Intelligent Data Analysis 17 (2013) 399–421 DOI 10.3233/IDA-130586 IOS Press

Semi-supervised learning on closed set lattices

Mahito Sugiyama^{a,b,*} and Akihiro Yamamoto^a

^aGraduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto, Japan ^bJapan Society for the Promotion of Science, Japan

Abstract. We propose a new approach for semi-supervised learning using *closed set lattices*, which have been recently used for frequent pattern mining within the framework of the data analysis technique of *Formal Concept Analysis* (FCA). We present a learning algorithm, called SELF (SEmi-supervised Learning via FCA), which performs as a multiclass classifier and a label ranker for *mixed-type data* containing both discrete and continuous variables, while only few learning algorithms such as the decision tree-based classifier can directly handle mixed-type data. From both labeled and unlabeled data, SELF constructs a closed set lattice, which is a partially ordered set of data clusters with respect to subset inclusion, via FCA together with discretizing continuous variables, followed by learning classification rules through finding *maximal* clusters on the lattice. Moreover, it can *weight* each classification rule using the lattice, which gives a partial order of preference over class labels. We illustrate experimentally the competitive performance of SELF in classification and ranking compared to other learning algorithms using UCI datasets.

Keywords: Semi-supervised learning, label ranking, mixed-type data, closed set lattice, formal concept analysis

1. Introduction

In various research domains from biology to economics, numerous *mixed-type data* including both discrete (binary or nominal) and continuous (real-valued) variables are collected by researchers. However, despite recent rapid development of many data analysis techniques in the fields of machine learning, data mining, and knowledge discovery, only few algorithms such as the decision tree-based classifier [29] can directly handle such mixed-type data. In particular, to the best of our knowledge, no learning algorithm treats mixed-type data in a *semi-supervised* manner.

Semi-supervised learning is a special form of classification [48,50]; a learning algorithm uses both labeled and unlabeled data to learn classification rules. In real tasks, it is often difficult to obtain enough labeled data since the task of labeling has a high cost in terms of time and money, while lots of unlabeled data can be collected easily. The goal of semi-supervised learning is to construct a better classifier using such large amount of unlabeled data together with labeled data in short supply.

^{*}Corresponding author: Mahito Sugiyama, Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, 606-8501 Kyoto, Japan. Tel.: +81 75 753 5628; Fax: +81 75 753 5628; E-mail: mahito@iip.ist.i.kyoto-u.ac.jp; Present address: Max Planck Institute for Intelligent Systems, AG K. Borgwardt, Spemannstr. 38, 72076, Tübingen, Germany. Tel.: +49 7071 601 1789; E-mail: mahito.sugiyama@tuebingen.mpg.de.

To effectively use unlabeled mixed-type data for learning, we in this paper propose a novel semisupervised learning algorithm, called SELF (SEmi-supervised Learning via Formal Concept Analysis), which can directly treat mixed-type data. SELF adopts a popular semi-supervised learning strategy, called cluster-and-label [6,9], where a clustering algorithm is first applied, followed by labeling each cluster using labeled data. One of the remarkable features of SELF is that it performs the clustering process using *Formal Concept Analysis* (FCA) [8,14], which is a mathematical theory for data analysis and knowledge representation introduced by Wille [45]. Recently, Pasquier et al. [30] proposed to use closed patterns (itemsets) obtained by FCA as condensed "lossless" representations of patterns. This new approach has been the subject of further research and extensions [1,31,36,46]. In SELF, the labeling process is performed on a *closed set lattice*, which is the result of FCA. Informally, this structure describes the maximally general classification rules that explain the training data, thus preventing overfitting. Moreover, each classification rule can be *weighted* using the closed set lattice by counting the number of clusters classified by the rule, resulting in the *preference* of class labels as a partial order of them for each unlabeled datum. Furthermore, FCA and closed set lattices enable us to naturally treat incomplete data including missing values.

- To summarize, this paper provides contributions to the respective fields of:
- 1. *To semi-supervised learning*: we present a novel approach based on an algebraic framework without assuming any data distribution.
- 2. To FCA: we study a novel application, semi-supervised learning, using FCA and closed set lattices.

The behavior of SELF is outlined as a flowchart in Fig. 1, and this paper is organized along it after discussing about related work in Section 2. The data preprocessing phase to construct a context from a given dataset to apply FCA is explained in Section 3.1. Missing values are handled in this phase. The learning phase is described in Sections 3.2 and 3.3; Section 3.2 shows data clustering and making closed set lattices by FCA and Section 3.3 explains the training algorithm of SELF to learn classification rules. Classification by learned rules is considered in Section 3.4. Section 4 gives empirical evaluation of SELF and, finally, key points and future work are summarized in Section 5.

2. Related work

Many studies have used FCA for machine learning and knowledge discovery [26], such as classification [12,13], clustering [47], frequent pattern and association rule mining [20,30,41], and bioinformatics [2,23,25]. In particular, Ganter and Kuznetsov [13] investigated the problem of binary classification of real-valued data and proposed algorithms based on the *JSM-method* that produces hypotheses (classifiers) using positive and negative examples. Their idea of using the lattice structure derived by FCA for classification is similar to our approach, but the way of treating continuous variables is different. Their method discretizes continuous variables by inequations, called *conceptual scaling* [14], that are given *a priori*, while SELF automatically discretizes them along with the learning process and no background knowledge and assumption about data are needed.

On the other hand, in machine learning context, decision tree-based algorithms such as C4.5 [32,33] can treat mixed-type data by discretizing continuous variables, and there are several discretization techniques [10,27,38] to treat continuous variables in a discrete manner. Our approach is different from them since we integrate discretization process into learning process and avoid overfitting using closed set lattices effectively. SELF uses cluster-and-label, or called label propagation, which is a popular approach in semi-supervised learning as mentioned in Introduction [4,6,9,19,44]. First SELF makes clusters without label information by FCA, followed by giving preferences of class labels for each cluster. To date,



Fig. 1. A flowchart of the proposed SELF algorithm. It learns classification rules from training data and applies them to classify test data. Here we say that a concept is consistent if all labels contained in the concept are same.

most of such approaches are designed for only continuous variables and, to the best of our knowledge, no semi-supervised learning algorithm based on cluster-and-label can treat mixed-type data including discrete variables appropriately. Since SELF uses FCA for clustering, it needs no distance calculation and no data distribution, which is one of the remarkable features of SELF.

There exists a study by Kok and Domingos [24] which is related to the idea of putting original data on lattices. They proposed a learning algorithm via hypergraph lifting, which constructs clusters by hypergraphs and learns on them. Their idea is thus similar to ours since we also "lift" raw data to the space of a closed set lattice via FCA. However, it is difficult to treat continuous variables in their approach, thereby our approach can be more useful for machine learning and knowledge discovery from mixed-type data.

SELF achieves not only semi-supervised learning but also label ranking using the preference for each class label. Recently, the concept of preference has attracted more and more attention in artificial intelligence including machine learning, resulting in formalization of the research topic of preference learning [49]. In particular, label ranking [5,18,43] has been treated in preference learning as an extension

	Table 1
	Notation used in this paper
TD	The set of real numbers
	The set of real numbers
$\tau = (H, X)$	Table, pair of header H and body X
set(X)	The set of tuples of body X
h	Feature (element in H)
x, y	Tuple
\perp	Missing value
n	Number of data (objects)
d	Number of features
Dom (h)	Domain of the feature h
G	The set of objects
М	The set of attributes
Ι	Binary relation between G and M
(G, M, I)	Context
8	Object, identified with tuple
т	Attribute
h.m	Qualified attribute generated from feature h
//	Closure operator
$\mathfrak{B}(G,M,I)$	Concept lattice
λ	Label
$\Lambda(g)$	Label of object g
$\Gamma(G)$	Set of labeled objects in G
R	Classification rule (pair of set of attributes and label)
$\boldsymbol{\omega}(\boldsymbol{R})$	Weight of classification rule R
$\psi(\lambda)$	Preference of label λ
\prec_*	True partial order
\prec	Predicted partial order
$CR(\prec,\prec_*)$	Correctness of \prec
CP(≺)	Completeness of \prec

of traditional supervised classification, where the objective is to obtain a ranker which gives a (partial) order of labels for each datum. SELF is the first algorithm that treats label ranking of mixed-type data by weighting each classification rule through closed set lattices.

3. SELF algorithm

We present the SELF algorithm in this section, which is the main part of this paper. The behavior of SELF is illustrated in Fig. 1; first it performs data preprocessing to make a context from a given dataset, second it constructs concept lattices by FCA, and third it learns the preference for each class label. Notations used in this paper are summarized in Table 1.

3.1. Data preprocessing

The aim of data preprocessing is to construct a (formal) context, a binary matrix specifying a set of objects and their attributes, to apply FCA to training data.

A dataset τ is given in the form of a *table*, or a *relation* [7,15,37]. Each table is a pair $\tau = (H, X)$ of a *header* H and a *body* X. We always denote the header size and the body size by d and n, respectively. An element of the header $h \in H$ is called a *feature*¹ and the *domain* of h is denoted by Dom(h). The

¹It is usually called an *attribute*, but to avoid confusion with an attribute in a context, we use the word "feature".

body *X* is a sequence of *tuples* $x_1, x_2, ..., x_n$, where each tuple x_i is a total function from *H* to $Dom(H) = \{Dom(h) \mid h \in H\}$ such that $x_i(h) \in Dom(h)$ for all $h \in H$. Informally, each tuple corresponds to a data point. Missing values in *X* are allowed and denoted by the special symbol \bot , that is, if the value $x_i(h)$ is missing, $x_i(h) = \bot$. In addition, we denote the body *X* by set(X) when we treat it as a set, that is, $set(X) = \{x_1, x_2, ..., x_n\}$. Thus we do not take the order and multiplicity into account in set(X). For each tuple *x* and a subset *J* of the header *H*, the *projection* of *x* on *J*, denoted by $x|_J$, is exactly the same as the restriction of *x* to *J*, i.e., the function from *J* to Dom(H) such that $x|_J(h) = x(h)$ for all $h \in J$.

We consider two types of variables, *discrete* and *continuous*, in this paper. If a feature $h \in H$ is discrete, $Dom(h) = S \cup \{\bot\}$ for some countable set *S*. For instance, $S = \{\mathbf{T}, \mathbf{F}\}$ if the feature *h* is binary and *S* is a (finite) set of symbols if *j* is nominal (categorical). If *h* is continuous, $Dom(h) = \mathbb{R} \cup \{\bot\}$, where \mathbb{R} is the set of real numbers.

In FCA, we call a triplet (G, M, I) *context*. Here *G* and *M* are sets and $I \subseteq G \times M$ is a binary relation between *G* and *M*. The elements in *G* are called *objects*, and those in *M* are called *attributes*. For a given table $\tau = (H, X)$, we identify the set of objects *G* with set $(X) = \{x_1, x_2, \dots, x_n\}$.

In the data preprocessing, for each feature $h \in H$ of a table τ , we independently construct a context (G, M_h, I_h) and combine them into a context (G, M, I). For this process, we always *qualify* attributes to be disjoint by denoting each element *m* of the attribute M_h by *h.m* following the notations used in the database systems literature [15].

First, we focus on preprocessing for discrete variables. Since a context is also a discrete representation of a dataset, this process is directly achieved as follows: For each feature *h*, the set of attributes $M_h = \{h.m|m \in \text{Dom}(h) \setminus \{\bot\}\}$ and, for each value $x_i(h)$, $(x_i, h.m) \in I_h$ if and only if $x_i(h) = m$. In this way, discrete values are translated into a context and missing values are naturally treated. Algorithm 1 performs this translation.

Algorithm 1: Data preprocessing for discrete variables					
Input: Table $\tau = (H, X)$ whose variables are discrete					
Output: Context (G, M_D, I_D)					
function CONTEXTD (τ)					
1: $G \leftarrow set(X)$					
2: for each $h \in H$					
3: $M_h \leftarrow \{h.m \mid m \in \text{Dom}(h) \setminus \{\bot\}\}$					
4: $I_h \leftarrow \{(x, h, x(h)) \mid x \in G \text{ and } x(h) \neq \bot\}$					
5: end for					
6: combine all (G, M_h, I_h) with $h \in H$ into (G, M_D, I_D)					
7: return (G, M_D, I_D)					

Example 1 Given a table $\tau = (H, X)$ with $H = \{1, 2, 3\}$ and $X = x_1, x_2$ such that

$$(x_1(1), x_1(2), x_1(3)) = (\mathbf{T}, \bot, \mathbf{C}),$$

$$(x_2(1), x_2(2), x_2(3)) = (\mathbf{F}, \mathbf{F}, \perp).$$

This table can be represented in the following manner.

Η		1	2	3
X	x_1	Т	\perp	С
	x_2	F	F	\perp

The domains are given as Dom (1) = Dom (2) = $\{T, F\}$ and Dom (3) = $\{A, B, C\}$. Here we have

$$G = \{x_1, x_2\},\$$

$$(M_1, I_1) = (\{1.\mathbf{T}, 1.\mathbf{F}\}, \{(x_1, 1.\mathbf{T}), (x_2, 1.\mathbf{F})\}),\$$

$$(M_2, I_2) = (\{2.\mathbf{T}, 2.\mathbf{F}\}, \{(x_2, 2.\mathbf{F})\}),\$$

$$(M_3, I_3) = (\{3.A, 3.B, 3.C\}, \{(x_1, 3.C)\}).$$

Thus we have the context (G, M, I) such that

$$M = M_1 \cup M_2 \cup M_3 = \{1.\mathbf{T}, 1.\mathbf{F}, 2.\mathbf{T}, 2.\mathbf{F}, 3.\mathbf{A}, 3.\mathbf{B}, 3.\mathbf{C}\},\$$
$$I = I_1 \cup I_2 \cup I_3 = \{(x_1, 1.\mathbf{T}), (x_1, 3.\mathbf{C}), (x_2, 1.\mathbf{F}), (x_2, 2.\mathbf{F})\}.$$

It is visualized as a cross-table as follows:

	1. T	1. F	2. T	2. F	3.A	3.B	3.C
x_1	×						Х
<i>x</i> ₂		×		×			

Second, we make a context from continuous variables using discretization. This process is embedded in the learning process (see Fig. 1) and discretizing resolution increases along with the process. The degree of resolution is denoted by a natural number k, called *discretization level* and, in the following, we explain how to discretize continuous variables at fixed level k. First we use min-max normalization [17] so that every datum is in the closed interval [0,1]. For every feature h, each value x(h) is mapped to a value y(h) such that

$$y(h) = \frac{x(h) - \min_{x \in set(X)} x(h)}{\max_{x \in set(X)} x(h) - \min_{x \in set(X)} x(h)}$$

Next, we discretize values in [0,1] and make a context using the binary encoding of real numbers, following the approach we have used [39]. At discretization level k, M_h for a feature $h \in H$ is always the set $\{h.1, h.2, \ldots, h.2^k\}$. For each value $x_i(h)$, if $x_i(h) = 0$, then $(x_i, h.1) \in I_h$. Otherwise if $x_i(h) \neq 0$, then $(x_i, h.a) \in I_h$ if and only if

$$\frac{a-1}{2^k} < x_i(h) \leqslant \frac{a}{2^k}.$$

If $x_i(h) = \bot$, then $(x_i, m) \notin I_h$ for all $m \in M_h$. This means that if we encode the value $x_i(h)$ as an infinite sequence $p = p_0 p_1 p_2 \dots$, a context at level k is decided by the first k bits $p_0 p_1 \dots p_{k-1}$. Each value is converted to exactly one relation of a context if it is not missing. Algorithm 2 shows the above process for making a context from continuous variables.

Example 2 Given a table $\tau = (H, X)$ with $H = \{1, 2, 3, 4\}$ and $X = x_1, x_2$ such that

 $(x_1(1), x_1(2), x_1(3)) = (\mathbf{T}, \mathbf{C}, 0.35, 0.78),$ $(x_2(1), x_2(2), x_2(3)) = (\bot, \bot, 0.813, \bot).$

Algorithm 2: Data preprocessing for continuous variables **Input:** Table $\tau = (H, X)$ whose variables are continuous, and discretization level k **Output:** Context $(G, M_{\rm C}, I_{\rm C})$ function CONTEXTC(τ , k) 1: $G \leftarrow set(X)$ 2: for each $h \in H$ 3: $M_h \leftarrow \{h.1, h.2, \ldots, h.2^k\}$ 4: Normalize values in the feature h by min-max normalization $I_h \leftarrow \emptyset$ 5: for each $x \in G$ 6: 7: if x(h) = 0 then $I_h \leftarrow I_h \cup \{(x, 1)\}$ 8: else if $x(h) \neq 0$ and $x(h) \neq \bot$ then $I_h \leftarrow I_h \cup \{(x, h.a)\}, \text{ where } (a-1)/2^k < x(h) \le a/2^k$ 9: end if 10: 11: end for 12: end for 13: combine all (G, M_h, I_h) with $h \in H$ into (G, M_C, I_C) 14: return $(G, M_{\rm C}, I_{\rm C})$

It can be represented as follows:

Η		1	2	3	4
X	x_1	Т	С	0.35	0.78
	x_2	\perp	\bot	0.813	\perp

where the first and second features are discrete with $Dom(1) = \{T, F\}$ and $Dom(2) = \{A, B, C\}$, and the third and forth are continuous. Assume that discretization level k = 1. We have

 $G = \{x_1, x_2\},\$ $(M_1, I_1) = (\{1.\mathbf{T}, 1.\mathbf{F}\}, \{(x_1, 1.\mathbf{T})\}),\$ $(M_2, I_2) = (\{2.\mathbf{A}, 2.\mathbf{B}, 2.\mathbf{C}\}, \{(x_1, 2.\mathbf{C})\}),\$ $(M_3, I_3) = (\{3.1, 3.2\}, \{(x_1, 3.1), (x_2, 3.2)\}),\$ $(M_4, I_4) = (\{4.1, 4.2\}, \{(x_1, 4.2)\}).$

Thus we have the context (G, M, I) such that $M = M_1 \cup M_2 \cup M_3 \cup M_4$ and $I = I_1 \cup I_2 \cup I_3 \cup I_4$, which is visualized as a cross-table as follows:

	1. T	1. F	2.A	2.B	2.C	3.1	3.2	4.1	4.2
x_1	×				×	×			×
<i>x</i> ₂							×		

3.2. Clustering and making lattices by FCA

From a context obtained by the data preprocessing, we generate closed sets as clusters of data points and construct closed set lattices by FCA. First we summarize FCA (see the articles [8,14] for detail). We always assume that a given table τ is converted into a context (*G*,*M*,*I*) by Algorithms 1 and 2.

For subsets $A \subseteq G$ and $B \subseteq M$, we define

$$A' := \{m \in M \mid (g,m) \in I \text{ for all } g \in A\},\$$
$$B' := \{g \in G \mid (g,m) \in I \text{ for all } m \in B\}.$$



Fig. 2. The closed set lattice (concept lattice) constructed from the context given in Example 3.2. In this diagram, each dot denotes a concept, which are treated as a cluster in SELF.

Using these mappings, we define a concept as follows: a pair (A,B) with $A \subseteq G$ and $B \subseteq M$ is called a *concept* of a context (G,M,I) if A' = B and A = B'. The set A is called an *extent* and B an *intent*. Each operator ' is a *Galois connection* between the power set lattices on G and M, respectively, hence the mapping " becomes a closure operator on the context (G,M,I). This means that, for each concept (A,B), A and B are (algebraic) *closed* sets. Note that a subset $A \subseteq G$ (resp. $B \subseteq M$) is the extent (resp. intent) of some concept if and only if A'' = A (resp. B'' = B). Thus a set of objects $A \subseteq G$ forms a cluster if and only if A'' = A. Each object usually belongs to more than one cluster, hence this method is not "crisp" clustering.

The set of concepts over (G, M, I) is written by $\mathfrak{B}(G, M, I)$ and called the *concept lattice*. If we focus on either one of the set of objects or attributes, this lattice is called the *closed set lattice*. In particular, in the context of frequent pattern mining, a set of attributes corresponds to an itemset and the lattice is called the closed itemset lattice. For a pair of concepts $(A_1, B_1) \in \mathfrak{B}(G, M, I)$ and $(A_2, B_2) \in \mathfrak{B}(G, M, I)$, we write $(A_1, B_1) \leq (A_2, B_2)$ if $A_1 \subseteq A_2$. Then $(A_1, B_1) \leq (A_2, B_2)$ holds if and only if $A_1 \subseteq A_2$ (and if and only if $B_1 \supseteq B_2$). This relation \leq becomes an order on $\mathfrak{B}(G, M, I)$ in the mathematical sense and $\langle \mathfrak{B}(G, M, I), \leqslant \rangle$ becomes a complete lattice. Let $\mathscr{C} \subseteq \mathfrak{B}(G, M, I)$. A concept $(A, B) \in \mathscr{C}$ is a *maximal element* of \mathscr{C} if $(A, B) \leq (X, Y)$ and $(X, Y) \in \mathscr{C}$ imply (A, B) = (X, Y) for all $(X, Y) \in \mathfrak{C}$. We write the set of maximal elements of \mathscr{C} by Max \mathscr{C} .

Many algorithms are available for constructing closed set lattices, or concept lattices, and the algorithm proposed by Makino and Uno [28] is known to be one of the fastest algorithms. Their algorithm enumerates all maximal bipartite cliques in a bipartite graph with $O(\Delta^3)$ delay, where Δ is the maximum degree of the given bipartite graph, that is,

$$\Delta = \max\left\{ \#J \mid J \subseteq I, \text{ where } \begin{array}{c} g = h \text{ for all } (g,m), (h,l) \in J, \text{ or} \\ m = l \text{ for all } (g,m), (h,l) \in J \end{array} \right\}$$

(#J is the number of elements in J) in the FCA context. A concept coincides with a bipartite graph, hence we can use their algorithm directly. For empirical experiments, we use the program LCM [40] provided by the authors to enumerate all concepts and construct the closed set lattice.

Example 3 Given the following context:



There exist eight concepts in total; $(\emptyset, \{1,2,3,4,5\}), (\{x_1\}, \{1,2,4\}), (\{x_2\}, \{2,4,5\}), (\{x_3\}, \{3\}), (\{x_1,x_2\}, \{2,4\}), (\{x_2,x_4\}, \{2,5\}), (\{x_1,x_2,x_4\}, \{2\}), and (\{x_1,x_2,x_3,x_4\}, \emptyset), and \Delta = 3.$ We show the closed set lattice in Fig. 2. Let $\mathscr{C} = \{(\emptyset, \{1,2,3,4,5\}), (\{x_1\}, \{1,2,4\}), (\{x_2\}, \{2,4,5\}), (\{x_1,x_2\}, \{2,4\}), (\{x_2,x_4\}, \{2,5\})\}$. Then Max $\mathscr{C} = \{(\{x_1\}, \{1,2,4\}), (\{x_1,x_2\}, \{2,4\}), (\{x_2,x_4\}, \{2,5\})\}$.

3.3. Learning classification rules

Here we present the main learning algorithm of SELF in Algorithm 3, which obtains a set of classification rules from a table τ for training. In this paper, a *classification rule* is a pair of a set of attributes and a label. Intuitively, every unlabeled tuple (datum) is classified to the associated label if it has the same attributes. SELF generates a set of classification rules at each discretization level. We give the precise algorithm of classification in the next subsection.

```
Algorithm 3: Main learning algorithm of SELF; learning classification rules
Input: Table \tau = (H, X)
Output: Classification rules \mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_K
function MAIN(\tau)
         Divide \tau vertically into two tables \tau_D and \tau_C, where \tau_D contains all discrete
 1:
         variables in \tau and \tau_{\rm C} contains all continuous variables in \tau
 2:
         (G, M_{\rm D}, I_{\rm D}) \leftarrow {\rm ContextD}(\tau_{\rm D})
         // make a context from discrete variables of \tau (see Section 3.1)
 3:
                    // k is discretization level
         k \leftarrow 1
                                                     // use this function recursively
 4 \cdot
         LEARNING(\tau_{\rm C}, G, M_{\rm D}, I_{\rm D}, k)
function LEARNING(\tau_{\rm C}, G, M_{\rm D}, I_{\rm D}, k)
         (G, M_{\rm C}, I_{\rm C}) \leftarrow {\rm ContextC}(\tau_{\rm C}, k)
 1:
         // make a context from continuous variables of \tau at level k (see Section 3.1)
 2:
         make (G, M, I) from (G, M_D, I_D) and (G, M_C, I_C)
         construct the concept lattice \mathfrak{B}(G,M,I) from (G,M,I) (see Section 3.2)
 3:
         \mathscr{C} \leftarrow \{(A,B) \in \mathfrak{B}(G,M,I) \mid (A,B) \text{ is consistent}\}
 4:
 5:
         \mathscr{R}_k \leftarrow \{(B, \Lambda(g)) \mid (A, B) \in \operatorname{Max} \mathscr{C} \text{and} g \in \Gamma(A)\}
 6:
         output \mathscr{R}_k
         G \leftarrow G \setminus \{g \mid g \in A \text{ for some } (A, B) \in \mathscr{C}\}
 7:
         remove corresponding attributes and relations from M_{\rm D} and I_{\rm D}, respectively
 8:
         remove corresponding tuples from \tau_{\rm C}
 9٠
         if \Gamma(G) = \emptyset then halt
10:
11:
         else LEARNING(\tau_{\rm C}, G, M_{\rm D}, I_{\rm D}, k+1)
12:
         end if
```

We introduce some notations. For each object $g \in G$, we denote a *label*, an identifier of a class, of g by $\Lambda(g)$, and if g is unlabeled; i.e., the label information is missing, we write $\Lambda(g) = \bot$. Moreover, we

define $\Gamma(G) := \{g \in G \mid \Lambda(g) \neq \bot\}$, hence objects in $\Gamma(G)$ are labeled objects, and those in $G \setminus \Gamma(G)$ are unlabeled objects. For a concept $(A, B) \in \mathfrak{B}(G, M, I)$, we say that it is *consistent* if $\Gamma(A) \neq \emptyset$ and $\Lambda(g) = \Lambda(h)$ for all $g, h \in \Gamma(A)$. Note that a concept with $\Gamma(A) = \emptyset$ (all labels are missing) is not consistent.

First SELF performs data preprocessing and makes the context (G, M, I) from a given table at each discretization level *k* using the algorithms given in Section 3.1. Second it constructs the concept lattice $\mathfrak{B}(G, M, I)$ using both labeled and unlabeled tuples and finds consistent concepts using labeled tuples (objects). Third it outputs the sets of classification rules such that

$$\mathscr{R}_{k} = \{(B,\lambda) \mid (A,B) \in \operatorname{Max}\mathscr{C}_{k} \text{ and } \lambda = \Lambda(g) \text{ with } g \in \Gamma(A)\}, \text{ where}$$
$$\mathscr{C}_{k} = \{(A,B) \in \mathfrak{B}(G,M,I) \mid (A,B) \text{ is consistent}\}$$

at discretization level k. The lattice enables us to avoid overfitting since, informally, attributes of maximal concepts correspond to the most general classification rules. If some objects that are not contained in consistent concepts remains, it refines discretization; i.e., increases discretization level, and repeats the above procedure for the remaining objects.

Moreover, SELF *weights* each classification rule. For a classification rule $R = (B, \lambda)$, the weight $\omega(R)$ is defined as follows:

$$\omega(R) := \#\{(C,D) \in \mathfrak{B}(G,M,I) \mid D \supseteq B\}.$$

Intuitively, the weight of a rule R means its importance since it is the number of clusters classified by the rule. Using the weight of rules, label ranking is realized (see the next subsection).

Example 4 Given a dataset $\tau = (H, X)$ and its labels as follows:

Η		1	2	3	Label
	<i>x</i> ₁	Т	С	0.28	1
	<i>x</i> ₂	F	А	0.54	1
X	<i>x</i> ₃	Т	В	\perp	\perp
	<i>x</i> ₄	F	А	0.79	2
	<i>x</i> ₅	Т	С	0.81	\perp

where $Dom(1) = \{T, F\} \cup \{\bot\}$, $Dom(2) = \{A, B, C\} \cup \{\bot\}$, and $Dom(3) = \mathbb{R} \cup \{\bot\}$. At discretization level 1, we have the following context:

	1. T	1. F	2.A	2.B	2.C	3.1	3.2
x_1	×				×	×	
x_2		×	×				×
<i>x</i> ₃	×			×			
x_4		×	\times				×
<i>x</i> ₅	×				×		×

We show the closed set lattice in the left-hand side in Fig. 3. By SELF, we obtain $\mathscr{R}_1 = \{(\{1,\mathbf{T}\},1)\}$ since the concept $(\{x_1,x_3,x_5\},\{1,\mathbf{T}\})$ is the maximal consistent concept, and there is no consistent concept that contains x_2 or x_4 . This classification rule means "For a tuple x, if $x(1) = \mathbf{T}$, then x is classified to the class 1". The weight is calculated as $\omega(\{1,\mathbf{T}\},1) = 6$. SELF removes objects x_1, x_3 , and x_5 contained in the consistent concepts and proceeds to the next level. At discretization level 2, we have the following context:



Fig. 3. The closed set lattices (concept lattices) at discretization levels 1 and 2 constructed during the learning phase in Example 4. In these diagrams, each black dot denotes the maximal consistent concept in the set of concepts covered by the dotted line.

	1. T	1. F	2.A	2.B	2.C	3.1	3.2	3.3	3.4
x_2		×	×					×	
<i>x</i> ₄		×	×						\times

The right-hand side in Fig. 3 shows the closed set lattice of the above context, and we obtain $\Re_2 = \{(\{1, \mathbf{F}, 2, A, 3, 3\}, 1), (\{2, A, 3, 4\}, 2)\}$. For instance, the first rule means "For a tuple *x*, if $x(1) = \mathbf{F}, x(2) = A$, and $0.5 < x(3) \leq 0.75$, its class label is 1". The weight are 2 for both rules.

We show that SELF always stops in finite time if there are no conflicting objects. Namely, for a table $\tau = (H, X)$, if there is no pair $x, y \in \text{set}(X)$ such that $\Lambda(x) \neq \Lambda(y)$ and x(h) = y(h) for all $h \in H$, Algorithm 3 stops in finite time. This statement is proved in the following way: if discretization level k is large enough, we have the concept lattice $\mathfrak{B}(G,M,I)$, where for every object $x \in G$, there exists a concept (A, B) such that $A = \{x\}$ since there is no pair $x, y \in G$ satisfying x(h) = y(h) for all $h \in H$. Thus each object x with $\Lambda(x) \neq \bot$ must be contained in some consistent concept, and the algorithm stops. Note that the algorithm works even if $\Gamma(G) = G$; i.e., all objects have labels, hence it also can be viewed as a supervised classification method.

The time complexity of learning by SELF is $O(nd) + O(\Delta^3 N)$ such that

$$N = \max_{k \in \{1,2,\dots,K\}} \#\mathfrak{B}(G_k, M_k, I_k)$$

where (G_k, M_k, I_k) is the context at discretization level k and K is the level where SELF stops since data preprocessing takes O(nd), making a concept lattice takes less than $O(\Delta^3 N)$, and obtaining classification rules takes less than O(N).

3.4. Classification

Now we have sets of classification rules $\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_K$ for each discretization level from training mixed-type data including labeled and unlabeled data using Algorithms 1, 2, and 3. In this section, we show how to classify a new unlabeled datum using the rules. We assume that such a new datum is given as a table v = (H, y), where the body y consists of only one tuple.

Algorithm 4 performs classification using the obtained rules $\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_K$. The algorithm is levelwise; i.e., at each level k, it makes a context (G, M, I) from the table v = (H, y) and apply the set of

Name	# Data	# Classes	# F	eatures
			Discrete	Continuous
Abalone	4177	28	1	7
Allbp	2800	3	2	3
Anneal	798	5	28	10
Arrhythmia	452	13	5	5
Australian	690	2	7	4
Crx	690	2	9	6
Echocardiogram	131	2	1	7
Heart	270	2	7	6
Hepatitis	155	2	13	6
Horse colic	368	2	8	2

Table 2 Statistics for UCI datasets used for experiments

Algorithm 4: Classification

Input: Classification rules $\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_K$, table $\upsilon = (H, y)$, and the set of labels \mathscr{L} Output: Preference of each label **function** CLASSIFY($\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_K, \upsilon$) Divide v vertically into two tables v_D and v_C , where v_D contains all discrete 1: variables in v and $v_{\rm C}$ contains all continuous variables in v $(G, M_{\rm D}, I_{\rm D}) \leftarrow \tilde{\rm ContextD}(v_{\rm D})$ 2: // make a context from discrete values of v3: for each $\lambda \in \mathscr{L}$ 4: $\psi(\lambda) \leftarrow 0$ 5: for each $k \in \{1, 2, ..., K\}$ $(G, M_{\rm C}, I_{\rm C}) \leftarrow {\rm CONTEXTC}(v_{\rm C}, k)$ 6: // make a context from continuous values of v at level k7: make a context (G, M, I) from (G, M_D, I_D) and (G, M_C, I_C) 8: $\psi(\lambda) \leftarrow \psi(\lambda) + \sum_{R \in \mathscr{Q}} \omega(R)$, where $\mathscr{Q} = \{(B, \lambda) \in \mathscr{R}_k \mid (y, b) \in I \text{ for all } b \in B\}$ 9: end for 10: output $\psi(\lambda)$ end for 11:

rules \mathscr{R}_k to it. Let \mathscr{L} be the domain of class labels. It checks all rules in \mathscr{R}_k and, for each label $\lambda \in \mathscr{L}$, it outputs the *preference* of the label λ , which is defined as

$$\psi(\lambda) := \sum_{k=1}^{K} \sum_{R \in \mathscr{Q}} \omega(R), \text{ where } \mathscr{Q} = \{(B, \lambda) \in \mathscr{R}_k \mid (y, b) \in I \text{ for all } b \in B\},\$$

by summing up weights of rules. Note that the set G is always a singleton $\{y\}$ in the classification phase. The result means that if $\psi(\lambda) > \psi(\lambda')$ for labels λ and λ' , λ is preferable than λ' , and vice versa, and if $\psi(\lambda) = \psi(\lambda)$, the preference of λ and λ' are same, resulting in the *partial order* over the set of labels \mathscr{L} . Thus the task of label ranking is achieved by the preference ψ . Moreover, if we pick up the label

$$\lambda \in \operatorname{argmax}_{\lambda \in \mathscr{L}} \psi(\lambda), \tag{1}$$

multiclass classification is also performed directly.

Example 5 Let us consider the case discussed in Example 4. A tuple y such that

 $(y(1), y(2), y(3)) = (\mathbf{T}, \mathbf{B}, 0.45)$



Fig. 4. Experimental results of accuracy for ten mixed-type datasets from UCI repository with varying the number of labeled data. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), and compared them to the decision tree-based classifier (Tree), SVM with the RBF kernel (SVM), and the *k*-nearest neighbor algorithm (1-NN, 5-NN). Data show mean \pm s.e.m.

satisfies only the rule $(\{1,\mathbf{T}\},1) \in \mathscr{R}_1$. Thus we have $\psi(1) = 6$ and $\psi(2) = 0$ for labels 1 and 2, respectively. A tuple *z* with

$$(z(1), z(2), z(3)) = (\mathbf{F}, \mathbf{A}, 0.64)$$

satisfies only the rule ({1.**F**, 2.**A**, 3.3}), hence $\psi(1) = 0$ and $\psi(2) = 2$.

4. Experiments

We empirically evaluate SELF. Our experiments consist of two parts: one is about multiclass classification, and the other is about label ranking.



Fig. 5. Experimental results of accuracy for ten mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 10 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), and compared them to the decision tree-based classifier (Tree), SVM with the RBF kernel (SVM), and the *k*-nearest neighbor algorithm (1-NN, 5-NN). Data show mean \pm s.e.m.

4.1. Methods

4.1.1. Environment

SELF was implemented in R version 2.12.1 [34] and all experiments were performed in the R environment. For enumeration of all concepts and construction of a closed set lattice from a context, we used LCM² distributed by Uno [40], which was implemented in C.

²http://research.nii.ac.jp/ uno/codes.htm.



Fig. 6. Experimental results of accuracy for ten mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 100 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), and compared them to the decision tree-based classifier (Tree), SVM with the RBF kernel (SVM), and the *k*-nearest neighbor algorithm (1-NN, 5-NN). Data show mean \pm s.e.m.

4.1.2. Datasets

We collected ten mixed-type datasets from UCI Machine Learning Repository [11]: *abalone*, *allbp*, *anneal*, *arrhythmia*, *australian*, *crx*, *echocardiogram*, *heart*, *hepatitis*, and *horse colic*. Their basic statistics are summarized in Table 2. Datasets *allbp*, *anneal*, *arrhythmia*, *australian*, *crx*, *echocardiogram*, *hepatitis*, and *horse colic* includes missing values, which were directly treated in SELF. In other learning algorithms, we ignored all tuples which have missing values since they cannot treat such datasets appropriately.

In label ranking, we used four datasets: *abalone*, *allbp*, *anneal*, and *arrhythmia*, which have more than three classes. The other datasets had only two classes and could not be used for label ranking evaluation.



Fig. 7. Experimental results of correctness (should be maximized) for four mixed-type datasets from UCI repository with varying the number of labeled data. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

4.1.3. Control learning algorithms

In multiclass classification, three learning algorithms were adopted: the decision tree-based classifier implemented in R supplied in the tree package [35], SVM with the RBF kernel (C = 5 and $\gamma = 0.05$) in the kernlab package [21], and the *k* nearest neighbor algorithm (k = 1 and 5) in the class package. Notice that only the decision tree-based algorithm can treat mixed-type data directly, which is one of typical such learning algorithms. All discrete values were treated as continuous in SVM and *k*NN.

4.1.4. Evaluation

In classification, for each dataset, the following procedure was repeated 20 times and the mean and s.e.m. (standard error of the mean) of accuracy was obtained: 1) the number of labeled data or features was fixed, where the range was from 10 to 100 and 2 to 10, respectively, 2) labeled training data were sampled randomly, 3) labels of the remaining data were predicted by respective learning algorithms, and 4) the accuracy was obtained.

The Eq. (1) was used to determine the most preferable label for each unlabeled datum. If there exists more than two such labels, we chose the smallest one.

We adopted two criteria: *correctness* and *completeness*, used in the literature [5] to evaluate partial orders of labels in label ranking. Correctness coincides with the *gamma rank correlation* [16], which is the normalized difference between the number of correctly ranked pairs and that of incorrectly ranked pairs. Let \mathcal{L} be the set of class labels and we denote by \prec_* the ground truth of the partial order over the set of labels \mathcal{L} . Assume that \prec is a predicated partial order. Here we define

$$C := \#\{(\lambda, \lambda') \in \mathscr{L} \times \mathscr{L} \mid \lambda \prec \lambda' \text{ and } \lambda \prec_* \lambda'\},\$$



Fig. 8. Experimental results of completeness (should be maximized) for four mixed-type datasets from UCI repository with varying the number of labeled data. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

$$D := \#\{(\lambda, \lambda') \in \mathscr{L} \times \mathscr{L} \mid \lambda \prec \lambda' \text{ and } \lambda' \prec_* \lambda\}.$$

Then, the correctness is defined by

$$\operatorname{CR}(\prec,\prec_*) := \frac{C-D}{C+D}.$$

Trivially, the correctness takes a value in [-1,1], and $CR(\prec,\prec_*) = 1$ if $\prec = \prec_*$ and $CR(\prec,\prec_*) = -1$ if \prec is the inversion of \prec_* . Thus the correctness should be maximized. Moreover, to evaluate the degree of completeness of a predicted partial order, we use the completeness defined as follows:

$$\operatorname{CP}(\prec) := \frac{C+D}{\#\{(\lambda,\lambda') \in \mathscr{L} \times \mathscr{L} \mid \lambda \prec_* \lambda' \text{ or } \lambda' \prec_* \lambda\}}$$

The completeness takes a value in [0, 1] and should be maximized.

4.2. Results

4.2.1. Multiclass classification

We evaluated SELF in multiclass classification. Specifically, we examined SELF's behavior with respect to the number of labeled data and the number of features; the number of labeled data was varied from 10 to 100, and the number of features from 2 to 10. When we fixed the number of labeled data,



Fig. 9. Experimental results of correctness (should be maximized) for mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 10 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

we used all features for *abalone*, *anneal*, *australian*, *crx*, *echocardiogram*, *heart*, and *hepatitis*, and only used features 1, 2, 3, 18, 20 in *allbp*, 1, 2, ..., 6, 22, 22, ..., 25 in *arrhythmia*, and 1, 2, 4, 5, ... 11 in *horse colic*, since we could not finish experiments in reasonable time for such dense datasets. The above features seem to be representative for each dataset. Otherwise if we fixed the number of features, we examined two cases in which the number of labeled data for training were 10 or 100. Such small amount of labeled data is typical in semi-supervised learning; for example, the numbers 10 and 100 were adopted in benchmarks in the literature³ [50, §21].

To analyze effectivity of unlabeled data in the semi-supervised manner, we trained SELF in two ways; one is using both labeled and the remaining all unlabeled data for training, and the other is using only labeled data for training without any unlabeled data. In the following, we denote "SELF" in the former case and "SELF (w/o)" in the latter case. All experiments were carried out in the *transductive setting* [42], that is, test data coincide with the unlabeled training data. This setting is common in empirical evaluation of semi-supervised learning methods [50, §21].

For control, three learning algorithms were adopted: the decision tree-based classifier, SVM with the RBF kernel, and the *k* nearest neighbor algorithm (k = 1 and 5). All the above algorithms are typical for supervised learning and hence did not use unlabeled data in training.

Figures 4 and 5, 6 show the accuracy with respect to changes in the number of labeled data and the number of features, respectively. In every case, the accuracy of SELF was much better than that of

³This content is available at http://olivier.chapelle.cc/ssl-book/benchmarks.pdf.



Fig. 10. Experimental results of completeness (should be maximized) for mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 10 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

SELF (w/o), and the accuracy was getting better according as the number of labeled data increases. Moreover, SELF's performance is getting better with increase in the number of features. SELF therefore can effectively use unlabeled data and features for learning.

In comparison with the tree algorithm which can treat mixed-type data directly, SELF showed better performance in all datasets in Fig. 4. Moreover, compared to other learning algorithms of SVM and kNN, SELF also achieved the best performance in *abalone*, *anneal*, and *horse colic*. When the number of labeled data is small (about 10–40), SELF outperformed other learning algorithms in all datasets except *allbp*, as shown in Figs 4 and 5.

4.2.2. Label ranking

We examined effectivity of SELF for label ranking. In consideration of the lack of benchmark data for label ranking, we adopted the following procedure for label ranking: We trained SELF using all labeled data on the respective dataset and obtained the ranking for each datum, and used them as the ground truth. Literature [5,18] which studied label ranking used the naïve Bayes classifier to make the ground truth of rankings from datasets. However, the mathematical theory is totally different from those of SELF, hence their approach is not appropriate to our case.

Figures 7 and 8 show the results of label ranking by SELF with varying the number of labeled data, and Figs 9–12 show those with respect to the number of features, where the number of labeled data is 10 for Figs 9 and 10, and 100 for Figs 11 and 12. The correctness of SELF is better than SELF (w/o) in *abalone*, and is similar between them in the other datasets for all conditions. In contrast, the



Fig. 11. Experimental results of correctness (should be maximized) for mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 100 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

completeness of SELF is much higher than that of SELF (w/o) in most cases. The main reason might be that lots of data are not classified to any class in SELF (w/o).

4.3. Discussion

Our experiments about classification (Figs 4, 5, 6) show that SELF has competitive performance compared to other machine learning algorithms, where unlabeled data can be used effectively in training. This result means that data clustering using the closed set lattices works well for semi-supervised learning of mixed-type data. Moreover, SELF can explicitly produce classification rules like the decision treebased algorithm, hence SELF's results can be easily interpreted. Furthermore, in label ranking (Figs 7– 12), SELF outperformed SELF (w/o) in most cases in terms of completeness, and the performance got higher with increase of the number of labeled data. Our results therefore show that unlabeled data are also effectively used in SELF in the task of label ranking.

5. Conclusion

We have proposed a novel semi-supervised learning method, called SELF, for mixed-type data including both discrete and continuous variables, and experimentally showed its competitive performance. The key strategy is data clustering with closed set lattices using FCA, and the present study shows the effectivity of the lattices in semi-supervised learning. To our best knowledge, this approach is the first



Fig. 12. Experimental results of completeness (should be maximized) for mixed-type datasets from UCI repository with varying the number of features. The number of labeled data was fixed at 100 in each experiment. We performed SELF using both labeled and unlabeled data (SELF) and using only labeled data (SELF (w/o)), Data show mean \pm s.e.m.

direct semi-supervised method for mixed-type data, and also the first one to exploit closed set lattices in semi-supervised learning. Moreover, we can directly treat missing values on SELF, meaning that SELF can be used for various practical datasets. To date, many semi-supervised learning methods use data distribution and probabilities, whereas SELF uses only the algebraic structure of data without any back-ground knowledge. Our results with lattice-based data analysis provide new insight to machine learning and knowledge discovery.

There are two future works; one is analysis of SELF from FCA point of view. Refinement of discretization of continuous variables must have some connection with *reduction* of a context [14] since if we extend a context by refining real-valued variables, the original attributes are removed by reduction. Thereby analysis of mathematical connection between them is a future work. The other is theoretical analysis in the computational learning theory context. de Brecht and Yamamoto [3] have proposed *Alexandrov concept space* for learning from positive data. Our proposed method might be an instance of the study, since the concept lattice is similar to the Alexandrov space. Thus theoretical analysis of our framework is also a future work.

Acknowledgments

We would like to thank Marco Cuturi for his helpful comments. This work was partly supported by Grant-in-Aid for Scientific Research (A) 22240010 and for JSPS Fellows $22 \cdot 5714$.

References

- G. Beslon, D.P. Parsons, J.M. Peña, C. Rigotti and Y. Sanchez-Dehesa, From digital genetics to knowledge discovery: Perspectives in genetic network understanding, *Intelligent Data Analysis* 14(2) (2010), 173–191.
- [2] V.G. Blinova, D.A. Dobrynin, V.K. Finn, S.O. Kuznetsov and E.S. Pankratova, Toxicology analysis by means of the JSM-method, *Bioinformatics* 19(10) (2003), 1201–1207.
- [3] M. de Brecht and A. Yamamoto, Topological properties of concept spaces (full version), *Information and Computation* 208 (2010), 327–340.
- [4] H. Cheng, Z. Liu and J. Yang, Sparsity induced similarity measure for label propagation. in: Proceedings of 12th IEEE International Conference on Computer Vision, (2009), 317–324.
- [5] W. Cheng, M. Rademaker, B. De Baets and E. Hüllermeier, Predicting partial orders: Ranking with abstention. in: *Machine Learning and Knowledge Discovery in Databases*, J. Balcázar, F. Bonchi, A. Gionis and M. Sebag, eds, Volume 6321 of Lecture Notes in Computer Science., Springer, 2010, pp. 215–230.
- [6] R. Dara, S.C. Kremer and D.A. Stacey, Clustering unlabeled data with SOMs improves classification of labeled realworld data, in: Proceedings of the 2002 International Joint Conference on Neural Networks. Volume 3, 2002, pp. 2237– 2242.
- [7] C.J. Date, An Introduction to Database Systems. 8 ed., Addison Wesley 2003.
- [8] B.A. Davey and H.A. Priestley, Introduction to lattices and order. 2 ed., Cambridge University Press 2002.
- [9] A. Demiriz, K.P. Bennett and M.J. Embrechts, Semi-supervised clustering using genetic algorithms. in: Proceedings of Artificial Neural Networks in Engineering, 1999, pp. 809–814.
- [10] U.M. Fayyad and K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1029.
- [11] A. Frank and A. Asuncion, UCI machine learning repository (2010).
- [12] B. Ganter and S. Kuznetsov, Hypotheses and version spaces, in: *Conceptual Structures for Knowledge Creation and Communication*, A.D. Moor, W. Lex and B. Ganter, eds, Volume 2746 of Lecture Notes in Computer Science., Springer 2003, pp. 83–95.
- [13] B. Ganter and S. Kuznetsov, Formalizing hypotheses with concepts, in: *Conceptual Structures: Logical, Linguistic, and Computational Issues*, B. Ganter and G.W. Mineau, eds, Volume 1867 of Lecture Notes in Computer Science, Springer 2000, pp. 342–356.
- [14] B. Ganter and R. Wille, Formal Concept Analysis: Mathematical Foundations. Springer 1998.
- [15] H. Garcia-Molina, J.D. Ullman and J. Widom, Database systems: The complete book, Prentice Hall Press 2008.
- [16] L. Goodman and W. Kruskal, Measures of Association for Cross Classifications. Springer 1979.
- [17] J. Han and M. Kamber, Data Mining. 2 edn. Morgan Kaufmann (2006).
- [18] E. Hülermeier, J. Fünkranz, W. Cheng and K. Brinker, Label ranking by learning pairwise preferences, Artificial Intelligence 172(16–17) (2008), 1897–1916.
- [19] T. Hwang and R. Kuang, A heterogeneous label propagation algorithm for disease gene discovery, in: Proceedings of SIAM International Conference on Data Mining, 2010, pp. 583–594.
- [20] R. Jaschke, A. Hotho, C. Schmitz, B. Ganter and G. Stumme, TRIAS–An algorithm for mining iceberg tri-lattices. in: Proceedings of the IEEE 6th International Conference on Data Mining, 2006, pp. 907–911.
- [21] A. Karatzoglou, A. Smola, K. Hornik and A. Zeileis, kernlab–an S4 package for kernel methods in R, *Journal of Statistical Software* 11(9) (2004), 1–20.
- [22] M. Kaytoue, S.O. Kuznetsov and A. Napoli, Revisiting numerical pattern mining with formal concept analysis. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011, pp. 1342–1347.
- [23] M. Kaytoue, S.O. Kuznetsov, A. Napoli and S. Duplessis, Mining gene expression data with pattern structures in formal concept analysis, *Information Sciences* 181 (2011), 1989–2001.
- [24] S. Kok and P. Domingos, Learning Markov logic network structure via hypergraph lifting. In: Proceedings of the 26th International Conference on Machine Learning, 2009, pp. 505–512.
- [25] S.O. Kuznetsov and M.V. Samokhin, Learning closed sets of labeled graphs for chemical applications, in: *Inductive Logic Programming*, S. Kramer and B. Pfahringer, eds, Volume 3625 of Lecture Notes in Computer Science., Springer 2005, pp. 190–208.
- [26] S.O. Kuznetsov, Machine learning and formal concept analysis, in: *Concept Lattices*, P. Eklund, ed., Volume 2961 of Lecture Notes in Computer Science, Springer 2004, pp. 287–312.
- [27] H. Liu, F. Hussain, C.L. Tan and M. Dash, Discretization: An enabling technique, *Data Mining and Knowledge Discovery* 6(4) (2002), 393–423.
- [28] K. Makino and T. Uno, New algorithms for enumerating all maximal cliques, in: SWAT 2004. Volume 3111 of Lecture Notes in Computer Science., Springer 2004, pp. 260–272.
- [29] S.K. Murthy, Automatic construction of decision trees from data: A multi-disciplinary survey, *Data Mining and Knowledge Discovery* 2(4) (1998), 345–389.

- [30] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Efficient mining of association rules using closed itemset lattices, *Information Systems* 24(1) (1999), 25–46.
- [31] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire and Y.W. Choong, Mining multidimensional and multilevel sequential patterns, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **4**(1) (2010), 4–37.
- [32] J.R. Quinlan, C4.5: programs for machine learning. Morgan Kaufmann (1993).
- [33] J.R. Quinlan, Improved use of continuous attributes in C4.5, Journal of Artificial Intelligence Research 4 (1996), 77–90.
- [34] R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, 2011.
- [35] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996.
- [36] J. Saquer and S. Jitender, Using closed itemsets for discovering representative association rules, in: *Foundations of Intelligent Systems*, Z. Ras and S. Ohsuga, eds, Volume 1932 of Lecture Notes in Computer Science, Springer 2010, pp. 495–504.
- [37] D.A. Simovici and C. Djeraba, Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics, Springer (2008).
- [38] M. Skubacz, J. Hollmén, Quantization of continuous input variables for binary classification, in: Intelligent Data Engineering and Automated Learning IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents. Volume 1983 of Lecture Notes in Computer Science., Springer 2000, pp. 42–47.
- [39] M. Sugiyama and A. Yamamoto, The coding divergence for measuring the complexity of separating two sets, in: Proceedings of 2nd Asian Conference on Machine Learning. Volume 13 of JMLR Workshop and Conference Proceedings, 2010, pp. 127–143.
- [40] T. Uno, M. Kiyomi and H. Arimura, LCM ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining, in: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, ACM, 2005, pp. 77–86.
- [41] P. Valtchev, R. Missaoui and R. Godin, Formal concept analysis for knowledge discovery and data mining: The new challenges, in: *Concept Lattices*, P. Eklund, ed., Volume 2961 of Lecture Notes in Computer Science, Springer 2004, pp. 352–371.
- [42] V. Vapnik and A. Sterin, On structural risk minimization or overall risk in a problem of pattern recognition, Automation and Remote Control 10(3) (1977), 1495–1503.
- [43] S. Vembu and T. Gärtner, Label ranking algorithms: A survey. in: Preference Learning, Springer-Verlag 2010, pp. 45–64.
- [44] F. Wang and C. Zhang, Label propagation through linear neighborhoods. In: Proceedings of the 23rd international conference on Machine learning, 2006, pp. 985–992.
- [45] R. Wille, Restructuring lattice theory: An approach based on hierarchies of concepts. In: Ordered Sets. D. Reidel Publishing Company (1982) pp. 445–470. This article is included in Formal Concept Analysis, LNCS 5548, 314–339, Springer (2009).
- [46] M.J. Zaki, Generating non-redundant association rules. In: Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 34–43.
- [47] Y. Zhang, B. Feng and Y. Xue, A new search results clustering algorithm based on formal concept analysis. In: Proceedings of 5th International Conference on Fuzzy Systems and Knowledge Discovery, IEEE 2008, pp. 356–360.
- [48] X. Zhu and A.B. Goldberg, Introduction to semi-supervised learning, Morgan and Claypool Publishers 2009.
- [49] J. Fürnkranz and E. Hüllermeier, eds, Preference learning, Springer 2010.
- [50] O. Chapelle, B. Schölkopf and A. Zien, eds, Semi-Supervised Learning, MIT Press (2006).