Semi-Supervised Learning on Closed Set Lattices

Mahito Sugiyama,[1,2*] Akihiro Yamamoto[1]

[1]Graduate School of Informatics, Kyoto University,
Yoshida Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
[2]Research Fellow of the Japan Society for the Promotion of Science

[*]To whom correspondence should be addressed;
E-mail: mahito@iip.ist.i.kyoto-u.ac.jp, Tel: +81-75-753-5628, Fax: +81-75-753-5628

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, whereas 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, whereas 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.

To effectively use unlabeled mixed-type data for learning, we in this paper propose a novel semi-supervised 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 a contribution to both the fields of semi-supervised learning and FCA:

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 Figure 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 Section 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. However, to date, 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 only one 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 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 Figure 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 $\mathrm{Dom}(h)$. The body X is a sequence of *tuples* $x_1, x_2, \dots, x_n$, where each tuple x_i is a total function from H to $\mathrm{Dom}(H) = \{\mathrm{Dom}(h) \mid h \in H\}$ such that $x_i(h) \in \mathrm{Dom}(h)$ 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 $\perp$, that is, if the value $x_i(h)$ is missing, $x_i(h) = \perp$. In addition, we denote the body X by $\mathrm{set}(X)$ when we treat it as a set, that is, $\mathrm{set}(X) = \{x_1, x_2, \dots, x_n\}$. Thus we do not take the order and multiplicity into account in $\mathrm{set}(X)$. 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 $\mathrm{Dom}(H)$ 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, $\mathrm{Dom}(h) = S \cup \{\perp\}$ 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, $\mathrm{Dom}(h) = \mathbb{R} \cup \{\perp\}$, 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 $\mathrm{set}(X) = \{x_1, x_2, \dots, x_n\}$.

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 $M_h = \{h.m \mid m \in \mathrm{Dom}(h) \setminus \{\perp\}\}$ and, for each value $x_i(h)$, $(x_i, h.m) \in I_h$ if and

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

Algorithm 1: Data preprocessing for discrete variables

```
Input: Table $\tau = (H, X)$ whose variables are discrete
Output: Context $(G, M_{\mathrm{D}}, I_{\mathrm{D}})$

function ContextD($\tau$)
1:  $G \leftarrow \mathrm{set}(X)$
2:  for each $h \in H$
3:      $M_h \leftarrow \{h.m \mid m \in \mathrm{Dom}(h) \setminus \{\perp\}\}$
4:      $I_h \leftarrow \{(x, h.x(h)) \mid x \in G$ and $x(h) \neq \perp\}$
5:  end for
6:  combine all $(G, M_h, I_h)$ with $h \in H$ into $(G, M_{\mathrm{D}}, I_{\mathrm{D}})$
7:  return $(G, M_{\mathrm{D}}, I_{\mathrm{D}})$
```

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.

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

$$(x_1(1), x_1(2), x_1(3)) = (\mathbf{T}, \perp, \mathrm{C}),$$
$$(x_2(1), x_2(2), x_2(3)) = (\mathbf{F}, \mathbf{F}, \perp).$$

This table can be represented in the following manner.

H		1	2	3
X	x_1	**T**	$\perp$	C
	x_2	**F**	**F**	$\perp$

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

$$\begin{aligned} G &= \{x_1, x_2\}, \\ (M_1, I_1) &= (\{1.\mathbf{T}, 1.\mathbf{F}\}, \{(x_1, 1.\mathbf{T}), (x_2, 1.\mathbf{F})\}), \\ (M_2, I_2) &= (\{2.\mathbf{T}, 2.\mathbf{F}\}, \{(x_2, 2.\mathbf{F})\}), \\ (M_3, I_3) &= (\{3.\mathrm{A}, 3.\mathrm{B}, 3.\mathrm{C}\}, \{(x_1, 3.\mathrm{C})\}). \end{aligned}$$

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

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

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	×						×
x_2		×		×			

Second, we make a context from continuous variables using discretization. This process is embedded in the learning process (see Figure 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

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

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 $\{h.1, h.2, \dots, h.2^k\}$. 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 $x_i(h) = \perp$, 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

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

It can be represented as follows:

H		1	2	3	4
X	x_1	**T**	C	0.35	0.78
	x_2	$\perp$	$\perp$	0.813	$\perp$

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

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

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:

Algorithm 2: Data preprocessing for continuous variables

Input: Table $\tau = (H, X)$ whose variables are continuous, and discretization level k

Output: Context (G, M_C, I_C)

```
function CONTEXTC($\tau$, $k$)
1:  $G \leftarrow \text{set}(X)$
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
5:      $I_h \leftarrow \emptyset$
6:      for each $x \in G$
7:          if $x(h) = 0$ then $I_h \leftarrow I_h \cup \{(x, 1)\}$
8:          else if $x(h) \neq 0$ and $x(h) \neq \perp$ then
9:              $I_h \leftarrow I_h \cup \{(x, h.a)\}$, where $(a-1)/2^k < x(h) \leqslant a/2^k$
10:         end if
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 $(G, M_C, I_C)$
```

	1.**T**	1.**F**	2.A	2.B	2.C	3.1	3.2	4.1	4.2
x_1	×				×	×			×
x_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 literatures [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

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

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 $(A_1,B_1) \leqslant (A_2,B_2)$ if $A_1 \subseteq A_2$. Then $(A_1,B_1) \leqslant (A_2,B_2)$ holds if and only if $A_1 \subseteq A_2$ (and if and only if $B_1 \supseteq B_2$). This relation $\leqslant$ 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 $(A,B) \leqslant (X,Y)$ and $(X,Y) \in \mathscr{C}$ imply $(A,B) = (X,Y)$ for all $(X,Y) \in \mathscr{C}$. We write the set of maximal elements of $\mathscr{C}$ by $\operatorname{Max}\mathscr{C}$.

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,

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

($\#J$ 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:

	1	2	3	4	5
x_1	×	×		×	
x_2		×		×	×
x_3			×		
x_4		×			×

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 Figure 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 $\mathscr{R}_1, \mathscr{R}_2, \dots, \mathscr{R}_K$

```
function MAIN($\tau$)
 1:  Divide $\tau$ vertically into two tables $\tau_{\mathrm{D}}$ and $\tau_{\mathrm{C}}$, where $\tau_{\mathrm{D}}$ contains all discrete
     variables in $\tau$ and $\tau_{\mathrm{C}}$ contains all continuous variables in $\tau$
 2:  $(G, M_{\mathrm{D}}, I_{\mathrm{D}}) \leftarrow$ CONTEXTD($\tau_{\mathrm{D}}$)
     // make a context from discrete variables of $\tau$ (see Section 3.1)
 3:  $k \leftarrow 1$    // $k$ is discretization level
 4:  LEARNING($\tau_{\mathrm{C}}, G, M_{\mathrm{D}}, I_{\mathrm{D}}, k$)    // use this function recursively

function LEARNING($\tau_{\mathrm{C}}, G, M_{\mathrm{D}}, I_{\mathrm{D}}, k$)
 1:  $(G, M_{\mathrm{C}}, I_{\mathrm{C}}) \leftarrow$ CONTEXTC($\tau_{\mathrm{C}}, k$)
     // make a context from continuous variables of $\tau$ at level $k$ (see Section 3.1)
 2:  make $(G, M, I)$ from $(G, M_{\mathrm{D}}, I_{\mathrm{D}})$ and $(G, M_{\mathrm{C}}, I_{\mathrm{C}})$
 3:  construct the concept lattice $\mathfrak{B}(G, M, I)$ from $(G, M, I)$ (see Section 3.2)
 4:  $\mathscr{C} \leftarrow \{(A, B) \in \mathfrak{B}(G, M, I) \mid (A, B) \text{ is consistent}\}$
 5:  $\mathscr{R}_k \leftarrow \{(B, \Lambda(g)) \mid (A, B) \in \mathrm{Max}\,\mathscr{C} \text{ and } g \in \Gamma(A)\}$
 6:  output $\mathscr{R}_k$
 7:  $G \leftarrow G \setminus \{g \mid g \in A \text{ for some } (A, B) \in \mathscr{C}\}$
 8:  remove corresponding attributes and relations from $M_{\mathrm{D}}$ and $I_{\mathrm{D}}$, respectively
 9:  remove corresponding tuples from $\tau_{\mathrm{C}}$
10:  if $\Gamma(G) = \emptyset$ then halt
11:  else LEARNING($\tau_{\mathrm{C}}, G, M_{\mathrm{D}}, I_{\mathrm{D}}, k + 1$)
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 $\Lambda(g) = \perp$. Moreover, we define $\Gamma(G) := \{g \in G \mid \Lambda(g) \neq \perp\}$, 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

$$\begin{aligned}&\mathscr{R}_k = \{(B,\lambda) \mid (A,B) \in \mathrm{Max}\mathscr{C}_k \text{ and } \lambda = \Lambda(g) \text{ with } g \in \Gamma(A)\}, \quad \text{where}\\ &\mathscr{C}_k = \{(A,B) \in \mathfrak{B}(G,M,I) \mid (A,B) \text{ is consistent}\}\end{aligned}$$

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:

H		1	2	3	Label
	x_1	**T**	C	0.28	1
	x_2	**F**	A	0.54	1
X	x_3	**T**	B	$\perp$	$\perp$
	x_4	**F**	A	0.79	2
	x_5	**T**	C	0.81	$\perp$

where $\mathrm{Dom}(1) = \{\mathbf{T},\mathbf{F}\} \cup \{\perp\}$, $\mathrm{Dom}(2) = \{\mathrm{A},\mathrm{B},\mathrm{C}\} \cup \{\perp\}$, and $\mathrm{Dom}(3) = \mathbb{R} \cup \{\perp\}$. 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		×	×				×
x_3	×			×			
x_4		×	×				×
x_5	×				×		×

We show the closed set lattice in the left-hand side in Figure 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:

	1.**T**	1.**F**	2.A	2.B	2.C	3.1	3.2	3.3	3.4
x_2		×	×					×	
x_4		×	×						×

The right-hand side in Figure 3 shows the closed set lattice of the above context, and we obtain $\mathscr{R}_2 = \{(\{1.\mathbf{F}, 2.\mathrm{A}, 3.3\}, 1), (\{1.\mathbf{F}, 2.\mathrm{A}, 3.4\}, 2)\}$. For instance, the first rule means "For a tuple x, if $x(1) = \mathbf{F}$, $x(2) = \mathrm{A}$, and $0.5 < x(3) \leqslant 0.75$, 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 $x, y \in \mathrm{set}(X)$ 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 $\Lambda(x) \neq \perp$ 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

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

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 $\upsilon = (H, y)$, 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 $\upsilon = (H, y)$ and apply the set of 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

Algorithm 4: Classification

```
Input: Classification rules $\mathscr{R}_1, \mathscr{R}_2, \dots, \mathscr{R}_K$, table $\upsilon = (H, y)$, and the set of labels $\mathscr{L}$
Output: Preference of each label

function CLASSIFY($\mathscr{R}_1, \mathscr{R}_2, \dots, \mathscr{R}_K$, $\upsilon$)
 1:  Divide $\upsilon$ vertically into two tables $\upsilon_D$ and $\upsilon_C$, where $\upsilon_D$ contains all discrete
     variables in $\upsilon$ and $\upsilon_C$ contains all continuous variables in $\upsilon$
 2:  $(G, M_D, I_D) \leftarrow$ CONTEXTD($\upsilon_D$)
     // make a context from discrete values of $\upsilon$
 3:  for each $\lambda \in \mathscr{L}$
 4:      $\psi(\lambda) \leftarrow 0$
 5:      for each $k \in \{1, 2, \dots, K\}$
 6:          $(G, M_C, I_C) \leftarrow$ CONTEXTC($\upsilon_C, k$)
             // make a context from continuous values of $\upsilon$ at level $k$
 7:          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 $\psi(\lambda)$
11:  end 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

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

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

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

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

4 Experiments

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

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 LCM[1] distributed by Uno [40], which was implemented in C.

[1]`http://research.nii.ac.jp/~uno/codes.htm`

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* included 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.

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 kNN.

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 equation (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 $\mathscr{L}$ be the set of class labels and we denote by $\prec_*$ the ground truth of the partial order over the set of labels $\mathscr{L}$. Assume that $\prec$ is a predicated partial order. Here we define

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

Then, the correctness is defined by

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

Trivially, the correctness takes a value in $[-1,1]$, and $\mathrm{CR}(\prec,\prec_*) = 1$ if $\prec = \prec_*$ and $\mathrm{CR}(\prec,\prec_*) = -1$ 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:

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

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, 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 literature [1] [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.

Figure 4 and Figures 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 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.

[1]This content is available at `http://olivier.chapelle.cc/ssl-book/benchmarks.pdf`.

In comparison with the tree algorithm which can treat mixed-type data directly, SELF showed better performance in all datasets in Figure 4. Moreover, compared to other learning algorithms of SVM and *k*NN, 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 Figures 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. Literatures [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 Figures 9 – 12 show those with respect to the number of features, where the number of labeled data is 10 for Figures 9 and 10, and 100 for Figures 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 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 (Figures 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 tree-based algorithm, hence SELF's results can be easily interpreted. Furthermore, in label ranking (Figures 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 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 background 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.

Acknowledgment

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 22·5714.

References

[1] G. Beslon, D. P. Parsons, J. M. Peña, C. Rigotti, Y. Sanchez-Dehesa: From digital genetics to knowledge discovery: Perspectives in genetic network understanding. Intelligent Data Analysis **14**(2) (2010) pp. 173–191

[2] V. G. Blinova, D. A. Dobrynin, V. K. Finn, S. O. Kuznetsov, E. S. Pankratova: Toxicology analysis by means of the JSM-method. Bioinformatics **19**(10) (2003) pp. 1201–1207

[3] M. de Brecht, A. Yamamoto: Topological properties of concept spaces (full version). Information and Computation **208** (2010) pp. 327–340

[4] H. Cheng, Z. Liu, J. Yang: Sparsity induced similarity measure for label propagation. in: 12th IEEE International Conference on Computer Vision, IEEE (2009) pp. 317–324

[5] W. Cheng, M. Rademaker, B. De Baets, E. Hüllermeier: Predicting partial orders: Ranking with abstention. in J. Balcázar, F. Bonchi, A. Gionis, M. Sebag, eds.: Machine Learning and Knowledge Discovery in Databases. Volume 6321 of Lecture Notes in Computer Science., Springer (2010) pp. 215–230

[6] R. Dara, S. C. Kremer, D. A. Stacey: Clustering unlabeled data with SOMs improves classification of labeled real-world 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 edn. Addison Wesley (2003)

[8] B. A. Davey, H. A. Priestley: Introduction to lattices and order. 2 edn. Cambridge University Press (2002)

[9] A. Demiriz, K. P. Bennett, 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, 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, A. Asuncion: UCI machine learning repository (2010)

[12] B. Ganter, S. Kuznetsov: Hypotheses and version spaces. in A. d. Moor, W. Lex, B. Ganter, eds.: Conceptual Structures for Knowledge Creation and Communication. Volume 2746 of Lecture Notes in Computer Science., Springer (2003) pp. 83–95

[13] B. Ganter, S. Kuznetsov: Formalizing hypotheses with concepts. in B. Ganter, G. W. Mineau, eds.: Conceptual Structures: Logical, Linguistic, and Computational Issues. Volume 1867 of Lecture Notes in Computer Science., Springer (2000) pp. 342–356

[14] B. Ganter, R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer (1998)

[15] H. Garcia-Molina, J. D. Ullman, J. Widom: Database systems: The complete book. Prentice Hall Press (2008)

[16] L. Goodman, W. Kruskal: Measures of Association for Cross Classifications. Springer (1979)

[17] J. Han, M. Kamber: Data Mining. 2 edn. Morgan Kaufmann (2006)

[18] E. Hülermeier, J. Fünkranz, W. Cheng, K. Brinker: Label ranking by learning pairwise preferences. Artificial Intelligence **172**(16–17) (2008) pp. 1897–1916

[19] T. Hwang, R. Kuang: A heterogeneous label propagation algorithm for disease gene discovery. in: SIAM International Conference on Data Mining. (2010) pp. 583–594

[20] R. Jaschke, A. Hotho, C. Schmitz, B. Ganter, G. Stumme: TRIAS–An algorithm for mining iceberg tri-lattices. in: Proceedings of the 6th International Conference on Data Mining, IEEE (2006) pp. 907–911

[21] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis: kernlab–an S4 package for kernel methods in R. Journal of Statistical Software **11**(9) (2004) pp. 1–20

[22] M. Kaytoue, S. O. Kuznetsov, 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, S. Duplessis: Mining gene expression data with pattern structures in formal concept analysis. Information Sciences **181** (2011) pp. 1989–2001

[24] S. Kok, 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, M. V. Samokhin: Learning closed sets of labeled graphs for chemical applications. in S. Kramer, B. Pfahringer, eds.: Inductive Logic Programming. Volume 3625 of Lecture Notes in Computer Science., Springer (2005) pp. 190–208

[26] S. O. Kuznetsov: Machine learning and formal concept analysis. in P. Eklund, ed.: Concept Lattices. Volume 2961 of Lecture Notes in Computer Science., Springer (2004) pp. 287–312

[27] H. Liu, F. Hussain, C. L. Tan, M. Dash: Discretization: An enabling technique. Data Mining and Knowledge Discovery **6**(4) (2002) pp. 393–423

[28] K. Makino, 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) pp. 345–389

[30] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Efficient mining of association rules using closed itemset lattices. Information Systems **24**(1) (1999) pp. 25–46

[31] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, Y. W. Choong: Mining multidimensional and multilevel sequential patterns. ACM Transactions on Knowledge Discovery from Data (TKDD) **4**(1) (2010) pp. 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) pp. 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, S. Jitender: Using closed itemsets for discovering representative association rules. in Z. Ras, S. Ohsuga, eds.: Foundations of Intelligent Systems. Volume 1932 of Lecture Notes in Computer Science., Springer (2010) pp. 495–504

[37] D. A. Simovici, 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, 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, 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, R. Godin: Formal concept analysis for knowledge discovery and data mining: The new challenges. in P. Eklund, ed.: Concept Lattices. Volume 2961 of Lecture Notes in Computer Science., Springer (2004) pp. 352–371

[42] V. Vapnik, A. Sterin: On structural risk minimization or overall risk in a problem of pattern recognition. Automation and Remote Control **10**(3) (1977) pp. 1495–1503

[43] S. Vembu, T. Gärtner: Label ranking algorithms: A survey. in: Preference Learning. Springer-Verlag (2010) pp. 45–64

[44] F. Wang, C. Zhang: Label propagation through linear neighborhoods. in: Proceedings of the 23rd international conference on Machine learning, ACM (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, Volume 5548 of Lecture Notes in Computer Science., Springer (2009) pp. 314–339

[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, 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, A. B. Goldberg: Introduction to semi-supervised learning. Morgan and Claypool Publishers (2009)

[49] J. Fürnkranz, E. Hüllermeier, eds.: Preference learning. Springer (2010)

[50] O. Chapelle, B. Schölkopf, A. Zien, eds.: Semi-Supervised Learning. MIT Press (2006)

Tables

Table 1. Notation used in this paper.

$\mathbb{R}$	The set of real numbers
$\tau = (H, X)$	Table, pair of header H and body X
$\mathrm{set}(X)$	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
$\mathrm{Dom}(h)$	Domain of the feature h
G	The set of objects
M	The set of attributes
I	Binary relation between G and M
(G, M, I)	Context
g	Object, identified with tuple
m	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)
$\omega(R)$	Weight of classification rule R
$\psi(\lambda)$	Preference of label λ
$\prec_*$	True partial order
$\prec$	Predicted partial order
$\mathrm{CR}(\prec, \prec_*)$	Correctness of $\prec$
$\mathrm{CP}(\prec)$	Completeness of $\prec$

Table 2. Statistics for UCI datasets used for experiments.

Name	# Data	# Classes	# Features	
			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

Figure Captions

Figure 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.

Figure 2

The closed set lattice (concept lattice) constructed from the context given in Example 3. In this diagram, each dot denotes a concept, which are treated as a cluster in SELF.

Figure 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.

Figure 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.

Figure 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 ± s.e.m.

Figure 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 ± s.e.m.

Figure 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 ± s.e.m.

Figure 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 ± s.e.m.

Figure 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.

Figure 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.

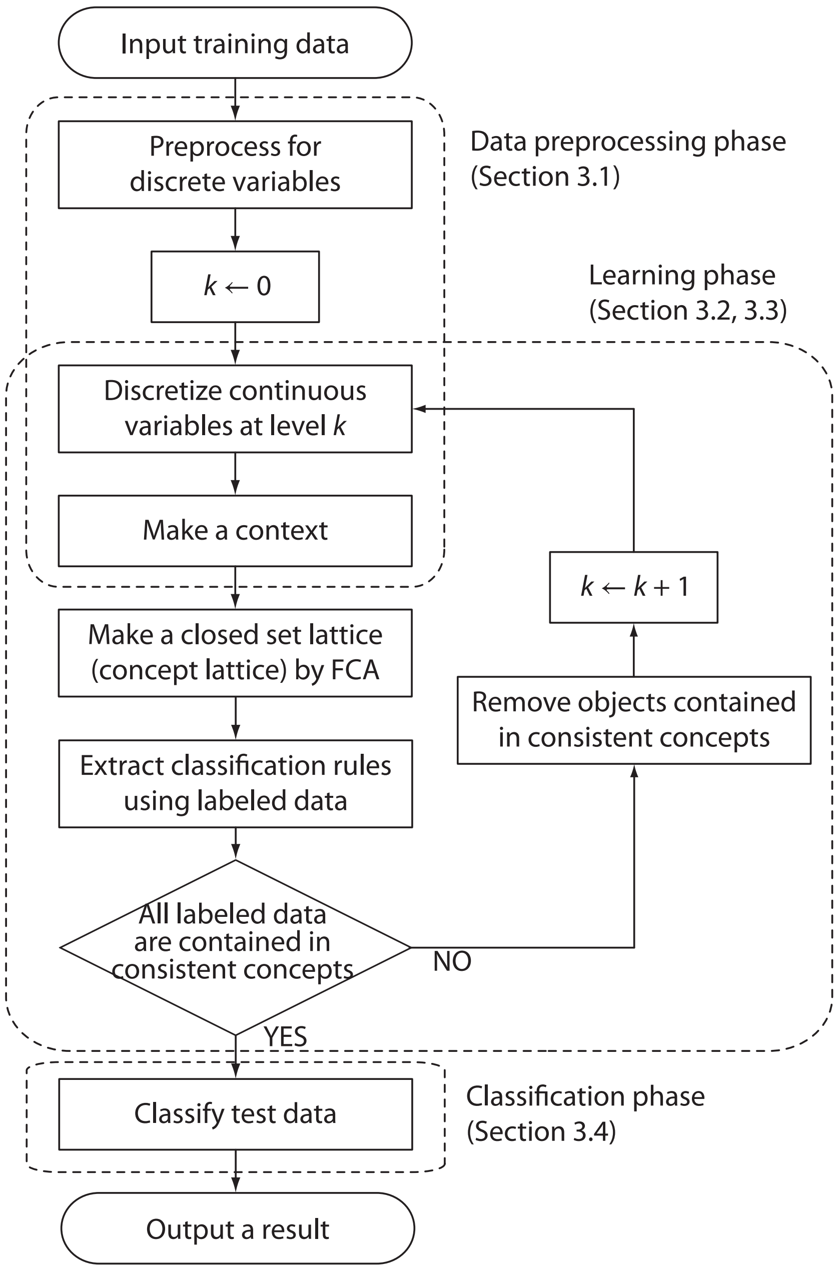
Figure 11

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.

Figure 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.

Figure 1



Input training data
Preprocess for discrete variables
Data preprocessing phase (Section 3.1)
$k \leftarrow 0$
Learning phase (Section 3.2, 3.3)
Discretize continuous variables at level k
Make a context
$k \leftarrow k+1$
Make a closed set lattice (concept lattice) by FCA
Remove objects contained in consistent concepts
Extract classification rules using labeled data
All labeled data are contained in consistent concepts
NO
YES
Classify test data
Classification phase (Section 3.4)
Output a result

Figure 2

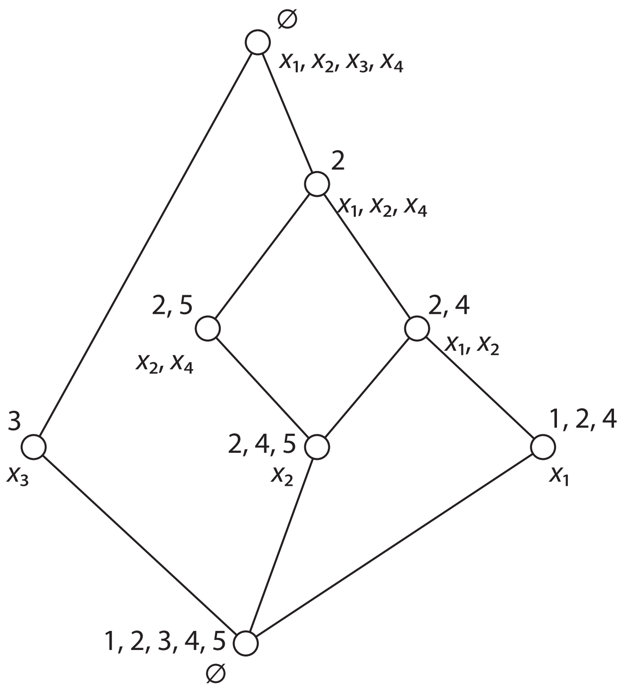
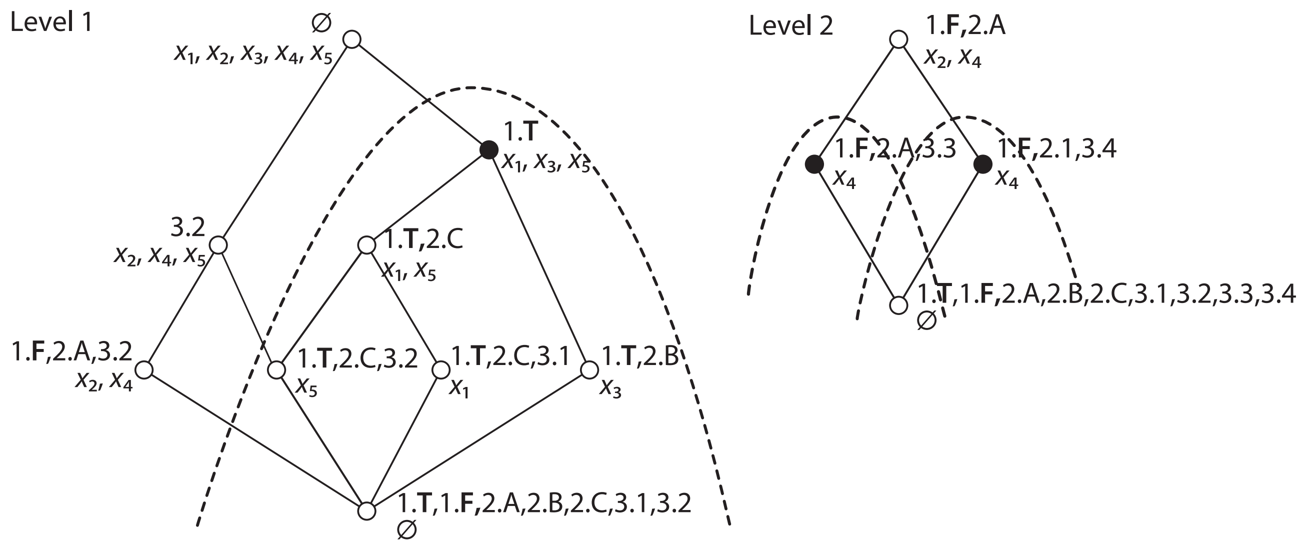


Figure 3



Level 1
∅
x_1, x_2, x_3, x_4, x_5
1.**T**
x_1, x_3, x_5
3.2
x_2, x_4, x_5
1.**T**,2.C
x_1, x_5
1.**F**,2.A,3.2
x_2, x_4
1.**T**,2.C,3.2
x_5
1.**T**,2.C,3.1
x_1
1.**T**,2.B
x_3
1.**T**,1.**F**,2.A,2.B,2.C,3.1,3.2
∅
Level 2
1.**F**,2.A
x_2, x_4
1.**F**,2.A,3.3
x_4
1.**F**,2.1,3.4
x_4
1.**T**,1.**F**,2.A,2.B,2.C,3.1,3.2,3.3,3.4
∅

Figure 4

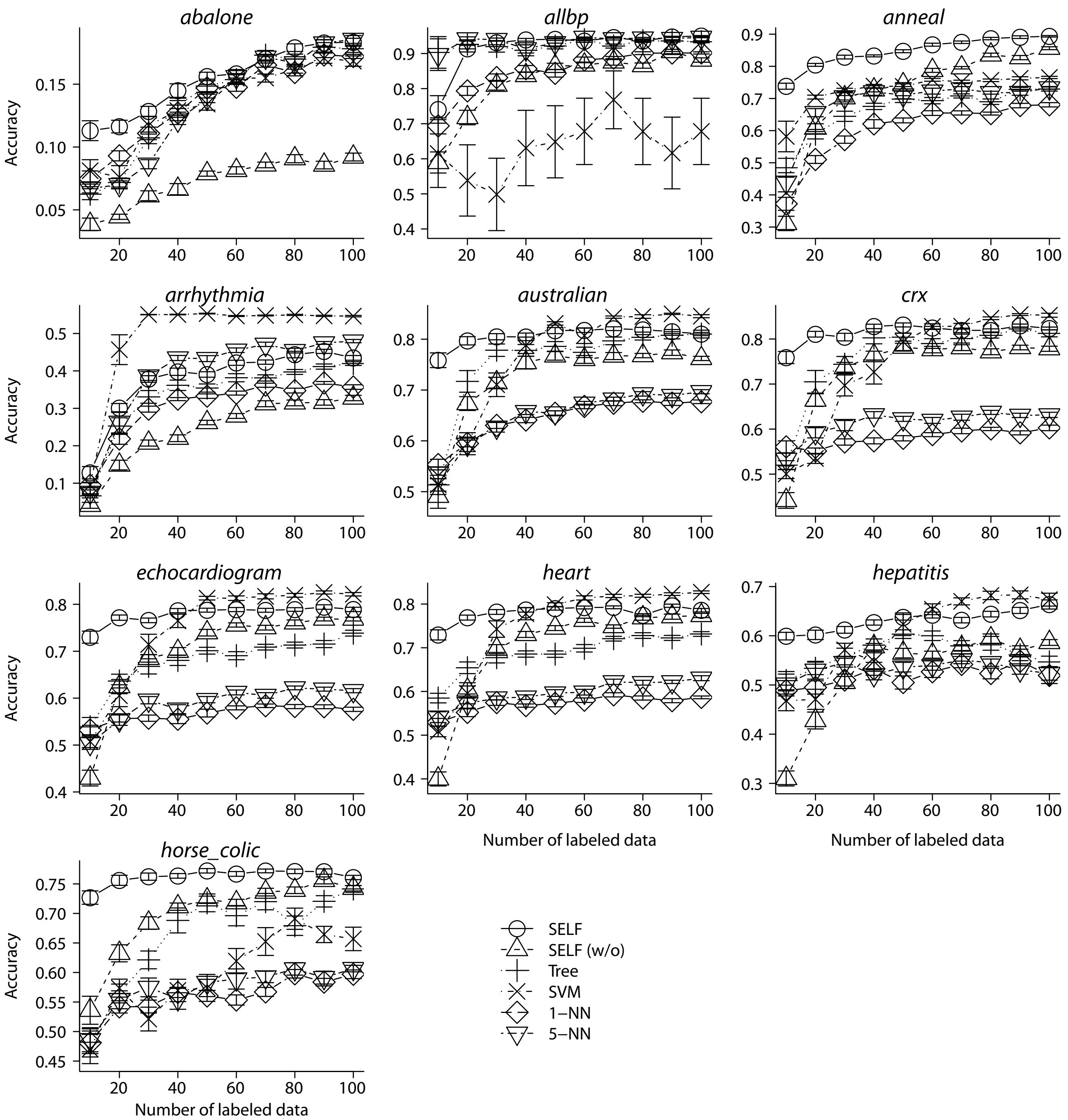
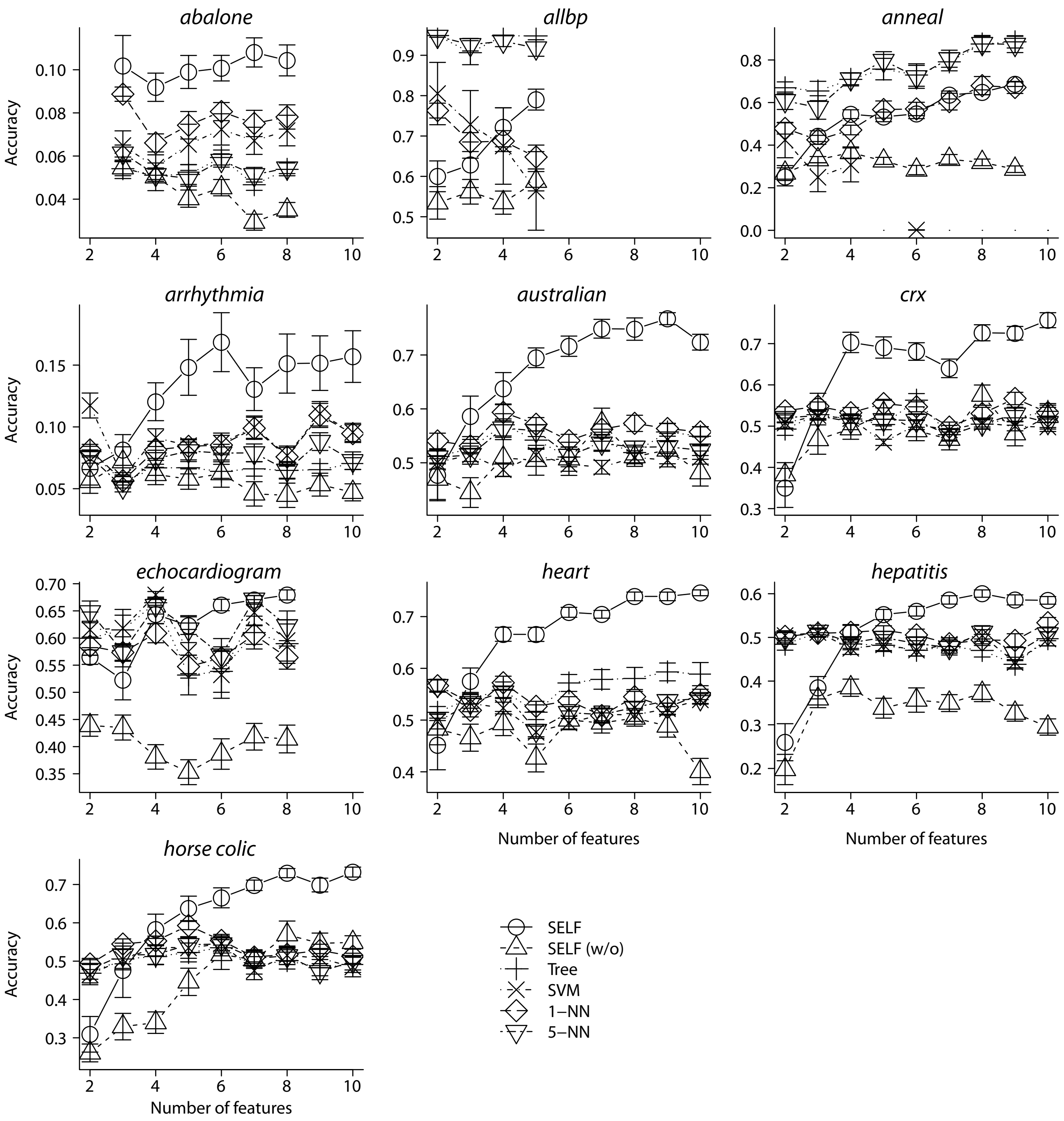
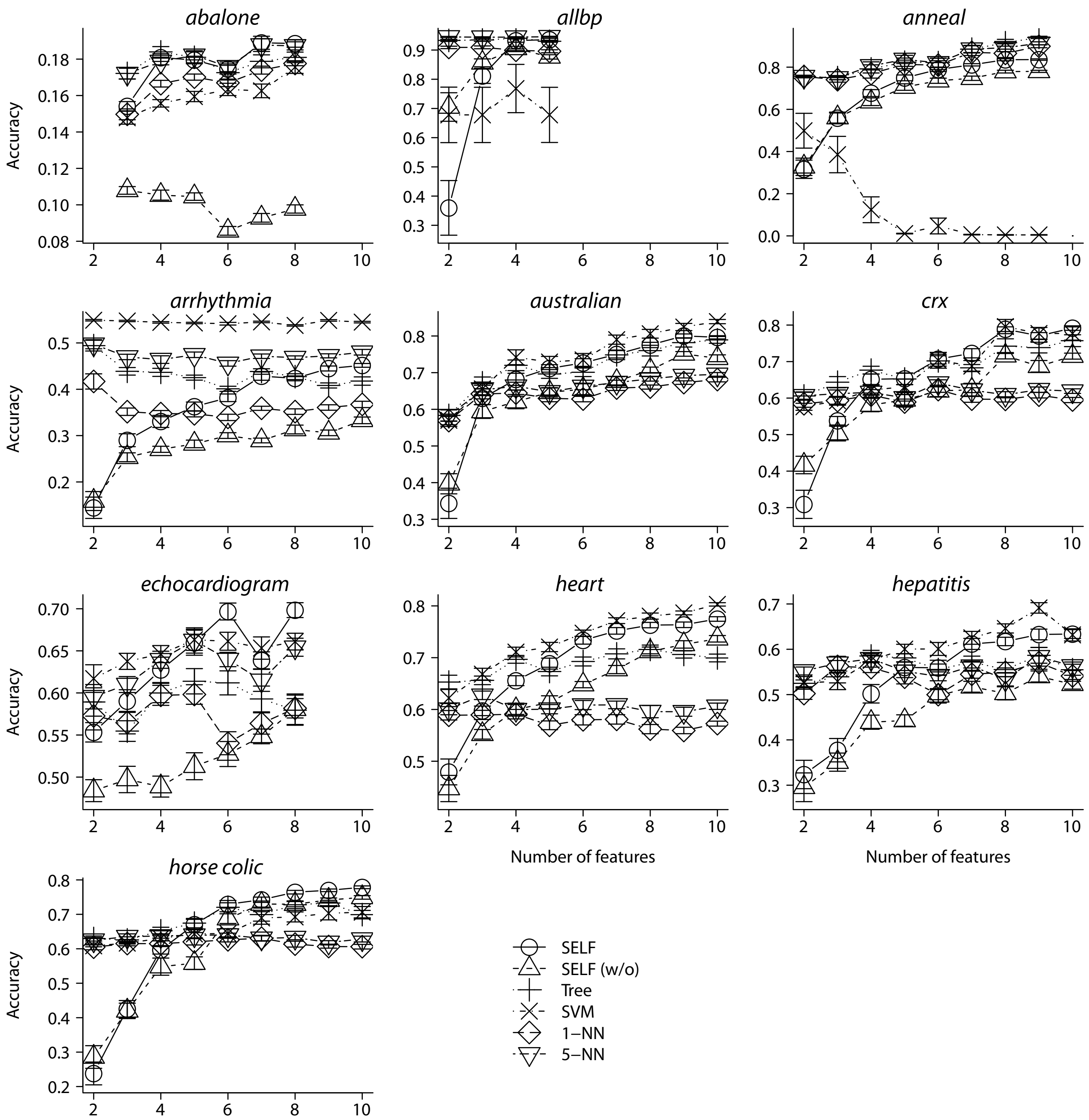

abalone
allbp
anneal
arrhythmia
australian
crx
echocardiogram
heart
hepatitis
horse_colic
Accuracy
Number of labeled data
SELF
SELF (w/o)
Tree
SVM
1−NN
5−NN

Figure 5



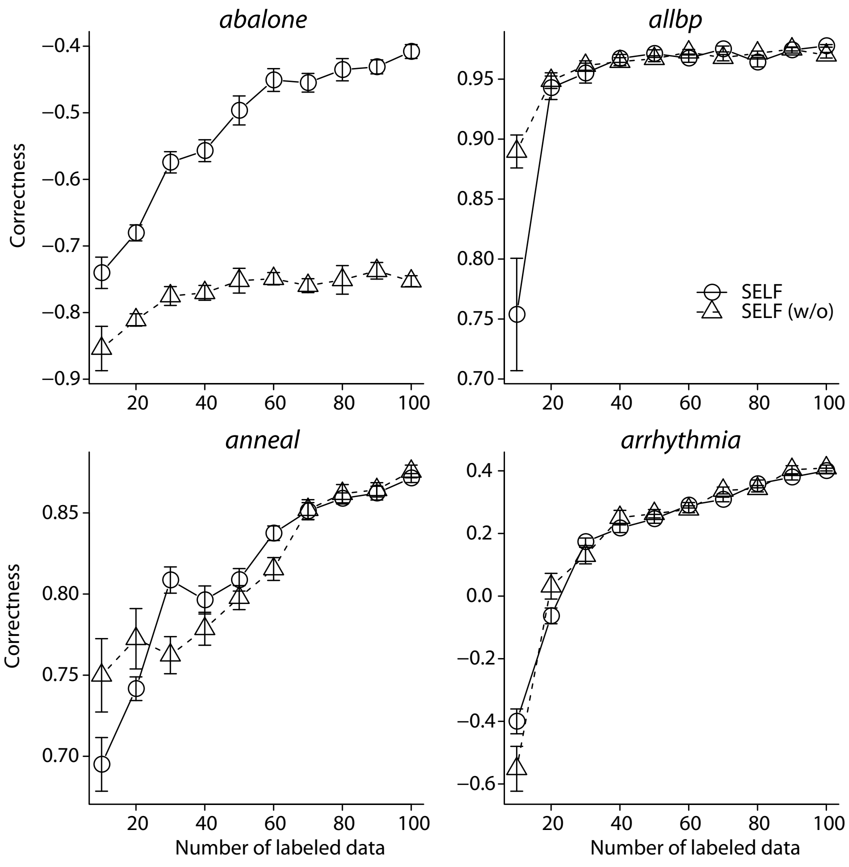
abalone
allbp
anneal
arrhythmia
australian
crx
echocardiogram
heart
hepatitis
horse colic
Accuracy
Number of features
SELF
SELF (w/o)
Tree
SVM
1–NN
5–NN

Figure 6



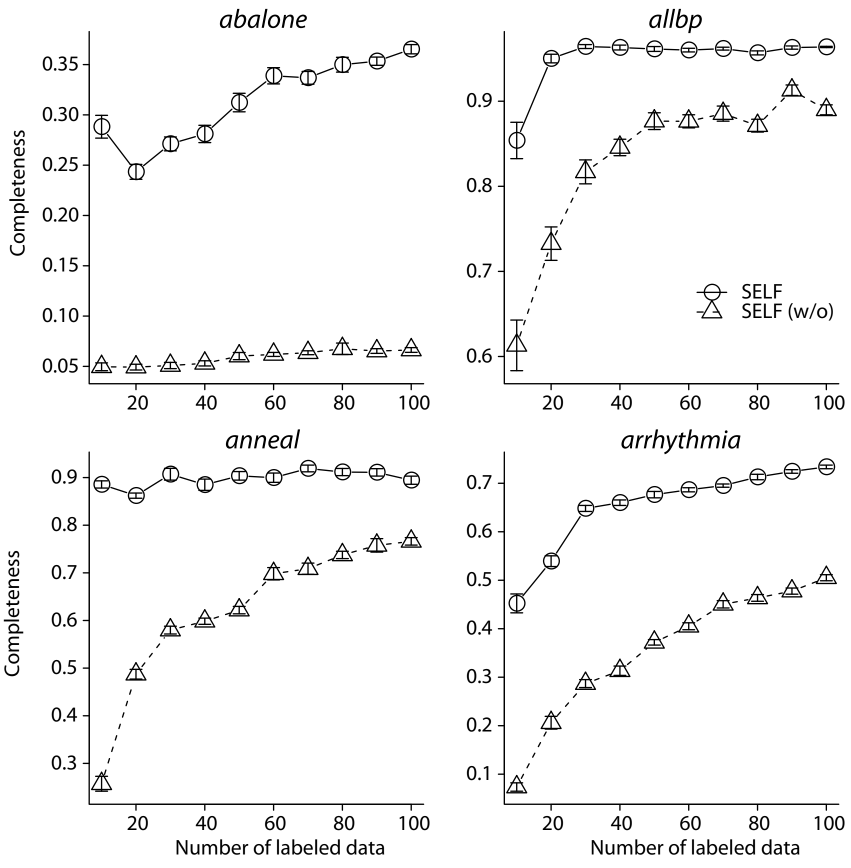
abalone
allbp
anneal
arrhythmia
australian
crx
echocardiogram
heart
hepatitis
horse colic
Accuracy
Number of features
SELF
SELF (w/o)
Tree
SVM
1–NN
5–NN

Figure 7



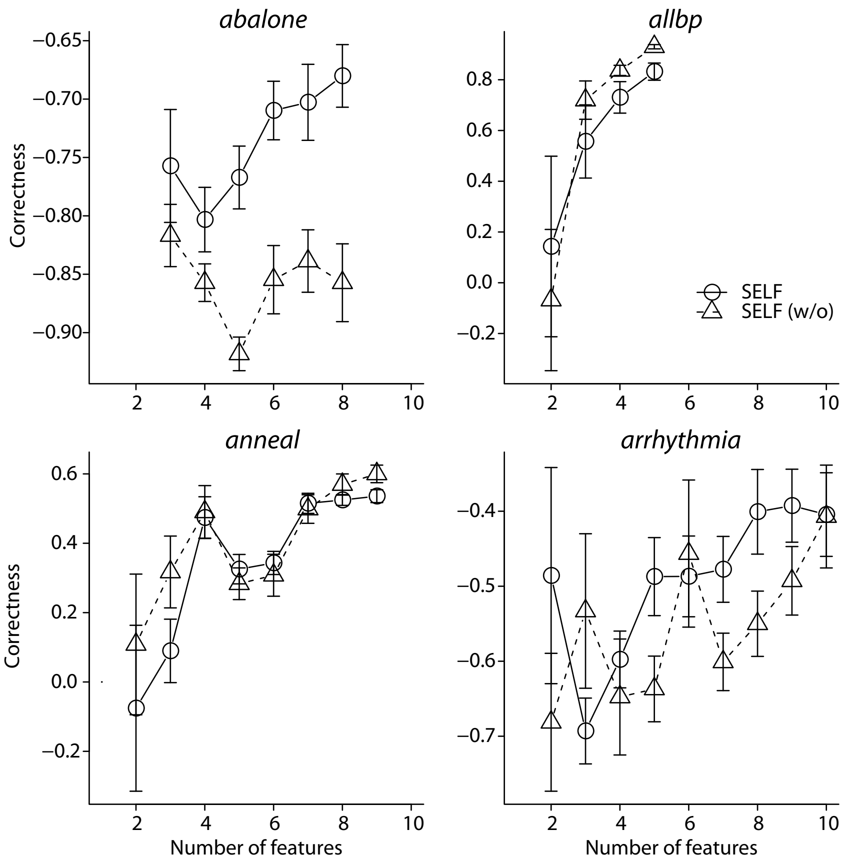
abalone
allbp
anneal
arrhythmia
Correctness
Correctness
Number of labeled data
Number of labeled data
SELF
SELF (w/o)

Figure 8



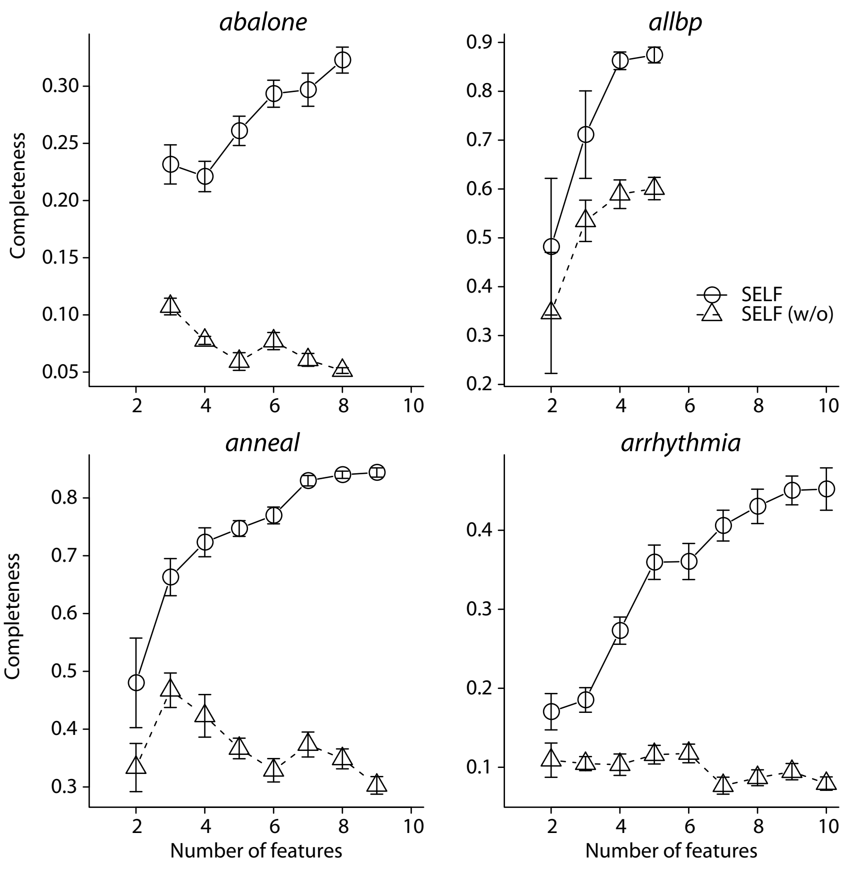
abalone
allbp
anneal
arrhythmia
Completeness
Number of labeled data
SELF
SELF (w/o)

Figure 9



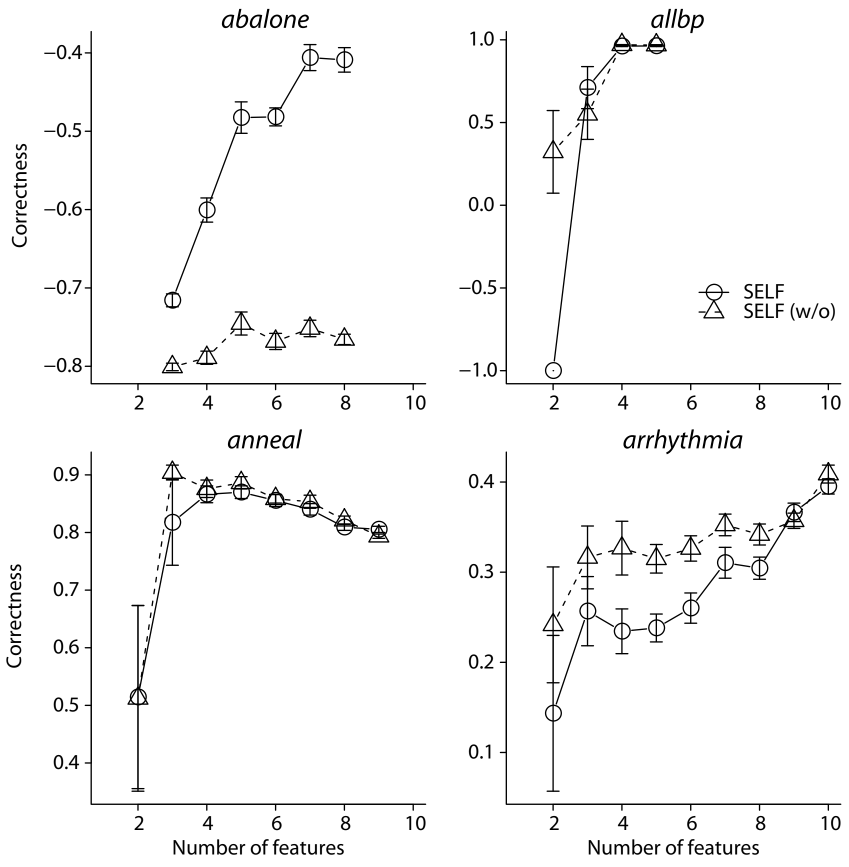
abalone
allbp
anneal
arrhythmia
Correctness
Number of features
SELF
SELF (w/o)

Figure 10



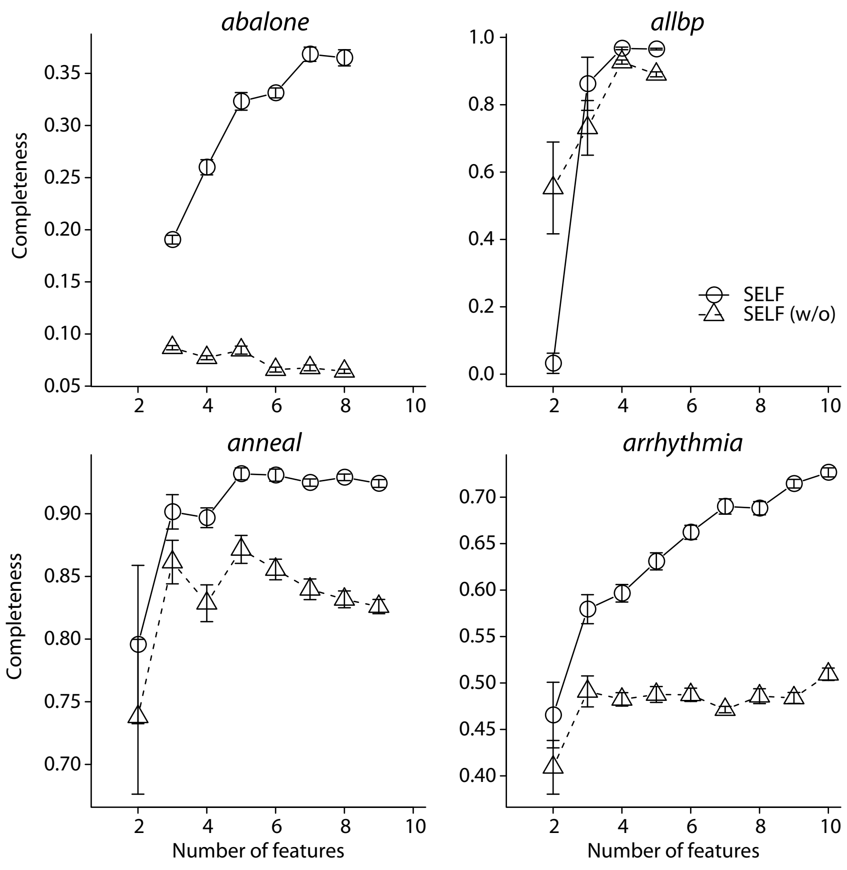
abalone
allbp
anneal
arrhythmia
Completeness
Number of features
SELF
SELF (w/o)

Figure 11



abalone
allbp
anneal
arrhythmia
Correctness
Number of features
SELF
SELF (w/o)

Figure 12


abalone
allbp
anneal
arrhythmia
Completeness
Number of features
SELF
SELF (w/o)