Object-Oriented Database Mining: Use of Object Oriented Concepts for Improving Data Classification Technique

Kitsana Waiyamai, Chidchanok Songsiri, and Thanawin Rakthanmanon

Computer Engineering Department, Kasetsart University, Thailand
Fengknw, fengtwr@ku.ac.th, schidchanok@kdl.cpe.ku.ac.th

Abstract. Complex objects are organized into class/subclass hierarchy where each object attribute may be composed of other complex objects. Almost of the existing works on complex data classification start by generalizing objects in appropriate abstraction level before the classification process. Generalization prior to classification produces less accurate result than integrating generalization into the classification process. This paper proposes CO4.5, an approach for generating decision trees for complex objects. CO4.5 classifies complex objects directly through the use of inheritance and composition relationships stored in object-oriented databases. Experimental results, using large complex datasets, showed that CO4.5 yielded better accuracy compared to traditional data classification techniques.

1 Introduction

Data classification, one important task of data mining, is the process of finding the common properties among a set of objects in a database and classifies them into different classes [2]. One of well-accepted classification method is the induction of decision trees. However, traditional decision tree based-algorithms [7], [8], [9] have failed to classify complex objects. Complex objects are organized into class/subclass hierarchy where each object attribute may be composed of other complex objects. The classification is usually performed at low-level concepts resulting in very bushy decision tress with meaningless results.

Several methods for generating decision trees from complex objects have been proposed. Wang et al. [3], [11] have integrated ID3 decision tree classification methods with AOI (A*ttribute-Oriented Induction)* [4], [5] to predict object distribution over different classes. Han et al. [6] have developed a level-adjustment process to improve the classification accuracy in large databases. These methods require users to provide an appropriate concept hierarchy before the classification process. This is a difficult task even for experts to identify constraint such as balanced concept hierarchies. Further, generalization prior to classification using domain knowledge at the pre-processing stage produces less accurate result than integrating generalization/specialization into the classification process.

This paper proposes CO4.5, an approach for generating decision trees from complex objects. CO4.5 is based on inheritance relationships and composition relation-

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 303–309, 2004.

ships stored in the object-oriented database. In the classification process, the method utilizes an object's attributes in different levels of the hierarchy by distinguishing object attributes and non-object attributes. For object attributes, they are reused several times in the classification process by generalizing them to their appropriate level through the use of inheritance relationships. By integrating generalization in the classification process, the method produces more efficient and accurate result compared to those methods using generalization prior to the classification process. Experimental results, using large complex datasets, showed that CO4.5 yielded better accuracy compared to the traditional data classification techniques.

The rest of the paper is organized as follows. Sect. 2 contains description of CO4.5 algorithm that generates decision trees through the use of inheritance and composition relationships. Sect. 3 analyzes the performance of the proposed algorithm. Finally, Sect. 4 contains conclusions and future work.

2 Using Object-Oriented Concepts for Complex Data Classification

Complex objects in object-oriented databases are objects that have inheritance and composition relationships between them. Inheritance relationship describes how a class reuses features from one another. Composition relationship describes how an object is composed of other objects. In this section, we present an algorithm CO4.5 for constructing decision trees. Object attributes and non- object attributes can be determined using composition relationships. Compared to non-object attributes, object attributes have more semantics; they are reused several times in the classification process by generalizing them to their appropriate level through the use of inheritance relationships. Once decision trees, of a given level, are obtained, the overall decision tree is obtained by combining all the level-based decision trees through the inheritance relationship. Level-based decision tree has *target class* called *determinant class*, which is lower-level class in the hierarchy, while the overall decision tree has traditional target class at the leaf nodes.

2.1 CO4.5 Algorithm

```
(1)   algorithm CO4.5(Node n)
(2)     begin
(3)        FindDTree(root_n, generalize-all-att-to-
appropriate-level(Data_n), target_n)
(4)        if n ∉ Leaf then
(5)          for c ∈ child-of-n do begin
(6)            CO4.5(c)
(7)          end for
(8)       end if
(9)   end.
```

Input of CO4.5 algorithm is set of complex objects stored in object-oriented databases. Notice that inheritance relationships can be inferred implicitly from the object-

oriented database schema, or can be specified by the user. Output of CO4.5 is the overall aggregated decision trees. It is a greedy tree-growing algorithm, which constructs decision trees using top-down recursive divide-and-conquer strategy.

CO4.5 starts by building decision tree, using *FindDTree()* function (described later in this section). Object-attributes are generalized into appropriate level by calling *generalize-all-att-to-appropriate-level()* function. In the case that user specifies the concept level, CO4.5 then generalizes object-attributes to that specific level. If concept level is not specified, the attributes are automatically generalized into the most general level. The tree starts with single node (node *n*) containing the training samples in the root node ($root_n$). If the samples ($Data_n$: set of tuples with value and target class of node *n*) are all of the same class, then the node becomes a leaf node and is labeled with that determinant class. Otherwise, hierarchical and composition relationships are used to select "test-attribute".

A branch is created for each value of the test-attribute, and the samples are partitioned accordingly. The algorithm uses the same process recursively to form a decision tree for each partition. The recursive partitioning stops only when all samples at a given node belong to the same class, or when there are no remaining attributes on which the samples may be further partitioned. After level-based decision trees with determinant classes are obtained, CO4.5 utilizes inheritance relationships recursively to form decision tree of the descending node (*child-of-node-n*). The recursive partitioning stops when node n becomes leaf node of the hierarchy.

Execution Example of CO4.5

CO4.5 starts by constructing the decision of the top level of hierarchy, where determinant classes are the classes in lower level of the hierarchy. Fig. 1a shows the determinant classes of A which are class B and class C.

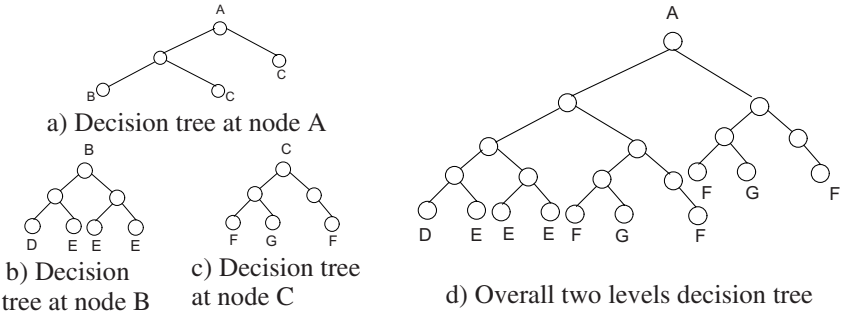

Fig. 1. Hierarchical decision tree

Then, decision trees of the lower level are constructed. Fig. 1b and 1c show, respectively, determinant classes of B which are D and E and determinant classes of C which are F and G. The overall decision tree in these two levels is shown in Fig. 1d. These steps are repeated until determinant classes of the bottom-level decision tree become user specified-target classes are obtained.

2.2 FindDTree Algorithm

Similar to C4.5, `FindDTree` is an algorithm for constructing a decision tree from data containing generalized complex attributes. All attributes (*att-of-Data)*, object attributes and non-object attributes, are used to calculate the information gain in order to find the test-attribute (*testAtt*) (line 5-10).

```
(1)   algorithm FindDTree(DTree d, ComplexData Data,
Target target)
(2)     begin
(3)       if all-prop>κ-in-one-class or no-more-att or
percent-object-subclass < ε then
(4)         d = mainDist(target)
(5)       end if
(6)       for each Att a ∈ att-of-Data do begin
(7)         g=Gain(Data,a)
(8)         if max<g
(9)           max=g
(10)          testAtt=a
(11)        end if
(12)       end for
(13)       for x ∈ value-of-testAtt do begin
(14)         NewData=newRound(Data_testAtt=x,testAtt)
(15)         d.value[i] = x
(16)         d.branch[i] = new DTree()
(17)       FindDTree(d.branch[i],NewData,target)
(18)       end for
(19)     end
```

Dataset is partitioned according to the value of the test-attribute using the *newRound()* function that works in the following manner:

a) If the test-attribute is a non-object attribute then use it for partitioning. It is not useful to reconsider it for finding information gain in the next round.
b) If the test-attribute is an object attribute, use it for partitioning. Then specialize the test-attribute one lower level of the concept hierarchy. If that object attribute cannot be specialized further, then do not use it to finding information gain in the next round.

For each dataset resulting of the *newRound()* function, *FindDTree()* is called recursively (line 14-17) until one of the following conditions is reached:

(1) If the portion of dataset in a given partition is less than the exception threshold ε, further partitioning of the dataset is halted
(2) Further partitioning of the dataset at a given node is terminated if the percentage of dataset belonging to any given class at that node exceeds the classification threshold κ [Kamber 1997] or less than exception threshold ε [Kamber 1997]. In this case the function *mainDist()* is called for determining target class with the main distribution
(3) No more attributes can be used for further classification (*no-more-att*)

3 Experimental Results

This section presents detailed evaluation of CO4.5 compared with AOI-based ID3 and the well-known C4.5 algorithms. The primary metric for evaluating classifier performance is classification accuracy. The comparison is made from different parameters such as number of data records, number of attributes, number of child/parents, number of target classes, and number of levels in the hierarchy.

In order to compare the three algorithms, two synthetic datasets are generated using randomizing functions with various parameters. The first one contains complete data with low noise, while the second one almost contains noisy data. Experiments have been realized on a Celeron CPU at clock rate of 1.2 GHz, 128 MB of main memory. In each experiment, results are the average percentage of accuracy using different data patterns. Number of training examples is fixed to 20,000, number of attributes: 11 (with 5 non-object attributes, 5 object attributes and 1 predictive attribute). For each attribute, number of distinct values is as follows: 3 for the non-object attributes, 3 for each level of the object attributes. Each object attribute has 5 levels. Predictive attribute has 2 target classes.

Fig. 2 shows how the total number of records affects the accuracy of the algorithms. Number of records is varying from 10000 to 80000. The experimental result demonstrates that all the three algorithms scale-up reasonably well with increasing number of records. Compared to AOI-based ID3 and C4.5, CO4.5, has the highest accuracy. Next experiment studies the accuracy of CO4.5, AOI-based ID3 and C4.5 as the number of attributes (object attributes for CO4.5 and AOI-based ID3 and non-object attributes for C4.5) increases. Fig. 3 shows that, with the use of semantics in object attributes, CO4.5 has better accuracy than AOI-based ID3 and C4.5.

Fig. 4 considers different levels of object attributes increasing from 3 to 6. Experimental result shows that accuracy of CO4.5 is increased with the number of object attribute levels. However, the accuracy continues to increase until a certain number of object attribute levels, and then it becomes steady. For AOI-based ID3, its accuracy depends on object attribute levels less than the threshold value, so the accuracy of AOI-based ID3 is rather steady. For C4.5, we notice that the number of object attribute levels does not affect its accuracy, which is rather steady as well. Fig. 5 shows the accuracy of the three algorithms as the number of child/parents increases. Through the semantics hidden in objects with large number of child/parents, CO4.5 provides better accuracy when the number of child/parents is increased.

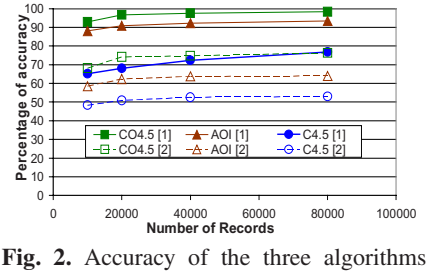

Fig. 2. Accuracy of the three algorithms with respect to number of training examples

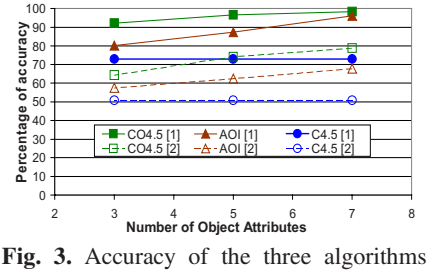

Fig. 3. Accuracy of the three algorithms with respect to number of object attributes

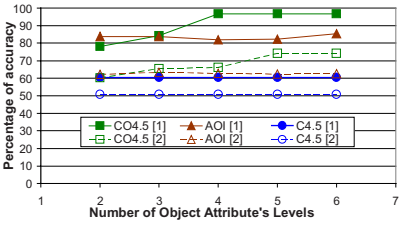

Fig. 4. Accuracy of the three algorithms with respect to number of object attribute levels

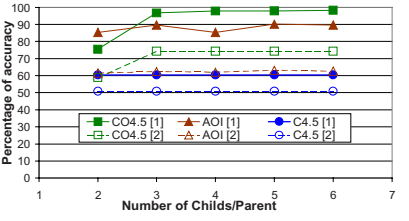

Fig. 5. Accuracy of the three algorithms with respect to number of child/parents

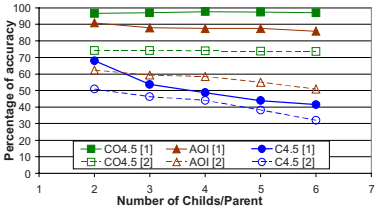

Fig. 6. Accuracy of the three algorithms with respect to number of target classes

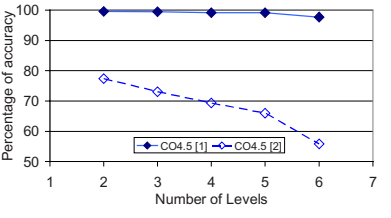

Fig. 7. Accuracy of CO4.5 with respect to number of levels in the hierarchy

Fig. 6 shows accuracy of the three algorithms with respect to number of target classes. Since CO4.5 uses the same object attribute several times, CO4.5 can classify data efficiently in any number of target classes. Unlike CO4.5, AOI-based ID3 and C4.5 have less accuracy when the number of target classes is increased. This can be explained by the fact that increasing target classes will decrease number of data per pattern, which leads to low accuracy. Fig. 7 shows accuracy of CO4.5 with respect to the number of levels in the hierarchy. Objective is to learn how the total number of hierarchical levels affects the accuracy of our algorithm. The number of hierarchical levels is varying from 3 to 6 with each node is divided to two nodes. This experiment shows that CO4.5 algorithm has less accuracy when the number of hierarchical levels is increased. This can be explained by the fact that there are errors in decision trees in each level. The more the number of levels, the more the overall error of the combining decision tree.

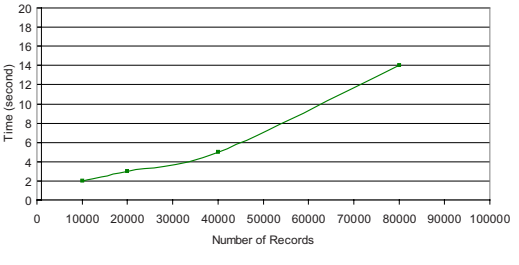

Fig. 8. Scale-Up Performance

In the last experiments, our objective is to learn how the total number of records affects the performance of CO4.5. In this experiment, only performance on the first dataset is shown since the two datasets produce the same execution time. Experimen-

tal result is shown in Fig. 8, where CO4.5 achieves nearly linear execution times. This explains the fact that the total classification time is dominated by the time used for specifying each object attribute.

4 Conclusion and Future Work

This paper proposes an approach for classification complex database. The main contribution is to use object's attributes in different levels of the hierarchy for the classification task. While almost of the existing works on complex data classification start by generalizing objects in appropriate abstraction level, the algorithm classifies complex objects directly through the use of inheritance relationships and composition relationship stored in object-oriented database. The proposed method solves the limitation of traditional algorithm in discovering complex object and in predicting multi-target class. The proposed algorithm can be further developed in the several ways such as increasing algorithm performance to support the complex hierarchical target class, apply the interesting object oriented database with this algorithm and use this new concept to develop on the current efficient data classification algorithm such as SLIQ, SPRINT.

References

1. Chen, M., Han, J., Yu, S. Data Mining: An Overview from Database Perspective. In IEEE Transactions on Knowledge and Data Engineering (1996)
2. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers (2001)
3. Han, J., Nishio, S., Kawano, H., Wang, W.: Generalization-based data mining in object-oriented databases using an object-cube model. In Data and Knowledge Engineering. 25 (1998) 55-97
4. Han, J., Fu, Y.: Exploration of the power of attribute-oriented induction in data mining. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 399-421
5. Han, J., Cai, Y., Cercone, N.: Data driven discovery of quantitative rules in relational databases. In IEEE Trans.Knowledge and Data Engineering. 5. (1993) 29–40
6. Kamber M., Winstone, L., Gong, W., Cheng, S., Han, J.: Generalization and decision tree induction: Efficient classification in data mining. Int. Workshop on Research Issues on Data Engineering , Birmingham, England (1997) 111-120
7. Mehta, M. Agrawal, R. Rissanen, J.: SLIQ: A Fast Scalable Classifier for Data Mining, Int Extending Database Technology. Avignon, France. (1996)
8. Quinlan, J. R.: Induction of decision trees. Machine Learning. 1. (1986) 81-106
9. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufman (1993)
10. Songsiri, C., Waiyamai, K, Rakthanmanonn, T.: An Object-oriented Data Classification Technique (in Thai)". In Proc. The National Computer Science and Engineering Conference (2002)
11. Wang, W.: Predictive Modeling Based on Classification and Pattern Matching Methods. M.Sc. thesis. Computing Science, Simon Fraser University. (1999)