Learning First-order Rules from Data with Multiple Parts: Applications on Mining Chemical Compound Data

Cholwich Nattee CHOLWICH@AI.SANKEN.OSAKA-U.AC.JP
Sukree Sinthupinyo SUKREE@AI.SANKEN.OSAKA-U.AC.JP
Masayuki Numao NUMAO@AI.SANKEN.OSAKA-U.AC.JP
The Institute of Scientific and Industrial Research, Osaka University, 8-1, Mihogaoka, Ibaraki, Osaka, 567-0047 Japan

Takashi Okada OKADA-OFFICE@KSC.KWANSEI.AC.JP
Department of Informatics, School of Science and Technology, Kwansei Gakuin University, 2-1 Gakuen-cho, Sanda, Hyogo, 669-1323, Japan

Abstract

Inductive learning of first-order theory based on examples has serious bottleneck in the enormous hypothesis search space needed, making existing learning approaches perform poorly when compared to the propositional approach. Moreover, in order to choose the appropiate candidates, all Inductive Logic Programming (ILP) systems only use quantitive information, e.g. number of examples covered and length of rules, which is insufficient for search space having many similar candidates. This paper introduces a novel approach to improve ILP by incorporating the qualitative information into the search heuristics by focusing only on a kind of data where one instance consists of several parts, as well as relations among parts. This approach aims to find the hypothesis describing each class by using both individual and relational characteristics of parts of examples. This kind of data can be found in various domains, especially in representing chemical compound structure. Each compound is composed of atoms as parts, and bonds as relations between two atoms. We apply the proposed approach for discovering rules describing the activity of compounds from their structures from two real-world datasets: mutagenicity in nitroaromatic compounds and dopamine antagonist compounds. The results were compared to the existing method using ten-fold cross validation, and we found that the proposed method significantly produced more accurate results in prediction.

1. Introduction

Inductive Logic Programming (ILP) is a learning approach which incorporates first-order logic to inductive learning. This knowledge representation provides comprehensibility in learning results and capability to handle more complex relational data which is difficult to be denoted using only the propositional representation. Yet, the richer, first-order representation generates huge space of possible hypotheses. Moreover, the existing ILP approaches use only quantitative information in order to select an appropriate candidate, i.e. using only a number of training examples covered without considering the quality of the examples covered. This makes the existing ILP approaches sometimes perform poorly compared to propositional approaches. In order to improve performance in predictive accuracy, we introduce a novel technique focusing on a kind of data called *multiple-part data*, i.e., one instance of data consists of several parts, as well as relations among parts. The objective of learning from multiple-part data is to find the hypothesis for describing class of each example by using characteristics of its part individually, as well as characteristics of relations among parts.

Though the existing first-order theory learning approaches can handle this kind of data because of the power of first-order representation, there is still limitation in the efficiency of the results, since the search

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

space becomes larger and contains the similar hypothesis candidates due to the numerous parts within one example. Thus, the search heuristics using only quantitative information cannot lead to good hypotheses. In order to solve this problem, we incorporate qualitative information to the search heuristics by weighing each part, based on its characteristics correlating to parts from other examples in the same class. This gives the parts with common characteristics higher weights than the uncommon parts, and helps search heuristics discriminate more efficiently. This weighing technique derives from the concept of multiple-instance learning, which is an extended two-class propositional learning approach for the data that cannot be labelled individually, albeit several instances of data are gathered and labelled as a group. Each positive group may consist of both positive and negative instances. Nevertheless, the multiple-instance learning aims to learn to predict instances not groups, thereby rendering itself similar to supervised learning where there are noisy examples in the positive examples. Most of the learning algorithms for multiple-instance data solve this ambiguity by using the similarity of data within the feature space in order to find the area in the feature space where several instances from various positive groups are located together and that this area is far from negative group instances. This method is modified and used as the weighing technique to evaluate each part of the multiple-part data containing similarity among parts before incorporating the weights into search heuristics to find the hypothesis that may consist of relations among parts.

In order to evaluate the proposed approach, we employed it to learn rules predicting activities of chemical compounds using their structure. The problem comes from the studies of Structure-Activity Relationship (SAR) aimed at finding the structure in chemical compounds describing their characteristics or activities. The knowledge discovered will be useful for developing new drugs. In recent years, the advance in High Throughput Screening (HTS) technology has produced vast amount of SAR data. Therefore, once the rules to predict the activities of existing SAR data is found, it will significantly help the screening process. The SAR data, which is represented by chemical compound structure, can be categorized as multiple-part data. Since we aim to find the substructure that can predict the activity of a compound, we apply the proposed system to learn the hypotheses from this kind of data. We also compared the learning result with the previous approaches in order to evaluate performance of the proposed system.

This paper is organized as follows. Section 2 describes FOIL and the multiple-instance learning which are the basic ideas for this research, and Section 3 explains the proposed approach to improve the learning performance. In Section 4, the experiments conducted on two real-world datasets are described and the experiment results are compared to the previous ILP approaches. We then consider the related works in Section 5. Finally, we summarize and conclude this research and consider our future directions in Section 6.

2. Background

2.1. FOIL

FOIL (Quinlan, 1990) is a top-down ILP system for learning function-free Horn clause definitions of a target predicate using background predicates. The learning process in FOIL starts with training examples containing all positive and negative examples, constructs a function-free Horn clause (a hypothesis) to cover some of the positive examples, and removes the covered examples from the training set. Next, it continues to search for the next clause. When the clauses covering all the positive examples have been found, they are reviewed to eliminate any redundant clauses and re-ordered so that all recursive clauses follow the non-recursive ones.

FOIL uses a heuristic function based on the information theory for assessing the usefulness of a literal. It provides effective guidance for clause construction. The purpose of this heuristic function is to characterize a subset of the positive examples. From the partial developing clause below

$$R(V_1, V_2, \ldots, V_k) \leftarrow L_1, L_2, \ldots, L_{m-1}$$

the training examples covered by this clause are denoted as T_i. The information required for T_i is given by

$$I(T_i) = -\log_2 \frac{|T_i^+|}{|T_i^+| + |T_i^-|} \qquad (1)$$

If a literal L_m is selected and yields a new set T_{i+1}, then the similar formula is given as

$$I(T_{i+1}) = -\log_2 \frac{|T_{i+1}^+|}{|T_{i+1}^+| + |T_{i+1}^-|} \qquad (2)$$

From the above, a heuristic used in FOIL is calculated as an amount of information gained when applying a new literal L_m;

$$Gain(L_i) = |T_i^{++}| \times (I(T_i) - I(T_{i+1})) \qquad (3)$$

T_i^{++} in this equation is the positive example extended in T_{i+1}.

This heuristic function is used over every candidate literal and the literal with the largest value is selected. The algorithm will continue until the generated clauses cover all positive examples.

2.2. Multiple-Instance Learning

In the supervised learning problem, we tried to design and create the algorithms that are able to generate the model from the training examples, in order to predict correct labels of unseen data, and each instance of the training examples has to be labelled beforehand. However, this framework may not be suitable for some applications. Dietterich et al. (1997) proposed the extended framework for the supervised learning to handle more ambiguities called *Multiple-Instance Learning*. In this new framework, unlabelled instances are grouped into a bag labelled as positive or negative. If a bag is positive, it means that the bag contains at least one positive instance; otherwise that bag would be labelled as negative. From this set-up, the target concept can be found from the area in the feature space, where the instances from various positive bags gather together.

After this framework and algorithm were presented, various approaches were proposed, some of which extend the existing supervised learning algorithm (Chevaleyre & Zucker, 2001; Gärtner et al., 2002). Maron and Lozano-Pérez (1998) proposed the original approach for multiple-instance learning using Diverse Density (DD). This approach is applied in the proposed system. This is followed by explanation of detail.

Diverse Density

The Diverse Density (DD) algorithm aims to measure a point in an n-dimensional feature space for multiple-instance domains. The DD at point p in the feature space shows how many *different* positive bags have an instance near p, and how *far* the negative instances are from p. Thus, the DD value is high in the area where instances from various positive bags are located together, and is rather far from instances from negative bags. It can be calculated as

$$\begin{aligned} DD(x) &= \prod_i (1 - \prod_j (1 - exp(-\|B_{ij}^{+} - x\|^2))) \cdot \\ & \prod_i \prod_j (1 - exp(-\|B_{ij}^{-} - x\|^2)) \end{aligned} \quad (4)$$

where x is a point in the feature space and B_{ij} represents the j^{th} instance of the i^{th} bag in training examples. For the distance, the Euclidean distance is adopted

$$\|B_{ij} - x\|^2 = \sum_k (B_{ijk} - x_k)^2 \quad (5)$$

In the previous approaches, several searching techniques were proposed for determining the value of features or the area in the feature space maximising DD value.

3. The Proposed Method

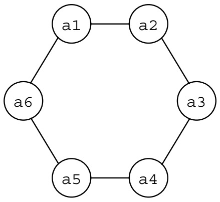
We present the top-down ILP system that is able to learn hypotheses more efficiently from the set of examples in which each example consists of several small parts, or when trying to predict class of data from the common substructure. The proposed system incorporates the existing top-down ILP system (FOIL) and applies the multiple-instance based measure to find the common characteristics among parts of positive examples. This measure is then used as a weight attached to each part of the example, so that the common parts among positive examples are attached with the high-valued weights. With these weights and the heuristic function based on example coverage, the system generates more precise and higher coverage hypotheses from the training examples. Next, Multiple-part data will be defined, followed by explanation of the modified heuristics.

3.1. Multiple-part Data

In this section, we define the multiple-part data, as well as the multiple-part learning problem on which this research focuses.

The multiple-part data is the data consisting of several components. Moreover, there are also relations among parts. In order to make the explanation easier to understand, we explain the multiple-part data using the example of chemical compound structure data which will be used in the experiments. From the chemical structure, each chemical compound or molecule represents one instance of multiple-part data. It consists of several atoms as parts as well as bonds which are relations between two atoms. Using the first-order representation, each compound can be denoted by using two predicates:

- *atom(Compound-ID, Atom-ID, Element)* for an atom.
- *bond(Compound-ID, Atom-ID$_1$, Atom-ID$_2$, Type)*



```
atom(c1,a1,c)         atom(c1,a2,c)
atom(c1,a3,c)         atom(c1,a4,c)
atom(c1,a5,c)         atom(c1,a6,c)
bond(c1,a1,a2,1)      bond(c1,a2,a3,2)
bond(c1,a3,a4,1)      bond(c1,a4,a5,2)
bond(c1,a5,a6,1)      bond(c1,a6,a1,2)
```

Figure 1. Examples of multiple-part data

for a bond.

Figure 1 shows the example of a chemical compound and its representation. We use *Compound-ID* and *Atom-ID* to identify each compound and each atom. A bond is a relation consisting of two atoms ($Atom\text{-}ID_1$ and $Atom\text{-}ID_2$). Moreover, we also include features to characterize atoms and bonds, which are *Element*, and *Type*. These features are useful for categorizing the atoms and bonds.

The multiple-part learning problem aims to find hypotheses describing the label or class of data from the substructure or the subset of parts. For example, in the case of chemical compound data, we want to find the substructure of the compound that is common among compounds with the same label or class, such as a group of atoms and bonds including their features. This problem is different from the traditional supervised learning that aims to predict class from the whole characteristics of data, such as predicting class of compound from its weight or some special value computed. Other real-world data can be categorized as multiple-part data, e.g. each time point in time-series data can be considered as a part with relations among parts, such as *before* and *after*.

3.2. Modified Heuristic Function

The heuristic function is used to control the way the algorithm explores hypothesis space. This function based on the information theory that counts the number of positive and negative tuples covered by the partially developing clause are used in FOIL (formula 1 and 3). With this, FOIL selects the literal that covers many positive tuples but few negative tuples. In order to help heuristics select better literals, we apply the DD value to each tuple, and we have to adapt the heuristic function, so that the parts with high DD values are selected first, making the hypothesis cover the common characteristics among parts from positive examples.

From formula 1, T_i^+ and T_i^- denote the set of positive and negative tuples respectively, as the DD value can be used to show the importance of the part of data by representing each instance of multiple-part data as a bag and each part as an instance in the bag. We then incorporate these to the heuristic function by altering $|T_i^+|$ to be the sum of the DD values of tuple. If this sum is high, it means that the literal can cover more common parts among positive examples. Thus, the heuristic function is adapted as follows:

$$DD_s(T) = \sum_{T_i \in T} DD(T_i) \tag{6}$$

$$I(T_i) = -\log_2 \frac{DD_s(T_i^+)}{DD_s(T_i^+) + |T_i^-|} \tag{7}$$

$$Gain(L_i) = DD_s(T_i^{++}) \times (I(T_i) - I(T_{i+1})) \tag{8}$$

This function is similar to weighing each part with the DD value and using the sum of these weights to select the literal, while the original heuristic function weighs all parts with the same value as 1. Nevertheless, we still use the number of negative tuples $|T_i^-|$ in the same way as the original heuristics, since we know that all parts of negative examples show the same strength. Therefore, it is similar to weighing all negative parts with value 1.

From the above function, there is still one problem left to be considered. This problem occurs when the learning algorithm tries to select relation among parts as a literal. Selecting the relations makes each tuple consist of more than one part, such as the system trying to append a bond into the clause, which makes each tuple contain two atoms. We then have to select the weight to represent each tuple from the DD value of the parts. We solve this problem by simply selecting the average DD value in the tuple as the weight of the tuple. The reason that we select the average value is that the proposed method aims to cover the tuples with high weight values.

3.3. Algorithm

From this modified function, we implement the prototype system called FOILMP (**FOIL** for **M**ultiple-**P**art data). This system basically uses the same algorithm as proposed in (Quinlan, 1990). Nevertheless, in order to construct accurate hypotheses, the beam search is applied so that the algorithm maintains a set of good

```
FindBestRule(Examples, Remaining)
  • Initialize $Beam$ with an empty rule.
  • Do
      – $NewBeam \leftarrow \{\}$
      – For each clause $C$ in $Beam$
          ∗ Generate $Candidates$ by adding all possible
            literals to $C$.
          ∗ For each new clause $nC$ in $Candidates$
              · Calculate heuristic of $nC$ using DD values.
              · Append $nC$ to $NewBeam$.
      – $Beam \leftarrow$ Best $BeamWidth$ clauses in $NewBeam$
      – $R \leftarrow$ Best clause in $Beam$
  • Until Accuracy($R$) $> \varepsilon$ and PositiveCoverage($R$) $> \gamma$
  • Return $R$
```

Figure 2. The algorithm for finding the best rule from the remaining positive examples.

candidates instead of selecting the best candidate at that time. This searching method makes the algorithm possible to backtrack to the right direction and finally get to the goal. Moreover, in order to obtain rules with high coverage, we define the coverage ratio, and the algorithm is set to select only the rules covering positive examples higher than that ratio. The modified subroutine for selecting rules is shown in figure 2. There are two user-defined parameters: ε for the minimum accuracy and γ for the minimum positive example coverage.

4. Experiments

We conducted the experiments on two datasets for SAR: Mutagenesis and Dopamine antagonist data. In order to evaluate the performance of the proposed system, these experiments were conducted in ten-fold cross validation, and we compared the results to the existing approaches.

4.1. Data

In this research, we aim to discover rules describing the activities of chemical compounds from their structures. Two kinds of SAR data were studied: the mutagenesis data (Srinivasan et al., 1994) and the dopamine antagonist data.

The mutagenesis data is used to test mutagenicity in nitroaromatic compounds, which are often known to be carcinogenic and also cause damage to DNA. These compounds occur in automobile exhaust fumes and are also common intermediates used in chemical industry. In this dataset, 230 compounds were obtained from the standard molecular modeling package QUANTA. Two predicates (atm and bond) are used to denote each compound:

- *atm(comp, atom, element, type, charge)*, stating that there is the atom *atom* in the compound *comp* that has element *element* of *type* and partial charge *charge*.
- *bond(comp, atom1, atom2, type)*, describing that there is a bond of *type* between the atoms *atom1* and *atom2* in the compound *comp*.

The background knowledge in this dataset is already formalized in the form of the multiple-part data, and thus, no preprocessing is necessary.

The dopamine antagonist dataset describes each compound as atoms and bonds when the compound is plotted in the 3-dimensional area. Each atom is represented by element type and its position in the 3-dimensional area. Each bond is represented by two atoms and bond type. From this information, it can be seen that the position of atom has no meaning, since a compound can be plotted in many different ways. Therefore, the positions are not used directly but are used for computing length of bond between atoms. Hence, after preprocessing, the background knowledge consists of two kinds of predicate.

- *atom* – element type (such as, C for Carbon, N for Nitrogen, O for Oxygen), position when it is plotted in 3D space (X, Y, and Z).
- *bond* – two atoms that are linked by the bond, bond type which can be 1 for a single bond, or 2 for a double bond.

To convert information above into predicates, we first found that the position (X, Y, and Z) in 3D space has no meaning, since we can rotate and re-plot the compound and it makes the position of atoms changed to other values. We then used the position of atom to compute length of bond which is a fixed feature not related to moving or rotating. Two predicates were generated:

- *atm(compound, atom, element)* – stating an atom *atom* in compound *compound* with element *element*.
- *bond(compound, atom1, atom2, bondtype, length)* – describing a bond *bond* in compound *compound*. This bond links atom *atom1* and atom *atom2* together with type *bondtype* and length *length*.

However, after discussion with the domain expert, we found that excepting bond, there are as well other kinds of link whose energy is not so strong as bond but it is frequently important to identify the compound structure. Therefore, we add another predicate in order to show this kind of information and call it *link* as below.

- *link(compound, atom1, atom2, length)* – describing a relation *link* in compound *compound*. It links atom *atom1* and atom *atom2* with length *length*.

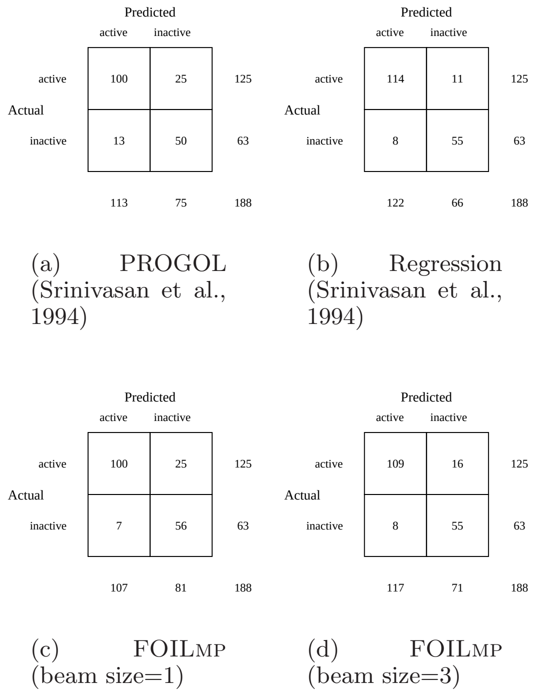
Finally, three kinds of predicate are used in the background knowledge. Nevertheless, there is only one feature to characterize each atom, that is, element type. This would not be enough to compute DD value if we use element type only. It means that we can separate dopamine antagonist compound by checking only elements included in that compound. Therefore, we need to add other features to characterize each atom. After discussing with the domain expert, the other features based on basic knowledge in chemistry are added: number of bonds linked to an atom, average length of bonds linked to an atom, connection to oxygen atom, minimum distance to oxygen and nitrogen.

Most of features are related to oxygen and nitrogen because the expert said that the position of oxygen and nitrogen has an effect to the activity of dopamine antagonist. Hence, the predicate *atm* is modified to *atm(compound, atom, element, number-bond, avg-bond-len, o-connect, o-min-len, n-min-len)*.

Although, the proposed method can handle only two-class data (only positive or negative), there are four classes for the dopamine antagonist compounds, however. Then, hypotheses for each class are learned by the one-against-the-rest technique, for instance, learning class D1 by using D1 as positive examples and D2,D3,D4 as negative examples.

4.2. Experiment Results and Discussion

For the mutagenesis data, we first compare the performance of FOILmp using all data consisting of 125 compounds for the positive class and 63 compounds for the negative class. In this experiment, the beam search strategy is applied in FOILmp by setting the beam size = 1 (hill-climbing search) and 3 subsequently. In this and following experiments, we set 0.9 for the minimum accuracy, and 0.1 for the minimum coverage. The results are compared to the existing results described in (Srinivasan et al., 1994). Figure 3 shows the performance tables.



		Predicted active	Predicted inactive	
Actual	active	100	25	125
	inactive	13	50	63
		113	75	188

(a) PROGOL (Srinivasan et al., 1994)

		Predicted active	Predicted inactive	
Actual	active	114	11	125
	inactive	8	55	63
		122	66	188

(b) Regression (Srinivasan et al., 1994)

		Predicted active	Predicted inactive	
Actual	active	100	25	125
	inactive	7	56	63
		107	81	188

(c) FOILmp (beam size=1)

		Predicted active	Predicted inactive	
Actual	active	109	16	125
	inactive	8	55	63
		117	71	188

(d) FOILmp (beam size=3)

Figure 3. Performance tables for Mutagenesis Data comparing FOILmp to PROGOL and the regression technique.

From the performance tables, even FOILmp with hill-climbing search strategy learns from this dataset better than PROGOL(Muggleton, 1995) with accuracy 83.0% for FOILmp and 79.8% for PROGOL. When compared to the regression technique based on the model called *logM* (Srinivasan et al., 1994), FOILmp with the beam size = 3 still showed worse performance than the regression model that predicts at 89.9%, whereas FOILmp predicts at only 87.2%. However, these experimental results still shows the advantage of FOILmp since the human expert is needed to choose the useful features to construct the model in order to use the regression model, and the results are based on those features, which are difficult to be comprehended by other chemists.

Table 1 shows the experiment results on Mutagenesis data. The prediction accuracy on test examples using ten-fold cross validation is compared to the existing approaches (FOIL and PROGOL). The proposed method predicts more accurately than the existing approaches.

For Dopamine Antagonist data, we conducted ten-fold cross validation to predict D1, D2, D3, and D4 activities. However, we compared the experimental results with Aleph (Srinivasan, 2001), since PROGOL cannot generate accurate rules from this dataset in reasonable

Table 1. Ten-fold cross validation test comparing the accuracy on Mutagenesis data

Approach	Accuracy
The proposed method	0.82
PROGOL	0.76
FOIL	0.61

time. Aleph is an ILP system based on inverse entailment and similar algorithm with PROGOL. However, Aleph has adopted several search strategies, such as randomized search which helps improve the performance of the system. In this experiment, we set Aleph to use GSAT (Selman et al., 1992), which is one of the randomized search algorithms where the best results can be generated. Table 2 shows the prediction accuracy computed for both positive and negative examples, and then, for only the positive examples. The table also shows the results of significance test using a one-paired t-test. The experiment results show that FOILmp predicts more accurately than Aleph in both accuracy computation methods. The significance tests also show the confidence level in the difference between accuracy. Figure 4 shows the details of the rules obtained by FOILmp. We also found that FOILmp generates rule with higher coverage than Aleph where the rule covers 36.5% of positive examples.

5. Related Work

In recent years, from the merits of ILP, there are many works proposed to learn from chemical compound structure data using ILP (Srinivasan et al., 1994; Srinivasan et al., 1996; Finn et al., 1998; Marchand-Geneste et al., 2002). These works proposed different kinds of background knowledge in order to discover the substructure of a molecule responsible for its activity.

King et al. (1995) has discussed whether propositional learner or ILP is better for learning from chemical structure. Actually, the first-order representation can denote chemical structure without losing any information, since denoting the relational data like chemical structure data using propositional logic is beyond its limit. Therefore, some special techniques are needed, such as for relations among parts. We may use only average value of features in that relation or use the domain-related knowledge to compute a new feature in order to help categorize. However, a learner using first-order representation performs worse than a propositional learner, since there are still some restrictions from the logic theory in ILP learners. However, using accuracy comparisons only may not be good as assessment, since the chemist's natural inclination is related to chemical structure, and the learning results from ILP is comprehensible to chemists.

Rule 1

```
d1(A) :- atm(A,B,C,D,E,F), E>=3.7, F=3.3,
         bond(A,L,B,H,M,N), bond(A,G,H,I,J,K),
         K=1.5, bond(A,O,B,P,Q,R),
         not_equal(H,P).
```

Accuracy = 93.2%   Coverage = 47.6%

This rule shows the molecule contains an atom `B` with the minimum distance to oxygen is greater or equal to 3.7, and the minimum distance to nitrogen is 3.3. From `B`, there are two bonds to two different atoms (`H` and `P`). Moreover, there is another bond from `H` to `I` with bond length is equal to 1.5.

Rule 2

```
d1(A) :- atm(A,B,C,D,E,F), F=3.0, E>=3.9,
         bond(A,G,B,H,I,J), J<1.4.
```

Accuracy = 93.8%   Coverage = 10.3%

This rule is similar to Rule 1 that there is one atom with specified minimum distance to oxygen and nitrogen. But there is only one bond with length less than 1.4.

Figure 4. Rules obtained by FOILmp using data for D1 activity.

Okada (2002) proposed the cascade model which can be considered an extension of association rule mining. In this approach, itemset lattice, is represented in the form of substructure pattern similar to the chemical structural formula. This system discovers some interesting rules, due to the co-occurrence of chemical structures among positive compounds. There is a limitation from using the propositional logic that the substructure has to be generated prior to testing for rule construction.

6. Conclusion and Future Works

We have presented the extension of FOIL for handling multiple-part data more efficiently by using Diverse Density from multiple-instance learning, in order to evaluate parts, so that the parts with common characteristics among positive examples have high weights. This enables the searching process to generate better results. We conducted the experiments on chemical compound data for structure-activity relationship studies. The experiment results showed that the pro-

Table 2. Ten-fold cross-validation test comparing the accuracy on dopamine antagonist data; Superscripts denote confidence levels for the difference in accuracy between FOILMP and Aleph, using a one-paired t-test: * is 95.0%, ** is 99.0%; no superscripts denote confidence levels below 95%.

Activity	FOILMP		Aleph	
	Accuracy(%) (overall)	Accuracy(%) (only positive)	Accuracy(%) (overall)	Accuracy(%) (only positive)
D1	97.0	85.5	96.0*	78.6**
D2	88.1	79.1	86.4*	70.5*
D3	93.4	78.4	93.1	75.1*
D4	88.4	85.1	87.6*	83.2*

posed method predicts the test examples more accurately than earlier ILP approaches.

For future work, the scaling factor of these features should be considered in heuristic value calculation, so that the system can produce more suitable heuristics from training data. Moreover, the current system uses only bonds as relations between atoms. We are planing to improve background knowledge by exploring more complex substructure of a molecule and evaluating the proposed system on other domains.

Acknowledgments

The authors thank anonymous reviewers for valuable comments. This research was supported by the Active Mining Project (Grant-in-Aid for Scientific Research on Priority Areas, No.759).

References

Chevaleyre, Y., & Zucker, J.-D. (2001). A framework for learning rules from multiple instance data. *Proc. 12th European Conf. on Machine Learning* (pp. 49–60). Springer.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Finn, P., Muggleton, S., Page, D., & Srinivasan, A. (1998). Pharmacophore discovery using the inductive logic programming system PROGOL. *Machine Learning*, *30*, 241–270.

Gärtner, T., Flach, P. A., Kowalczyk, A., & Smola, A. J. (2002). Multi-instance kernels. *Proc. 19th International Conf. on Machine Learning* (pp. 179–186). Morgan Kaufmann.

King, R. D., Sternberg, M. J. E., & Srinivasan, A. (1995). Relating chemical activity to structure: An examination of ILP successes. *New Generation Computing*, *13*, 411–433.

Marchand-Geneste, N., Watson, K. A., Alsberg, B. K., & King, R. D. (2002). New approach to pharmacophore mapping and qsar analysis using inductive logic programming. application to thermolysin inhibitors and glycogen phosphorylase b inhibitors. *Journal of Medicinal Chemistry*, *45*, 399–409.

Maron, O., & Lozano-Pérez, T. (1998). A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*. The MIT Press.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, *13*, 245–286.

Okada, T. (2002). Active user's response: Lessons from the structure-activity relationship analysis of dopamine antagonists. *Proc. IEEE International Conference on Data Mining* (pp. 103–107).

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*, 239–266.

Selman, B., Levesque, H. J., & Mitchell, D. (1992). A new method for solving hard satisfiability problems. *Proceedings 10th National Conference on Artificial Intelligence* (pp. 440–446).

Srinivasan, A. (2001). The Aleph manual. http://web.comlab.ox.ac.uk/oucl/research/areas-/machlearn/Aleph/.

Srinivasan, A., Muggleton, S., King, R., & Sternberg, M. (1994). Mutagenesis: ILP experiments in a non-determinate biological domain. *Proc. 4th International Workshop on Inductive Logic Programming* (pp. 217–232).

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*, 277–299.