. Freqs. : 0.017498
```

Figure 4: Five association patterns found in analysis phase 1 of proof of concept experiment; patterns were found between discovery rounds 5 and 15 after promoting some simpler (more trivial) associations and deleting some subgroup patterns.

terns, i.e., $\mathcal{L}_{bcm} = \mathcal{L}_{asd} \cup \mathcal{L}_{sgd}$. The feature functions $\mathcal{F}_{\text{bcm}}$ can be separated into three groups, i.e.,

$$\mathcal{F}_{\text{bcm}} = \mathcal{F}_{\text{sgd}} \cup \mathcal{F}_{\text{asd}} \cup \mathcal{F}_{\text{dsc}}$$

where $\mathcal{F}_{\text{sgd}}$, $\mathcal{F}_{\text{asd}}$, and $\mathcal{F}_{\text{dsc}}$ are sets of subgroup discovery, association discovery, and descriptor functions, respectively. The subgroup discovery features $\mathcal{F}_{\text{sgd}}$ contain the functions given by Eq. (1) for the three choices of b equal to 0, 1/2, and 1. Analogously, the association functions contain the same trade-offs with frequency, but with the target deviation measure replaced by the lift measure f_{ass}. Also pattern frequency is included for both pattern classes. Finally, the descriptor features contain the relative shortness f_{sho} along with binary indicator functions f^d_{cns} that signal whether attribute d is present in the descriptor or not, i.e., $f^d_{\text{cns}}(s) = 1$ if $d \in \text{attr}(s)$ and $f^d_{\text{cns}}(s) = 0$ otherwise. For subgroup patterns there are in addition similar features that can express affinity for a specific target attribute, i.e., for all $t \in [n]$ the feature $f^t_{\text{trg}}(s, t')$ that takes on the value 1 if and only if $t = t'$. For the resulting feature space $\mathcal{F}_{\text{bcm}}$ we have for all patterns $p \in \mathcal{P}$ that $\varphi_f(p) \in [0, 1]$. Hence, we can use the bound $S = c^{1/d}$ for setting the learning rate for the utility updates as defined in Eq. (2) where c denotes as usual the capacity of the pattern cache.

The employed set of algorithm $\mathcal{M}_{\text{bcm}}$ consists of 4 direct pattern sampling and 4 beam search algorithm such that from each group there are two algorithms for each discovery task. Direct pattern sampling produces random pattern collections as the outcome of fast appropriately biased random experiments without constructing auxiliary parts of the pattern space (see Boley et al. [2012] from which we also use the complementary pattern sampling library[1]). All algorithm preprocess the data by discretizing numerical attributes into high and low bins. In the case of subgroup discovery all algorithm draw the target attribute at random according to the distribution given by the current weights for the target preference features. The beam search algorithms then directly optimize either subgroup or association interestingness; finding the top-10 patterns with a beam size of 5 or 10, respectively. For the sampling algorithms appropriately constructed pattern distributions are chosen: For association discovery we use distributions that are biased towards patterns with a high frequency on the one side but that contain individual constraints with a low frequency on the other side. This favors the production of patterns with a high lift. For subgroup discovery we split the dataset into to parts corresponding to high and low values of the target attribute and sample patterns discriminating these parts.

4.2 German Socio-economical Data

For this proof of concept we used data from the domain of socio-economics and politics, which can be used to investigate a diverse set of understandable and interpretable analysis questions. Specifically, we constructed a table from publicly available database provided by the German Federal Office of Statistic[2]. This database provides a wide range of statistical variables mapped to regional units of Germany.

For our table, we let the *data records* correspond to the 413 administrative districts of Germany (Landkreise), which is the second finest spatial resolution provided in the database. Each district is described by 39 *attributes* that can be roughly grouped into socio-economical and political variables. In terms of socio-economic attributes we selected variables from the following categories: age structure and education of the population, economic indicators (e.g., GDP growth, unemployment), and structure of the labor market (workforce in different sectors such as production, public service, etc.). In terms of political attributes, we added the election results of the five major political parties for the federal elections in 2005 and 2009, respectively: CDU (conservative), SPD (center-left), GREEN (center-left), FDP (liberal), and LEFT (left-wing).

4.3 Results

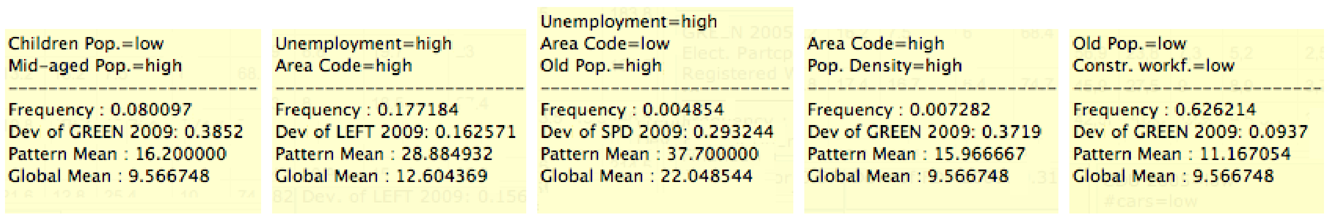
We report some results of an exemplary analysis session to illustrate the system behavior. For this we assume that the user starts at first with a very general purely exploratory analysis intent on order to get an overview before she turns to attack more specific questions. That is, we assume the following analysis question for phase 1.

Phase 1—general question:
What attribute/value combinations show a strong correlation in the data?

While following this intent, the user investigates and promotes mostly fairly general association patterns, while she deletes too specific and in particular subgroup patterns. During the first discovery rounds the produced matching candidate patterns are dominated by those that reflect the well-known fact that political parties have relatively stable regional strongholds. This means there are a lot of patterns of the form "party 2005=high/low, party 2009=high/low". Then, after a few rounds, more space in the candidate area is devoted to association patterns, and consequently a higher

[1]`http://www-kd.iai.uni-bonn.de/index.php?page=software_details&id=23`

[2]`www.regionalstatistik.de`



```
Children Pop.=low
Mid-aged Pop.=high
--------------------------
Frequency : 0.080097
Dev of GREEN 2009: 0.3852
Pattern Mean : 16.200000
Global Mean : 9.566748
```

```
Unemployment=high
Area Code=high
--------------------------
Frequency : 0.177184
Dev of LEFT 2009: 0.162571
Pattern Mean : 28.884932
Global Mean : 12.604369
```

```
Unemployment=high
Area Code=low
Old Pop.=high
--------------------------
Frequency : 0.004854
Dev of SPD 2009: 0.293244
Pattern Mean : 37.700000
Global Mean : 22.048544
```

```
Area Code=high
Pop. Density=high
--------------------------
Frequency : 0.007282
Dev of GREEN 2009: 0.3719
Pattern Mean : 15.966667
Global Mean : 9.566748
```

```
Old Pop.=low
Constr. workf.=low
--------------------------
Frequency : 0.626214
Dev of GREEN 2009: 0.0937
Pattern Mean : 11.167054
Global Mean : 9.566748
```

Figure 5: Subgroup patterns found in analysis phase 2 of proof of concept experiment; patterns were found between discovery rounds 10 and 30 after promoting specific party targets and demoting sufficiently many subgroup patterns with other parties in descriptor.

diversity of them with more non-trivial correlation are offered. For instance, the patterns shown in Fig. 4 have been produced between discovery rounds 5 and 15. These patterns confirm some known associations between attributes. CDU is strong in the economically strong regions of the south of Germany that show low unemployment. Conversely, CDU also and its economically liberal coalition partner FDP are weaker in economically problematic areas with high unemployment and little industry. Other patterns in the set appear to be plausible and could be interesting for further investigation.

After this initial exploratory phase, we now turn to a much more specialized analysis question, in which we try to find explanations for the 2009 election results of some specific parties.

Phase 2—specific question:
What socio-economic and regional factors favored parties from the left spectrum in the German 2009 federal election?

This question implies that we are interest in subgroup patterns with the three targets SPD, GREEN, and LEFT. At the same time, we are interested in socio-economic and regional explanations only, which means do not want to see descriptor elements of other parties (or the same party in the 2005 election). Correspondingly, while following this analysis question, the user deletes general association patterns and those subgroup patterns that do comply to the descriptor restrictions mentioned above. Again, just as in phase 1, in the beginning the produced candidates are dominated by the obvious correlations between party attributes. Additionally, now we also have a much more narrow focus of suitable pattern forms, because we are not interested just in any subgroup pattern but only in those with having a target from a set of 3 out of 39 attributes. Consequently, this time it takes longer until suitable patterns show up in the result area. The patterns shown in Fig. 5 have been produced between discovery rounds 10 and 30. Again the patterns partially confirm some known party preferences (note that a high area code corresponds to the south/east and a low area code to north/west regions). For instance it is known that SPD has a relatively high result among older voters, while GREEN is strong in the densely populated urban areas. Finally, LEFT is known to be strong among areas in the east and particularly in economically weak areas with high unemployment.

5. CONCLUSION

We presented a general framework for combining different pattern discovery methods and algorithm into a single one-click mining system. As a proof of concept of these ideas, we constructed a prototype that includes association and subgroup discovery. In a preliminary evaluation we saw that the resulting system is able to produce patterns corresponding to certain simple analysis goals without exposing the user to the technical side of the pattern mining method and its algorithms. The system is now ready to advance to the next stage of evaluation with a full scale empirical user study.

One limitation of the current approach is related to the expressiveness of the user utility model. Currently, the model weights as well as the feature functions are all positive and the learning takes place in primal space without kernel-based learning. On the one hand, this results in a sub-modular utility function for rankings, which can be efficiently optimized by the greedy algorithm within a reasonable approximation ratio. On the other hand, it prevents the learning of negative influence of some base features. Also the user is currently not able to learn, e.g., conjunctive conditions on the utility. For example she can not express that she is interested in a certain descriptive attribute only in the context of a specific target and otherwise not.

Another direction for future research is to closer investigate and extend the feedback options of the user. For instance, it appears to be desirable for negative feedback to differentiate between the cases of patterns that are simply invalid for the current analysis task and between patterns that are interesting in a vacuum but not novel. One approach to realize this is to introduce an explicit model of the user's background knowledge. This would allow to include subjective interestingness functions (see De Bie [2011]) into the utility model.

Acknowledgment

This work was supported by the German Science Foundation (DFG) under the reference number 'GA 1615/2-1'.

APPENDIX

A. SUBGROUP DEVIATION MEASURE

The subgroup deviation measure can be computed differently depending on the nature of the target attribute. For a categorical targets a_t with domain $V_t = \{v_1, \ldots, v_k\}$ the measure is given as

$$f^t_{\mathrm{dv}}(s) = \sum_{i=1}^{k} |\mathbf{p}^t_S(i) - \mathbf{p}^t_D(i)|$$

where $S = s(D)$ is the extension of s and $\mathbf{p}^t_X \in [0,1]^k$, defined by

$$\mathbf{p}^t_X(i) = |\{d \in X \colon a_t(d) = v_i\}| \,/\, |X| \enspace ,$$

is the empirical distribution of target values with respect to some subset of the data $X \subseteq D$. Hence, Eq. (1) includes well-known subgroup discovery measures such as Weighted Relative Accuracy and the Binomial Test. In case of a numerical target, the deviation can be computed by fitting a continuous distribution and then approximating the total variation distance. Here, for simplicity we assume a normal distribution with mean $\mu = \sum_{d \in D} a_t(d) / |D|$ and variance

$$\sigma_X^2 = \sum_{d \in X} (a_t(d) - \mu_X)^2 / (|X| - 1)$$

for the global data; and for the pattern a normal distribution with $\mu_S = \sum_{d \in s(D)} a_t(d) / |s(D)|$ and the same variance as in the global data. The target deviation can then be expressed much simpler as

$$f_{\mathrm{dv}}^t(s) = 2 \left| \operatorname{erf}\left(\frac{\mu_S - \mu_D}{2\sqrt{2\sigma_D^2}} \right) \right| ,$$

where erf denotes the Gauss error function. This is equal to twice the probability mass of $\mathcal{N}(\mu_D, \sigma_D^2)$ that is at most $(\mu_S - \mu_D)/2$ standard deviations away from the mean μ_D.

References

Abraham Bernstein, Foster Provost, and Shawndra Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):503–518, 2005.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. *KNIME: The Konstanz information miner.* Springer, 2008.

M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2012.

Longbing Cao. Actionable knowledge discovery and delivery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):149–163, 2012.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006.

Tijl De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery*, 23(3):407–446, 2011.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

Bart Goethals, Sandy Moens, and Jilles Vreeken. Mime: a framework for interactive visual pattern mining. In *Machine Learning and Knowledge Discovery in Databases*, pages 634–637. Springer, 2011.

Henrik Grosskreutz, Mario Boley, and Maike Krause-Traudes. Subgroup discovery for election analysis: a case study in descriptive data mining. In *Discovery Science*, pages 57–71. Springer, 2010.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996.

Ingo Mierswa. Rapid miner. *KI*, 23(2):62–63, 2009.

Katharina Morik and Martin Scholz. The miningmart approach to knowledge discovery in databases. In *Intelligent Technologies for Information Analysis*, pages 47–65. Springer, 2004.

K. Raman, P. Shivaswamy, and T. Joachims. Online learning to diversify from implicit feedback. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 705–713. ACM, 2012.

Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. In *Proceedings of the 29th International Conference on Machine Learning, (ICML 2012)*, 2012.

Koen Smets and Jilles Vreeken. Slim: Directly mining descriptive patterns. In *Proceedings of the Twelfth SIAM International Conference on Data Mining (SDM 2012)*, pages 236–247, 2012.

Matthijs van Leeuwen and Arno J. Knobbe. Non-redundant subgroup discovery in large and complex data. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III*, pages 459–474, 2011.

Geoffrey I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *TKDD*, 4(1), 2010.

Geoffrey I. Webb. Filtered-top-k association discovery. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 1(3):183–192, 2011.

Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 773–778. ACM, 2006.