UNDERSTANDING WHAT MACHINE LEARNING PRODUCES

Part I: Representations and their comprehensibility

Sally Jo Cunningham, Matt Humphrey and Ian H. Witten
Department of Computer Science, University of Waikato,
Hamilton, New Zealand.
{matth, sallyjo, ihw}@cs.waikato.ac.nz

Abstract—*The aim of many machine learning users is to comprehend the structures that are inferred from a dataset, and such users may be far more interested in understanding the structure of their data than in predicting the outcome of new test data. Part I of this paper surveys representations based on decision trees, production rules and decision graphs, that have been developed and used for machine learning. These representations have differing degrees of expressive power, and particular attention is paid to their comprehensibility for non-specialist users. The graphic form in which a structure is portrayed also has a strong effect on comprehensibility, and Part II of this paper develops knowledge visualization techniques that are particularly appropriate to help answer the questions that machine learning users typically ask about the structures produced.*

The result of machine learning can be evaluated from two quite different points of view: how the knowledge that is acquired performs in new situations; and how well users comprehend the explicit descriptions of knowledge that are generated.

In the literature, machine learning schemes are usually assessed on their performance alone. Techniques such as evaluation on test sets and cross-validation are specifically designed to measure performance. Methods based on minimum description length, which are often used to control pruning, do have some connection with comprehensibility in that succinct descriptions are—other things being equal—easier to understand than lengthy ones. However, they are generally employed to compare alternative descriptions couched in the same terms, rather than being used across representations.

People who apply machine learning in practice frequently discover that their clients adopt the second perspective. What interests them most is not classification performance on new examples, but the perspicuity of the knowledge that is derived from the data and the light that this sheds on the original problem. Representation is of crucial concern to these users: they want to understand the knowledge structures and relate them to the data from whence they came.

Perspicuity of knowledge and knowledge representations is inherently subjective and impossible to treat quantitatively. Moreover, one cannot adopt the standard method of comparing machine learning algorithms by testing them on benchmark data sets, because only users of the data can judge perspicuity, and they are not available. No doubt for these reasons, researchers shy away from the second kind of evaluation. Nevertheless, the importance of user understanding is often of overriding concern in practice, and comparisons should not be based on the wrong thing just because it is easy to measure. The purpose of this paper is to begin to redress the balance.

We define "learning" as the acquisition of structural descriptions from examples, descriptions that can either be used as a basis for performance or studied in their own right—and it is the latter that interests us. In this paper we restrict attention to classification learners, as they are generally the system of choice when the user's aim is to understand an empirical data set through explicit knowledge representation.

The first part of this paper surveys knowledge representations that are used in classification systems. Whereas one often thinks in terms of trees *versus* rules, this is an over-simplistic

view, for many variants of both are in common use and they are likely to significantly influence comprehensibility. The second part of the paper stands back and studies the kind of questions that people ask about the knowledge structures generated. It turns out that these questions require ways of visualizing knowledge structures in tandem with the data from which they are derived. In the second part of the paper we go on to describe such visualizations and give examples of their use.

1 Decision trees

Decision trees are perhaps the oldest, and certainly the most popular, way of representing the outcome of classification learning. They consist of a set of nodes and branches, each node being labeled with an attribute name and each branch leading out of it being labeled with one or more possible values for that attribute. Each node has just one incoming branch, except for one, the root, which is designated as the starting point for traversing the tree. Leaves are labeled with values of the classification attribute.

A data item is classified by traversing the tree. The attribute value of the data item corresponding to the label of the root of the tree is compared to the values on the root's outgoing branches, and the matching branch is selected. This node label matching and branch selection process continues until a leaf is encountered, at which point the data item is classified according to the label of the leaf.

This representation is often augmented by adding two measures to each leaf, one giving the number of data items in the training set that are associated with that leaf, and the other giving either the number of these that are incorrectly classified or the predicted proportion of errors based on a statistical model (Quinlan, 1993). These measures are of particular importance when the user is attempting to tease out significant subtrees rather than simply using the decision tree as a whole for classification. The number of data items covered by a leaf gives a rough indication of the importance or generality of the associated path through the tree, while the error measure provides an approximation of the confidence level or trustworthiness of the path.

A given decision tree may be correct, but incomprehensible. Sometimes, this way of organizing information is just too hard for people to assimilate. Michie (1986) describes how an early induction program (ID3) created a decision tree describing a chess end game. Although it was 100% accurate and computationally efficient, the tree could not be understood by human chess experts—"It was not a question of a few glimmers of sense here and there scattered through a large obscure structure, but just a total blackout." Part of the problem may be simply the large tree size needed to represent the patterns underlying complex data sets. As discussed below, simplifying the tree or rearranging its nodes may be sufficient to restore comprehensibility. Another tack is to adopt graphical visualization techniques; we take this up in Part II.

Simplified decision trees

Real data are invariably noisy and incomplete. They contain errors in measurements, omit factors that contribute to the classification, include irrelevant attributes, etc. To cope with this noise, decision tree induction is usually accomplished in two phases: the creation of an initial tree, which is often very large, and its subsequent pruning to remove branches with low statistical validity (Mingers, 1989). Pruning is intended to solve the problem of overfitting to the training data: the goal is to produce a tree that will have higher classification accuracy on test data, even though its performance on the training data may decline. A number of techniques for simplifying trees have been proposed and investigated (Breiman *et al.*, 1984; Mingers, 1987; Niblett, 1986; Quinlan, 1986, 1987).

An additional advantage of pruning is that because the tree reduces in size, it becomes more comprehensible to humans (Quinlan, 1987). While there is no cognitive litmus test for "understandability," common sense indicates that trees with fewer nodes—and hence fewer possible paths—are more readily understandable than larger ones. An empirical test of five pruning techniques over five test data sets indicates that the effects of pruning can be very

strong, in some cases generalizing large, nearly incomprehensible trees to a few leaves while maintaining an acceptable level of classification performance (Mingers, 1989).

If the accuracy requirement is sufficiently relaxed, pruning may be able to further illuminate implicit concepts in the data. Bohanec and Bratko (1994) "summarize" a decision tree by removing branches that contribute relatively small percentages of the overall total accuracy. Although the accuracy of the decision tree may decline, the primary ideas embodied in the data emerge more clearly, no longer obscured by the plethora of detail supplied by little-used nodes and leaves. By setting decreasing accuracy levels, the pruned tree can be made to provide higher and higher levels of generalization.

Decision stumps

The tree simplification process is carried to an extreme with the 1R procedure (Holte, 1993), which produces decision "stumps"—decision trees of only one level, branching on a single attribute. This was originally developed not as a machine learning technique in its own right, but rather to illustrate the lack of complexity of the standard datasets used to test new machine learning algorithms. It works by building separate rules based on each attribute of a dataset, a procedure that is trivial for discrete-valued attributes and involves a simple discretization scheme for continuous-valued ones. These single-attribute tests are then ranked on their classification ability. Holte showed that for twenty of the standard UCI datasets, the best one-attribute rule achieves performance similar to state-of-the-art machine learning methods!

The set of single-attribute trees can be used to rank the attributes according to their predictive accuracy. This gives an indication of the relative "importance" of each attribute, taken in isolation, to the data as a whole. When the data set contains an excessive number of attributes—too many for conventional ML algorithms to process efficiently—this can be applied as a weeding mechanism, using the ranking to eliminate very weakly predictive attributes. Of course, a set of attributes that, considered individually, are weakly predictive may, taken as a whole, be strongly predictive; though it is not clear that this phenomenon is common in practical data sets.

Trees upon trees: representing structured attributes

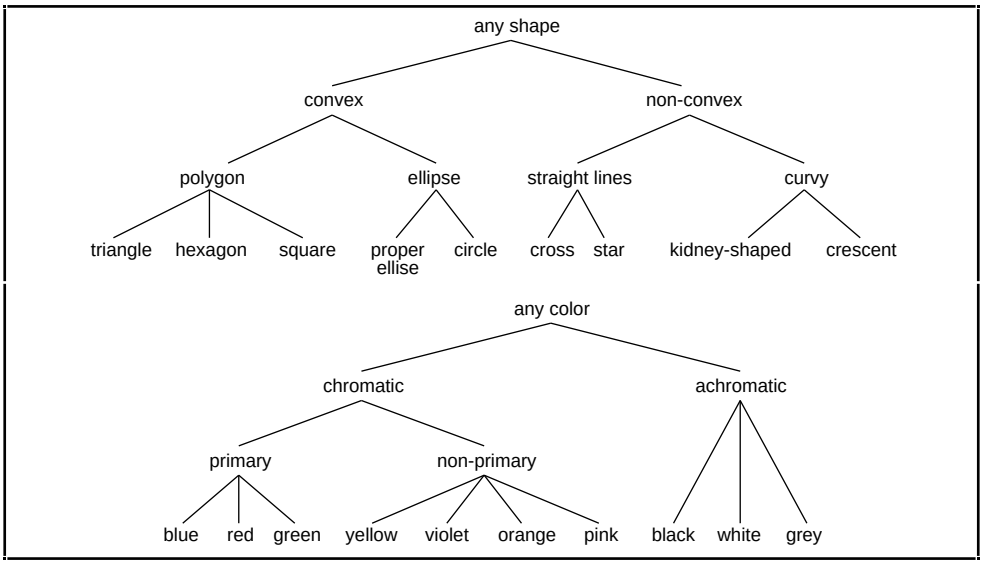
Decision trees can be extended to support tree-structured attributes: that is, discrete attributes whose values are drawn from hierarchies rather than from a simple list of possibilities. For example, Figure 1a presents hierarchies for the *color* and *shape* attributes. While such attributes often occur in natural domains, most decision tree induction schemes insist on attributes being nominal, so that hierarchies must be "flattened." A common technique is to define a new nominal attribute for each level in the hierarchy (Almuallim *et al.*, 1995), as shown for the *color* and *shape* hierarchies in Figure 1c. Using it, the first data item in Figure 1b becomes

[chromatic, non-primary, yellow, convex, polygon, square, +]

The flattening process inevitably increases the number of attributes and obscures the semantics of the data representation.

It is possible for a decision tree induction algorithm to handle hierarchies directly, supporting both a more natural presentation of training data and a decision tree embodying a greater degree of generalization (Almuallim *et al.*, 1995). The tree produced for the training data is shown in Figure 1e. Arc labels are understood to represent that value *and all its ancestors* in the tree. This tree is much smaller than the one C4.5 produces (Figure 1d), and makes tests over more meaningfully expressed attributes. This approach enhances the readability of decision trees by supporting induction over structures that are easily assimilated by people.

a



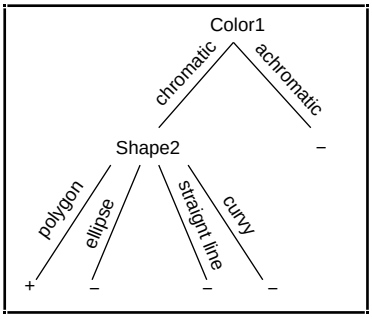
b

shape	color	class
yellow	square	+
green	hexagon	–
white	cross	–
primary	circle	–
black	circle	–
pink	convex	+

c

shape-1	{convex, non-convex}
shape-2	{polygon, ellipse, straight lines, curvy}
shape-3	{triangle, hexagon, square, proper ellipse, circle, cross, star, kidney-shaped, crescent}
color-1	{chromatic, achromatic}
color-2	{primary, non-primary, achromatic}
color-3	{blue, red, green, yellow, violet, orange, pink, black, white, gray}

d



e

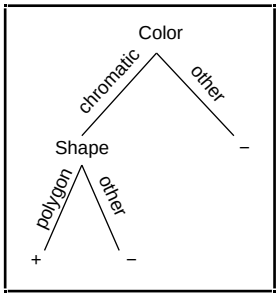


Figure 1 Induction over tree-structured attributes (after Almuallim et al., 1995)
(a) Tree-structured attributes for shape and color
(b) Training examples
(c) Encoding shape and color hierarchies into nominal attributes
(d) Decision tree based on nominal encoding of attributes
(e) Decision tree based on hierarchical attributes

Decision forests: SE-trees

The node splitting mechanism applied by conventional tree induction algorithms introduces a search bias into the learning process, by ordering the attributes for splitting. This bias can be mitigated by using a more general tree representation, the SE-tree, which embeds many alternative decision trees (Rymon, 1993; Rymon and Short, 1994). Multiple attributes can appear at each node, and a complete SE-tree for a given set of attributes contains a representation of all sets of attribute-value pairs. For example, Figure 2 illustrates a complete tree for three binary attributes A, B, and C. An indexing scheme orders the attributes considered at a node, and a node can be expanded only by attributes whose index value exceeds its own. In Figure 2, the attributes are indexed alphabetically.

Classifying a new instance presents two problems. An SE-tree may be incomplete, in that no path through the tree matches the instance, or inconsistent, in that several paths match. Incompleteness is handled by using partial matching to assign classifications. Rymon (1993) argues that inconsistency is, in fact, the SE-tree's greatest advantage, and stems from its ability to express multiple perspectives on a data set. If desired, a single classification can be produced by weighted averaging of the matches. In terms of intelligibility, however, both partial matches and weighted matching are difficult for humans to interpret.

Each SE-tree embeds one *primary* decision tree: the one for which the ordering of the attributes for testing is the same as the attribute index used by the SE-tree construction algorithm. By re-indexing the attributes, an SE-tree can be constructed so as to embed *any* decision tree for a given data set as the primary tree. The user, then, can use the indexing scheme to specify a desired bias for exploring the hypothesis space—essentially starting off with a "favorite" decision tree and building upon it to increase classification accuracy.

Building a complete tree for a given data set may be computationally infeasible. In this case, the user-specified bias can be used as an exploration policy. Tree construction is interrupted when time or space resource limits are exceeded, and the algorithm returns the SE-tree with the best classification accuracy so far.

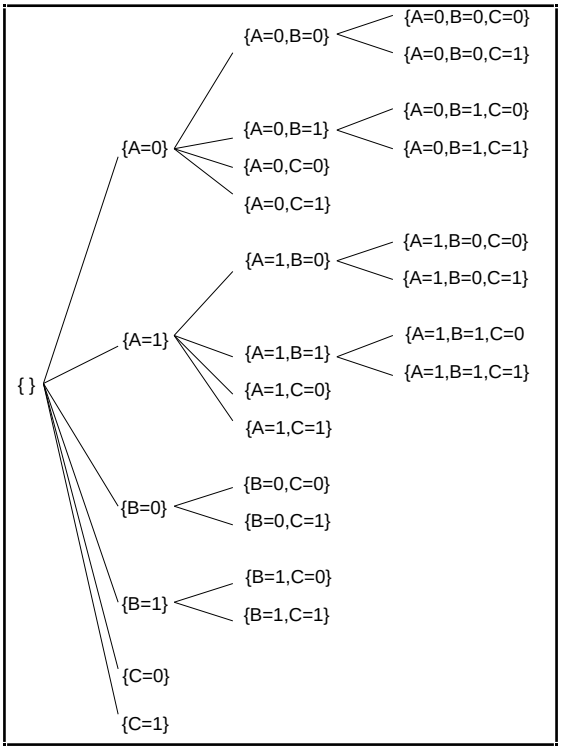


Figure 2 Complete SE-tree for three binary attributes, under alphabetic ordering

Subtree replication

Some knowledge structures cannot be easily or compactly represented as a decision tree (Cendrowska, 1987; Oliver, 1993). For example, there may be no single attribute that is common to all rules implicit in a dataset. In that case, a more appropriate representation would be a "decision forest" of several trees, rather than a single one, since the attribute chosen as the root of the tree cannot provide a clean classificatory split. The only way to force such a structure into tree form is to add extraneous terms to conjoin these disjoint trees into a single unit.

Consider the proposition (A=1 and B=1) or (C=1 and D=1). If the four attributes A, B, C, and D each have three possible values, the proposition can be represented by three rules:

Rule 1: A=1 and B=1 → +
Rule 2: C=1 and D=1 → +
Rule 3: otherwise → –

If attribute A is arbitrarily selected as the partitioning criterion for the root node, the most compact single decision tree representation for this ruleset is shown in Figure 3—obviously less understandable than the rule set above.

This readability problem corresponds to an explosion in the number of possible paths through the tree. Because the decision tree contains duplicated sub-trees, rules 1 and 2 above generate five paths in Figure 3:

A=1 and B=1 → +
A=1 and B=2 and C=1 and D=1 → +
A=1 and B=1 and C=1 and D=1 → +
A=2 and C=1 and D=1 → +
A=3 and C=1 and D=1 → +

In general, many Boolean functions with small propositional or rule representations have corresponding decision trees that are large, redundant, and inefficient (Pagallo and Haussler, 1990).

Subtree fragmentation

It is hard to comprehend "bushy" decision trees in which nodes have many children (Oliver, 1993). Standard decision-tree induction algorithms produce such trees when attributes have many possible values, as illustrated in Figure 4a. Bushy trees fragment the training set into many leaves, each covering only a small proportion of the data.

The problem is often alleviated by clustering attribute values to test for more than one value. Figure 4b shows alternative structures for the eight-way node on the left, reducing the

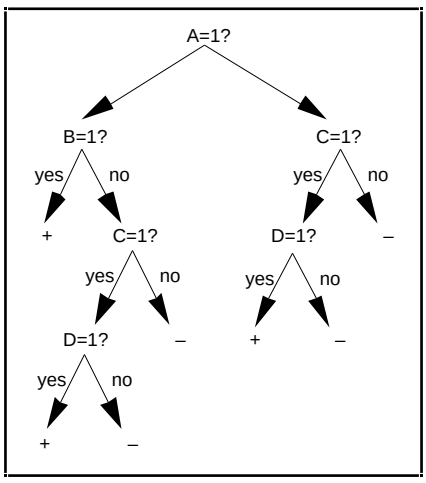


Figure 3 Decision tree representation for (A=1 and B=1) or (C=1 and D=1)

number of branches to four and two respectively. Obviously there are many such possibilities, and the subset construction process is expensive computationally (Quinlan, 1986). Unfortunately, there is no way of reducing the search by first selecting between value subsets of different sizes—for example, choosing whether to branch eight, four, or two ways in Figure 4b (Oliver, 1993).

2 Production rules

Production rules are a popular alternative to decision trees for representing the outcome of machine learning. Ideally, each member of a set of rules is a self-contained, cognitively manageable unit. The modular nature of rules is very helpful when isolating parts of a rule set for study. In practice, however, the rule execution mechanism dictates the semantics of a rule set. Are rules tested all at once?—and if so, how is conflict handled? Are they tested in sequence until one fires?—in which case, their effect is not independent. What if no rule fires?—can a default be specified?

A rule set is sometimes far more comprehensible than the corresponding decision tree, particularly when it contains relatively disjoint pieces of information. Moreover, if rules really are independent then different rule sets can be merged into a composite one, whereas there is no straightforward way to merge decision trees (Quinlan, 1987). This supports a more incremental approach to induction.

It is trivial to convert a decision tree to a set of production rules by writing each path through the tree, from root to leaf, as a separate rule. As we have seen, the process of forming a tree and converting it in this simple way can produce an unnecessarily large set of rules. This initial set of candidate rules can be simplified by generalizing rules, dropping irrelevant conditions and removing redundant ones (Quinlan, 1987). As rules are generalized, some can be removed because they become duplicates. Others may become vacuous because their conditions are completely eliminated. The next step is to evaluate the contribution of each rule to the overall accuracy of the rule set. Rules that do not reduce the number of classification errors (or ones that actually increase it) are omitted. These measures can greatly reduce the perceived complexity of a rule set.

A disadvantage of rules as compared to decision trees is that there is no indication of interactions or relationships between them. As the number of rules grows, the proportion of

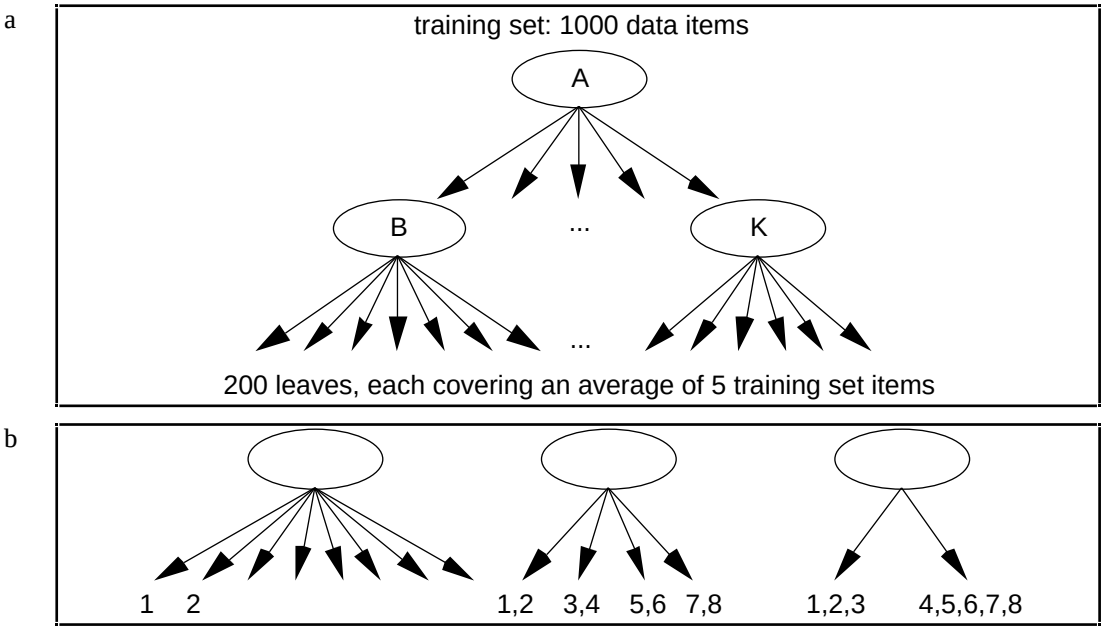


Figure 4 Decision tree fragmentation and attribute partitioning (from Oliver, 1993)
(a) Fragmentation of the data set
(b) Alternative partitionings of an attribute's values

the training set covered by each individual one decreases, and it becomes more difficult to gain an understanding of the structure of the system as a whole (Corruble *et al.*, 1995). One measure of the extent to which rules are related is the similarity between their conditions. This can be visualized by converting rules to a decision tree or graph (see below).

Another approach is to group rules by analyzing similarities between the training examples they cover (Riddle *et al.*, 1994). There are two ways a data item can match a rule: it may satisfy the rule's conditions, and it may satisfy the rule's consequence as well. This provides a basis for calculating the distance between any two rules, which in turn allows the rules to be clustered hierarchically by distance. Figure 5 shows a structure that has been obtained by clustering rules in this way. Although this representation resembles a decision tree in appearance, it cannot be used effectively for prediction. Instead, the hierarchy is intended as a visual inspection aid for detecting rules that process similar portions of the training set.

Decision lists

The decision list representation is akin to an extended "if – then – else if – ..." rule (Rivest, 1987; Clark and Niblett, 1989). It is essentially an ordered set of rules, as illustrated in

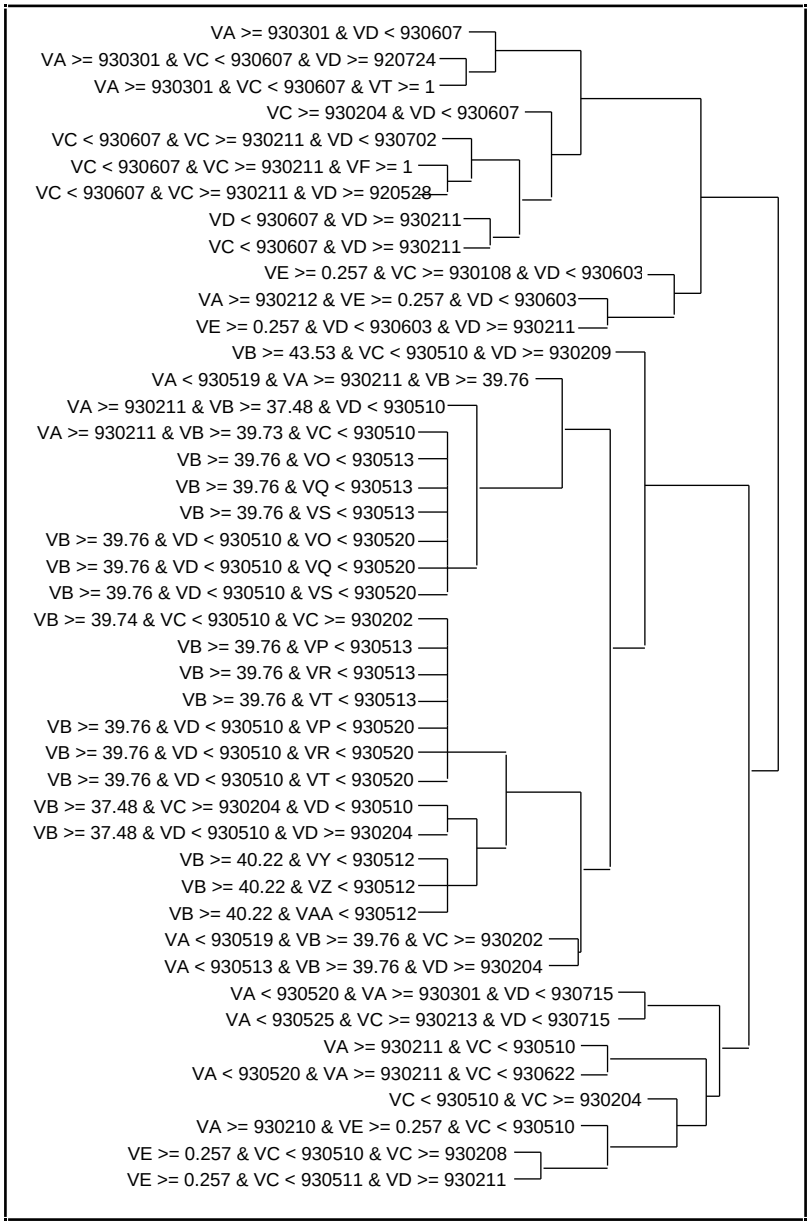


Figure 5 Hierarchical clustering of rules by similarities in data coverage (after Riddle, 1994)

Figure 6. When used predictively, the rules are tested in sequence until one fires; the remainder are ignored. The final rule in the list has a null condition and serves as a default classification. With this execution strategy, specific rules—corresponding to exceptions to the general pattern—appear early in the list, while widely applicable rules appear later.

The rules in a decision list lack the modularity of unordered production rules; a given rule is meaningful only on the context of its predecessors. Furthermore, the ordering can only capture very crude interactions between rules. For example, an exception is global to the entire collection of rules beneath it (Holsheimer and Siebes, 1994).

Ripple-down rules

Ripple-down rules can be viewed as an extension of decision lists that provide a way of specifying exceptions to each rule individually. Unlike decision lists, however, general rules *precede* their exceptions. Exceptions are "rippled down" by nesting their if-then tests within the more general rule. Figure 7 shows a set of ripple-down rules for the concept (A=1 A=2) ¬ (A=1 A=2).

This representation expresses relationships between rules by arranging things so that more specific rules are presented as exceptions to more general ones. Compton and Jansen (1990) argue that this format provides a sense of context by making apparent the relative generality of substructures and the degree to which one sub-rule overlaps or subsumes others.

The "depth" of a rule is measured by the extent to which it is nested in other rules. For example, in decision lists all rules are at depth 1, while the innermost rules in Figure 7 have depth 2. The greater the depth, the more difficult it will be to interpret a given rule. In practice, the depth must remain fairly small because both run time and sample complexity of ripple-down rule induction algorithms grow exponentially with depth (Kivinen *et al.*, 1992).

3 Decision graphs

When considering the shortcomings of decision trees and rules, one is naturally drawn to consider the use of graph representations. Recent research has shown how graphs can be derived from trees by unifying common substructures, while a second line of work has arrived at graph structures by considering a generalization of ripple-down rules.

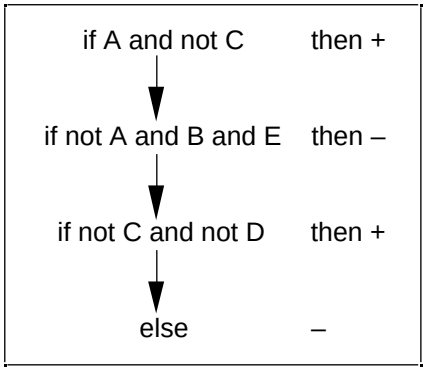


Figure 6 A decision list (from Rivest, 1987)

```
if A=1 then +
   except that if A=2 then –
if A=2 then +
   except that if A=1 then –
default: –
```

Figure 7 Ripple-down rules for the concept (A=1 A=2) ¬ (A=1 A=2) (from Kivinen *et al.*, 1992)

Deriving graphs from trees

Decision graphs address the problems of redundancy and fragmentation by merging duplicate nodes of decision trees. Formally, a decision graph is a directed acyclic graph with two kinds of nodes: *category* nodes and *decision* nodes (Kohavi, 1994). All category nodes have outdegree zero; they are the classification "leaves" of the graph. Each decision node is labeled by an attribute, and has one outgoing arc for each possible value of that attribute. A distinguished decision node, the *root* node, is the only one with indegree zero; it provides a unique entrance point into the graph.

A decision graph, like a decision tree, assigns a classification to a data item by traversing that item through the structure. A unique path is traced from the root to a category node by matching data attribute values to arc labels in the graph. Like decision trees, graphs are generally "read-once" in that each variable occurs at most once along a decision path.

Figure 8 shows a decision graph representation of Quinlan's "probabilistic classification over disjunctions" dataset (Quinlan, 1987), an artificial dataset containing ten binary attributes (A0–A9). One of the attributes (A0) is irrelevant, and is included primarily to evaluate the power of induction algorithms to detect irrelevant attributes. The data is generated as follows:

```
if (A1   A2   A3)   (A4   A5   A6)   (A7   A8   A9)
    then   class = + with probability 0.9, – with probability 0.1
    else   class = – with probability 0.9, + with probability 0.1
```

The most compact decision tree for this dataset has 39 internal nodes and 40 leaves (Oliver, 1993). The graph of Figure 8 renders the underlying function more comprehensible by avoiding node duplication.

Decision graphs can be derived from decision trees (Mahoney and Mooney, 1991; Kohavi and Li, 1995; Oliveira and Sangiovanni-Vincentelli, 1995), or induced directly (Oliver, 1993; Kohavi, 1994; Corruble *et al.*, 1995). Rather than working with a general decision graph, some algorithms induce a special case of "oblivious" graphs (Kohavi, 1994; Kohavi and Li, 1995). A decision graph is *leveled* when the nodes are partitioned into a sequence of pairwise disjoint sets—the levels—such that all outgoing edges from each level terminate at the next level. (The

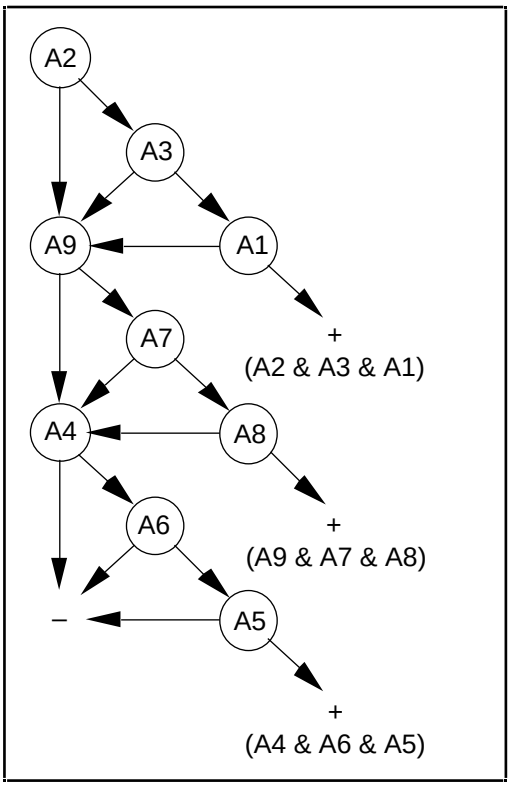


Figure 8 Decision graph for a problem of Quinlan's (after Oliver, 1993)

example in Figure 8 is not a leveled graph.) An *oblivious* decision graph is not only leveled, but all nodes at a given level are labeled by the same variable.

Moreover, an oblivious decision graph can be *reduced* so that no two distinct nodes on the same level branch in exactly the same way on the same values by the simple procedure of merging such nodes if they exist. It can be shown that for any decision structure, there is a reduced oblivious, read-once decision graph that is no larger than the smallest general decision graph representing that structure (Kohavi and Li, 1995). If "size" is correlated with comprehensibility, as seems intuitively likely, then these restricted structures will tend to be more understandable than the general form.

Exception directed acyclic graphs

An exception directed acyclic graph (EDAG) is a kind of generalized decision graph that, like ripple-down rules, can explicitly represent exceptions to a more broadly applicable concept, while at the same time expressing relationships between rules (Gaines, 1995, 1996). The decision graph representation is expanded so that:

- a graph can have more than one root, and may be composed of several disconnected parts;
- test conditions are associated with nodes rather than with arcs;
- a decision node can contain both a test condition and a category;
- branching tests do not have to be mutually exclusive, so that multiple conclusions may be inferred.

Figure 9 shows an EDAG representation of the well-known contact lens prescription problem (Cendrowska, 1987). The single root node contains a default conclusion—that no contact lens should be prescribed. All paths from the root are explored; paths are not labeled at all. A path ends when the test condition of a node along the path fails, or when a leaf is encountered. For example, if the "tear production = normal" test fails, the system falls back on the default classification, "none". For each node encountered on the path, the conclusion of the node (if

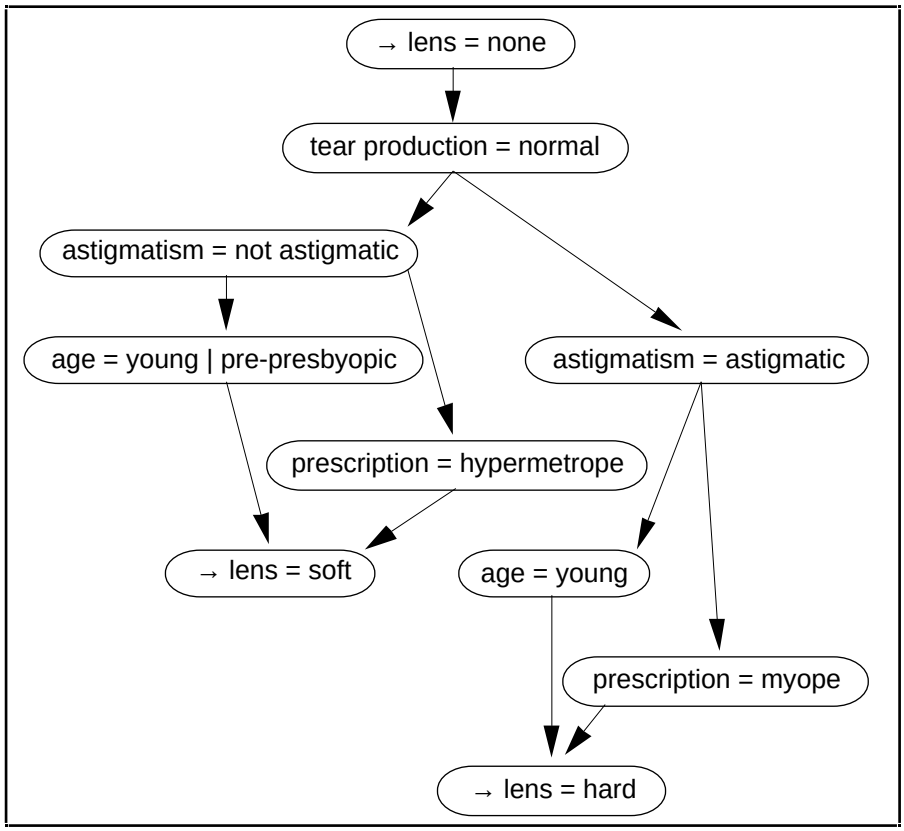


Figure 9 EDAG for the contact lens prescription problem (from Gaines, 1996)

any) replaces any previous conclusion noted along that path. If the graph contains more than one root, the subgraphs for each root are traced. Since multiple paths may succeed, more than one classification can be produced for a given data item.

Note that rules, decision trees, and rules with exceptions are all subsumed by the EDAG representation. A set of production rules forms a trivial graph with one subgraph per rule and no connections. A decision tree is already a graph in tree form, with no disconnected parts and with node replication. Rules with exceptions form a graph whose internal nodes contain conclusions. An EDAG representation of a given induction result is therefore guaranteed to be no larger than its corresponding set of rules or decision tree, and can be significantly smaller (Gaines, 1996). If size and comprehensibility are indeed related, then an EDAG representation will be no less intelligible than the corresponding tree, rules, or graph, and may be more understandable.

Conclusions

The aim of many machine learning users is to comprehend the structures that are abstracted from a dataset, and such users may be far more interested in understanding the structure of their data than in predicting the outcome of new examples. In contrast, the emphasis in the machine learning research literature tends to focus very much on prediction ability.

Several different representations have been developed, and used in machine learning, that have differing degrees of expressive power—and therefore comprehensibility. The basic representations are trees, rule sets, and graphs. Within each are many variants, and a development can be traced that proceeds from simple structures to ones with higher degrees of semantic expressiveness. In general, one would expect an expressive language to permit more compact representations of complex decisions, but impose greater demands on the user's ability and motivation to learn how to interpret the result. Sophisticated structures like EDAGs are likely to appeal most to users who realize that their problem is complex and are prepared to devote some effort to understanding the result, while more rudimentary ones like plain trees and rules are more suitable for simple problems and for users with a casual or fleeting interest in the results.

The graphic form in which a structure is portrayed has a strong effect on comprehensibility. Part II of this paper develops knowledge visualization techniques that are particularly appropriate to help answer the questions that machine learning users typically ask about the structures produced.

References

Almuallim, H., Akiba, Y. and Kaneda, S. (1995) "On handling tree-structured attributes in decision tree learning." *Proceedings of the 12th International Conference on Machine Learning*, 12–20.

Bohanec, M. and Bratko, I. (1994) "Trading accuracy for simplicity in decision trees." *Machine Learning* 15(3), 223–250.

Breiman, L., Freidman, J., Olshen, R. and Stone, C. (1984) *Classification and regression trees*. California: Wadsworth International.

Cendrowska, J. (1987) "PRISM: An algorithm for inducing modular rules." *International Journal of Man-Machine Studies* 27, 349–370.

Clark, P. and Niblett, T. (1989) "The CN2 induction algorithm." *Machine Learning* 3(4), 261–284.

Compton, P. and Jansen, B. (1990) "A philosophical basis for knowledge acquisition." *Knowledge Acquisition* 2(3), 179–278.

Corruble, V., Thire, F. and Ganascia, J-G. (1995) "Comprehensible exploratory induction with decision graphs." *IJCAI'95 Workshop on Machine Learning and Comprehensibility*. Available at <URL: `http://www.lri.fr/~cn/web-ijcai/vincent.ps`>.

Gaines, B.R. (1995) “Structured and Unstructured Induction with EDAGs.” *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95).* Available at <URL: http://ksi.cpsc.ucalgary.ca/articles/Induct/EDAG95/>.

Gaines, B.R. (1996) “Transforming rules and trees into comprehensible knowledge structures.” in *Advances in Knowledge Discovery and Data Mining*, edited by U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy. Cambridge, Massachusetts: MIT Press.

Holsheimer, M. and Siebes, A.P.J.M. (1994) Data mining: the search for knowledge in databases. Technical report CS-R9406, CWI, Amsterdam, The Netherlands.

Holte, R.C. (1993) “Very simple classification rules perform well on most commonly used datasets.” *Machine Learning*, 11, 63–90.

Kivinen, J., Mannila, H. and Ukkonen, E. (1992) “Learning rules with local exceptions,: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 37–44.

Kohavi, R. (1994) “Bottom-up induction of oblivious read-once decision graphs: strengths and limitations.” *Proceedings of AAAI-94*. Available at <URL: ftp://starry.stanford.edu/pub/ronnyk/aaai94.ps>.

Kohavi, R. and Li, Chia-Hsin (1995) “Oblivious decision trees, graphs, and top-down pruning.” *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95).* Available at <URL: ftp://starry.stanford.edu/pub/ronnyk/eodg.ps>.

Mahoney, J.J. and Mooney, R.J. (1991) Initializing ID5R with a domain theory: some negative results. Technical Report 91-154, Computer Science Dept., University of Texas at Austin, Austin, TX, USA.

Michie, D. (1986) *On Machine Intelligence*, second edition. Chichester, England: Ellis Horwood.

Mingers, J. (1987) “Expert systems—rule induction with statistical data.” *Journal of the Operational Research Society* 38, 347–352.

Mingers, J. (1989) “An empirical comparison of pruning methods for inductive learning.” *Machine learning* 4(2), 227–243.

Niblett, T. (1986) “Constructing decision trees in noisy domains”. In *Progress in machine learning*, edited by I. Bratko and N. Lavrac. England: Sigma Press.

Oliveira, A.L. and Sangiovanni-Vincentelli, A. (1995) “Inferring reduced ordered decision graphs of minimum description length.” *Proceedings of the 12th International Conference on Machine Learning* (Tahoe City, CA, USA), 421–429.

Oliver, J.J. (1993) “Decision graphs—an extension of decision trees.” *Proceedings of the 4th International Workshop on AI and Statistics*, 343–350.

Pagallo, G. and Haussler, D. (1990) “Boolean feature discovery in empirical learning.” *Machine Learning* 5, 71–99.

Quinlan, J.R. (1986) “Induction of decision trees.” *Machine Learning* 1, 81–106.

Quinlan, J.R. (1987) “Simplifying decision trees.” *International Journal of Man-Machine Studies*, 221–234.

Quinlan, J.R. (1993) *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Riddle, P., Fresnedo, R. and Newman, D. (1994) “Framework for a generic knowledge discovery toolkit.” *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, 457–468.

Rivest, R.L. (1987) “Learning decision lists.” *Machine Learning* 2(3), 229–246.

Rymon, R. (1993) “An SE-tree based characterization of the induction problem.” *Proceedings of the 10th international conference on machine learning* (Amherst, MA, USA), 268–275.

Rymon, R. and Short, N.M. (1994) “Automatic cataloguing and characterization of earth science data using SE-trees.” Coddard Conference on Space Applications of Artificial Intelligence (Greenbelt, MD, USA). Available at <URL: http://www.isp.pitt.edu/~rymon/papers/tele94.ps>.