Vol.2011-MPS-86 No.28
Vol.2011-BIO-27 No.28
2011/12/2

Semi-Supervised Ligand Finding Using Formal Concept Analysis

Mahito Sugiyama,[†1,*1] Kentaro Imajo,[†1]
Keisuke Otaki[†1] and Akihiro Yamamoto[†1]

To date, enormous studies have been devoted to investigate biochemical functions of *receptors*, which have crucial roles for signal processing in organisms, and *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* costs high. Here we challenge to ligand finding 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 the *semi-supervised* manner. The key to LIFT is to realize 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 our task compared to other machine learning algorithms.

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. 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* costs high 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*[5], 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[15]. 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 costs high 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.

Information about receptors and ligands is donated to various databases, such as KEGG[⋆1], and in this paper we use the IUPHAR database[11)⋆2]. In the database, every ligand is characterized by seven features. 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.

Our proposed algorithm, called LIFT (Ligand FInding via Formal ConcepT Analysis), uses "label propagation", or cluster-and-label, which is a typical approach in semi-supervised learning[2),4)]. 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 realized by *Formal Concept Analysis*

†1 Graduate School of Informatics, Kyoto University
∗1 Presently with Research Fellow of the Japan Society for the Promotion of Science

⋆1 http://www.genome.jp/kegg/
⋆2 http://www.iuphar-db.org/index.jsp

(FCA)[3),6)], which is a mathematical data analysis technique originally proposed by Wille[14)]. One of successful applications of FCA in data mining is for frequent pattern and association rule mining proposed by Pasquier *et al.*[9)], 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, 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.*[7)] 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.

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 works in Section 4.

2. The LIFT Algorithm

We present the LIFT algorithm in this section.

2.1 Database Formalization

We treat a ligand database using the notion of a relational database. 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 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) context, a binary matrix specifying a set of objects and their attributes, to apply FCA.

FCA is a mathematical data analysis technique[3),6)], which is applied to a triple (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 $\mathrm{set}(X) = \{x_1, x_2, \ldots, x_n\}$. 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).

First, we focus on preprocessing for discrete values of features HBA, HBD, RB, and NLR. For each feature $h \in H$, the set of attributes $M_h = \{h.m \mid m \in x(h)$ 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 consider how to make a context from continuous values using *discretization*. 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* so that every value is in the closed interval $[0, 1]$. Next we discretize values in $[0, 1]$ and make a context using the *binary encoding* of real numbers, which is the same approach as the literature 12). At discretization level k, $M_h = \{h.1, h.2, \ldots, h.2^k\}$. For each tu-

Vol.2011-MPS-86 No.28
Vol.2011-BIO-27 No.28
2011/12/2

Algorithm 1: Data preprocessing for making context

Input: Table $\tau = (H, X)$ and discretization level k
Output: Context (G, M^k, I^k)

function CONTEXT(τ, k)
1: $G \leftarrow \mathrm{set}(X)$
2: **for each** feature $h \in H$
3: **if** $\mathrm{Dom}(h) = \mathbb{N}$ **then** $(M_h, I_h) \leftarrow$ CONTEXTD(X, h)
4: **else if** $\mathrm{Dom}(h) = \mathbb{R}$ **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, I^k)
7: **return** (G, M^k, I^k)

function CONTEXTD(X, h)
1: $M \leftarrow \{h.m \mid m \in x(h)$ such that $x \in \mathrm{set}(X)\}$
2: $I \leftarrow \{(x, h.m) \mid x \in \mathrm{set}(X)$ 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 $\{x(h) \mid x \in \mathrm{set}(X)\}$
3: **for each** $x \in \mathrm{set}(X)$
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)

ple 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 $(a-1)/2^k < x(h) \leq a/2^k$. 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.

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 literature 3), 6) 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

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

The set A is called an *extent* and B an *intent*. □

The set of concepts over (G, M, I), called the *concept lattice*, is written by $\mathcal{B}(G, M, I)$. In the context of frequent pattern mining, a set of attributes corresponds to an itemset and the lattice is called the closed itemset lattice[9]. 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 $\mathcal{B}^k(\tau)$.

To obtain concept lattices, we use the algorithm developed by Makino and Uno[8], which is known to be one of the fastest such algorithms. Its time complexity is theoretically bounded as $O(\Delta^3)$, where Δ denotes the maximum degree of the given bipartite graph, *i.e.*, $\Delta = \max\{\#J \mid J \subseteq I$, where $g = h$ for all $(g, m), (h, l) \in J$ or $m = l$ for all $(g, m), (h, l) \in J\}$ ($\#J$ is the number of elements in J).

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.

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*.

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

and $B \neq \emptyset$ hold. □

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 $\bigcup_{x \in A} \Lambda(x)$. Note that $\Lambda(x)$ could be empty, meaning that the object x is unlabeled; LIFT allows unlabeled data for training. 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 $\psi_y^k(\lambda|\tau) := \sum\{\#\Lambda(A)^{-1} \mid y \text{ is consistent with } (A, B) \in \mathcal{B}^k(\tau)$ such that $\lambda \in \Lambda(A)\}$, where we assume $\#\Lambda(A)^{-1} = 0$ if $\#\Lambda(A) = 0$ for simplicity. □

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.

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.

Example 1 Let us consider a table $\tau = (H, X)$ with $H = \{$HBD, RB, TPS, MW$\}$, where labels are associated with each tuple as shown in Table 1, and a table $\upsilon = (H, y)$ with an unlabeled tuple y. Assume that $k_{\max} = 2$. At discretization level 1, we have $\psi_y^1(\mathrm{A}) = 1.5$, $\psi_y^1(\mathrm{B}) = 0$, and $\psi_y^1(\mathrm{C}) = 0.5$, since y is consistent with two concepts $(A_1, B_1) = (\{x_1, x_3\}, \{\mathrm{MW}.2\})$ 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 Figure 1). Remember that we always ignore the concept whose attribute is empty. At discretization level 2, we have $\psi_y^2(\mathrm{A}) = 1, \psi_y^2(B) = 0$, 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 $\psi_y(\mathrm{A}) = 2.5, \psi_y(\mathrm{B}) = 0$, and $\psi_y(\mathrm{C}) = 0.5$ for each label. □

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**

From the preference obtained by LIFT, multi-label classification can be realized, 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 realize the label ranking problem.

The time complexity of LIFT is $O(nd) + O(\Delta^3) + O(N)$, 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, since data preprocessing takes $O(nd)$, making a concept lattice takes $O(\Delta^3)$, and obtaining the preference takes less than $O(N)$.

Example 2 For training and test data given in Example 1, 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 $\mathrm{B} \leq \mathrm{C} \leq \mathrm{A}$ of label ranking for the tuple y. □

Vol.2011-MPS-86 No.28
Vol.2011-BIO-27 No.28
2011/12/2

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

HBD	TPS	MW	Labels	
0	0.98	0.88	A	
1	0.41	0.48	B	C
2	0.12	0.71	A	C
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	×						×				×

3. Experiments

3.1 Materials and Methods

Environment. LIFT was implemented in R and all experiments were performed in R version 2.12.2[10]. LIFT uses LCM distributed by Uno[13] 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.

Databases. We collected the entire 1,782 ligand data in the IUPHAR database[11]⋆1. Receptors, which corresponds to class labels, are classified into families, hence we picked up the eleven largest families from the database and used respective families as datasets for each training. In semi-supervised learning of LIFT, entire ligands were used as unlabeled training data.

Evaluation. 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 Figure 2), 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 a control method for evaluation of LIFT, we adopted SVM with the RBF kernel and the decision tree-based method implemented in R. Mean and s.e.m. (standard error of the mean) of accuracy was obtained for each dataset (receptor family) by 10-fold crossvalidation.

⋆1 http://www.iuphar-db.org/index.jsp

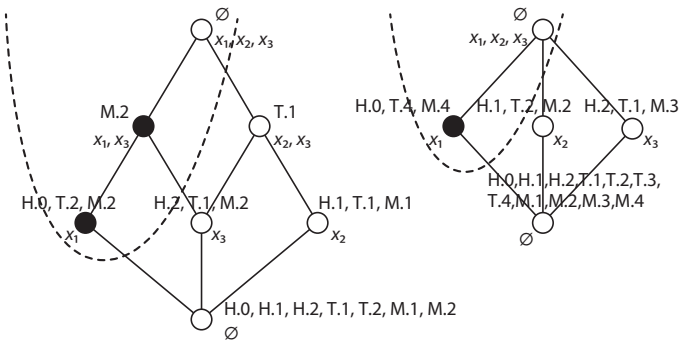

Fig. 1 Concept lattices constructed from contexts $\mathcal{B}^1(\tau)$ (left) and $\mathcal{B}^2(\tau)$ (right) in Table 1. The tuple y is consistent with concepts denoted by black dots.

3.2 Results and Discussion

Results are shown in Figure 2. These results clearly show that LIFT is more effective than the typical classification algorithms of SVM and the tree algorithm for ligand finding. 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 a future work.

4. Conclusion

In this paper, we have proposed the semi-supervised learning algorithm, called LIFT, for ligand finding from databases. LIFT realizes preference learning, that is, multi-label classification and ranking, in the semi-supervised manner. First,

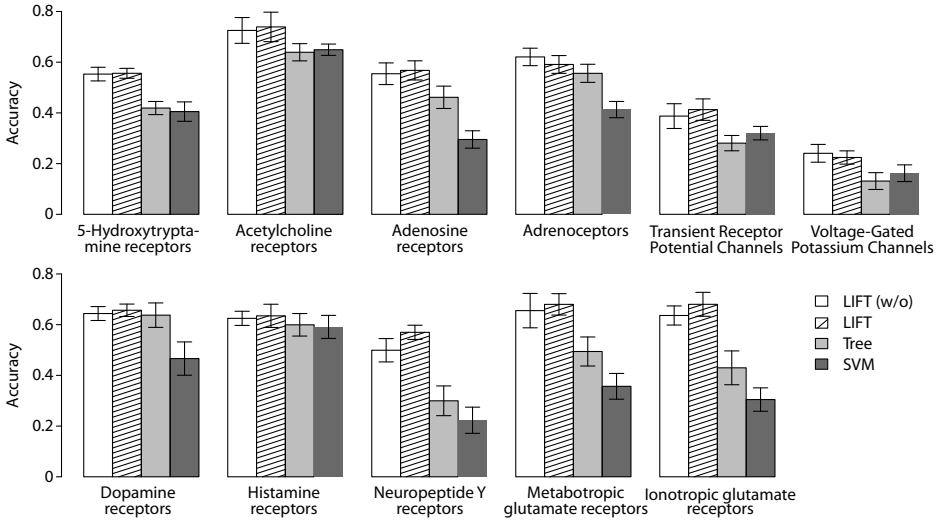

Fig. 2 Accuracy for each receptor family obtained by LIFT without unlabeled training data (LIFT (w/o)), LIFT, the tree algorithm, and SVM with the RBF kernel. Data show mean ± s.e.m.

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.

Acknowledgment

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) Dara, R., Kremer, S.C. and Stacey, D.A.: Clustering unlabeled data with SOMs improves classification of labeled real-world data, *Proceedings of the 2002 International Joint Conference on Neural Networks*, Vol.3, pp.2237–2242 (2002).
3) Davey, B.A. and Priestley, H.A.: *Introduction to lattices and order*, Cambridge University Press, 2 edition (2002).
4) Demiriz, A., Bennett, K.P. and Embrechts, M.J.: Semi-supervised clustering using genetic algorithms, *Proceedings of Artificial Neural Networks in Engineering*, pp. 809–814 (1999).
5) Fürnkranz, J. and Hüllermeier, E.(eds.): *Preference learning*, Springer (2010).
6) Ganter, B. and Wille, R.: *Formal Concept Analysis: Mathematical Foundations*, Springer (1998).
7) 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, *Proceedings of the National Academy of Sciences*, Vol.93, No.1, pp.438–442 (1996).
8) Makino, K. and Uno, T.: New algorithms for enumerating all maximal cliques, *SWAT 2004*, Lecture Notes in Computer Science, Vol.3111, Springer, pp.260–272 (2004).
9) Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L.: Efficient mining of association rules using closed itemset lattices, *Information Systems*, Vol.24, No.1, pp.25–46 (1999).
10) R Development Core Team: *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing (2011).
11) 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.
12) 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).
13) Uno, T., Kiyomi, M. and Arimura, H.: LCM ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining, *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, ACM, pp.77–86 (2005).
14) 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, 314–339, Springer (2009).
15) Zhu, X. and Goldberg, A.B.: *Introduction to semi-supervised learning*, Morgan and Claypool Publishers (2009).