Inferring Reduced Ordered Decision Graphs of Minimal Description Length

Arlindo L. Oliveira*
Alberto Sangiovanni-Vincentelli
Dept. of EECS, UC Berkeley, Berkeley CA 94720

May 17, 1994

1 Introduction

The induction of decision trees from labeled training set data [BFOS84, BFOS84, Qui86, QR89] has been a successful approach for the induction of classification rules. However, although decision trees can, in principle, represent any concept, they are not concise representations for some concepts of interest. In particular, the quality of the generalization performed by a decision tree induced from data suffers because of two well known problems: the replication of subtrees required to represent some concepts and the rapid fragmentation of the training set data when high arity attributes are tested at a node. (See, for instance, [Oli93] for a more detailed description of these limitations).

Decision graphs have been proposed as one way to alleviate this problems, but the algorithms proposed so far for the construction of these graphs suffer from serious limitations. [MM91] propose to identify related subtrees in a decision tree obtained using standard methods, but reported limited success. [Oli93] proposed a greedy algorithm that performs either a join or a split, according to which one reduces more the description length. They reported some good results in relatively simple problems but our experiments using a similar approach failed in more complex problems.

In this paper, we propose to use ordered decision graphs (ODGs) as the underlying description. In ODGs, the variables can only be tested in accordance with a previously specified order. By using ODGs, we can effectively avoid some of the problems encountered by other researchers when trying to identify common subtrees. We describe a simple but relatively effective algorithm that derives an ordered decision graph with minimal description length from a decision tree built using standard techniques. The approach proposed here is very different from the one proposed in [Koh94] that also used RODGs. His approach as described in the reference, suffers from serious limitations when solving problems with medium to large numbers of attributes although it performs well for small problems.

The complexity of the generated decision graph and the quality of the generalization performed are evaluated using problems from the machine learning literature that are known to be hard to solve using decision trees.

Section 2 presents ODGs and briefly describes some of the interesting properties they exhibit. Section 3 describes the coding methods used and the cost function that is minimized. Section 4 describes how a an ODG with a small description length can be obtained from a decision tree. Some of the results obtained using this algorithm are described in section 5.

2 Decision trees and ordered decision graphs

We address the problem of inferring a classification rule given a set of objects, the training set. Each object is described in terms of a collection of discretely-valued attributes $(x_1 \in X_1, ..., x_n \in X_n)$ and the class it belongs to, $c \in C$. The objective is to infer a classification rule (a function $F : X_1 \times X_2 ... \times X_n \rightarrow C$) that can be used to classify unseen objects with the highest possible accuracy.

A decision graph (DG) is a rooted, directed, acyclic graph where each non-terminal node is labeled with the index of the attribute being tested at that node. Each terminal node is labeled with one label in C. From each non-terminal

*To whom all correspondence should be sent: 207-194 Cory Hall, Dept. of EECS, UC Berkeley, Berkeley CA 94720

node an arch labeled with each possible value of the attribute being tested at that node leads to another node in the graph.

As decision trees, decision graphs can be used to classify any object by tracing a path from the root to a terminal node that follows the edges labeled with the values present in that object.

A decision graph is called ordered if there is an ordering of the variables such that, for all possible paths in the graph, the variables are always tested in that order (possibly skipping some of them). A decision graph is called reduced if only one terminal node exists for each class, no two nodes exist that branch exactly in the same way and is never the case that all outgoing edges of a given node terminate in the same node. The level of a node is defined as the position of the variable tested at that node in the variable ordering used.

Given an ordering, reduced ordered decision graphs are a canonical representation [Bry86], for functions in that domain. This means that given a function $F : X_1 \times X_2 ... \times X_n \rightarrow C$ and an ordering of the variables, there is one and only one representation for the function F.

Packages that manipulate reduced ordered decision graphs are widely available and have become the most commonly used tool for function manipulation in the logic synthesis community. Some of these packages are restricted to Boolean functions [BRB89] (each non-terminal graph has two and only two outgoing edges) while others [KB90] can accept multi-valued attributes.

All these packages provide at least the same basic functionality: the ability to combine functions using basic Boolean and arithmetic operations and the ability to test for containment or equivalence of two functions. Remarkably, and due to the canonicity property described above, the equivalence test (and, therefore, the tautology test) can be performed in constant time.

3 Minimizing message length and encoding of RODGs

Let t be the description length of an RODG and d be the length of the message required to describe the exceptions to this RODG in a given training set. According to the MDL principle, one should choose the RODG that minimizes the total message length, $t + d$. However, as pointed out in [QR89], different linear combinations of t and d are also consistent with a Bayesian interpretation of the MDLP and may be chosen according to different beliefs about the data. In order to simplify the problem, we will consider, in this paper, a particular case of the approach, where d needs to be 0 and only the value of t is minimized. This is to say that we will always build decision trees and RODGs that will classify correctly 100% of the examples present in the training set.

The approach can be easily generalized to minimize a different linear combination of t and d, with only minor modifications required in the algorithms described in section 4.

Since we will be encoding RODGs using an approach inspired in the one proposed in [QR89] for decision trees, we briefly review the later. [QR89] define the encoding of a tree recursively using the following rules:

1. A terminal node is encoded as a 0 followed by an encoding of the default class.
2. A non-terminal node is encoded starting with a 1, followed by the encoding of the variable tested at that node, followed by the encodings of the subtrees.

When encoding an RODG, we use the fact that some nodes will be visited more than once. Instead of re-describing them, we can simply make a reference to the description of that node.

1. A terminal node is encoded as a 0 followed by an encoding of the default class.
2. A non-terminal node that was never visited is encoded starting with 10 followed by an encoding of the variable tested at that node, followed by the encodings of the subgraphs.
3. A non-terminal node that was visited before is encoded starting with 11 followed by a reference to the (already described) function implemented by that node.

Following [QR89], we ignore the issues related with the use of non-integral numbers of bits and we make the description less redundant by noting that when one is deeper in the decision graph not all variables can be usefully tested (only the ones that were node tested previously). Also, when a reference to an already described function is given, one needs only use $\log_2(n)$ bits, where n is the number of non-terminal nodes described up to that point.

4 Obtaining a small decision graph from a decision tree

For clarity, and without loss of generality, we will henceforth assume that the attributes are Boolean valued and there are only two classes. This considerably simplifies the description of the algorithm.

We assume, for now, that a given variable ordering was defined. Section 4.3 details how this ordering is obtained.

4.1 Obtaining a RODG from a decision tree

Let T be a decision tree created using one of the approaches proposed in the literature (for example, [Qui86]). Let v_i denote the attribute tested in node i, t_i (the *then* node) denote the index of the node pointed by the arc leaving node i when attribute v_i is 1 and e_i (the *else* node) denote the index of the node pointed by the arc leaving node i when attribute v_i is 0. Finally, let node 0 be the root of the decision tree.

Each node in the decision tree implements a well defined Boolean function, that has a unique representation in RODG form. Let f_i be the function implemented by node i. By definition, $f_i = \overline{v_i} \wedge f_{e_i} \ \vee \ v_i \wedge f_{t_i}$.

The RODG is built recursively by simply computing $f_0 = \overline{v_0} \wedge f_{e_0} \ \vee \ v_0 \wedge f_{t_0}$.

Since this RODG implements exactly the same function as the original decision tree, there is no gain in using it instead of the original tree to perform classification. However, the description length of the RODG obtained in this fashion can usually be reduced.

4.2 Reducing an RODG

A node in an RODG is redundant (wrt to a given training set) if it has the following property: all incoming edges can be moved to some other node while still preserving an exact classification for all examples in the training set[1].

Given this definition, the following algorithm is used to reduce the description complexity of an RODG:

1. Select the lowest level redundant node. If no redundant nodes exist, stop.
2. Redirect all edges that are pointing to this node.
3. If nodes with no incoming edges exist, remove them and goto 3. If no such nodes exist, goto 1.

It is very important to note that the algorithm does much more that identifying subtrees that can replace other subtrees. In the process of removing redundant nodes, new and possibly very different functions are created. This enables the approach to work in problems that simply cannot be solved by decision trees, no matter how large the training sets, like, for instance, the n-bit exclusive or problem.

4.3 Selecting the best ordering

The selection of a good ordering for the variables is of critical importance if the target is to obtain a compact RODG. Regrettably, selecting the optimal ordering for a given function is hard and cannot be solved exactly in many cases. However, many heuristic algorithms have been proposed for this problem [FS90, Rud93].

In our setting, the problem is even more complex because one wants to select the ordering that minimizes the final RODG and this ordering may not be the same as the one that minimizes the original RODG obtained from the decision tree.

Our implementation uses the *sift* algorithm for dynamic RODG ordering [Rud93]. A complete description of this algorithm is impossible here due to space limitations. However, the basic idea is simple. It is known that, given a RODG, swapping the order of two adjacent variables can be done very efficiently because only the nodes in those two levels are affected. The *sift* algorithm selects the best position in the ordering for a given variable by moving that variable up and down (using the inexpensive swap operation) and recording the smaller size observed. This procedure is applied once to all variables and can be, optionally, iterated to convergence.

This algorithm is extremely efficient since it was designed to be used to be applied to RODGs with several millions of nodes. Therefore, we can afford to apply it after the removal of each redundant node.

[1] An incoming edge can only be redirected if it doesn't violate the variable ordering restrictions.

5 Experiments

To evaluate the efficacy of the approach, we selected a set of problems that are know to be hard to learn using a decision tree representation. The *monks2* problem was proposed in [TBB+91]. *Dnf2, dnf3, mux11 and par4* are described in [PH90]. *Tictactoe* is from [MA91] and *legal*, en encoding of legal and illegal positions in chess endings is described in more detail in [Oli94]. For all problems, multi-valued attributes were encoded using a binary encoding.

For each problem, we performed 10 runs with randomly generated training sets. The resulting decision tree and RODG were tested using one independent test set (that included, possibly, some or all of the examples present in the training sets). Statistics about the problems and the sizes of the training and test sets used are listed in table 1.

Problem	# inputs	training	testing
Monks2	10	216	432
Dnf2	40	2187	2000
Dnf3	32	1652	2000
Par4	16	640	2000
Mux11	32	802	2000
Legal	18	905	1813
TicTacToe	18	479	958

Table 1: Problem statistics

The graphs in figure 1 show the evolution of the classification accuracy as the description length of the RODG is reduced.

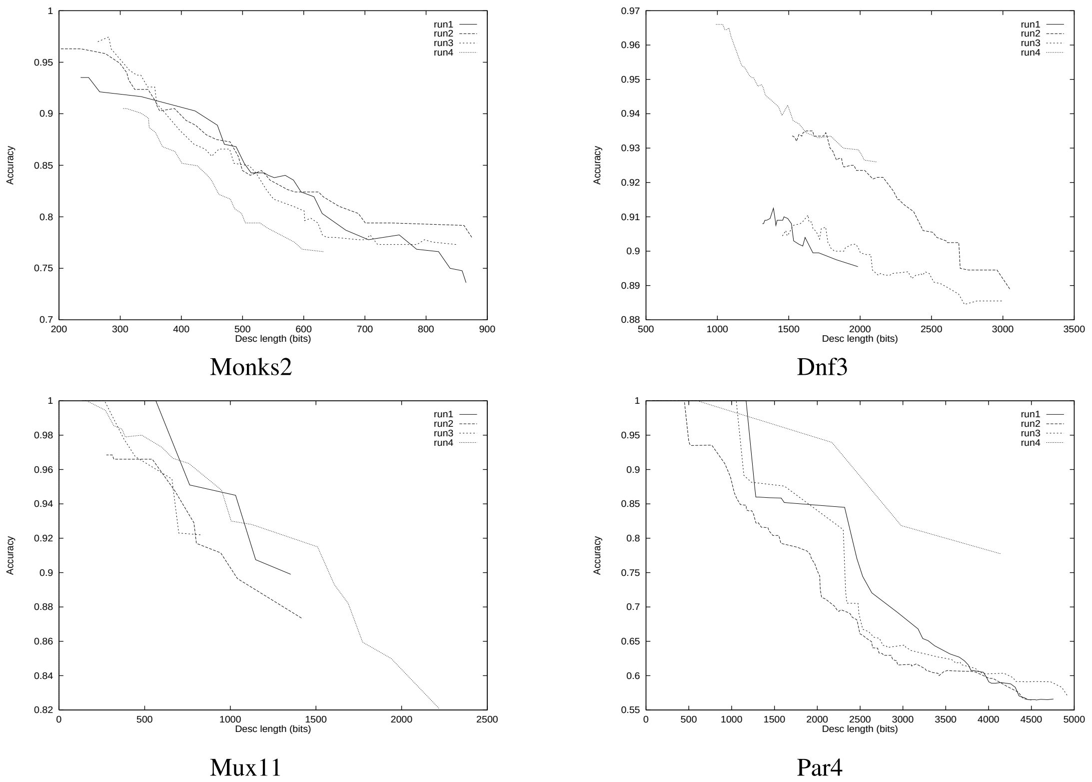

Figure 1: Accuracy versus description length

For the problems *monks2*, *mux11*, *dnf3* and *par4*, we plotted the accuracy in the test set versus de description length of the successives RODGs obtained by the algorithm described in section 4. For each line, the point furthest to the right is obtained with the RODG derived directly from the decision tree and the accuracy at that point is the same as the one obtained with the decision tree. The leftmost points of each line correspond to the last RODG obtained. The figure shows that, for these problems, a significant increase in classification accuracy is always obtained by reducing the size of the RODG.

Table 2 lists the results obtained for all the problems. The second and third columns list the description length of the original decision tree and the first RODG. The fourth column lists the description length of the final RODG obtained. These values are computed as described in section 3. The following two columns list the generalization accuracy of the first and last RODG. Finally, the last column lists the improvement in generalization accuracy obtained for each example, together with the standard deviations.

Problem	Description length (bits)			Generalization accuracy		
	Dec Tree	Init RODG	Final RODG	DT	RODG	Improv $\pm\sigma$
Monks2	567.3	829.6	272.0	75.48	93.91	18.43 $\pm$ 5.61
Dnf2	1552.3	3777.8	2178.8	89.13	90.39	1.26 $\pm$ 0.57
Dnf3	1088.2	2318.0	1327.2	89.89	92.12	2.22 $\pm$ 1.34
Par4	1580.0	4981.8	64.64	59.09	100.0	40.91 $\pm$ 4.19
Mux11	788.6	968.20	149.46	90.92	99.69	8.76 $\pm$ 4.65
Legal	1901.2	7568.5	2377.70	81.88	82.51	0.63 $\pm$ 1.56
TicTacToe	340.7	483.2	379.2	96.18	96.66	0.48 $\pm$ 0.90

Table 2: Description lengths and accuracy

There are several interesting points that should be noted. First, the description of the RODG that results from encoding the decision tree usually requires more bits than the original decision tree. This is more a consequence of the ordering requirement than the slightly less efficient encoding of each node when no reconvergence exists in the graph.

The reduction performed by the algorithm can vary widely, from almost no gain to several orders of magnitude. Not surprisingly, the larger gains in classification accuracy are obtained when more compression is achieved in the description of the RODG.

In all examples the accuracy of the final RODG was, in the average, better than the accuracy of the decision tree. However, the improvement observed for the last two problems is not statistically significant.

Finally, figure 2 shows the first and last RODG obtained in the first run of the *mux11* problem. The interesting point to note, apart from the reduction in size, is the fact that the ordering algorithm also obtained an almost optimal order of the variables (just one node larger than the absolute minimum size required to represent this problem). Decision trees have troubles with problems like this (a 3-bit multiplexer with 21 irrelevant variables) because, as pointed out in [Qui88], the greedy strategy for node splitting tends to favor testing the data attributes first instead of the selection bits.

6 Conclusions and future work

The results presented in section 5 show that the accuracy of the generalization performed by a decision tree algorithm can be improved, in some cases significantly, by using the algorithms described in section 4.

The algorithm used to reduce the RODG found is too simple and, in many cases, it got stuck very early in the process. It also suffers from other severe limitations (for example, it never creates a new node at a given level). Current research is concentrated on the improvement of this algorithm. In particular, it is possible to create redundant nodes when none exists by moving some edges of the RODG to other nodes (while still keeping perfect performance in the training set data). The best way to implement this procedure, however, is still under investigation, because moving an edge between nodes can have complicated implications, both above and below the affected nodes levels. We were able to improve significantly the results by performing random shuffles of the edges (between compatible nodes) but this results are not reported here as that doesn't seem, at the moment, the more efficient way to proceed.

Finally, it must be stated that many other good alternatives have been proposed that aim at increasing the quality of the generalization performed by a decision tree. Both constructive induction techniques and different forms of post-processing of the decision tree have been proposed. Specially worth mentioning are the *fringe*-like algorithms

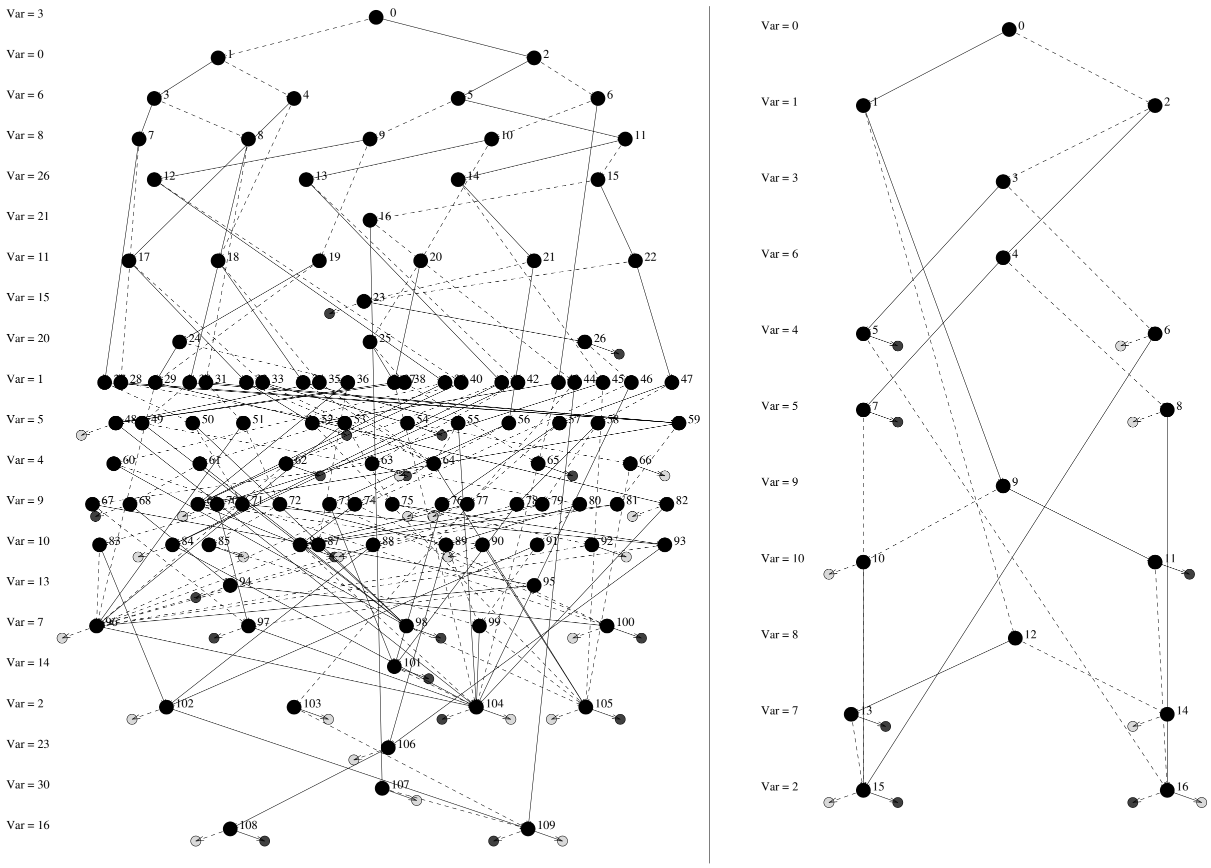

Figure 2: First and last RODG for the first mux11 run

[PH90, YRB91, OSV93] that incrementally build new composite attributes by looking at the tests performed at the fringes of decision trees. Our own experiments with these algorithms have shown that they are remarkably efficient at learning concepts that accept a compact DNF representation. We conjecture that the performance of the approach described in this paper cannot beat this type of algorithms in these problems. Furthermore, it is know that there are functions that have compact DNF representations but require exponentially large RODGs, no matter what ordering is selected [Dev93]. *Fringe*-like algorithms, however, have a much harder time when the problems at hand don't have a compact DNF representation. In that case, we believe an approach like the one outlined here can be one of the best alternatives available.

It must be noted, however, that using ODGs instead of unordered decision graphs comes at a price, since there are functions that cannot be represented in compact form in RODG form but accept compact representations if the fixed variable ordering requirement is not enforced. We believe this limitation if a small price to pay for the superior ease of manipulation that is obtained when one uses RODGs.

References

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[BRB89] K. Brace, R. Rudell, and R. Bryant. Efficient implementation of a BDD package. In *Design Automation Conference*, June 1989.

[Bry86] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 1986.

[Dev93] S. Devadas. Comparing two-level and ordered binary decision diagrams representations of logic functions. *IEEE Transaction on Computer-Aided Design*, 12(5), 1993.

[FS90] Steven J. Friedman and Kenneth J. Supowit. Finding the optimal variable ordering for binary decision diagrams. *IEEE Trans. Comput.*, 39(5):710–713, May 1990.

[KB90] T. Kam and R.K. Brayton. Multi-valued decision diagrams. *Tech. Report No. UCB/ERL M90/125*, December 1990.

[Koh94] Ron Kohavi. Bottom-up induction of oblivious read-once decision graphs. In *European Conference in Machine Learning*, 1994.

[MA91] P. M. Murphy and D. W. Aha. *Repository of Machine Learning Databases - Machine readable data repository*. University of California, Irvine, 1991.

[MM91] J. J. Mahoney and R. J. Mooney. Initializing ID5R with a domain theory: some negative results. Technical Report 91-154, CS Dept., University of Texas at Austin, Austin, TX, 1991.

[Oli93] J. J. Oliver. Decision graphs - an extension of decision trees. Technical Report 92/173, Monash University, Clayton, Victoria 3168, Australia, 1993.

[Oli94] Arlindo L. Oliveira. *Inductive Learning by Selection of Minimal Representations*. PhD thesis, UC Berkeley, 1994. In preparation.

[OSV93] Arlindo L. Oliveira and A. Sangiovanni-Vincentelli. Learning complex boolean functions : Algorithms and applications. In *Advances in Neural Information Processing Systems 6*, Denver, CO, 1993. Morgan Kaufmann.

[PH90] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100, 1990.

[QR89] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the Minimum Description Length Principle. *Inform. Comput.*, 80(3):227–248, March 1989. (An early version appeared as MIT LCS Technical report MIT/LCS/TM-339 (September 1987).).

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui88] J. R. Quinlan. An empirical comparison of genetic and decision-tree classifiers. In *Fifth International Conference on Machine Learning*, pages 135–141, 1988.

[Rud93] Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *ICCAD*, pages 42–47. IEEE Computer Society Press, 1993.

[TBB+91] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. de Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufaman, S. Keller, I. Kononenko, J. Kreusiger, R. S. Michalski, T. Mitchell, P. Pachowitz, Y. Reich, H. Vafaic, W. Van de Weldel, W. WEnzel, J. Wnek, and J. Zhang. The monk's problems: a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[YRB91] D. S. Yang, L. Rendell, and G. Blix. Fringe-like feature construction: A comparative study and a unifying scheme. In *Proceedings of the Eight International Conference in Machine Learning*, pages 223–227, San Mateo, 1991. Morgan Kaufmann.