Constructing a Decision Tree for Graph Structured Data

Warodom Geamsakul, Takashi Matsuda, Tetsuya Yoshida, Hiroshi Motoda and Takashi Washio

Institute of Scientific and Industrial Research,
Osaka University
8-1 Mihogaoka, Ibaraki, Osaka 567-0047, JAPAN
{warodom,matsuda,yoshida,motoda,washio}@ar.sanken.osaka-u.ac.jp

Abstract. Decision tree Graph-Based Induction (DT-GBI) is proposed that constructs a decision tree for graph structured data. Substructures (patterns) are extracted at each node of a decision tree by stepwise pair expansion (pairwise chunking) in GBI to be used as attributes for testing. Since attributes (features) are constructed while a classifier is being constructed, DT-GBI can be conceived as a method for feature construction. The predictive accuracy of a decision tree is affected by which attributes (patterns) are used and how they are constructed. A beam search is employed to extract good enough discriminative patterns within the greedy search framework. Pessimistic pruning is incorporated to avoid overfitting to the training data. Experiments using a DNA dataset were conducted to see the effect of the beam width, the number of chunking at each node of a decision tree, and the pruning. The results indicate that DT-GBI that does not use any prior domain knowledge can construct a decision tree that is comparable to other classifiers constructed using the domain knowledge.

1 Introduction

Since structure is represented by proper relations and a graph can easily represent relations, knowledge discovery from graph structured data poses a general problem for mining from structured data. Examples amenable to graph mining problems are finding typical web browsing patterns, identifying typical substructures of chemical compounds, finding typical subsequences of DNA and discovering diagnostic rules from patient history records.

Graph-Based Induction (GBI) [9, 3], on which DT-GBI is based, discovers typical patterns in general graph structured data by recursively chunking two adjoining nodes. It can handle a graph data having loops with colored/uncolored nodes and links. GBI is very efficient because it employs greedy search. GBI does not lose any information of graph structure after chunking, and it can use various evaluation functions in so far as they are based on frequency. It is not, however, suitable for graph structured data where many nodes share the same label because of its greedy recursive chunking without backtracking, but it is still effective in extracting patterns from data where each node has a distinct label (*e.g.,* World Wide Web browsing data) or where some typical structures exist even if some nodes share the same labels (*e.g.,* chemical structure data containing benzene rings etc). The decision tree construction method [5, 6] is a widely used technique for data classification and prediction, but the data must be represented by or transformed into attribute-value pairs. However, it is not trivial to define proper attributes for graph-structured data beforehand.

We have proposed a method called Decision Tree Graph-Based Induction (DT-GBI), which constructs a classifier (decision tree) for graph-structured data while constructing the attributes during the course of tree building using GBI recursively and did preliminary performance evaluation [8].

A pair extracted by GBI, consisting of nodes and links among them[1] that is treated as an attribute and the existence/non-existence of the pair in a graph is treated as its value for the graph. Thus, attributes (pairs) that split the data effectively are extracted by GBI while a decision tree is being constructed. To classify unseen graph-structured data by the constructed decision tree, attributes that appear in the nodes of the tree are produced from data before the classification.

In this paper we first report an improvement made on DT-GBI after the preliminary analysis [8] to increase its predictive accuracy. After that, we report on the performance evaluation of the improved DT-GBI through experiments using a DNA dataset from the UCI repository and show that the results are comparable to the results that are obtained by using the domain knowledge [7].

Section 2 briefly describes the framework of DT-GBI and Section 3 describes the improvement made on DT-GBI. Evaluation of the improved DT-GBI is reported in Section 4. Section 5 concludes the paper with a summary of the results and the planned future work.

2 Decision Tree Graph-Based Induction

2.1 Graph-Based Induction Revisited

GBI employs the idea of extracting typical patterns by stepwise pair expansion. In the original GBI an assumption was made that typical patterns represent some concepts/substructure and "typicality" is characterized by the pattern's frequency or the value of some evaluation function of its frequency. Repeated chunking enables to extract typical patterns of various sizes. The search is greedy and no backtracking is made. Because of this, all the "typical patterns" that exist in the input graph are not necessarily extracted. The problem of subgraph isomorphism is known to be NP-complete. GBI aims at extracting only meaningful typical patterns of certain sizes. Its objective is not finding all the typical patterns nor finding all the frequent patterns.

For finding a pattern that is of interest any of its subpatterns must be of interest because of the nature of repeated chunking, *i.e.*, a larger pattern can only

[1] Repeated chunking of pairs results in subgraph structure

be constructed by pairing two smaller subpatterns, which must have been constructed at earlier steps. Frequency measure satisfies this monotonicity. However, if the criterion chosen does not satisfy this monotonicity, repeated chunking may not find good patterns even though the best pair based on the criterion is selected at each iteration. To resolve this issue GBI was improved to use two criteria, one for frequency measure for chunking and the other for finding discriminative patterns after chunking. The latter criterion does not necessarily hold monotonicity property. Any function that is discriminative can be used, such as Information Gain [5], Gain Ratio [6] and Gini Index [2], and some others.

The improved stepwise pair expansion algorithm is summarized in Fig. 1. The output of the improved GBI is a set of ranked typical patterns. These patterns are typical in the sense that they are more discriminative than non-selected patterns in terms of the criterion used.

```
GBI($G$)
  Enumerate all the pairs $P_{all}$ in $G$
  Select a subset $P$ of pairs from $P_{all}$ (all the pairs
    in $G$) based on typicality criterion
  Select a pair from $P_{all}$ based on chunking criterion
  Chunk the selected pair into one node $c$
  $G_c$ := contracted graph of $G$
  while termination condition not reached
    $P := P \cup$ GBI($G_c$)
  return $P$
```

Fig. 1. Algorithm of GBI

2.2 Feature Construction by GBI

We regard a subgraph in a graph as an attribute so that graph-structured data can be represented with attribute-value pairs according to the existence of particular subgraphs.

However, it is difficult to identify and extract those subgraphs selectively which are effective for classification task beforehand. If pairs are extended in a step-wise fashion by GBI and discriminative ones are selected and further extended while constructing a decision tree, discriminative patterns (subgraphs) can be constructed simultaneously during the construction of a

```
DT-GBI($D$)
  Create a node $DT$ for $D$
  if termination condition reached
    return $DT$
  else
    $P$ := GBI($D$) (with the number of chunking
      specified)
    Select a pair $p$ from $P$
    Divide $D$ into $D_y$ (with $p$) and $D_n$ (without $p$)
    Chunk the pair $p$ into one node $c$
    $D_{yc}$ := contracted data of $D_y$
    for $D_i := D_{yc}, D_n$
      $DT_i$ := DT-GBI($D_i$)
      Augment $DT$ by attaching $DT_i$ as its child
        along yes(no) branch
  return $DT$
```

Fig. 2. Algorithm of DT-GBI

decision tree. The algorithm of DT-GBI is summarized in Fig. 2. Since the value for an attribute is yes (contains pair) and no (does not contain pair), the constructed decision tree is represented as a binary tree. Each time when an attribute (pair) is selected to split the data, the pair is chunked into a larger node in size. Thus, although initial pairs consist of two nodes and the link between them, attributes useful for classification task are gradually grown up into larger pair (subgraphs) by applying chunking recursively. In this sense the proposed DT-GBI method can be conceived as a method for feature construction.

3 Enhancement of DT-GBI

3.1 Beam Search for Expanding Search Space

Since the search in GBI is greedy and no backtracking is made, which patterns are extracted by GBI depends on which pair is selected for chunking in Fig. 2. To increase the search space and extract good enough patterns still keeping the computational complexity within a tolerant level, a beam search is incorporated to GBI within the framework of greedy search [4]. A certain fixed number of pairs ranked from the top are allowed to be chunked individually in parallel. To prevent each branch from growing exponentially, the total number of pairs to chunk, thus the beam width, is fixed at each level of branch. Thus, at any iteration step, there is always a fixed number of chunking that is performed in parallel.

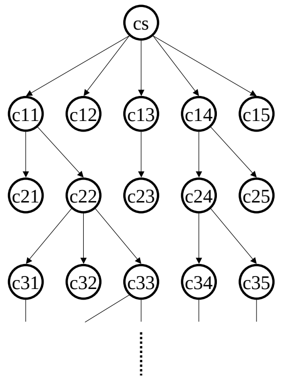

Fig. 3. An Example of state transition with beam search when the beam width = 5

An example of state transition with beam search is shown in Fig.3 in the case where the beam width is 5. The search starts with a single state cs. All pairs in cs are enumerated and ranked according to both the frequency measure and the typicality measure (*e.g.*, discriminatory measure). The top 5 pairs according to the frequency measure are selected, and each of them is used as a pattern to chunk, branching into 5 children c_{11}, c_{12}, ..., c_{15}, each rewritten by the chunked pair. All pairs within these 5 states are enumerated and ranked according to the two measures, and again the top 5 ranked pairs according to the frequency measure are selected. The state c_{11} is split into two states c_{21} and c_{22} because two pairs are selected, but the state c_{12} is deleted because no pair is selected. This is repeated until the stopping condition is satisfied. Increase in the search space improves the pattern extraction capability of GBI and thus that of DT-GBI.

3.2 Pruning Decision Tree

Recursive partitioning of data until each subset in the partition contains data of a single class results in overfitting to the training data and thus degrades the predictive accuracy of decision trees. Our previous approach [8] used a very naive *prepruning* method by limiting the threshold number of graphs in leaves to 10. To improve the predictive accuracy, a pessimistic pruning used in C4.5 [6] is implemented by growing an overfitted tree first and then pruning it based on the confidence interval for binomial distribution. The current algorithm has a step for postpruning in Fig. 2.

4 Performance Evaluation of DT-GBI

The proposed method is tested against the promoter dataset in UCI Machine Learning Repository[1]. This dataset consists of strings that represent nucleotides (one of A, G, T, or C). The input features are 57 sequential DNA nucleotides and the total number of instances is 106 including 53 positive instances (sample promoter sequence) and 53 negative instances (non-promoter sequence). This dataset was explained and analyzed in [7]. In their analysis, they first configured a neural network using the domain knowledge and refined it to best fit the results. The rules were then extracted from the converged network. Thus, their method needs the domain knowledge to guide the search.

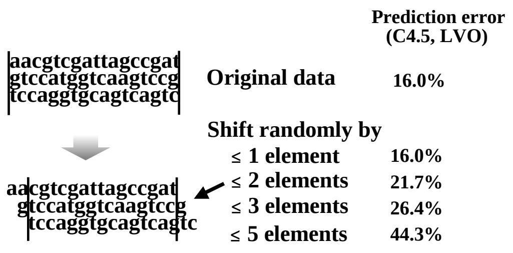

Fig. 4. Change of error rate by shifting the sequence in the promoter dataset

One important thing is that the data is so prepared that each sequence of nucleotides is aligned at a reference point, which makes it possible to assign the n-th attribute to the n-th nucleotide in the attribute-attribute value representation (thus applicable to a neural network). In a sense, this dataset is encoded using the domain knowledge. This is confirmed by the following experiment. Running C4.5 gives a predictive error of 16.0% by leaving one out cross validation. Randomly shifting the sequence by 3 elements gives 21.7% and by 5 elements 44.3% (Fig. 4). If the data is not properly aligned, standard classifiers such as C4.5 that use attribute-attribute value representation does not solve this problem. One of the advantages of graph representation is that it does not require the data to be aligned around a reference point. In this paper, each sequence is converted to a graph representation assuming that an element interacts up to 10 elements on both sides (See Fig. 5). Each sequence thus results in a graph with 57 nodes and 515 lines.

In the experiment, frequency (chunking measure) was used to select a pair to chunk in GBI and information gain [5] (typicality measure) was used in DT-GBI to select a pair from the pairs that are returned by GBI. A decision tree was constructed in either of the following two ways: 1) apply chunking n_r times only at the root node and only once at other nodes of a decision tree, and 2) apply chunking n_e times at every node of a decision tree. Note that n_r and n_e are defined along the depth in Fig. 3. Thus, there is more chunking taking place during the search when the beam width is larger. The pair (subgraph) that is selected for each node of the decision tree is the one which maximizes the informaion gain among all the pairs that are enumerated. Pruning of decision tree was conducted either by a) prepruning: set the termination condition in DT-GBI in Fig. 2 to whether the number of graphs in D is equal to or less than 10 or b) postpruning: conduct the pessimistic pruning in Subsection 3.2 by setting the confidence level to 25%. The beam width was changed from 1 to 15. The prediction error rate of a decision tree constructed by DT-GBI was evaluated by the average of 10 runs of 10 fold cross validation.

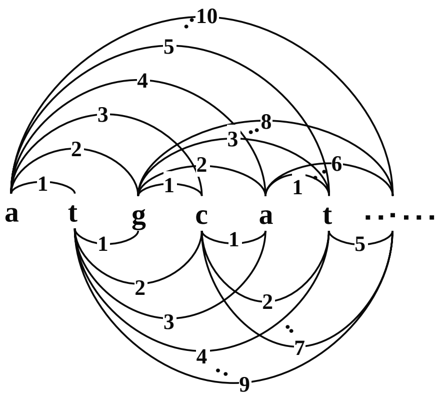

Fig. 5. Conversion of DNA Sequence Data to a graph

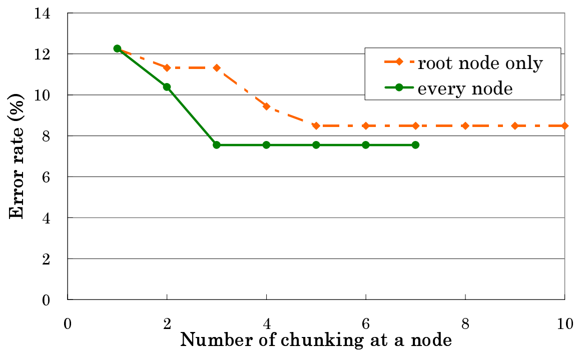

Fig. 6. Result of experiment (beam width=1, without pessimistic pruning)

The first experiment focused on the effect of the number of chunking at each node of a decision tree and thus beam width was set to 1 and the prepruning was used. The parameters n_r and n_e were changed from 1 to 10 in accordance with 1) and 2) explained in the previous paragraph, respectively. Fig. 6 shows the result of experiments. In this figure the dotted line indicates the error rate for 1) and the solid line for 2). The best error rate was 8.49% when $n_r = 5$ for 1) and 7.55% when $n_e = 3$ for 2). The corresonding induced decision trees are shown shown in Fig. 7 ($n_r = 5$) and Fig. 8 ($n_e = 3$). The decrease of error rate levels off when the the number of chunking increases for both 1) and 2). The result shows that repeated application of chunking at every node results in constructing a decision tree with better predictive accuracy.

The second experiment focused on the effect of beam width, changing its value from 1 to 15 using pessimistic pruning. The number of chunking was fixed at the best number which was determined by the first experiment in Fig. 6, namely, $n_r = 5$ for 1) and $n_e = 3$ for 2).

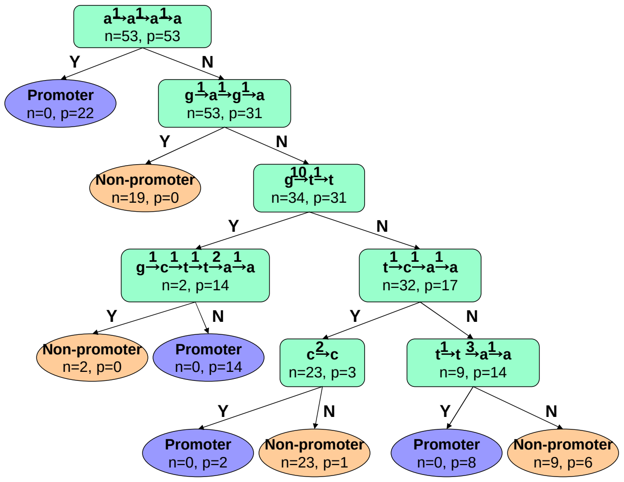

Fig. 7. Example of constructed decision tree (chunking applied 5 times only at the root node, beam width = 1, with prepruning)

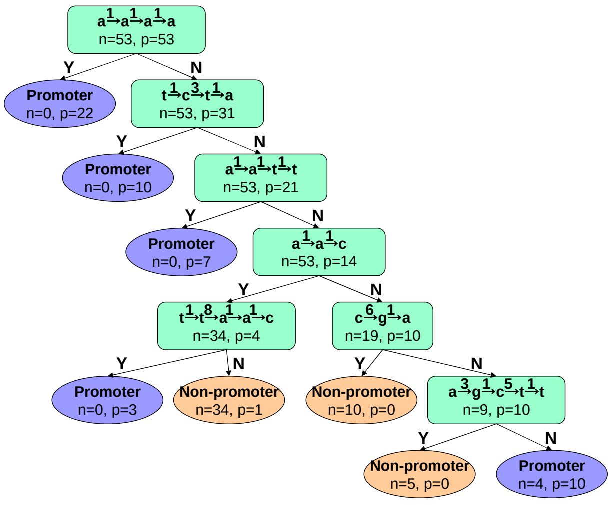

Fig. 8. Example of constructed decision tree (chunking applied 3 times at every node, beam width = 1, with prepruning)

The result is summarized in Fig. 11. The best error rate was 4.06% when the beam width = 12 for 1) ($n_r = 5$) and 3.77% when the beam width = 8 for 1) ($n_e = 3$). Examples of decision tree are shown in Fig. 9 and Fig. 10. Fig. 12 shows yet another result when prepruning was used.

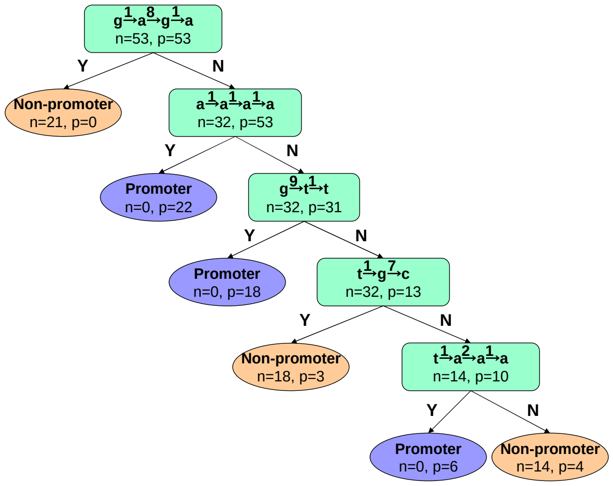
$g \xrightarrow{1} a \xrightarrow{8} g \xrightarrow{1} a$
n=53, p=53
Y
N
Non-promoter
n=21, p=0
$a \xrightarrow{1} a \xrightarrow{1} a \xrightarrow{1} a$
n=32, p=53
Y
N
Promoter
n=0, p=22
$g \xrightarrow{9} t \xrightarrow{1} t$
n=32, p=31
Y
N
Promoter
n=0, p=18
$t \xrightarrow{1} g \xrightarrow{7} c$
n=32, p=13
Y
N
Non-promoter
n=18, p=3
$t \xrightarrow{1} a \xrightarrow{2} a \xrightarrow{1} a$
n=14, p=10
Y
N
Promoter
n=0, p=6
Non-promoter
n=14, p=4

Fig. 9. Example of constructed decision tree (chunking applied 5 times only at the root node, beam width = 12, with pessimistic pruning)

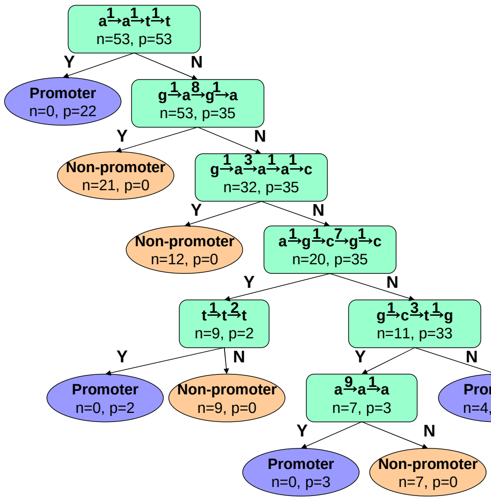
$a \xrightarrow{1} a \xrightarrow{1} t \xrightarrow{1} t$
n=53, p=53
Y
N
Promoter
n=0, p=22
$g \xrightarrow{1} a \xrightarrow{8} g \xrightarrow{1} a$
n=53, p=35
Y
N
Non-promoter
n=21, p=0
$g \xrightarrow{1} a \xrightarrow{3} a \xrightarrow{1} a \xrightarrow{1} c$
n=32, p=35
Y
N
Non-promoter
n=12, p=0
$a \xrightarrow{1} g \xrightarrow{1} c \xrightarrow{7} g \xrightarrow{1} c$
n=20, p=35
Y
N
$t \xrightarrow{1} t \xrightarrow{2} t$
n=9, p=2
$g \xrightarrow{1} c \xrightarrow{3} t \xrightarrow{1} g$
n=11, p=33
Y
N
Promoter
n=0, p=2
Non-promoter
n=9, p=0
Y
N
$a \xrightarrow{9} a \xrightarrow{1} a$
n=7, p=3
Promoter
n=4, p=30
Y
N
Promoter
n=0, p=3
Non-promoter
n=7, p=0

Fig. 10. Example of constructed decision tree (chunking applied 3 times at every node, beam width = 8, with pessimistic pruning)

The result reported in [7] is 3.8 % (they also used 10-fold cross validation) which is obtained by the M-of-N expression rules extracted from KBANN

(Knowledge Based Artificial Neural Network). The obtained M-of-N rules are too much complicated and not easy to interprete. Since KBANN uses domain knowledge to configure the initial artificial neural network, it is worth mentioning that DT-GBI that does not use any domain knowledge induced a decision tree with comparable predictive accuracy. Comparing the decision trees in Figs. 9 and 10, the trees are not stable. Both gives a similar predictive accuracy but the patterns in the decision nodes are not the same. According to [7], there are many pieces of domain knowledge and the rule conditions are expressed by the various combinations of these pieces. Among these many pieces of knowledge, the pattern (a → a → a → a) in the second node in Fig. 9 and the one (a → a → t → t) in the root node in Fig. 10 match their domain knowledge, but the others do not match. We have assumed that two nucleotides that are apart more than 10 nodes are not directly correlated. Thus, the extracted patterns have no direct links longer than 9. It is interesting to note that the first node in Fig. 9 relates two pairs (g → a) that are 7 nodes apart as a discriminatory pattern. Indeed, All the sequence having this pattern are concluded to be non-promoter from the data. It is not clear at this stage whether the DT-GBI can extract the domain knowledge or not. The data size is too small to make any strong claims.

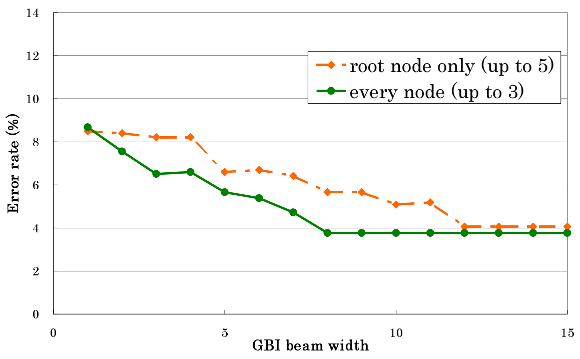

Fig. 11. Result of experiment (with pessimistic pruning)

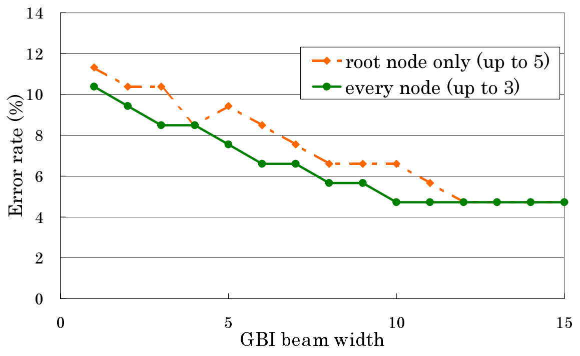

Fig. 12. Result of experiment (with prepruning)

5 Conclusion

This paper reports the current status of DT-GBI, which constructs a classifier (decision tree) for graph-structured data by GBI. Substructures useful for classification task, are constructed on the fly by applying repeated chunking in

GBI during the construction process of a decision tree. Newly introduced beam search is very effective in increasing the predictive accuracy. The performance of DT-GBI is evaluated through experiments on a classification problem of DNA promoter sequences from the UCI repository and the results show that DT-GBI is comparable to other method that uses domain knowledge in modeling the classifier.

Immediate future work includes to incorporate more sophisticated method for determining the number of cycles to call GBI at each node to improve prediction accuracy. Utilizing the rate of change of information gain by successive chunking is a possible way to automatically determine the number. Another important direction is to explore how the partial domain knowledge is effectively incorporated to constrain the search space. DT-GBI is currently being applied to much larger medical dataset.

Acknowledgment

This work was partially supported by the grant-in-aid for scientific research on priority area "Active Mining" (No. 13131101, No. 13131206) funded by the Japanese Ministry of Education, Culture, Sport, Science and Technology.

References

1. C. L. Blake, E. Keogh, and C.J. Merz. Uci repository of machine leaning database, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
3. T. Matsuda, T. Horiuchi, H. Motoda, and T. Washio. Extension of graph-based induction for general graph structured data. In *Knowledge Discovery and Data Mining: Current Issues and New Applications, Springer Verlag, LNAI 1805*, pages 420–431, 2000.
4. T. Matsuda, H. Motoda, T. Yoshida, and T. Washio. Knowledge discovery from structured data by beam-wise graph-based induction. In *Proc. of the 7th Pacific Rim International Conference on Artificial Intelligence, Springer Verlag, LNAI 2417*, pages 255–264, 2002.
5. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
6. J. R. Quinlan. *C4.5:Programs For Machine Learning*. Morgan Kaufmann Publishers, 1993.
7. G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.
8. G. Warodom, T. Matsuda, T. Yoshida, H. Motoda, and T. Washio. Classifier construction by graph-based induction for graph-structured data. In *Advances in Knowledge Discovery and Data Mining, Springer Verlag, LNAI 2637*, pages 52–62, 2003.
9. K. Yoshida and H. Motoda. Clip : Concept learning from inference pattern. *Journal of Artificial Intelligence*, 75(1):63–92, 1995.