社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ICS－138　(39)
2004／12／7

Partial Rule Weighting Using Single-Layer Perceptron

Sukree Sinthupinyo,† Cholwich Nattee,† Masayuki Numao,†
Takashi Okada†† and Boonserm Kijsirikul†††

Inductive Logic Programming (ILP) has been widely used in Knowledge Discovery in Databases (KDD). The ordinary ILP systems work in two-class domains, not in multi-class domains. We have proposed the method which is be able to help ILP in multi-class domains by using the partial rules extracted from the ILP's rules combined with weighting algorithm to classify unseen examples. In this paper, we improve the weighting algorithm by using single layer perceptron. The learned weights from the perceptrons and the partial rules are then combined to represent the knowledge extracted from the domains. The accuracy of the proposed method on classification of a real-world data set, dopamine antagonist molecules, shows that our approach can remarkably improve the previous weighting algorithm and the original ILP's rules.

1. Introduction

Various ILP techniques and systems have been recently applied to Knowledge Discovery in Databases (KDD).[2),3),5),11),12)] Standard ILP systems are designed to suit two-class problems in which the output rules are constructed to cover all positive examples but none of negative examples. In classifying process, to predict the class of the examples, the examples which are covered by the rule(s) are classified as positive class, while the ones which are not covered by any rules are classified as negative class. However, in using ILP in multi-class domains, an example may either match with the rules from different classes or it may not match with any rules. Thus, ILP alone cannot classify such examples.

In[15)], we have proposed the technique which can help ILP in multi-class domains. In our approach, the partial rules are first extracted from the original ILP's rules and then are collaboratively used to classify unseen examples. Finally, the weights are attached to the partial rules to represent the knowledge from the domains. To classify an unseen example, we first apply all partial rules to the example. Only covering partial rules are considered. The weights for each class from all covering partial rules are then totalized. Finally, the class with the highest value is selected as the class of the example.

The main idea of our work is that the partial rules which are more general than the original rules can easily cover the example. Hence, the problem is now how to classify the examples which are covered by several rules from different classes. In this paper, we propose an approach which can determine the weights of the partial rules using simple single layer perceptron.

We also evaluated our approach on a real-world data set, dopamine antagonist molecules. In human brain, the dopamine molecules function as the neurotransmitters. They bind to the dopamine receptors in order to send the signal between neurons. The excessive levels of the dopamines have been implicated in schizophrenia. Hence, for the medical treatment of schizophrenic patients, the dopamine antagonist molecules which can block the binding between dopamines and dopamine receptors are used to reduce the signal transfer level. This can limit the effect of the high density of the dopamines. Therefore, the knowledge extracted from this data set may be useful in schizophrenic drug discovery.

The paper is organized as follows. In the next section, we present the concept of partial rule. Weighting algorithm is expressed in Section 3. The details of the experiments are presented in Section 4. The paper ends with the conclusion in Section 5.

† The Institute of Scientific and Industrial Research, Osaka University
†† School of Science and Technology, Kwansei Gakuin University
††† Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University

2. Partial Rule Concept

A *partial rule* is *a rule whose body contains a valid sequence of the literals, from the body of the original rule, which starts with the literal consuming the input variables in the head of the rule*. The partial rule extraction algorithm is based on the idea of the newly introduced variables, similar idea as the feature extraction in BANNAR[4]. The partial rule extraction strategy is described below:

- Given the original rule $l_0 \leftarrow l_1, l_2, ..., l_n$ where l_0 is the literal in the head and $l_1, l_2, ..., l_n$ are the literals in the body of the rule.
- Construct all possible *primitive partial rules* $p_0 \leftarrow p_1, p_2, ..., p_m$, where literal p_0 is l_0; p_i where $i = 1...m$ is the literal selected from $l_1, l_2, ..., l_n$; literal p_{i+1} consumes the variable(s) newly introduced in p_i; and literal p_m does not introduce new variable or there is no literal in $l_1, l_2, ..., l_n$ consuming new variable introduced in p_m.
- Make all possible combinations of the primitive partial rules, constructed from the previous step, which have the common variables not occurring in the head l_0.

For example, given the original rule:

```
molecule(A) :- atm(A, B, C, D, E, F),
   C=n, E=2.8, bond(A, G, B, H, I, J),
   gteq(J, 1.5).
```

The primitive partial rules are:

```
molecule(A) :- atm(A, B, C, D, E, F),
   C=n.
molecule(A) :- atm(A, B, C, D, E, F),
   E=2.8.
molecule(A) :- atm(A, B, C, D, E, F),
   bond(A, G, B, H, I, J),
   gteq(J, 1.5).
```

All combinations of the primitive partial rules which have the common variables not occurring in the head of the rule are:

```
molecule(A) :- atm(A, B, C, D, E, F),
   C=n, E=2.8.
molecule(A) :- atm(A, B, C, D, E, F),
   C=n, bond(A, G, B, H, I, J),
   gteq(J, 1.5).
molecule(A) :- atm(A, B, C, D, E, F),
   E=2.8, bond(A, G, B, H, I, J),
   gteq(J, 1.5).
molecule(A) :- atm(A, B, C, D, E, F),
   C=n, E=2.8, bond(A, G, B, H, I, J),
   gteq(J, 1.5).
```

¿From the above extraction strategy and the shown example, we can see that the partial rules are some parts of the original rule. The set of literals in the body of the partial rule is subset of the literals in the body of the original rule. So that the partial rules are more general than the original rule. Fig.1 shows the coverage of the partial rules and the original rules.

In the figure, assume that there are three classes and the rules for each class are constructed. From Fig.1(a), we can see that there are some examples that are not covered by any rules and some that are covered by the rules from different classes. In the case of the examples covered by the rules from different classes, we can use some tie breaker methods, for example Majority Class[1),6)], to determine the class of them. However, in the other case, for the examples that are not covered by any rules, we have no coverage information for determining the class of them.

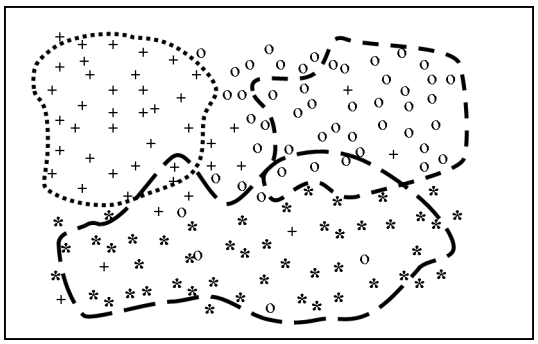
In Fig.1(b), when we extract the partial rules from the original rules and use them instead of the original ones, the covered area becomes larger. The number of the uncovered examples decreases, while the number of the multiple covered examples increases. Thus, the problem is now how to predict the class of the examples when there are many rules covering them.

3. Partial Rule Weighting Algorithm

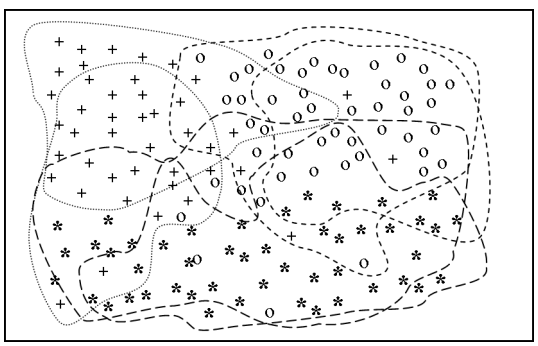
In the previous section, we presented the extraction strategy of the partial rule and showed that we now need the algorithm to determine the classes of the examples which are covered by many rules.

The idea of our work is that we assign the importance to each partial rule in the form of the weights for each class. In[15)], we have proposed the idea of using the Winnow-based algorithm[7)] to assign the weights to the partial rules. In order to use the Winnow-based algorithm, we only focus on the partial rules which cover the example. The weights for each class of the covering partial rules are totalized. The class with the highest value is selected as the class of the example.

In this paper, we propose the different weighting algorithm. We employ the single layer per-



(a)



(b)

Fig. 1 Problem space of (a) original rules and (b) partial rules

ceptron to determine the weight of each partial rule. One perceptron represents one class. Thus, in our domain (as will be described in Section 4), four perceptrons can be used to represent four classes of dopamine antagonist molecules. The input nodes of the perceptron represent the boolean value of the partial rules when we apply each partial rule to the example.

In training perceptron process, to construct the input vector, we apply all partial rules to the example . The applicable partial rules are represented by 1, while the inapplicable partial rules are represented by -1. To construct the output vector, we assign 1 for the node that represents the class of the example and 0 for the others.

After the perceptrons are trained, the knowledge of the domain is hidden in the form of the weights of the link and the structure of the perceptrons. We thus extract the knowledge from the perceptrons by directly attaching the weight of the link, from the input node to the perceptrons, to each partial rule represented by that input node. For example, the weights and the partial rules from the perceptrons in Fig.2 can be represented as:

$$\begin{array}{ll} (& ((w_{0,1}, w_{0,2}, ..., w_{0,n}), true), \\ & ((w_{1,1}, w_{1,2}, ..., w_{1,n}), P_1), \\ & ((w_{2,1}, w_{2,2}, ..., w_{2,n}), P_2), \\ & ((w_{3,1}, w_{3,2}, ..., w_{3,n}), P_3), \\ & ... \\ & ((w_{m,n}, w_{m,n}, w_{m,n}, w_{m,n}), P_m) \quad) \end{array}$$

where $w_{0,j}$ is the bias value of output node j, and $w_{i,j}$ is the weight of the link between input node i and output node j.

To classify unseen examples, we use the following strategy:

Given a problem with m partial rules and n classes. W_i is a vector of length n, where the element $w_{i,j}$ of W_i is the weight of the link between input node i and output node j, P_i is the partial rule represented by input node i, W_0 is a vector of the bias of output nodes, and $P_0 = true$.

To classify example e, we use the following strategy.

- Let V is a vector of length n, initially with $V = W_0$.
- Apply all P_i to e. If P_i is applicable, $V \leftarrow V + W_i$ else, $V \leftarrow V - W_i$.
- Let v_l be the highest value in V, return class l.

4. Experiments

We evaluated our approach on a real-world data set, dopamine antagonist molecules. The data set contained 1366 molecules of 4 classes, D1, D2, D3, and D4[10]. The information of the molecules was originally described in the form of the position in three dimension space of atoms, types of atoms, types of bonds, and dopamine antagonist activity of molecules. We converted the positions of atoms to the relations between atoms and bonds. We instead represented the information of atoms, bonds, and distances between atoms in term of 3 predicates, `atm/6`, `bond/6`, and `link/4`, respectively. The details of these three predicates are described below:

- `atm(A,B,C,D,E,F)` represents that the

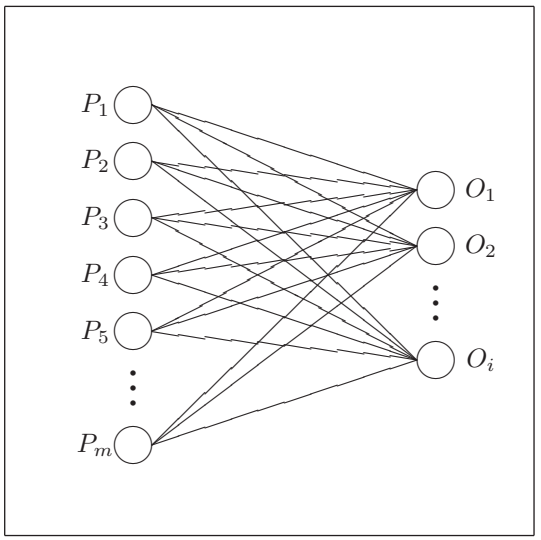


Fig. 2 Single Layer Perceptrons Structure

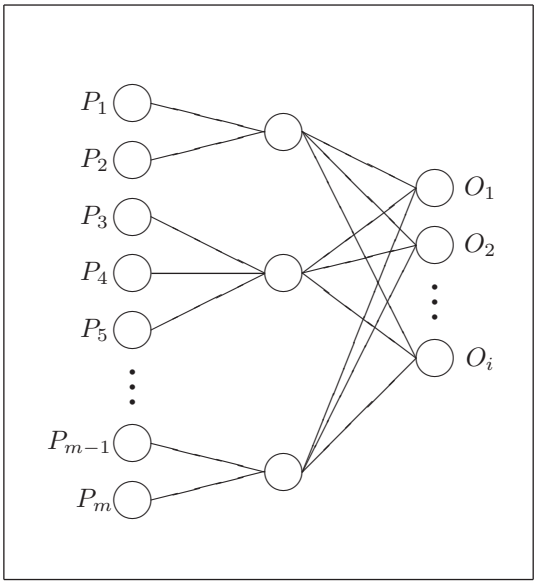


Fig. 3 Multi-Layer Perceptrons Structure

atom `B` is in molecule `A`, is type `C`, forms a bond with oxygen atom if `D` is 1, otherwise it does not link to any oxygen atom, has distance `E` to the nearest oxygen atom, and has distance `F` to the nearest nitrogen atom.

- `bond(A,B,C,D,E,F)` represents that the bond `B` is in molecule `A`, has atoms `C` and `D` on each end, is type `E`, and has length `F`.
- `link(A,B,C,D)` represents that in the molecule `A`, the distance between atoms `B` and `C` is `D`.

The rules were constructed using Aleph system[16]. In our experiments, we selected the randomized search method, provided by the system, using the GSAT algorithm[14] that was originally proposed for solving propositional satisfiability problems.

4.1 Compared Approaches

We compared our approach with other four approaches, i.e. Multi-Layer Perceptrons, Winnow-Based algorithm[7], Majority Class[1),6)] method and Decision Tree Learning algorithm. The brief review of each approach is described below.

Multi-Layer Perceptrons

We conducted the experiment using Multi-Layer Perceptrons (MLP) to compare the effect of using the complex neural network structure to the simple structure used in our approach. The structure of our MLP contained three layers, i.e. *input layer*, *hidden layer*, and *output layer*. The input layer represented the partial rules and the output layer represented the classes, same representation as the structure of the single layer perceptrons, while the hidden layer represented the original rules. We linked the input nodes representing the partial rules to the hidden nodes representing their own original rules. The hidden nodes and output nodes were fully connected. We trained this MLP by using the standard backprapagation algorithm. The structure of the MLP is shown in Fig.3.

Winnow-Based Algorithm

Each partial rule can be viewed as an expert which can vote for the class of test examples. The weights of the partial rules which cover the example are totalized and the class with the highest value is selected as the class of the example. The weights are obtained by using the training strategy below:

Given a problem with m partial rules, n classes, and promotion factor α. P is a vector of length m, where element p_i of P is a partial rule. W_i is a vector of length n, where element $w_{i,j}$ of W_i is the weight of class j of partial rule p_i. V is a summation vector of length m, where v_i of V is the summation of the weights of class i. The weight vector W_i are updated by using the following procedure:

- Initialize all $w_{i,j} = 1$
- Until termination condition is met, Do
  - For each training example e, Do
    - Initialize all $v_i = 0$ and c as the class

of e

- For all partial rules p_i which match with e, add corresponding W_i to V,
$$V = V + W_i$$
- Let v_k be the maximum element in V, predict the example e as class k
- If $c = k$, no update is required; otherwise the weight w_i corresponding to p_i which matches with e is updated by,
$$w_{i,j} = \begin{cases} \alpha w_{i,j} & \text{if } j = k, \\ \alpha^{-1} w_{i,j} & \text{if } j = c. \end{cases}$$

For more details please refer to[15].

Majority Class Method

In the Majority Class method, we selected the class which had the maximum number of examples in training set as the default class. An example which matched with only rule(s) from one class was classified as that class, while an example which could not match with any rule was classified as the default class. In case of the examples which matched with the rules from two or more classes, we selected the class of which the matched rules covered maximum number of examples.

Decision Tree Learning Algorithm

Decision Tree Learning (DTL) is a well-known propositional Machine Learning technique which employs the Information Theory to guide in searching for the best theory. DTL has been successfully applied to many attribute-value real-world domains[8),9),17]. The decision tree learner used in our experiments was C4.5 system[13].

We trained C4.5 using the features obtained by comparing the partial rules with an example, and these features were a set of truth values (either $true$ or $false$). The features were the same as those used as the input vector of the neural network. The reason that we selected C4.5 to apply to the same feature set is to compare the efficiency of the weights from the neural network which is employed in the proposed method with the decision tree.

4.2 Experimental Results

We ran 10-fold cross validation method. The proposed approach (PR+SLP) was compared to the other four approaches, i.e. ILP with Majority Class method (ILP+Majority Class), Partial Rule and C4.5 (PR+C4.5), Partial Rule and Winnow (PR+Winnow), and Partial Rule and Multi-Layer Perceptron (PR+MLP). The results are shown in Table 1.

Table 1 The accuracy of the compared approaches.

Method	Accuracy (%)
ILP+Majority Class	79.11±4.37
PR+C4.5	85.71±3.41
PR+Winnow	88.65±3.85
PR+MLP	89.65±3.78
PR+SLP	92.03±3.14

The accuracy of combining partial rules and single layer perceptron (PR+SLP) is higher than combining with multi-layer perceptron with 95% confidence level using the standard paired t-test method, while it is higher than the other approaches with 99.5% confidence level using the same method.

In Table 2, the numbers of correctly classified examples on three portions, i.e. exactly covered, multiple covered and uncovered, are shown. The Exactly Covered column indicates the number of the examples covered by the rule(s) from only one class, the Multiple Covered column indicates the number of the examples covered by the rules from different classes, and the Uncovered column indicates the number of the examples which are not covered by any rule. Furthermore, we ran another experiment to show the effect of using partial rule by combining only the original rules and the single layer perceptrons. The structure of the perceptrons was same as shown in Fig.2 but we used the boolean value obtained when we applied the orignal rules, instead of the partial rules, to the examples.

The results in Table 2 show that the improvement of our approach over the original rules came from both multiple covered and uncovered portions. Additionally, we can see from the second row that when we combined the original rules and the single layer perceptrons the accuracy was much improved on multiple covered portion, while the accuracy was slightly improved on uncovered portion.

5. Conclusion

We have proposed an approach that employs single layer perceptron to determine the importance of the partial rules. The partial rules are used instead of the original rules to en-

Table 2 Improvements of using partial rules and single layer perceptrons over the original rules with Majority Class and Multi-Layer Perceptron, reported according to exactly covered examples, multiple covered examples, and uncovered examples.

Method	Exactly Covered	Multiple Covered	Uncovered
ILP+Majority Class	965/1049	49/97	68/220
ILP+SLP	962/1049	66/97	85/220
PR+SLP	991/1049	81/97	185/220

large the coverage on the examples. The partial rules and the weights from the perceptrons are combined to classify unseen examples. The results show that the proposed method remarkably improved the accuracy of the original rules and achieved a better result than the other approaches, evaluated on classifying the activities of the dopamine antagonist molecules.

References

1) P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. Fifth European Working Session on Learning*, pages 151–163, Berlin, 1991. Springer.
2) S. Džeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 118–152. 1996.
3) P.W. Finn, S.Muggleton, D.Page, and A.Srinivasan. Pharmacophore Discovery Using the Inductive Logic Programming System (PROGOL). *Machine Learning*, 30(2-3), 1998.
4) B.Kijsirikul, S.Sinthupinyo, and K.Chongkasemwongse. Approximate match of rules using backpropagation neural networks. *Machine Learning*, 44(3):273–299, 2001.
5) S. Kramer, B. Pfahringer, and C. Helma. Mining for causes of cancer: Machine learning experiments at various levels of detail. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, Menlo Park, CA, 1997. AAAI Press.
6) W. V. Laer, L. D. Raedt, and S. Dzeroski. On multi-class problems and discretization in inductive logic programming. In *International Syposium on Methodologies for Intelligent Systems*, pages 277–286, 1997.
7) N. Littlestone. Learning quickly when irrelevant attributes abound: A new linearthreshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
8) H. Milde, L. Hotz, J. Kahl, B. Neumann, and S. Wessel. MAD: A real world application of qualitative model-based decision tree generation for diagnosis. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 246–255, 1999.
9) H.Milde, L.Hotz, J.Kahl, and S.Wessel. Qualitative model-based decision tree generation for diagnosis in real world industrial application. *KI - Kunstliche Intelligenz*, 13(3):30–35, 1999.
10) C. Nattee, S. Sinthupinyo, M. Numao, and T. Okada. Knowledge discovery using inductive logic programming in dopamine antagonist structure data. Technical report, I.S.I.R., Osaka University, 2004. http://libra.ai.sanken.osaka-u.ac.jp/usr/cholwich/techreport.pdf.
11) U.Pompe, I.Kononenko, and T.Makše. An application of ILP in a musical database: Learning to compose the two-voice counterpoint. In *Proceedings of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programing*, pages 1–11, 1996.
12) R. Quiniou, M.-O. Cordier, G. Carrault, and F.Wang. Application of ILP to cardiac arrhythmia characterization for chronicle recognition. *Lecture Notes in Computer Science*, 2157:220–227, 2001.
13) J.Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, 1993.
14) B. Selman, H. J. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proc. 10th National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.
15) S. Sinthupinyo, C. Nattee, M. Numao, T. Okada, and B. Kijsirikul. Combining partial rules and winnow algorithm: Results on classification of dopamine antagonist molecules. In *The Third Internation Workshop on Active Mining*, 2004.
16) A. Srinivasan. The Aleph Manual, 2001.
17) N. Ye, X. Li, and S. M. Emran. Decision tree for signature recognition and state classification. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop June 6-7, 2000 at West Point, New York*, pages 194–199, June 2000.