

Proposed architectural model for optimal transformation of decision table and decision tree into knowledge base

M. Shuaib Qureshi, M. Imran Saeed and Syed M. Saqlain

Dept. of Computer Science, Faculty of Basic & Applied Sciences, International Islamic Univ., Islamabad, Pakistan
 qureshi.shuaib @gmail.com

Abstract

Knowledge is one of the most precious resources of an organization. Every organization wishes to preserve and fully utilize its knowledge. Within organization knowledge is present in various forms, may be in the minds of workers or in documented form. In documented form the knowledge has various representation schemes such as frames, scripts, lists, decision trees and decision tables etc. We have proposed a transformation method to optimally transform knowledge present in decision trees and decision tables into knowledge base. According to our proposal, the knowledge present in these two representation schemes should first be converted into corresponding set of human interpretable rules by using some existing transformation algorithms. The decision tree should be converted directly into set of human interpretable rules but for decision table transformation, two ways are adopted; either it should be converted first into decision tree and then to set of rules or, should be converted directly into set of rules. Once the set of rules is obtained then it will be optimized by using some existing optimization algorithms and unnecessary conditions will be omitted. After rules optimization process, comparison with the existing rules in the knowledge base is carried out. If these optimized rules are not found in the knowledge base, it should be added. If some rules need updation, then these should be added after updation. During comparison those rules which already exist in the knowledge base should be omitted.

Keywords: Decision table; decision tree; knowledge base; transformation; optimization.

Introduction

Knowledge is one the most precious resources of an organization. Every organization wishes to preserve and fully utilize its knowledge. Once the knowledge is acquired, it must be organized in an application's knowledge base for later use. The collection of knowledge related to a problem is organized and is called Knowledge base (KB). Knowledge can be organized in different configurations to facilitate the inferencing. There are different ways for KB organization (Efraim & Jay, 2003). Many different knowledge representations, such as semantic nets, frames etc, have been proposed for use over the years. Within an organization knowledge is present in various form, may be in the minds of workers or in documented form. In documented form the knowledge has various representation schemes such as frames, scripts, lists, decision tree and decision tables etc. Every knowledge representation scheme has certain inherent strengths and weaknesses. A decision table (DT) is composed of rows and columns. In majority of cases, the DT method allows common corroboration and confirmation checking such as inconsistencies, redundancy, incompleteness, contradiction etc in the problem specification (Nguyen *et al.*, 1987).

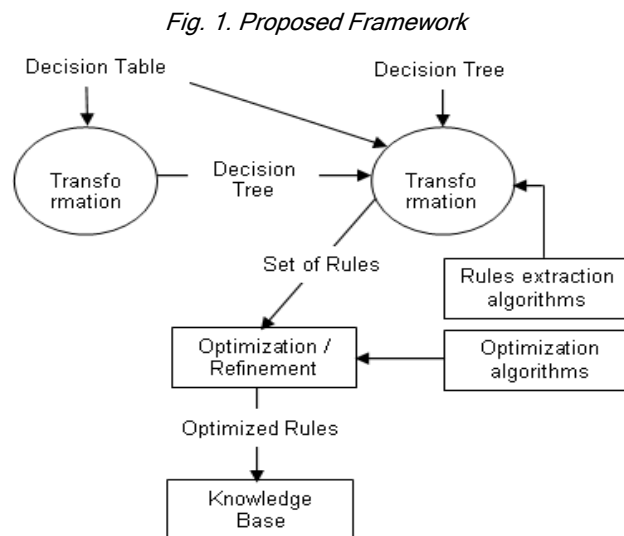
A decision tree (DTree) is a tool using model or graph for taking different decisions

(Wikipedia, 2009). Goals are represented by the nodes of the DTree while as decisions are represented by its links. Conversion of DTree into other formats can be done easily. The DT, DTree and Rues in knowledge gaining or completion stage, however, do not inevitably remain alike. So there is no need to use one and only one representation throughout the entire knowledge development life cycle. For occasion, it is possible to start from DT in the knowledge acquirement process and achieve rule-based stipulation in the accomplishment stage (Vanthienen & Wets, 1994).

Knowledge usually transforms from one layout to another suitable layout for getting quicker response. One knowledge depiction format is appropriate for one type of computation while other for another type. Therefore there is a need to map knowledge from one depiction to another. This mapping gives earlier response and decrease in computational amount. The DT might be transformed into DTree and DTree into set of human interpretable rules. Any client can recognize and alter a rule set without any difficulty than he/she can recognize and alter a DT or DTree.

Problem Description

Classification algorithm is a kind of important technology in data mining. At present there are a variety of



classification modules, some of which are, Value Reduction, DTree, DT, Neural Network, Statistical Model, Bayesian Classifier etc.

DTree, DT and Rules are commonly used methods (Yan Li *et al.*, 2009). The major issue when produce complete knowledge based application is, "how to apply the decision logic" (Vanthienen & Dries, 1995). DT and DTrees are optimized before its conversion into Rules, i.e. in earlier stages. Construction of fully optimized DTs and DTrees are NP-hard problems (Hyafil & Rivest, 1976; Garey & Johnson, 1979; Louis *et al.*, 1989; Murphy & Mccraw, 1991; Naumov, 1991; Vanthienen & Wets, 1994; Decai & Lingli, 2006; Jun & Shu Li, 2007; Yan Li *et al.*, 2009). Hence the need arises "how to optimally transform DT and DTree into KB; and how to store the optimized knowledge in the KB?. So what strategy should be adopted?"

Proposed Framework (Fig. 1)

Transformation of DTree into rules can easily be done by using some top-down/left-to-right approach but for DT transformation, there exist two possibilities. First, it is possible to convert the DT to a DTree and then transform DTree into set of rules. Second, the DT is transformed directly into a set of rules. Once the set of rules is formed, the next step is the refinement of newly created rules to optimize the knowledge representation. In this refinement stage, unnecessary and unwanted conditions are removed from each rule using some existing optimization algorithm.

During this refinement stage, rules are optimized; as a result efficiency is achieved. After rules refinement/

Table 1. Results of the proposed framework

Technique	Diabetes data set		Sonar data set		Zoo date set	
	No. of rules	Accuracy (%)	No. of rules	Accuracy (%)	No. of rules	Accuracy (%)
Decision table	32	71.224	15	69.2308	15	86.1386
Decision tree	39 (size)	73.8281	35 (size)	71.1538	17 (size)	92.0792
Rules	13	75.2604	8	80.2885	8	92.0792
Optimized rules	8	80.5194	7	85.7142	7	100

Fig.2. Rules covering algorithm (Jun Zhou & Shu-You Li, 2007)

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.

Input:

- D , a data set class-labeled tuples;
- Att_vals , the set of all attributes and their possible values.

Output: A set of IF-THEN rules.

Method:

- (1) $Rule_set = \{\}$; // initial set of rules learned is empty
- (2) for each class c do
- (3) repeat
- (4) $Rule = Learn_One_Rule(D, Att_vals, c)$;
- (5) remove tuples covered by $Rule$ from D ;
- (6) until terminating condition;
- (7) $Rule_set = Rule_set + Rule$; // add new rule to rule set
- (8) endfor
- (9) return $Rule_Set$;

Fig. 3. Decision Tree Algorithm (Jun Zhou & Shu-You Li, 2007)

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition D .

Input:

- Data partition, D , which is a set of training tuples and their associated class labels;
- $attribute_list$, the set of candidate attributes;
- $Attribute_selection_method$, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a $splitting_attribute$ and, possibly, either a $split_point$ or $splitting_subset$.

Output: A decision tree.

Method:

- (1) create a node N ;
- (2) if tuples in D are all of the same class, C then
- (3) return N as a leaf node labeled with the class C ;
- (4) if $attribute_list$ is empty then
- (5) return N as a leaf node labeled with the majority class in D ; // majority voting
- (6) apply $Attribute_selection_method(D, attribute_list)$ to find the "best" $splitting_criterion$;
- (7) label node N with $splitting_criterion$;
- (8) if $splitting_attribute$ is discrete-valued and multiway splits allowed then // not restricted to binary trees
- (9) $attribute_list = attribute_list - splitting_attribute$;
- (10) for each outcome j of $splitting_criterion$
// partition the tuples and grow subtrees for each partition
- (11) let D_j be the set of data tuples in D satisfying outcome j ;
- (12) if D_j is empty then
- (13) attach a leaf labeled with the majority class in D to node N ;
- (14) else attach the node returned by $Generate_decision_tree(D_j, attribute_list)$ to node N ;
- (15) endfor
- (15) return N ;

optimization, we compare the refined rules with the already existing rules in the KB. If the refined rule already exists then there is no need to update the KB. It might be a case when some existing rules are updated or modified. If the KB has deficiency of these refined rules then the rules should be added. During comparison those rules should be omitted which already exist in the KB.

For the whole process some of the existing algorithms are used (Fig. 2, 3, 4).

Results and discussions

For investigating the research problem in depth, different steps

were followed during this study. The first step involved data collection. The second step centered on DTree and DT frameworks. Existing frameworks are exposed and got knowledge about their potential and flexibility. In the third step, data analyses were carried out, by which the ideas, concepts and materials related to the research problem have been recognized. The data was examined to give a synopsis of knowledge transformation frameworks, identifying components and interconnections. Finally, a knowledge transformation framework was proposed.

A case study has been conducted for justifying the efficiency of the newly proposed framework. In this case study three examples of different data sets were taken.

The different data sets were Diabetes data set having 768 instances and 9 attributes, Sonar data set consisting of 208 instances and 61 attributes and Zoo data set having 101 instances

and 18 attributes. 10-fold cross-validation test mode was used. These data sets were passed through the proposed framework and the results achieved are presented in Table 1 and Fig. 5, 6 and 7.

Conclusion and Future Work

In this paper we have proposed an architectural framework for optimal transformation of knowledge

present in DTrees and DTs into KB. Some of the advantages of the proposed framework are: reduction in computational amount, fast response, redundant knowledge removal, optimization, accuracy, reduction in time and space complexities.

The combination of various classification techniques with the proposed architecture for better efficiency is a choice for future work.

References

1. Decai H and Lingli W (2006) Analysis on the drawbacks of the commonly used measures of the significance of attributes in decision table and a new measure. *Proc. 1st IEEE Intl. Multi-Sym. on Computer & Computational Sciences (MSCCS)*.
2. Efraim T and Jay E (2003) Decision support systems and intelligent systems. 6th Ed. New Dehli: Prentice -Hall of India Pvt. Ltd.
3. Hyafil L and Rivest RL (1976) Constructing optimal binary decision trees is NP-Complete. *Infor. Process. Lett.* 51,15-17.
4. Jun Zou and Shu-You Li (2007) Design of the knowledge system based on basic rules. *Proc. IEEE 6th Intl. Conf. on Machine Learning & Cybernetics*, Hong Kong, August 19-22. pp: 3704-3707.
5. Louis AC, Yuping Q and Warren K (1989) Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annal. Oper. Res.* 21(1), 1-30.
6. Garey MR and Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. San Francisco, CA.
7. Murphy OJ and Mccraw RL (1991) Designing storage efficient decision trees. *IEEE Transact. Computers.* 40(3), 315-319.
8. Naumov GE (1991) NP-completeness of problems of construction of optimal decision trees. *Soviet Physics, Doklady*, 36(4), 270-271.
9. Nguyen TP, Laffey W and Pecora T (1987) Knowledge base

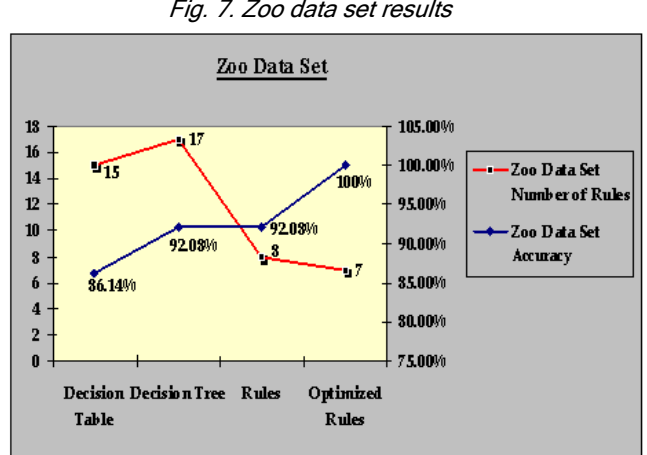
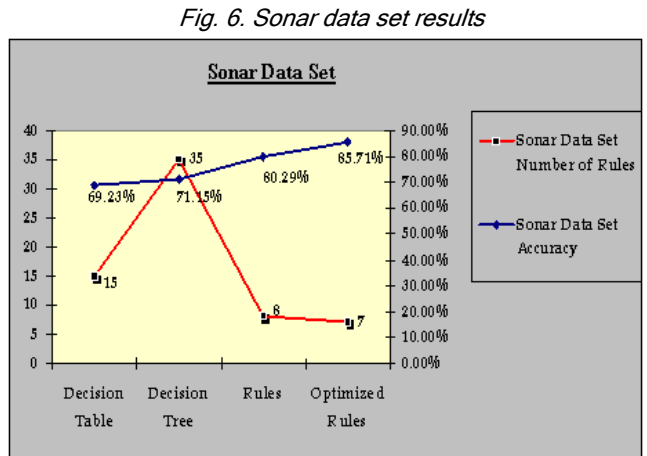
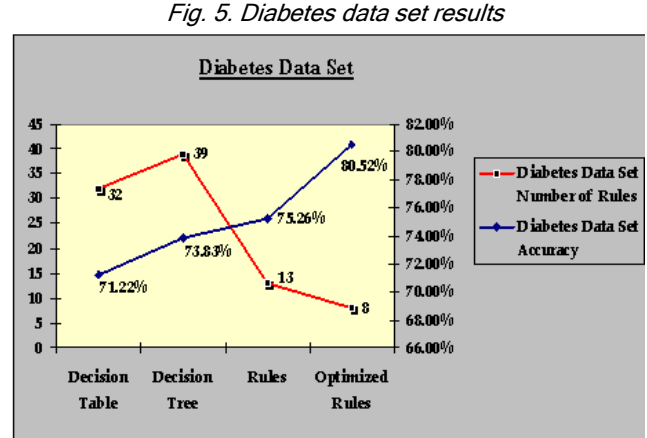
Fig. 4. Ant Miner Algorithm (Lie et al., 2004)

Algorithm : High level pseudo-code of Ant-Miner.

```

begin Ant-Miner
  training_set ← all training examples;
  rule_list ← ∅;
  while |training_set| > max_uncovered_training_examples do
    τ ← initializes pheromones;
    rule_best ← ∅;
    i ← 1;
    repeat
      rule_i ← CreateRule();
      ComputeConsequent(rule_i);
      Prune(rule_i);
      UpdatePheromones(τ, rule_i);
      if Q(rule_i) > Q(rule_best) then
        rule_best ← rule_i;
      end
      i ← i + 1;
    until i ≥ max_number_rules OR convergence ;
    rule_list ← rule_list ∪ rule_best;
    training_set ← training_set \ CorrectlyCoveredExamples(rule_best);
  end
end

```



- verification. *AI Magazine.* 8(2), 69-75.
10. Vanthienen J and Dries E (1995) Restructuring and simplifying rule bases. *IEEE*, 10(3), 484-485.
11. Vanthienen J and Wets G (1994) Restructuring and optimizing knowledge representations. In: *Proc. IEEE 6th Intl. Conf. on Tools with Artificial Intelligence*, Nov. 6-9. pp: 768-771.
12. Wikipedia (2009) Decision tree. Retrieved Aug. 05, 2009, from http://en.wikipedia.org/wiki/Decision_tree.
13. Yan Li, Fa-Chao Li, Chen-Xia Jin and Tao Feng, (2009) A rule extraction algorithm based on attribute importance. *Proc. IEEE 8th Intl. Conf. on Machine Learning & Cybernetics, Boding, July 12-15.* pp: 127-132.