Improving Multiclass ILP by Combining Partial Rules with Winnow Algorithm: Results on Classification of Dopamine Antagonist Molecules

Sukree Sinthupinyo[1], Cholwich Nattee[1], Masayuki Numao[1], Takashi Okada[2], and Boonserm Kijsirikul[3]

[1] Department of Architecture for Intelligence, The Institute of Scientific and Industrial Research, Osaka University,
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{sukree,cholwich,numao}@ai.sanken.osaka-u.ac.jp
[2] Center for Information & Media Studies, Kwansei Gakuin University
okada@kwansei.ac.jp
[3] Department of Computer Engineering, Chulalongkorn University
boonserm.k@chula.ac.th

Abstract. In this paper, we propose an approach which can improve Inductive Logic Programming in multiclass problems. This approach is based on the idea that if a whole rule cannot be applied to an example, some partial matches of the rule can be useful. The most suitable class should be the class whose important partial matches cover the example more than those from other classes. Hence, the partial matches of the rule, called *partial rules*, are first extracted from the original rules. Then, we utilize the idea of Winnow algorithm to weight each partial rule. Finally, the partial rules and the weights are combined and used to classify new examples. The weights of partial rules also show another aspect of the knowledge which can be discovered from the data set. In the experiments, we apply our approach to a multiclass real-world problem, classification of dopamine antagonist molecules. The experimental results show that the proposed method gives the improvement over the original rules and yields 88.58% accuracy by running 10-fold cross validation.

1 Introduction

In recent years, Inductive Logic Programming (ILP) has been widely applied to various real-world applications [1, 2]. Standard ILP are usually two-class classifier (positive and negative classes). A test example which matches with some rules is classified as positive class, while the example which does not match with any rule are classified as negative class. This causes some troubles when we need to use ILP in multiclass problems. In such problems, when a test example does not match with any rule or matches with some rules from more than two classes, we cannot determine which class is most suitable for the example.

In this paper, we propose an approach which can utilize the standard ILP's rules in multiclass problems. Our approach is based on the idea that if a whole

rule cannot be applied to an example, some parts of rule may match with that example. Thus, we can make use of these matches to determine the class for the example. The most suitable class should be the class whose the number of important matches is higher than those of other classes. Thus, in our approach, we first extract some part of rule which will be used as *partial rule*. Then, all partial rules are given the importance in term of weights using Winnow-based approach [3]. Finally, the partial rules and the weights are combined and used to classify new examples. Moreover, the weights assigned to the partial rules also show another aspect of the characteristic of data set that is very useful in knowledge discovery fashion.

We apply our approach to a real-world problem, classification of dopamine antagonist molecules. Dopamine antagonist molecule is a kind of molecules which can block the binding between dopamines and dopamine receptors in the signal transfer process in human brain. The excessive levels of the dopamine have been implicated in schizophrenia. Hence, in the medical treatment of schizophrenic patients, the dopamine antagonist molecules are used to decrease the signal transfer level which can limit the effect of the high density of dopamines. The knowledge discovered from this domain may be useful for schizophrenic drug development.

1.1 Related Work

Srinivasan and King [4] proposed the work using ILP to discover new attributes or *features* which are then used by linear regression to predict chemical activity. Our work is different in that we focus on using ordinary ILP's rules in multiclass problems; the extracted features in our work are the parts of the original rules and aimed to be combined with other method to classify new examples in multiclass fashion. The following are the works proposed to help ILP in multiclass problems. Dietterich and Bakiri [5] proposed the method which employs the error correcting code to represent the class of examples and tries to predict the most suitable class for test examples. Round Robin Rule Learning proposed by Fürnkranz [6] focuses on training examples rearrangement. The training examples from each class are used to train the learner several times. A test example is tested with all trained classifiers. The most winning class is selected as the class of the test example. Eineborg and Bostr [7] proposed the method for selecting the class for the uncovered examples, Rule Stretching. The method aims to deal with an uncovered example by generalising the original rules to cover the example and select the most accurate rule as the rule which best matches with the example.

1.2 Paper Outline

The paper is organized as follows. In the next section, we present a concept of ILP and the obstacles when ILP is applied to multiclass problems. The partial rules extraction strategy and the weights adjustment are expressed in Section

3 and Section 4, respectively. The details of the experiments are presented in Section 5. The paper ends with the conclusion in Section 6.

2 Using ILP in Multiclass Problems

ILP is the Machine Learning technique which is originally proposed as a two-class classifier. ILP aims to construct a rule set that covers all positive examples and none of the negatives. The output of ILP is the first-order rules which will be used to classify new examples. This causes some troubles when we need to use ILP in multiclass problems, i.e. (1) how to construct the rule for each class, and (2) how to select the class for each example. In the former case, as mention earlier, ILP systems search for the rules which cover positive examples, however, in multiclass problems, we need to construct the rules for each class. Hence, the additional technieques must be used to help ILP to construct the rules, such as one-against-all, round robin rule learning [6], and loss-based decoding [8]. Nevertheless, in this work, we emphasize on the latter case. Thus, we employ the common method, one-against-all, to construct the rules for each class.

In the one-against-all algorithm, a k-class problem is reduced to k two-class problems. To generate the rules for class i, the training examples are organized by using the training examples of class i as positive examples and using the training examples of class j where $j = 1, ..., k$ and $j \neq i$ as negative examples. For example, our data set contains 4 classes, i.e. D1, D2, D3, and D4 (as will be described in Section 5). We use the training examples of class D1 as the positive examples and use those of classes D2, D3, and D4 as the negatives for learning rules of class D1. Using this strategy, the obtained rules are unordered.

The problem of class selection arises when an example does not exactly match with any rule or matches with some rules from more than two classes, especially in case of unordered rules. In case of ordered rules, the class selection is not complicated, the example which does not exactly match with any rule can be classified as the default class, while the example which matches with multiple rules from different classes can be classified as the class of the higher order rule. However, in case of unordered rules, as constructed in this work, ILP's rules alone cannot select the class of the example which does not match with any rule or matches with multiple rules from different classes. Hence, we propose an approach which is based on the idea that if the whole rule cannot be applied to the example, we can utilize some partial matches of the rule to determine the most appropriate class.

In our experiments, we employ an ILP system, Aleph [9], to construct the rules for each class. The rule construction of Aleph starts with building the most specific clause, called *bottom clause*. Then, to seek for the best generalised clause, Aleph provides many search algorithms which users can select the most suitable one for their domain. In our experiments, we selected the randomized search method using an altered form of the GSAT algorithm [10] that was originally proposed for solving propositional satisfiability problems. The GSAT algorithm provided by Aleph is modified to suit the clause searching process in ILP fashion.

3 Partial Rules

As described in the previous section, our approach is based on the idea that some partial matches in the rule can be used to classify the unclassifiable examples. Hence, several parts of a rule or *partial rules* are first extracted from the original rules. Then, they are used to classify unseen examples collaboratively. The following describes our partial rule extraction algorithm.

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 [11]. As shown in Fig. 1, each input variable in a literal is introduced as a new variable in some preceding literals. Thus, we group the literal which consumes the new variable and the literal which introduces that variable into the same sequence. For example, in Fig. 1, the new variable `D` introduced in literal `link(A, B, C, D)` is used as the input variable of literal `D=2.7`. Thus, we group these two literals into the same sequence, `link(A, B, C, D), D=2.7`.

Our 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_i when $i = 1...n$ is the literal in the body of the rule.
- Construct all possible primitive partial rules $p_0 \leftarrow p_1, p_2, ..., p_m$ where p_0 is l_0; p_i when $i = 1...m$ is the literal selected from l_i; p_{i+1} consumes the variable(s) newly introduced in p_i; p_m does not introduce new variable or there is no l_i 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, from the original rule, as shown in Fig. 1:

```
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).
```

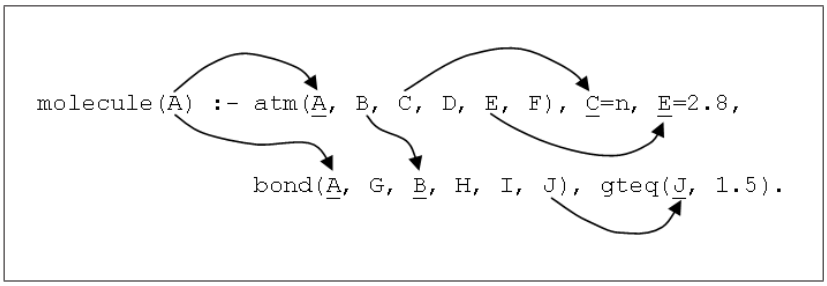

Fig. 1. New variables consumption. The underlined characters show the input variables of literals

4 Weights Adjustment Using Winnow-Based Approach

As described earlier, our approach is based on the idea that some partial matches can be used to classified new examples. Thus, we extract the partial rules from the original rules and use them collaboratively for classifying examples. The idea of using many partial rules to classify an example is that the partial rules are assigned the importance in form of the weights of each class and all applicable partial rules are combined with their weights for determining the class of the example. When we need to classify an example, we determine the summation of the weight of each class of all partial rules which match with the example. The class which has the highest summation of weights is selected as the class of the example.

The Winnow algorithm [3] is originally proposed as a linear threshold algorithm. For an input vector x, weight vector w, promotion factor $\alpha > 1$, and threshold $\theta > 0$, the algorithm predicts 1 if $w \cdot x \geq \theta$. Intuitively speaking, the Winnow algorithm activates the output if the input x is high enough. If $w \cdot x$ is too low, the weight vector w is increased by updating $w_i \leftarrow \alpha^{x_i} w_i$. On the other hand, if $w \cdot x$ is too high, the weight vector w is decreased by updating $w_i \leftarrow \alpha^{-x_i} w_i$. However, in our approach, the concept of prediction scheme is different. In our class prediction, instead of comparing the summation to the threshold we need only the highest summation of the weight from each class, so that we can make use of the Winnow algorithm by employing the following strategy.

Given a problem with n partial rules, m classes, and promotion factor α. P is a vector of length n, where element p_i of P is a partial rule. W_i is a vector of length m, 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,

$$w_{i,j} = \begin{cases} \alpha w_{i,j} & \text{if } j = k, \\ \alpha^{-1} w_{i,j} & \text{if } j = c. \end{cases}$$

Each partial rule is weighed by a weight vector of which elements are for each class and we classify an example as the class which has the highest summation of the weights of the applicable partial rules. When an example is incorrectly classified, the output class is different from the target class. This means the summation of the weight of the output class of all applicable partial rules is higher than that of the target class. Thus, we decrease the weights of the output class of all applicable partial rules by using Winnow algorithm's weight updating equation, $w_{i,j} = \alpha^{-1} w_{i,j}$ and increase the weights of the target class of all applicable partial rules by using promotion factor, $w_{i,j} = \alpha w_{i,j}$.

To classify an unseen example e, we use the following strategy.

- Initialize all $v_i = 0$
- 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, classify the example e as class k

5 Experiments

The data set used in the experiments contained 1366 molecules of dopamine antagonist molecules of 4 classes, D1, D2, D3, and D4. Information of the molecules was originally described in term of the position in three dimension space of atoms, types of atoms, types of bonds, and dopamine antagonist activity of molecules. However, the position in three dimension space was not useful for discriminating

examples because a molecule could rotate or move to other positions in the space. Hence, we converted the position of atoms to the relation 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 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 following is an example of rules obtained from the experiments.

```
molecule(A,d1) :- link(A, B, C, D), bond(A, E, F, C, G, H), D=6.9,
    H=1.4, bond(A, I, J, F, K, H), bond(A, L, M, J, G, H),
    bond(A, N, B, O, G, P).
[Positive cover = 53 Negative cover = 5]

molecule(A,d2) :- atm(A, B, C, D, E, F), C=n, bond(A, G, B, H, I,
    J), gteq(J, 1.5), atm(A, L, M, D, N, O), N=5.1, O=1.5.
[Positive cover = 42 Negative cover = 1]

molecule(A,d3) :- link(A, B, C, D), D=4.1, atm(A, B, E, F, G, H),
    H=4.1, bond(A, I, B, J, K, L), bond(A, M, C, N, K, L).
[Positive cover = 56 Negative cover = 1]

molecule(A,d4) :- link(A, B, C, D), D=4.4, bond(A, E, C, F, G, H),
    bond(A, I, F, J, G, H), bond(A, K, L, C, G, H), bond(A, M, J,
    N, G, H).
[Positive cover = 130 Negative cover = 8]
```

We compared our approach with other two approaches, i.e. Majority Class [12, 13] and Decision Tree Learning. As described in Section 2, ILP's rules alone cannot classify the examples which match with multiple rules from different classes or do not match with any rule, so that we make the rules be fairly compared to other methods by using the Majority Class in such cases.

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.

Another method compared in our experiment is the Decision Tree Learning (DTL) algorithm. DTL is a well-known propositional Machine Learning technique which employs the Information Theory to guide in searching for the best theories. The decision tree learner used in our experiments is C4.5 system [14]. To compare our method to C4.5, we used the truth values obtained by comparing the partial rules with examples as the attributes of the examples. By using this attribute set, the examples originally represented as first order logic were transformed into propositional representation. Finally, these transformed examples are used as the training examples for C4.5. For example, assume that we have 8 partial rules. When we compare the example, `molecule(m06497)`, with all partial rules, the second and fifth partial rules are *true* while the other partial rules are $false$. The training examples for C4.5 of `molecule(m06497)` will be $< false, true, false, false, true, false, false, false >$.

We ran 10-fold cross validation experiment using three methods, the original ILP system with the Majority Class method (ILP+Majority Class), Partial Rules and DTL (PR+DTL), and our approach, Partial Rules and Winnow algorithm (PR+Winnow).

The accuracy shown in Table 1 was separately evaluated when the rules were used as in two-class fashion. The covered examples were classified as positive, while the uncovered examples were classified as negative. The accuracy of each class was obtained from the test set consisting of only the examples from the test set of that class. The accuracy in Table 1 shows the accuracy of the rules from each class. The average accuracy of all classes is 76.42%. Furthermore, this percentage of accuracy also shows the coverage ratio on the test set.

Table 2 shows the accuracy of each approach in classifying test examples in multi-class fashion. The accuracy of ILP+Majority Class approach is 79.21%. This shows that the only the Majority Class method can slightly improve the accuracy of the original rules. The accuracy of PR+DTL is 85.72%, higher than ILP+Majority with 99.5% confidence level using the standard paired t-test method. The accuracy of PR+Winnow is 88.58%, higher than ILP+Majority and PR+DTL methods with 99.5% and 99.0% confidence level respectively using the same comparing method.

Table 3 shows the ratio between the number of examples correctly classified and the number of examples for each portion. Exactly Covered column indicates the number of the examples covered by the rule(s) from only one class, Multiple Covered column indicates the number of the examples covered by the rules from different classes, and Uncovered column indicates the number of the examples which are not covered by any rule. The results show that our approach much improved the accuracy in Multiple Covered and Uncovered portion. In Multiple Covered portion, PR+Winnow correctly classified 77 of 97 examples, whereas only 49 examples were correctly classified by the Majority Class method. For the uncovered examples, PR+Winnow correctly classified 171 of 220 examples, while only 68 examples were correctly classified by the Majority Class method.

An example of some partial rules which are highly weighed is shown below.

```
molecule(A) :- atm(A, E, F, G, H, I), bond(A, N, E, O, P, M),
```

Table 1. The accuracy of the output rules used to classify only the positive examples of each class

Class	Accuracy (%)
D1	77.42
D2	70.30
D3	74.80
D4	83.16
Average	76.42

Table 2. The accuracy of the compared methods.

Method	Accuracy (%)
ILP+Majority Class	79.11±4.37
PR+DTL	85.71±3.41
PR+Winnow	88.65±3.85

Table 3. Improvements over the original rules with Majority Class method, 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
PR+Winnow	962/1049	77/97	171/220

```
               atm(A, O, F, G, Q, R), H=2.4.
               [0.1999, 33.5451, 0.2812, 0.5303]
[The original rule is
molecule(A) :- link(A, B, C, D), atm(A, E, F, G, H, I), D=5.6,
               H=2.4, gteq(I, 3.8), bond(A, J, B, K, L, M),
               bond(A, N, E, O, P, M), atm(A, O, F, G, Q, R),
               lteq(Q, 2.9), lteq(M, 1.4), bond(A, S, C, T, P, U).]

molecule(A) :- atm(A, B, C, D, E, F), bond(A, G, B, H, I, J),
               atm(A, H, K, D, L, M), L=6.5.
               [0.9524, 0.1566, 35.2224, 0.1904]
[The original rule is
molecule(A) :- atm(A, B, C, D, E, F), bond(A, G, B, H, I, J),
               atm(A, H, K, D, L, M), link(A, H, N, O),
               atm(A, N, K, D, L, M), atm(A, H, K, D, L, M),
               L=6.5, F=1.3.]
```

The weights in the above example show another advantage of our approach. We can see that when an example matches with these highly weighed partial rules, the example has the high probability of being classified as the class whose weighted is very high. This provides us some knowledge which can be discovered from the dataset, different from the original rules which sometimes are too specific and not useful. Our approach can seek for some pieces of knowledge which are more important than the others in the original rule. For example, the second partial rule in the above example shows that if an unseen example matches with this partial rule, that example has the high probability of being classified as class 3 which is very highly weighed.

6 Conclusion

We have proposed an approach that can improve ILP in multiclass problem. Our method is based on the idea that the unequal important partial rule matching with an example can be useful for classifying the example. The partial rules are extracted from the original rules and are assigned the importance in term of the weights obtained from Winnow-based approach. The experimental results on classifying the activity of the dopamine antagonist molecules show that our approach was successfully applied to such domain by yielding 88.58% accuracy. The accuracy obtained from the experiments also shows that using only the matching of the partial rules and an attribute learner C4.5 improved the accuracy over using the original rules with the majority class method, and the accuracy was further much improved when using the proposed method. Furthermore, the weights of the partial rules also show some pieces of knowledge which are previously hidden in the original rules.

References

1. Enot, D.P., King, R.D.: Application of Inductive Logic Programming to Structure-based Drug Design. In Lavrac, N., Gamberger, D., Todorovski, L., Blockeel, H., eds.: Proc. 7th European Conf. On Principles and Practice of Knowledge Discovery in Databases., Springer-Verlag (2003)
2. Quiniou, R., Cordier, M.O., Carrault, G., Wang, F.: Application of ILP to cardiac arrhythmia characterization for chronicle recognition. Lecture Notes in Computer Science **2157** (2001) 220–227
3. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linearthreshold algorithm. Machine Learning **2** (1988) 285–318
4. Srinivasan, A., King, R.: Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In Muggleton, S., ed.: Proceedings of the 6th International Workshop on Inductive Logic Programming, Stockholm University, Royal Institute of Technology (1996) 352–367
5. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research **2** (1995) 263–286
6. Fürnkranz, J.: Round robin rule learning. In Brodley, C.E., Danyluk, A.P., eds.: Proceedings of the 18th International Conference on Machine Learning (ICML-01), Williamstown, MA, Morgan Kaufmann Publishers (2001) 146–153
7. Eineborg, M., Boström, H.: Classifying uncovered examples by rule stretching. Lecture Notes in Computer Science **2157** (2001)
8. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. In: Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2000) 9–16
9. Srinivasan, A.: The Aleph Manual (2001)
10. Selman, B., Levesque, H.J., Mitchell, D.: A New Method for Solving Hard Satisfiability Problems. In: Proc. 10th National Conference on Artificial Intelligence, AAAI Press (1992) 440–446
11. Kijsirikul, B., Sinthupinyo, S., Chongkasemwongse, K.: Approximate match of rules using backpropagation neural networks. Machine Learning **44** (2001) 273–299
12. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In: Proc. Fifth European Working Session on Learning, Berlin, Springer (1991) 151–163
13. Laer, W.V., Raedt, L.D., Dzeroski, S.: On multi-class problems and discretization in inductive logic programming. In: International Syposium on Methodologies for Intelligent Systems. (1997) 277–286
14. Quinlan, J.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers (1993)