IPSJ Transactions on Mathematical Modeling and Its Applications Vol.5 No.2 39–48 (June 2012)

Regular Paper

Semi-Supervised Ligand Finding Using Formal Concept Analysis

Mahito Sugiyama[1,†1,a)] Kentaro Imajo[1] Keisuke Otaki[1]
Akihiro Yamamoto[1]

Received: November 4, 2011, Revised: December 7, 2011,
Accepted: February 28, 2012

Abstract: To date, enormous studies have been devoted to investigate biochemical functions of *receptors*, which have crucial roles for signal processing in organisms. *Ligands* are key tools in experiments since receptor specificity with respect to them enables us to control activity of receptors. However, finding ligands is difficult; choosing ligand candidates relies on expert knowledge of biologists and conducting test experiments *in vivo* or *in vitro* has a high cost. Here we investigate the ligand finding problem with a machine learning approach by formalizing the problem as *multi-label classification* mainly discussed in the area of *preference learning*. We develop in this paper a new algorithm LIFT (Ligand FInding via Formal ConcepT Analysis) for multi-label classification, which can treat ligand data in databases in a *semi-supervised* manner. The key to LIFT is to achieve clustering by putting an original dataset on *lattices* using the data analysis technique of *Formal Concept Analysis* (FCA), followed by obtaining the preference for each label using the lattice structure. Experiments using real data of ligands and receptors in the IUPHAR database show that LIFT effectively solves the task compared to other machine learning algorithms.

Keywords: semi-supervised learning, preference learning, formal concept analysis, ligand, receptor

1. Introduction

A *receptor* is a protein molecule located at the surface of a cell, which receives chemical signals from outside of the cell. Since receptors have crucial roles for signal processing in organisms, to date, enormous studies have been devoted to investigate their biochemical functions. The key approach in an experiment is to use receptor specificity with respect to a *ligand*, which triggers a cellular response by binding to a receptor, for controlling the receptor actions (**Fig. 1**). However, finding new convenient ligands is difficult; choosing ligand candidates relies on expert knowledge of biologists and conducting experiments to test whether or not candidates work *in vivo* or *in vitro* has a high cost in terms of time and money. Thus an *in silico* approach is required for helping biologists.

In this paper, we adopt a machine learning, or knowledge discovery and data mining, approach to find candidates of ligands. Specifically, we formulate the problem of ligand finding as *multi-label classification* recently discussed in the field of *preference learning* [9], where each training datum used for learning is associated with not a single class label but a set of possible labels. Here, for each ligand, receptors to which it binds correspond to class labels of the ligand, and our goal is to predict labels (i.e., receptors) of ligands from databases of receptors and ligands. A ligand can often bind to more than two receptors; this is why our problem is not traditional single-label but multi-label classification. Moreover, we try to predict labels in a *semi-supervised* manner [3], [31]. Semi-supervised learning is a special form of classification, where a learning algorithm uses both labeled and unlabeled data in training. Commonly, only few labeled data are assumed to be available since the labeling task has a high cost in a real situation. Semi-supervised learning therefore fits to our goal since, in our problem, only few ligands for each receptor have been discovered yet lots of ligands for other receptors are available.

Formally, the problem of semi-supervised multi-label classification is stated as follows: Given a sequence $X = x_1, x_2, \ldots, x_n$, where each x_i is a *tuple*, or a *feature vector*, and a domain of labels $\mathcal{L}$. Each tuple x_i is associated with a set of labels $L_i \subseteq \mathcal{L}$. Since we consider semi-supervised learning, $L_i = \emptyset$ is allowed. The goal is, for any tuple (test datum) y, to predict the *preference* of labels with respect to y, and we can decide whether or not y is associated with a label λ for each $\lambda \in \mathcal{L}$.

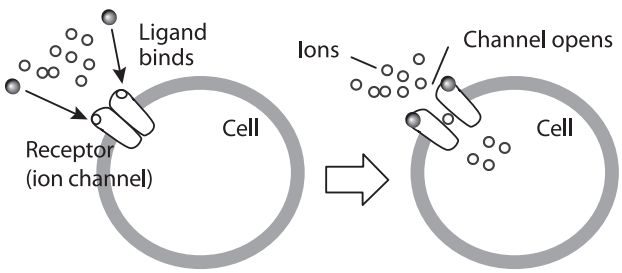


Fig. 1 A ligand-gated ion channel, which is a typical receptor.

[1] Graduate School of Informatics, Kyoto University, Kyoto 606–8501, Japan
[†1] Presently with the Institute of Scientific and Industrial Research, Osaka University
[a)] mahito@ar.sanken.osaka-u.ac.jp

Information about receptors and ligands is donated to various databases, such as KEGG[*1], and in this paper we use the IUPHAR database [25][*2]. In the database, every ligand is characterized by seven features as follows: hydrogen bond acceptors, hydrogen bond donors, rotatable bonds, topological polar surface area, molecular weight, XLogP, and number of Lipinski's rules broken. We abbreviate them in this paper HBA, HBD, RB, TPS, MW, XLogP, and NLR, respectively. Here, TPS, MW, and XLogP take *continuous* (real-valued) values while the others, HBA, HBD, RB, and NLR, take *discrete* values. Thus to design an effective classifier for the IUPHAR database, we have to appropriately treat *mixed-type data* including both discrete and continuous variables.

Recently, semi-supervised learning is one of active research fields in machine learning and knowledge discovery, and now various semi-supervised learning algorithms have already been developed [3], [31]. However, most of them are designed for real-valued variables and cannot be applied to mixed-type data directly and, moreover, they do not treat multi-label classification. Therefore, in this paper, we construct a new learning algorithm, called LIFT (Ligand FInding via Formal ConcepT Analysis), which is designed for semi-supervised multi-label classification of mixed-type data and hence it solves the ligand finding problem from the IUPHAR database. The basic strategy of LIFT is similar to the learning method SELF [27], proposed by two of the authors, which directly handles mixed-type data in the semi-supervised manner. Since SELF cannot treat multi-label classification, we redesign the essential algorithm of SELF to fit to multi-label classification.

LIFT uses "label propagation," or cluster-and-label, which is a typical approach in semi-supervised learning [5], [8]. This means that it first makes clusters without label information, followed by giving preferences of class labels for each cluster. In LIFT, the clustering process is performed by *Formal Concept Analysis* (FCA) [7], [11], which is a mathematical data analysis technique originally proposed by Wille [29]. One of successful applications of FCA in data mining is for frequent pattern and association rule mining proposed by Pasquier et al. [22], where closed patterns (itemsets) obtained by FCA is used as condensed "lossless" representations of original patterns. Using FCA, informally, we can introduce a lattice structure, called a *concept lattice* or a *closed set lattice*, which is a partially ordered set of data clusters with respect to subset inclusion, into original data. Many studies used FCA for machine learning and knowledge discovery, such as classification [10], clustering [30], and bioinformatics [2], [17], [19], but ligand finding presented in this paper is a novel application of FCA.

To date, no study treats machine learning for ligand finding in the (multi-class) classification point of view. Recently, to the best of our knowledge, there exists only one related study by Ballester and Mitchell [1], which investigated a machine learning approach to predict the *affinity* of ligands, the strength of docking. Another approach was performed by King et al. [18] for modeling structure-activity relationships (SAR), which can be applied to ligand finding. However, their goal is to understand the chemical model by describing relations using inductive logic programming (ILP), thus their approach is different from ours. On the other hand, most *in silico* studies about receptors and ligands tried to construct predictive models using domain-specific knowledge, such as the potential energy of a complex, the two-dimensional co-ordinates, and the free energy of binding [14], [21], and lots of scoring methods were proposed; e.g., AMBER [4], AutoDock [15], and DrugScore [12]. However, to use such a method, some domain-specific background knowledge is required and results strongly depend on them. In contrast, our approach relies on only databases, thereby the user does not need any background knowledge and can easily use and understand results.

This paper is organized as follows: Section 2 presents the LIFT algorithm. Section 3 gives experimental results with methodologies and discussion. Finally we summarize our results and discuss our future work in Section 4.

2. LIFT Algorithm

We present the algorithm, LIFT (LIgand Finding via Formal ConcepT Analysis), which is the main part of the paper. Notations used in this paper are summarized in **Table 1**.

2.1 Database Formalization

We treat a ligand database using the notion of a relational database [6], [26]. A set of ligands is treated as a *table*, or *relation*, τ which is a pair (H, X) of a *header* H and a *body* X. A header H is a finite set of feature[*3] names, where each $h \in H$ is referred to as the *domain* of h, denoted by $\mathrm{Dom}(h)$; a body X is a sequence of *tuples* $x_1, x_2, \ldots, x_n$, where each tuple x_i is defined as 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 $h \in H$. We denote the number of tuples, the table size, n by $|\tau|$. When we treat the body X as a set, we denote it by $\mathrm{set}(X)$, that is, $\mathrm{set}(X) = \{x_1, x_2, \ldots, x_n\}$. This means that we do not take the order and multiplicity into account in $\mathrm{set}(X)$.

In the IUPHAR database, the header H is always the set {HBA, HBD, RB, TPS, MW, XLogP, NLR}, and

$$\mathrm{Dom(HBA)} = \mathrm{Dom(HBD)} = \mathrm{Dom(RB)} = \mathrm{Dom(NLR)} = \mathbb{N},$$
$$\mathrm{Dom(TPS)} = \mathrm{Dom(MW)} = \mathrm{Dom(XLogP)} = \mathbb{R},$$

where $\mathbb{N}$ and $\mathbb{R}$ denote the set of natural numbers and real numbers, respectively.

Let J be a subset of the header H. For each tuple x, the *projection* of x on J, denoted by $x|_J$, is exactly the same as the restriction of x to J, which is the function from J to $\mathrm{Dom}(H)$ such that $x|_J(h) = x(h)$ for all $h \in J$. For a table $\tau = (H, X)$, the projection of τ is the table $\tau|_J = (J, X|_J)$, where $X|_J$ is defined by $X|_J := x_1|_J, x_2|_J, \ldots, x_n|_J$.

2.2 Data Preprocessing for FCA

First LIFT performs data preprocessing to construct a (formal)

[*1] http://www.genome.jp/kegg/
[*2] http://www.iuphar-db.org/index.jsp
[*3] It is usually called an *attribute*, but we use the word *feature* to avoid confusion with an attribute of a context in FCA.

Table 1 Notation.

$\tau = (H, X)$	Table, which is pair of header H and body X
set(X)	The set of tuples of body X
h	Feature (element in H)
x, y	tuple
$x\|_J$	Projection of x on $J \subseteq H$
$\|\tau\|$	Number of tuples
Dom(h)	Domain of feature h
G	The set of objects
M	The set of attributes
I	Binary relation between G and M
(G, M, I)	Context
$\mathcal{B}(G, M, I)$	Concept lattice (the set of concepts) of context (G, M, I)
k	Discretization level
$G(\tau)$	The set of objects generated from τ
$M^k(\tau)$	The set of attributes generated from τ at discretization level k
$I^k(\tau)$	Binary relation generated from τ at discretization level k
g	Object
m	Attribute
$h.m$	Qualified attribute generated from feature h
$''$	Closure operator
$\mathcal{B}(\tau)$	Concept lattice generated from table τ
$k_{\max}$	Maximum level
$\Lambda(x)$	Set of labels associated with tuple x
λ	Label
$\mathcal{L}$	The domain of labels
$\psi_y^k(\lambda\|\tau)$	Preference of label λ at level k for tuple y with respect to τ
$\psi_y(\lambda\|\tau)$	Preference of label λ for tuple y with respect to τ (abbreviated as $\psi_y(\lambda)$ if τ is understood from context)

Algorithm 1: Data preprocessing for making context
Input: Table $\tau = (H, X)$ and discretization level k
Output: Context $(G(\tau), M^k(\tau), I^k(\tau))$

function ContextC(τ, k)
1: $G \leftarrow$ set(X)
2: **for each** feature $h \in H$
3: **if** Dom(h) = $\mathbb{N}$ **then** $(M_h, I_h) \leftarrow$ ContextD(X, h)
4: **else if** Dom(h) = $\mathbb{R}$ **then** $(M_h, I_h) \leftarrow$ ContextC(X, h, k)
5: **end if**
6: combine $(M_{\mathrm{HBA}}, I_{\mathrm{HBA}}), (M_{\mathrm{HBD}}, I_{\mathrm{HBD}}), \ldots, (M_{\mathrm{NLR}}, I_{\mathrm{NLR}})$ into $(M^k(\tau), I^k(\tau))$
7: **return** $(G(\tau), M^k(\tau), I^k(\tau))$

function ContextD(X, h)
1: $M \leftarrow \{h.m \mid m \in x(h)$ such that $x \in$ set(X)$\}$
2: $I \leftarrow \{(x, h.m) \mid x \in$ set(X) and $x(h) = m\}$
3: **return** (M, I)

function ContextC(X, h, k)
1: $M \leftarrow \{1, 2, \ldots, 2^k\}$, $I \leftarrow \emptyset$
2: Normalize the set $\{x(h) \mid x \in$ set(X)$\}$
3: **for each** $x \in$ set(X)
4: **if** $x(h) = 0$ **then** $I \leftarrow I \cup \{(x, h.1)\}$
5: **else if** $x(h) \neq 0$ **then**
6: $I \leftarrow I \cup \{(x, h.a)\}$, where $(a-1) \cdot 2^{-k} < x(h) \leq a \cdot 2^{-k}$
7: **end if**
8: **end for**
9: **return** (M, I)

context, a binary matrix specifying a set of objects and their attributes, to apply FCA to training data. As described in the previous subsection, a ligand database is represented by a sequence of tuples with seven features including both discrete and continuous variables, and LIFT applies a different preprocess to each feature type.

FCA is a mathematical data analysis technique [7], [11], which is applied to a triplet (G, M, I), called a (formal) *context*, where 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*. In this paper, we identify a tuple with an object, hence the set of objects G is always set(X) $= \{x_1, x_2, \ldots, x_n\}$.

From a given table (dataset), LIFT independently constructs seven pairs of attributes and binary relations $(M_{\mathrm{HBA}}, I_{\mathrm{HBA}})$, $(M_{\mathrm{HBD}}, I_{\mathrm{HBD}})$, …, $(M_{\mathrm{NLR}}, I_{\mathrm{NLR}})$ for each feature in the header H and combines 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 $h.m$.

First, we focus on preprocessing for discrete values of features HBA, HBD, RB, and NLR. Since a context is also a discrete representation of a dataset, this process is directly achieved in the following manner. For each feature $h \in H$, the set of attributes

$$M_h = \{h.m \mid m \in x(h) \text{ such that } x \in \mathrm{set}(X)\}$$

and, for each $x \in \mathrm{set}(X)$, $(x, h.m) \in I_h$ if and only if $x(h) = m$. In this way, discrete values are translated into a context. The function ContextD in Algorithm 1 performs this translation.

Second, we describe how to make a context from continuous values using *discretization*. Since a context has a discrete structure, we need to discretize original continuous values. We embed such a discretization process in the learning process and increase discretizing resolution along with the learning process. The degree of resolution is denoted by a natural number k, called *discretization level*, and we explain the method of discretization at fixed level k in the following. First we use *min-max normalization* [13] so that every value is in the closed interval $[0, 1]$. Namely, for every feature $h \in H$ and tuple x, the value $x(h)$ is mapped to the 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, which is the same approach as Ref. [27]. At discretization level k,

$$M_h = \{h.1, h.2, \ldots, h.2^k\}.$$

For each tuple x and feature h, if $x(h) = 0$, then $(x, h.1) \in I_h$. Otherwise if $x(h) \neq 0$, then $(x, h.a) \in I_h$ if and only if

$$\frac{a-1}{2^k} < x(h) \leq \frac{a}{2^k}. \tag{1}$$

This means that continuous values are encoded by the binary encoding scheme, that is, if we encode a real number $x(h)$ into an infinite sequence $p = p_0 p_1 p_2 \ldots$, a context at discretization level k is decided by the first k bits $p_0 p_1 \ldots p_{k-1}$. Each value is converted to exactly one relation of a context.

The function ContextC in Algorithm 1 performs the above process to make a context from continuous variables. In the following, for a given table τ, we write $G(\tau)$, $M^k(\tau)$, and $I^k(\tau)$ for the set of objects, attributes, and binary relations at discretization level k obtained by Algorithm 1, respectively.

Example 1 Given a table $\tau = (H, X)$, where $H = \{\mathrm{HBD}, \mathrm{TPS}, \mathrm{MW}\}$ and $X = x_1, x_2, x_3$ such that

$$(x_1(\mathrm{HBD}), x_1(\mathrm{TPS}), x_1(\mathrm{MW})) = (0, 0.61, 0.98),$$
$$(x_2(\mathrm{HBD}), x_2(\mathrm{TPS}), x_2(\mathrm{MW})) = (0, 0.44, 0.74),$$
$$(x_3(\mathrm{HBD}), x_3(\mathrm{TPS}), x_3(\mathrm{MW})) = (1, 0.72, 0.34)$$

as shown in **Table 2**. Let discretization level $k = 1$. Then, for each feature, we have the context as follows:

$$M_{\mathrm{HBD}} = \{\mathrm{HBD}.0, \mathrm{HBD}.1\},$$
$$I_{\mathrm{HBD}} = \{(x_1, \mathrm{HBD}.0), (x_2, \mathrm{HBD}.0), (x_3, \mathrm{HBD}.1)\},$$
$$M_{\mathrm{TPS}} = \{\mathrm{TPS}.1, \mathrm{TPS}.2\},$$
$$I_{\mathrm{TPS}} = \{(x_1, \mathrm{TPS}.2), (x_2, \mathrm{TPS}.1), (x_3, \mathrm{TPS}.2)\},$$
$$M_{\mathrm{MW}} = \{\mathrm{MW}.1, \mathrm{MW}.2\},$$
$$I_{\mathrm{MW}} = \{(x_1, \mathrm{MW}.2), (x_2, \mathrm{MW}.2), (x_3, \mathrm{MW}.1)\}.$$

Thus we have the context such that

Table 2 A table and a context in Example 1.

H		HBD	TPS	MW
	x_1	0	0.61	0.98
X	x_2	0	0.44	0.74
	x_3	1	0.72	0.34

	HBD.0	HBD.1	TPS.1	TPS.2	MW.1	MW.2
x_1	×			×		×
x_2	×		×			×
x_3		×		×	×	

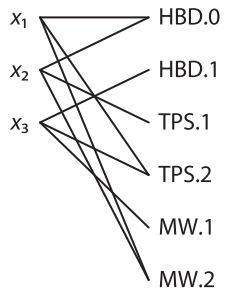


Fig. 2 The bipartite graph corresponding to the upper context.

$$G(\tau) = \{x_1, x_2, x_3\},$$
$$M^1(\tau) = M_{\mathrm{HBD}} \cup M_{\mathrm{TPS}} \cup M_{\mathrm{MW}},$$
$$I^1(\tau) = I_{\mathrm{HBD}} \cup I_{\mathrm{TPS}} \cup I_{\mathrm{MW}},$$

which is visualized as a cross-table in Table 2. □

2.3 Lattice Construction Using FCA

From a context obtained by the data preprocessing, LIFT generates closed sets as clusters and constructs lattices of closed sets (concept lattices) by FCA. We first summarize FCA. See Refs. [7], [11] for detail explanation.

Definition 1 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 $A = B'$, where

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

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 a set of objects $A \subseteq G$ is *closed* and (A, B) is a concept for some set of attributes B if and only if $A'' = A$, and vice versa. Thus a set of objects A of a concept (A, B) can be viewed as a *cluster* determined by the algebraic property of "closed."

The set of concepts over (G, M, I), called the *concept lattice*, is written by $\mathcal{B}(G, M, I)$. In frequent pattern mining, a set of attributes corresponds to an itemset and the lattice is called the closed itemset lattice [22]. For a pair of concepts (A_1, B_1), $(A_2, B_2) \in \mathcal{B}(G, M, I)$, 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 $\mathcal{B}(G, M, I)$ in the mathematical sense and $\langle \mathcal{B}(G, M, I), \leq \rangle$ becomes a complete lattice. For a table τ, we denote the set of concepts $\mathcal{B}(G(\tau), M^k(\tau), I^k(\tau))$ by $\mathcal{B}^k(\tau)$. If domains of all features are discrete; i.e., $\mathrm{Dom}(h) = \mathbb{N}$ for all $h \in H$, we abbreviate the superscript k.

Table 3 A table τ in Example 3 and the context $(G(\tau), M^1(\tau), I^1(\tau))$, where feature names TPS, MW, and XLogP are abbreviated as T, M, and X, respectively.

H		TPS	MW	XLogP
	x_1	0.23	0.12	0.18
X	x_2	0.35	0.03	0.74
	x_3	0.41	0.79	0.91

	T.1	T.2	M.1	M.2	X.1	X.2
x_1	×		×		×	
x_2	×		×			×
x_3	×			×		×

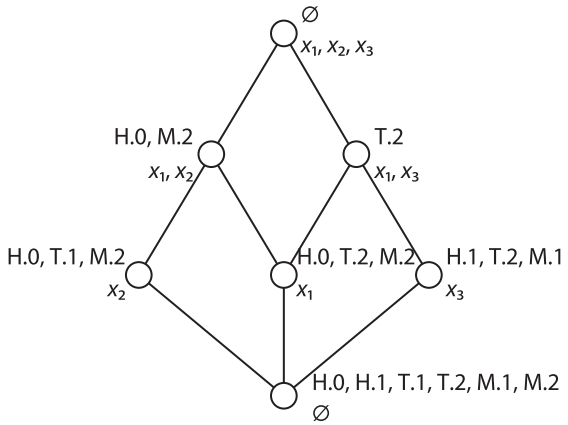


Fig. 3 A concept lattice constructed from the context in Table 2. Feature names HBD, TPS, and MW are abbreviated as H, T, and M, respectively.

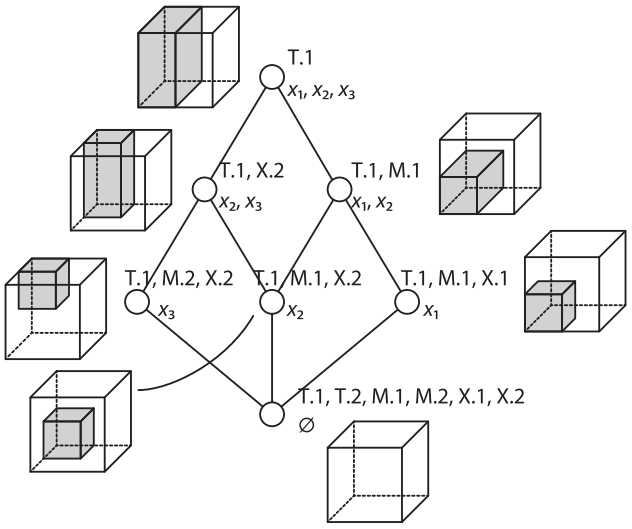


Fig. 4 The concept lattice $\mathcal{B}^1(\tau)$ with its geometric interpretation.

Example 2 Let us consider the table τ given in Example 1 and the context $(G(\tau), M^1(\tau), I^1(\tau))$ generated from τ. The set $\mathcal{B}^k(\tau)$ consists of seven concepts in total: ($\emptyset$, {HBD.0, HBD.1, TPS.1, TPS.2, MW.1, MW.2}), ($\{x_1\}$, {HBD.0, TPS.2, MW.2}), ($\{x_2\}$, {HBD.0, TPS.1, MW.2}), ($\{x_3\}$, {HBD.1, TPS.2, MW.1}), ($\{x_1, x_2\}$, {HBD.0, MW.2}), ($\{x_1, x_3\}$, {TPS.2}), and ($\{x_1, x_2, x_3\}$, $\emptyset$). We show the concept lattice in **Fig. 3**. □

Example 3 Let us consider a table $\tau = (H, X)$ such that H = {TPS, MW, XLogP} and $X = x_1, x_2, x_3$, where

$$(x_1(\text{TPS}), x_1(\text{MW}), x_1(\text{XLogP})) = (0.23, 0.12, 0.18),$$
$$(x_2(\text{TPS}), x_2(\text{MW}), x_2(\text{XLogP})) = (0.35, 0.03, 0.74),$$
$$(x_3(\text{TPS}), x_3(\text{MW}), x_3(\text{XLogP})) = (0.41, 0.79, 0.91),$$

which consists of only continuous values (**Table 3**). The concept lattice corresponds to a hierarchy of cubes in three-dimensional Euclidean space (**Fig. 4**). □

To obtain concept lattices, we use the algorithm developed by Makino and Uno [20], which is known to be one of the fastest such algorithms. Their algorithm enumerates all maximal bipartite cliques in a bipartite graph with $O(\Delta^3)$ delay, where Δ denotes the maximum degree of the given bipartite graph, i.e.,

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

($\#J$ is the number of elements in J). For instance, $\Delta = 3$ in the context $(G(\tau), M^1(\tau), I^1(\tau))$ in Example 1 (shown in Table 2). Since we can easily check that each context and concept exactly coincide with a bipartite graph and a maximal bipartite clique, respectively (**Fig. 2**), we can use their algorithm directly.

2.4 Classification and Ranking

Here we discuss classification on concept lattices using label information. Our strategy is to design *preference*, a kind of *weight*, for each label of a given test datum (unlabeled tuple) y based on concepts produced by FCA, and achieve multi-label classification based on the preference. Moreover, we show that label ranking can be achieved using the preference.

First LIFT translates y into a context with just one object using Algorithm 1; i.e., $G(\upsilon)$, $M^k(\upsilon)$, and $I^k(\upsilon)$, where $\upsilon = (H, y)$. We always assume that the header H is exactly the same as that of a table $\tau = (H, X)$ of training data.

The key idea is, for each concept $(A, B) \in \mathcal{B}^k(\tau)$ obtained from a table τ of training data, to treat the set of attributes B as a *classification rule*. For an unlabeled tuple y, we check whether or not the object y has the all attributes of the concept (A, B), since this condition means that the object y has the same properties of the objects A, meaning that y is classified to the same class of objects in A. We call this property *consistency* which is formally defined as follows:

Definition 2 (Consistency) For a context $(\{y\}, M, I)$ and a concept (A, B), the object y is *consistent* with (A, B) if two conditions $B \subseteq \{m \in M \mid (y, m) \in I\}$ and $B \neq \emptyset$ hold. □

Consistency has the monotonicity with respect to the order $\leq$ on a concept lattice. If an object y is consistent with a concept (A, B), it is consistent with any concept (C, D) such that $(A, B) \leq (C, D)$, and if an object y is not consistent with a concept (A, B), it is not consistent with any concept (C, D) such that $(C, D) \leq (A, B)$. Thus, for the set of concepts $\mathcal{B}^k(\tau)$, if we define

$$C(y) := \{(A, B) \in \mathcal{B}^k(\tau) \mid y \text{ is consistent with } (A, B)\},$$

there always exist finite concepts (A_1, B_1), (A_2, B_2), …, (A_l, B_l) such that

$$\bigcup_{i \in \{1,2,\ldots,l\}} \uparrow(A_i, B_i) = C(y),$$

where

$$\uparrow(A, B) = \{(C, D) \in \mathcal{B}^k(\tau) \mid (A, B) \leq (C, D)\}.$$

Here we give the formal definition of the preference of a label. We denote the set of labels associated with a tuple x by $\Lambda(x)$. Thereby, for a set of tuples (objects) A, $\Lambda(A)$ denotes the set $\bigcup_{x\in A} \Lambda(x)$. LIFT allows unlabeled data for training, hence $\Lambda(x)$ could be empty, meaning that the object x is unlabeled. This is why LIFT is a semi-supervised learning algorithm.

Definition 3 (Preference at discretization level k) Given tables $\tau = (H, X)$ and $\upsilon = (H, y)$ with $|\upsilon| = 1$. For each discretization level k and each label $\lambda \in \mathcal{L}$, we define the *preference of λ at discretization level k* with respect to the tuple y by

$$\psi_y^k(\lambda|\tau) := \sum\{\#\Lambda(A)^{-1} \mid y \text{ is consistent with } (A, B) \in \mathcal{B}^k(\tau) \text{ such that } \lambda \in \Lambda(A)\},$$

where $\#\Lambda(A)$ denotes the number of elements in $\Lambda(A)$, and we assume $\#\Lambda(A)^{-1} = 0$ if $\#\Lambda(A) = 0$ for simplicity. □

We do not take the size $\#A$ of the extent A into account, since the distribution of training data with respect to labels is often biased, especially in biological data.

Example 4 Let $\tau = (H, X)$ be a table in Example 3 (see Fig. 4) and tables $\upsilon = (H, y)$ and $\sigma = (H, z)$ be

$$(y(\text{TPS}), y(\text{MW}), y(\text{XLogP})) = (0.12, 0.41, 0.31),$$
$$(z(\text{TPS}), z(\text{MW}), z(\text{XLogP})) = (0.31, 0.22, 0.89).$$

Assume that $\Lambda(x_1) = \{\text{A}\}$, $\Lambda(x_2) = \{\text{B}\}$, and $\Lambda(x_3) = \{\text{C}\}$ in X. Binary relations $I^1(\upsilon)$ and $I^1(\sigma)$ at discretization level 1 for objects y and z are

$$I^1(\upsilon) = \{(y, \text{TPS.1}), (y, \text{MW.1}), (y, \text{XLogP.1})\},$$
$$I^1(\sigma) = \{(z, \text{TPS.1}), (z, \text{MW.1}), (z, \text{XLogP.2})\}.$$

The object y is consistent with three concepts ($\{x_1, x_2, x_3\}$, {TPS.1}), ($\{x_1, x_2\}$, {TPS.1, MW.1}), and ($\{x_1\}$, {TPS.1, MW.1, XLogP.1}), and z is consistent with four concepts ($\{x_1, x_2, x_3\}$, {TPS.1}), ($\{x_1, x_2\}$, {TPS.1, MW.1}), ($\{x_2, x_3\}$, {TPS.1, XLogP.2}), and ($\{x_2\}$, {TPS.1, MW.1, XLogP.2}). Thus we have the preference

$$\psi_y^1(\text{A}|\tau) = \frac{1}{3} + \frac{1}{2} + 1 = 1.83,\ \psi_y^1(\text{B}|\tau) = \frac{1}{3} + \frac{1}{2} = 0.83,$$
$$\psi_y^1(\text{C}|\tau) = \frac{1}{3} = 0.33,$$
$$\psi_z^1(\text{A}|\tau) = \frac{1}{3} + \frac{1}{2} = 0.83,\ \psi_z^1(\text{B}|\tau) = \frac{1}{3} + \frac{1}{2} + \frac{1}{2} + 1 = 2.33,$$
$$\psi_z^1(\text{C}|\tau) = \frac{1}{3} + \frac{1}{2} = 0.83.$$

These results of preferences reflect the similarity between data, since y and z are most similar to the first and second tuples of X, respectively. □

It is easy to perform multi-class classification from the preference at some fixed discretization level. However, this preference would not be enough to exploit information from obtained data. We show a simple representative case in the following, which shows the *anti-monotonicity* of the notion of consistency with respect to discretization level.

Table 4 A table τ for training with labels and a table υ as a test datum in Example 5, shown at the bottom of τ, and contexts at discretization levels 1 and 2, where HBD and TPS are abbreviated as H and T, respectively.

H		HBD	TPS	Labels
X	x_1	0	0.56	A
	x_2	0	0.91	B
y		0	0.11	

	H.0	T.1	T.2
x_1	×		×
x_2	×		×
y	×	×	

	H.0	T.1	T.2	T.3	T.4
x_1	×			×	
x_2	×				×
y	×	×			

Example 5 Let $\tau = (H, X)$ be a table such that $H = \{\text{HBD, TPS}\}$ and $X = x_1, x_2$, where

$$(x_1(\text{HBD}), x_1(\text{TPS})) = (0, 0.56),\ \Lambda(x_1) = \{\text{A}\}$$
$$(x_2(\text{HBD}), x_2(\text{TPS})) = (0, 0.91),\ \Lambda(x_2) = \{\text{B}\}$$

and $\upsilon = (H, y)$ be a table such that

$$(y(\text{HBD}), y(\text{TPS})) = (0, 0.11).$$

At discretization level 1, we have the context shown in **Table 4** and there are two concepts

$$(\{x_1, x_2\}, \{\text{HBD.0, TPS.2}\}),\ (\emptyset, \{\text{HBD.0, TSP.1, TPS.2}\}).$$

The object y is not consistent with any concept, hence

$$\psi_y^1(\text{A}|\tau) = 0,\ \psi_y^1(\text{B}|\tau) = 0.$$

However, at discretization level 2, there are four concepts

$$(\{x_1, x_2\}, \{\text{HBD.0}\}),\ (\{x_1\}, \{\text{HBD.0, TPS.3}\}),$$
$$(\{x_2\}, \{\text{HBD.0, TPS.4}\}),\ (\emptyset, \{\text{HBD.0, TSP.1, TPS.2}\}),$$

and y is consistent with the concept ($\{x_1, x_2\}$, {HBD.0}), thus

$$\psi_y^2(\text{A}|\tau) = 0.5,\ \psi_y^2(\text{B}|\tau) = 0.5.$$

Therefore y can be classified to both classes A and B. □

Ideally, all discretization levels should be taken into account to obtain the preference of labels. One of straightforward ways is to obtain the preference of a label by summing up preferences for each discretization level. However, if we define the preference by

$$\psi_y(\lambda|\tau) := \sum_{k\geq 1} \psi_y^k(\lambda|\tau),$$

this preference goes to infinity in many cases. We therefore introduce the maximum level $k_{\max}$ of discretization as a parameter.

Definition 4 (Preference) Given tables τ and υ, where $|\upsilon| = 1$, and a natural number $k_{\max}$. For each label $\lambda \in \mathcal{L}$, we define the *preference* of λ by

$$\psi_y(\lambda|\tau) := \sum_{k=1}^{k_{\max}} \psi_y^k(\lambda|\tau)$$

for a tuple y. □

We abbreviate "$|\tau$" of the expression $\psi_y(\lambda|\tau)$ if it is understood from context. We give the LIFT algorithm in Algorithm 2, which calculates the preference for each label.

Example 6 Let us consider a table $\tau = (H, X)$ with $H = \{\text{HBD,}$

Table 5 A table τ for training with labels and a table υ as a test datum, shown at the bottom of τ, in Example 6 and contexts at discretization levels 1 and 2, where HBD, TPS, and MW are abbreviated as H, T, and M, respectively.

H		HBD	TPS	MW	Labels	
	x_1	0	0.98	0.88	A	
X	x_2	1	0.41	0.48	B	C
	x_3	2	0.12	0.71	A	C
y		0	0.77	0.79		

	H.0	H.1	H.2	T.1	T.2	M.1	M.2
x_1	×				×		×
x_2		×		×		×	
x_3			×	×			×
y	×				×		×

	H.0	H.1	H.2	T.1	T.2	T.3	T.4	M.1	M.2	M.3	M.4
x_1	×						×				×
x_2		×			×				×		
x_3			×	×						×	
y	×						×				×

Algorithm 2: The LIFT algorithm
Input: Tables $\tau = (H, X)$ and $\upsilon = (H, y)$, and maximum level $k_{\max}$
Output: Preference ψ_y for each label $\lambda \in \mathcal{L}$

function LIFT(τ, υ, $k_{\max}$)
1: $k \leftarrow 1$ // k is discretization level
2: **for each** label $\lambda \in \mathcal{L}$
3: $\psi_y(\lambda|\tau) \leftarrow 0$ // initialization
4: **end for**
5: **return** LEARNING($\tau, \upsilon, k, k_{\max}$)

function LEARNING($\tau, \upsilon, k, k_{\max}$)
1: $(G(\tau), M^k(\tau), I^k(\tau)) \leftarrow$ CONTEXT(τ, k) // make a context from τ
2: $(G(\upsilon), M^k(\upsilon), I^k(\upsilon)) \leftarrow$ CONTEXT(υ, k) // make a context from υ
3: make a concept lattice $\mathcal{B}^k(\tau)$ from $(G(\tau), M^k(\tau), I^k(\tau))$ by FCA
4: **for each** label $\lambda \in \mathcal{L}$
5: compute the preference $\psi_y^k(\lambda|X)$ at discretization level k
6: $\psi_y(\lambda|X) \leftarrow \psi_y(\lambda|X) + \psi_y^k(\lambda|X)$
7: **end for**
8: **if** $k = k_{\max}$ **then**
9: **return** $(\psi_y(\lambda|\tau))_{\lambda\in\mathcal{L}}$
10: **else**
11: **return** LEARNING($\tau, \upsilon, k+1, k_{\max}$)
12: **end if**

RB, TPS, MW}, where labels are associated with each tuple as shown in **Table 5**, and a table $\upsilon = (H, y)$ with an unlabeled tuple y. Assume that $k_{\max} = 2$. At discretization level 1, we have

$$\psi_y^1(\mathrm{A}) = 1.5,\ \psi_y^1(\mathrm{B}) = 0,\ \text{and}\ \psi_y^1(\mathrm{C}) = 0.5,$$

since y is consistent with two concepts

$$(A_1, B_1) = (\{x_1, x_3\}, \{\mathrm{MW.2}\}) \text{ and}$$
$$(A_2, B_2) = (\{x_1\}, \{\mathrm{HBD.0}, \mathrm{TPS.2}, \mathrm{MW.2}\}),$$

where $\Lambda(A_1) = \{\mathrm{A}, \mathrm{C}\}$ and $\Lambda(A_2) = \{\mathrm{A}\}$ (see **Fig. 5**). Remember that we always ignore the concept whose attribute is empty. At discretization level 2, we have

$$\psi_y^2(\mathrm{A}) = 1, \psi_y^2(B) = 0,\ \text{and}\ \psi_y^2(\mathrm{C}) = 0,$$

since y is consistent with only one concept $(\{x_1\}, \{\mathrm{HBD.0}, \mathrm{TPS.4}, \mathrm{MW.4}\})$. Finally we have

$$\psi_y(\mathrm{A}) = 2.5, \psi_y(\mathrm{B}) = 0,\ \text{and}\ \psi_y(\mathrm{C}) = 0.5$$

for each label. □

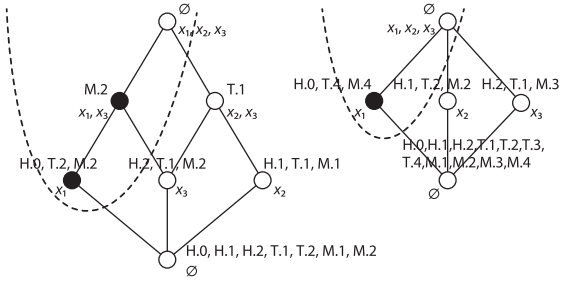


Fig. 5 Concept lattices constructed from contexts $\mathcal{B}^1(\tau)$ (left) and $\mathcal{B}^2(\tau)$ (right) in Table 5. The tuple y is consistent with concepts denoted by black dots.

From the preference obtained by LIFT, multi-label classification can be achieved, that is, an unlabeled tuple y is associated with a set of labels $L \subseteq \mathcal{L}$ such that $L = \{\lambda \in \mathcal{L} \mid \psi_y(\lambda) \neq 0\}$. Furthermore, a partial order $\preceq$ of labels can be derived from preferences, where $\lambda_i \preceq \lambda_j$ (λ_j is preferable than λ_i) if $\psi_y(\lambda_i) \leq \psi_y(\lambda_j)$. Thus we can also achieve the label ranking problem using the preference.

The time complexity of LIFT is $O(nd) + O(\Delta^3 N)$, where n is the number of tuples, d is the number of features, and N is the maximum number of concepts in concept lattices constructed in the learning process of LIFT; i.e.,

$$N = \max_{1\leq k\leq k_{\max}} \#\mathcal{B}^k(\tau),$$

since data preprocessing takes $O(nd)$, making a concept lattice takes less than $O(\Delta^3 N)$, and obtaining the preference takes less than $O(N)$.

Example 7 For training and test data given in Example 6, labels A and C are associated with y since both $\psi_y(\mathrm{A})$ and $\psi_y(\mathrm{C})$ are larger then 0. Moreover, we have the order $\mathrm{B} \preceq \mathrm{C} \preceq \mathrm{A}$ of label ranking for the tuple y. □

3. Experiments

We evaluate the LIFT algorithm using real data of ligands and receptors compared to SVM and the decision tree-based algorithm. We also experimentally measure the effectiveness of unlabeled data for training in semi-supervised learning by LIFT.

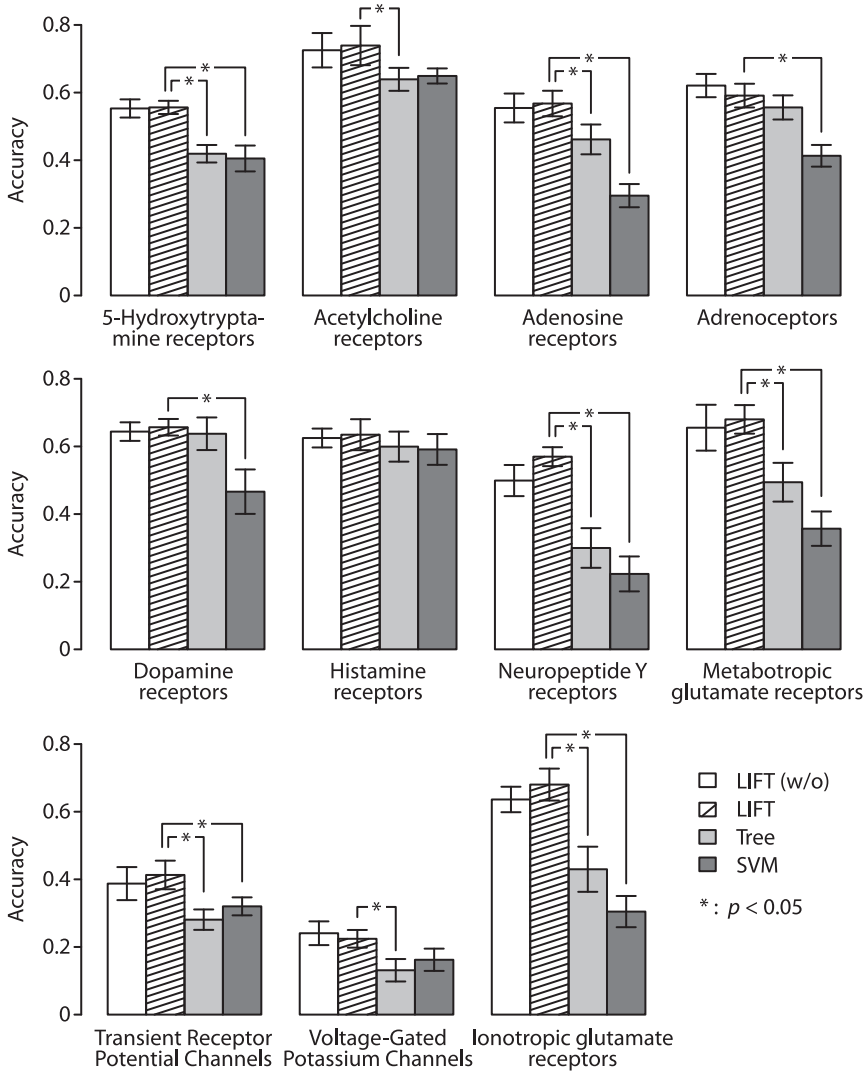


Fig. 6 Accuracy for each receptor family obtained by LIFT without unlabeled training data (LIFT (w/o)), LIFT, the tree algorithm, and SVM with the RBF kernel. Data show mean ± s.e.m.

3.1 Materials and Methods

3.1.1 Environment

LIFT was implemented in R and all experiments were performed in R version 2.12.2 [23]. LIFT uses LCM[*4] distributed by Uno [28] to construct a concept lattice $\mathcal{B}^k(\tau)$, which was implemented in C. In all experiments, we used Mac OS X version 10.6.5 with two 2.26-GHz Quad-Core Intel Xeon CPUs and 12 GB of memory.

3.1.2 Databases

We collected the entire 1,782 ligand data in the IUPHAR database [25][*5]. In the database, each ligand is characterized by seven features: HBA, HBD, RB, TPS, MW, XLogP, and NLR as described in Section 2.1. Receptors, which corresponds to class labels, are classified into families, such as 5-Hydroxytryptamine receptors and Acetylcholine receptors, hence we picked up the eleven largest families from the database and used respective families as datasets for each training. Statistics of receptor families is shown in **Table 6**. In semi-supervised learning of LIFT, entire ligands except the focusing receptor family were used as unlabeled training data.

Table 6 Families of receptors. The number of ligands and receptors correspond to the data size and the number of class labels, respectively.

Family name	# of Ligands	# of Receptors
5-Hydroxytryptamine receptors	286	53
Acetylcholine receptors	100	68
Adenosine receptors	162	40
Adrenoceptors	111	35
Dopamine receptors	69	40
Histamine receptors	120	37
Neuropeptide Y receptors	76	34
Metabotropic glutamate receptors	73	9
Transient receptor potential channels	78	58
Voltage-gated potassium channels	61	71
Ionotropic glutamate receptors	81	14

3.1.3 Learning Algorithms

To measure the effectiveness of unlabeled ligand data, we used LIFT in two cases: Only labeled data were used in training in the first case (denoted by LIFT (w/o) in **Fig. 6**), and all ligands except test data were used as unlabeled training data in the second case. The maximum level $k_{\max}$ was set at 5 throughout all experiments. As control for evaluation of LIFT, we adopted SVM with the RBF kernel and the decision tree-based method imple-

[*4] http://research.nii.ac.jp/~uno/codes.htm
[*5] http://www.iuphar-db.org/index.jsp

mented in R [24], since the tree method is a typical learning algorithm which can be applied to mixed-type data containing both discrete and continuous variables. We used the function `ksvm` in the kernlab package for SVM [16], where all discrete values are treated as continuous. Note that these control methods are typical supervised learning methods and cannot use unlabeled data in the learning phase. Moreover, since they are algorithms designed for single-label classification, we just used the first label for each training datum. LIFT thus has some advantage compared to SVM and the decision tree-based method in learning in terms of label information.

3.1.4 Evaluation

Let $\upsilon = (H, Y)$ be a test table with $Y = y_1, y_2, \ldots, y_n$ and the domain of labels $\mathcal{L}$ be $\{\lambda_1, \lambda_2, \ldots, \lambda_l\}$. Assume that we have the preference $\{\psi_y(\lambda_i|\tau) \mid 1 \leq i \leq l\}$ for each label $\lambda_i \in \mathcal{L}$ by LIFT, where

$$\psi_y(\lambda_{p_1}|\tau) \geq \psi_y(\lambda_{p_2}|\tau) \geq \psi_y(\lambda_{p_3}|\tau) \geq \ldots \geq \psi_y(\lambda_{p_l}|\tau).$$

for each tuple $y \in \mathrm{set}(Y)$. In LIFT, we define the accuracy $\mathrm{acc}(\upsilon)$ by

$$\mathrm{acc}(\upsilon) := \frac{\sum_{i=1}^{n} \#\left\{\lambda_j \in \Lambda(y_i) \mid j \in \{p_1, p_2, \ldots, p_{\#\Lambda(y_i)}\}\right\}}{\sum_{i=1}^{n} \#\Lambda(y_i)},$$

which takes values in $[0, 1]$ to be maximized. This means that when y is associated with q labels, we check whether or not each label is in top-q labels determined by the preference. Notice that we do not take labels $\lambda_{p_{\#\Lambda(y)+1}}, \ldots, \lambda_{p_l}$ into account to obtain the accuracy since the database has only positive information and $\lambda \notin \Lambda(y)$ *does not* means that the ligand y does not bind to the receptor λ.

For the decision-tree based method and SVM, the accuracy is obtained by

$$\mathrm{acc}(\upsilon) := \frac{\#\{y_i \mid 1 \leq i \leq n,\ f(y_i) \in \Lambda(y_i)\}}{n},$$

where $f(y_i)$ is the output for the tuple y_i by respective learning algorithms.

Mean and s.e.m. (standard error of the mean) of accuracy was obtained for each dataset (receptor family) by 10-fold cross-validation.

3.2 Results and Discussion

Results are shown in Fig. 6. These results clearly show that LIFT is more effective than the typical classification algorithms of SVM and the tree algorithm for ligand finding. Accuracy obtained by LIFT is significantly higher than that by SVM and the tree algorithm in eight cases out of eleven cases (checked by paired t-test, $\alpha = 0.05$). One of reasons of the difference might be that LIFT can treat multi-labels effectively whereas SVM and the tree algorithm cannot. Moreover, SVM treated discrete values as continuous, presumably resulting in lower accuracy.

Comparison with other preprocessing methods is valuable. For example, we can convert the original database into a single-label database by duplicating tuples with multiple labels. It is thus future work to investigate the relationship between LIFT and such techniques.

Since each family has many classes from 9 to 71, accuracy of LIFT, which is more than 50% in most cases, is high enough. Furthermore, unlabeled training data can be used effectively in LIFT in the semi-supervised manner. Our results therefore indicate that LIFT should be valuable for finding new ligands and contribute to biology and biochemistry.

By using LIFT, we can find new ligand candidates from any training data, hence LIFT can be used as a tool for actual biological experiments to narrow down new ligand candidates. Checking such candidates obtained by LIFT in biological experiments is future work.

4. Conclusion

In this paper, we have proposed a semi-supervised learning algorithm, called LIFT, for ligand finding from databases. LIFT performs preference learning, that is, multi-label classification and ranking, in the semi-supervised manner. First, every dataset is translated into a (formal) context, followed by clustering of it by FCA by putting on a concept lattice, where each continuous (real-valued) value is discretized based on the binary encoding scheme. Then, on the lattice, the preferences of class labels for unlabeled test data are obtained by taking labels of training data into account.

Since LIFT is a flexible learning algorithm, it can be applied to any databases in various domains. Thus considering contributions to other domains is one of future work. Another future work is to treat incremental databases in LIFT, because lots of databases are frequently updated whereas LIFT cannot directly treat such incremental databases. LIFT can display weighted classification rules, which are easily-interpreted, thus analysis of learned rules from biological point of view is also a future work. Furthermore, using biological background knowledge such as the structure of a receptor for learning is interesting future work.

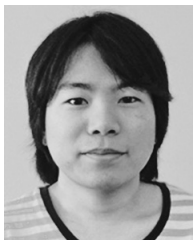
Acknowledgments This work is inspired by insightful ideas of Professor Shigeo Kobayashi. This work was partly supported by Grant-in-Aid for Scientific Research (A) 22240010 and for JSPS Fellows 22·5714.

References

[1] Ballester, P.J. and Mitchell, J.B.O.: A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking, *Bioinformatics*, Vol.26, No.9, pp.1169–1175 (2010).
[2] Blinova, V.G., Dobrynin, D.A., Finn, V.K., Kuznetsov, S.O. and Pankratova, E.S.: Toxicology analysis by means of the JSM-method, *Bioinformatics*, Vol.19, No.10, pp.1201–1207 (2003).
[3] Chapelle, O., Schölkopf, B. and Zien, A. (Eds.): *Semi-Supervised Learning*, MIT Press (2006).
[4] Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W. and Kollman, P.A.: A second generation force field for the simulation of proteins, nucleic acids, and organic molecules, *Journal of the American Chemical Society*, Vol.117, No.19, pp.5179–5197 (1995).
[5] Dara, R., Kremer, S.C. and Stacey, D.A.: Clustering unlabeled data with SOMs improves classification of labeled real-world data, *Proc. 2002 International Joint Conference on Neural Networks*, Vol.3, pp.2237–2242 (2002).
[6] Date, C.J.: *An Introduction to Database Systems*, Addison Wesley, 8 edition (2003).
[7] Davey, B.A. and Priestley, H.A.: *Introduction to Lattices and Order*, 2nd edition, Cambridge University Press (2002).
[8] Demiriz, A., Bennett, K.P. and Embrechts, M.J.: Semi-supervised clustering using genetic algorithms, *Proc. Artificial Neural Networks in Engineering*, pp.809–814 (1999).

[9] Fürnkranz, J. and Hüllermeier, E. (Eds.): *Preference learning*, Springer (2010).
[10] Ganter, B. and Kuznetsov, S.: Hypotheses and version spaces, *Conceptual Structures for Knowledge Creation and Communication*, de Moor, A., Lex, W. and Ganter, B. (Eds.), Lecture Notes in Computer Science, Vol.2746, pp.83–95, Springer (2003).
[11] Ganter, B. and Wille, R.: *Formal Concept Analysis: Mathematical Foundations*, Springer (1998).
[12] Gohlke, H., Hendlich, M. and Klebe, G.: Knowledge-based scoring function to predict protein-ligand interactions1, *Journal of Molecular Biology*, Vol.295, No.2, pp.337–356 (2000).
[13] Han, J. and Kamber, M.: *Data Mining*, 2nd edition, Morgan Kaufmann (2006).
[14] Huang, N., Kalyanaraman, C., Bernacki, K. and Jacobson, M.P.: Molecular mechanics methods for predicting protein-ligand binding, *Physical Chemistry Chemical Physics*, Vol.8, No.44, pp.5166–5177 (2006).
[15] Huey, R., Morris, G.M., Olson, A.J. and Goodsell, D.S.: A semiempirical free energy force field with charge-based desolvation, *Journal of Computational Chemistry*, Vol.28, No.6, pp.1145–1152 (2007).
[16] Karatzoglou, A., Smola, A., Hornik, K. and Zeileis, A.: kernlab—an S4 package for kernel methods in R, *Journal of Statistical Software*, Vol.11, No.9, pp.1–20 (2004).
[17] Kaytoue, M., Kuznetsov, S.O., Napoli, A. and Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis, *Information Sciences*, Vol.181, pp.1989–2001 (2011).
[18] King, R.D., Muggleton, S.H., Srinivasan, A. and Sternberg, M.J.E.: Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming, *Proc. National Academy of Sciences*, Vol.93, No.1, pp.438–442 (1996).
[19] Kuznetsov, S.O. and Samokhin, M.V.: Learning closed sets of labeled graphs for chemical applications, *Inductive Logic Programming*, Kramer, S. and Pfahringer, B. (Eds.), Lecture Notes in Computer Science, Vol.3625, pp.190–208, Springer (2005).
[20] Makino, K. and Uno, T.: New algorithms for enumerating all maximal cliques, *SWAT 2004*, Lecture Notes in Computer Science, Vol.3111, pp.260–272, Springer (2004).
[21] Moitessier, N., Englebienne, P., Lee, D., Lawandi, J. and Corbeil, C.R.: Towards the development of universal, fast and highly accurate docking/scoring methods: A long way to go, *British Journal of Pharmacology*, Vol.153, No.S1, pp.S7–S26 (2008).
[22] Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L.: Efficient mining of association rules using closed itemset lattices, *Inf. Syst.*, Vol.24, No.1, pp.25–46 (1999).
[23] R Development Core Team: *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing (2011).
[24] Ripley, B.D.: *Pattern Recognition and Neural Networks*, Cambridge University Press (1996).
[25] Sharman, J.L., Mpamhanga, C.P., Spedding, M., Germain, P., Staels, B., Dacquet, C., Laudet, V., Harmar, A.J. and NC-IUPHAR: IUPHAR-DB: New receptors and tools for easy searching and visualization of pharmacological data, *Nucleic Acids Research*, Vol.39, pp.D534–D538 (2011). Database Issue.
[26] Simovici, D.A. and Djeraba, C.: *Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics*, Springer (2008).
[27] Sugiyama, M. and Yamamoto, A.: Semi-Supervised Learning for Mixed-Type Data via Formal Concept Analysis, *Conceptual Structures for Discovering Knowledge*, Andrews, S., Polovina, S., Hill, R. and Akhgar, B. (Eds.), Lecture Notes in Computer Science, Vol.6828, pp.284–297 (2011).
[28] Uno, T., Kiyomi, M. and Arimura, H.: LCM ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining, *Proc. 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, pp.77–86, ACM (2005).
[29] Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts, *Ordered Sets*, D. Reidel Publishing Company, pp.445–470 (1982). This article is included in Formal Concept Analysis, LNCS 5548, pp.314–339, Springer (2009).
[30] Zhang, Y., Feng, B. and Xue, Y.: A New Search Results Clustering Algorithm Based on Formal Concept Analysis, *Proc. 5th International Conference on Fuzzy Systems and Knowledge Discovery*, pp.356–360, IEEE (2008).
[31] Zhu, X. and Goldberg, A.B.: *Introduction to semi-supervised learning*, Morgan and Claypool Publishers (2009).

Mahito Sugiyama received his B.E., M.E., and Ph.D. degrees from Kyoto University in 2006, 2008, and 2012, respectively. He is currently a researcher at Osaka University. Since 2010 until 2012 he has been a research fellow of the Japan Society for the Promotion of Science. His research interest includes machine learning, data mining, and knowledge discovery. In particular, he is addressing computational and discrete approaches to machine learning. He is a member of JSAI.



Kentaro Imajo received his B.E. degree in Information Science and Technology from Osaka University in 2011. He is a graduate student in Master's Program at Graduate School of Informatics, Kyoto University. His research interest is image processing.

Keisuke Otaki received his B.E. degree in Engineering from Kyoto University in 2011. He is now a master course student at Graduate School of Informatics, Kyoto University. His research interests are knowledge discovery, in particular, privacy-preserving data mining, and analyzing generative models of data. He is a student member of IPSJ, JSAI, and IEICE.

Akihiro Yamamoto received his B.S. degree from Kyoto University in 1985, and Dr.Sci. degree from Kyushu University in 1990. Currently, he is a professor of the Department of Intelligence Science and Technology, Graduate School of Informatics at Kyoto University. He has made research contributions to foundations of intelligence science, with a particular focus on application of mathematical logic to machine learning. His recent research interest includes developing machine learning theory with discrete mathematics, computational algebra, and computational calculus. He is a member of JSAI, IPSJ, and JSSST.