ID5: An Incremental ID3

Paul E. Utgoff

COINS Technical Report 87-95

December 30, 1987

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Telephone: 413-545-4843

Abstract

This paper describes ID5, an incremental algorithm that produces decision trees similar to those built by Quinlan's ID3. The principal benefit of ID5 is that new training instances can be processed by revising the decision tree instead of building a new tree from scratch. ID3, ID4, and ID5 are described in detail, compared, and contrasted. Experiments illustrate observed performance in terms of training expense and classification accuracy.

Contents

1. Introduction

One of the problems that continues to receive much attention by scientists in the field of Machine Learning is that of mechanical induction. The induction problem is:

Given: A stream or batch of instances in which each instance is described by a conjunction of features, and each instance is labelled as a positive instance (example) or as a negative instance (counterexample) of the target concept (concept to be induced),

Find: A classification rule that correctly determines whether an unobserved instance is positive or negative.

A number of induction algorithms have been invented, including those of Winston (Winston, 1975), Michalski (Michalski and Chilausky, 1980), Mitchell (Mitchell, 1978), and Vere (Vere, 1980).

The ability to learn a reliable classification rule is fundamental to intelligent behavior. Concepts such as "dangerous", "edible", "ally", "profitable", and others, are routine and essential. In addition, classifiers can provide decisions that drive other intelligent processes. For example, the Meta-Dendral program learned the concept of a chemical bond that is likely to break in a mass-spectrometer (Buchanan and Mitchell, 1978). The LEX program learned concepts that distinguished good and bad candidate problem states for particular operator applications (Mitchell et al, 1983). The AQ11 program learned to diagnose soybean disease according to observable symptoms (Michalski and Chilausky, 1980).

2. ID3

One inductive algorithm that has performed well is Quinlan's ID3 (Quinlan, 1983; Quinlan, 1986). His algorithm builds a decision tree (Moret, 1982; Breiman et al, 1984) that can be used to classify instances. To build a decision tree, the algorithm chooses a *test attribute* that best partitions the instances into smaller sets for which decision subtrees are constructed recursively. The algorithm is:

1. If all the instances are positive, return the decision tree "+".[1]
2. If all the instances are negative, return the decision tree "−".[2]
3. Define A_{best} to be the attribute with the lowest $INFO$ value.
4. For each value $V_{best,j}$ of A_{best}, recursively construct a new decision tree $D_{best,j}$ from those instances with value $V_{best,j}$.
5. Return the decision tree with test attribute A_{best}, and branches $V_{best,j}$ each connected to subtree $D_{best,j}$.

[1] meaning "the classification is positive"
[2] meaning "the classification is negative"

The $INFO$ function is an information-theoretic measure that Quinlan adapted from Shannon's entropy measure (Shannon, 1948). The value returned by $INFO$ is to be considered a measure of the average amount of decision making that would need to be done in order to determine the classification of an instance. The strategy of picking the apparent best single attribute does not guarantee that an optimal[3] decision tree is constructed. However, experience with the algorithm has shown that it usually builds simple decision trees that generalize well to unobserved instances.

The $INFO$ function, given in the appendix, is common to each of ID3, ID4, and ID5. It need not be understood carefully to follow the discussion. The main point to note is that $INFO$ is a function of the number of positive and negative instances seen for each value of the given attribute. It is used as a heuristic for picking the test attribute for the root of a decision tree or subtree. Throughout this paper, the following notation is employed:

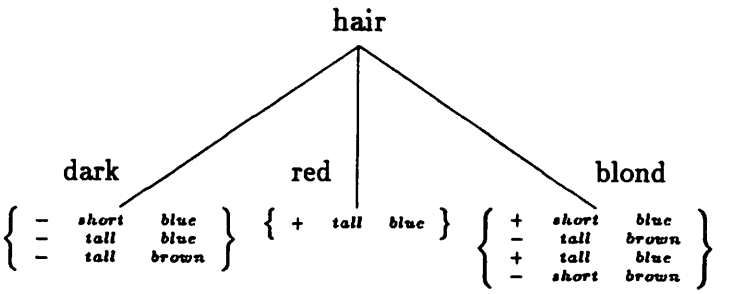
A_x	The name of an attribute
$V_{x,y}$	The name of a value for attribute A_x
$[p,n]$	Indicates p positive instances and n negative instances observed
$P(name)$	Returns the number of positive instances observed.
$N(name)$	Returns the number of negative instances observed.
m_x	Number of values for attribute A_x

Observe how ID3 builds a decision tree for the 8 instance example from Quinlan (Quinlan, 1983). Each instance is described as a conjunction of three attribute-value pairs, using the attributes: *height*, *hair* color, and color of *eyes*. The instances are:

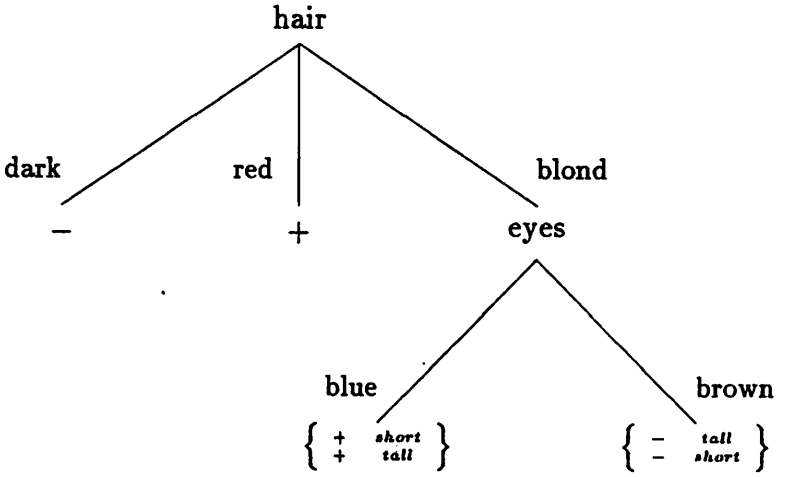
class	height	hair	eyes
−	short	blond	brown
−	tall	dark	brown
+	tall	blond	blue
−	tall	dark	blue
−	short	dark	blue
+	tall	red	blue
−	tall	blond	brown
+	short	blond	blue

ID3 determines *hair* to be the best test attribute because $INFO(hair) = .5000$, $INFO(eyes) = .6068$, and $INFO(height) = .9512$. This produces the initial tree:

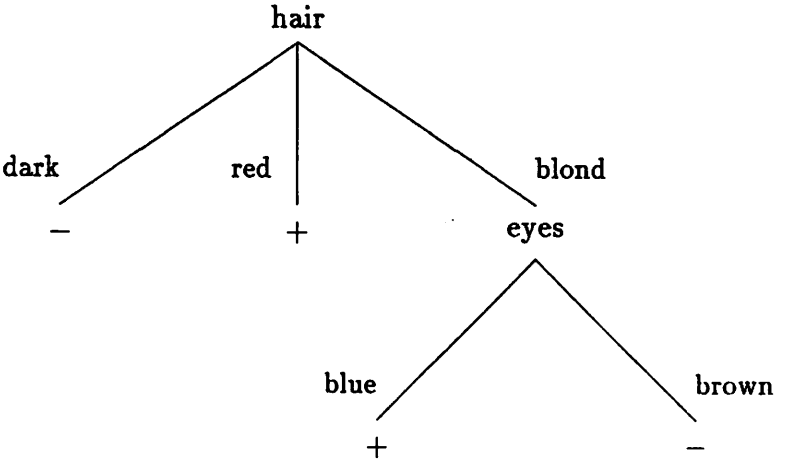
[3]Here, *optimal* means containing the fewest nodes (tests) needed to be able to correctly classify all the training instances.



The algorithm is applied recursively at each leaf node, producing the intermediate tree:



The final step produces the decision tree:



The ability of ID3 to construct decision trees that are efficient classifiers and that generalize well is attractive. For learning problems in which a database of instances is

available and is not likely to change, ID3 is a good choice for building a classification rule. For problems in which new instances are expected to become available on a regular basis, it would be preferable to accept instances incrementally, without needing to build a new decision tree from scratch each time.

3. ID4

Schlimmer and Fisher constructed ID4 (Schlimmer & Fisher, 1986), which incrementally builds a decision tree similar to that which ID3 would build. Instead of building a decision tree from a batch of instances, ID4 updates a decision tree based on each individually observed instance. The algorithm for handling a training instance is:

1. For each potential test attribute at a node, update the count of positive and negative instances for that attribute, and the positive and negative count for the observed value of that attribute.

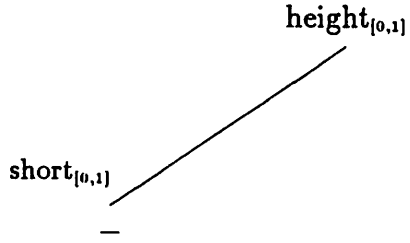
2. If the lowest INFO measure for a non-test attribute is lower than that of the current test attribute, then discard the subtrees below the current test attribute, and redefine the test attribute to be A_{best}.

3. If $P(A_{best}) = 0$, return the decision tree "−".

4. If $N(A_{best}) = 0$, return the decision tree "+".

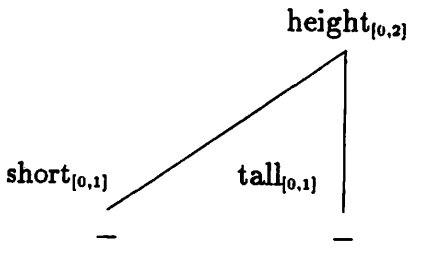
5. Update the decision tree below the value of A_{best} that occurs in the instance description. Return the decision tree with test attribute A_{best}.

Observe how ID4 updates a decision tree incrementally, as each of the 8 instances of the example set is processed. After the first instance (−,short,blond,brown), the decision tree is:



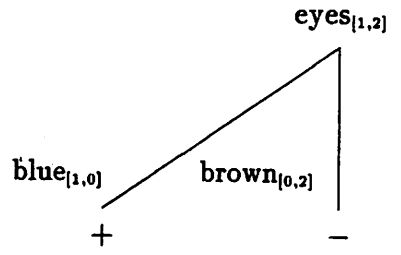
Note the count of 0 positive instances and 1 negative instance indicated in brackets. Here, the attribute "height" was chosen randomly because all three possible attributes have the *INFO* value 0.0.

The second instance (−,tall,dark,brown) updates the tree to:



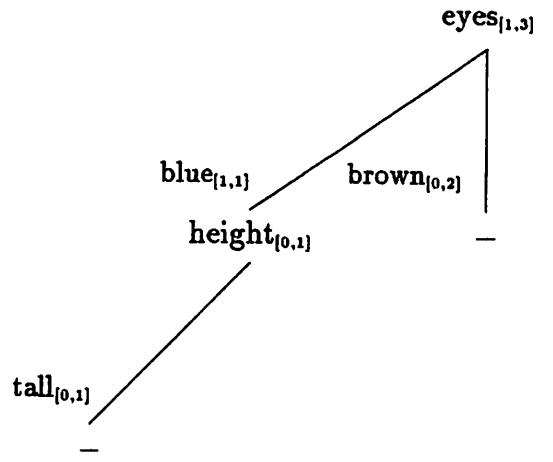
"Height" remains the test attribute because neither of the other two attributes has a lower *INFO* measure.

The third instance (+,tall,blond,blue) updates the tree to be:



Note that the test attribute has been revised to be "eyes" instead of "height". This is because "eyes" now has an *INFO* score of 0.0 while both "height" and "hair" now have scores of 0.667.

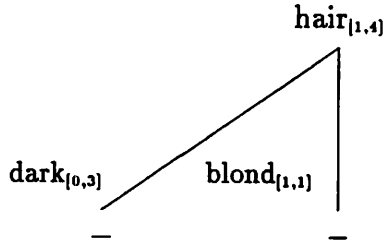
The fourth instance (−,tall,dark,blue) updates the tree to:



It may seem that there is an error in the count of positive and negative instances below "blue". This is because one positive and one negative instance have been observed. However, the subtree with test attribute "height" was not created until the negative instance was observed. A subtree is needed only when at least one positive and at least one negative instance have been observed. So, the negative instance is the first one to have been

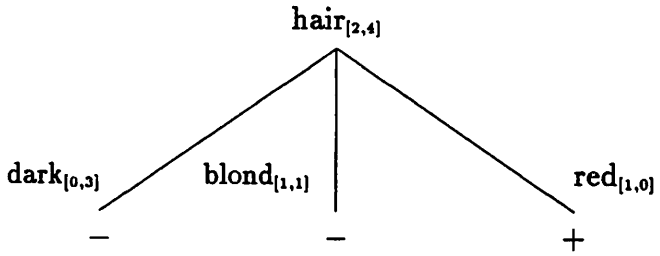
recorded at node "height" and below.

The fifth instance (−,short,dark,blue) causes the tree to be revised to:

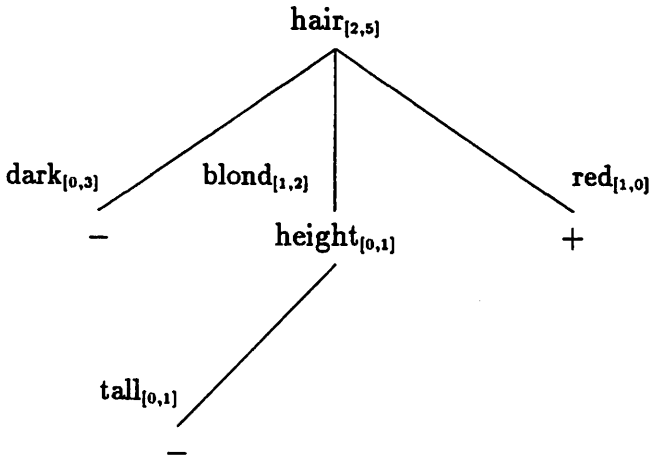


This revision demonstrates how ID4 discards the subtrees of an attribute that formerly was the test attribute. This aspect of the algorithm results in periodic and potentially expensive setbacks during training.

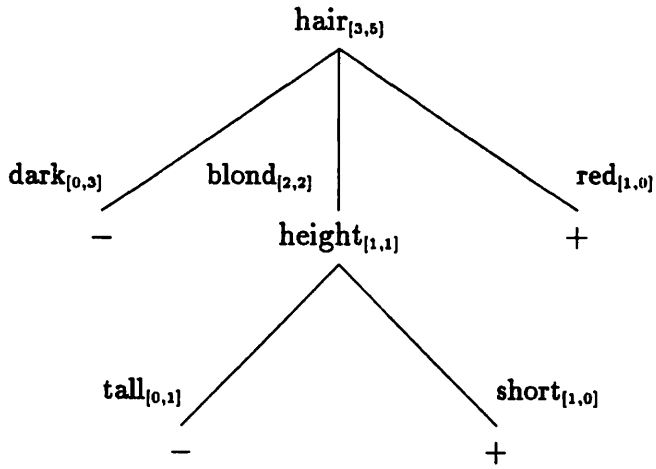
The sixth instance (+,tall,red,blue) updates the tree to:



The seventh instance (−,tall,blond,brown) leads to:

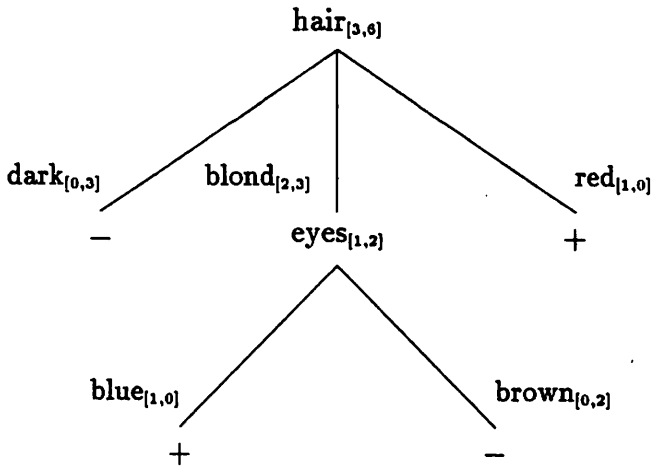


The eighth instance (+,short,blond,blue) updates the tree to:



Although all eight instances have been processed, the decision tree still does not classify all eight instances correctly.

Cycling to the first instance again, ID4 updates the decision tree to be:



Note that the subtree below "blond" formerly had test attribute "height" but now has "eyes" as the test attribute. At this point, the decision tree classifies all eight training instances correctly.

The algorithm offers an approach to incremental learning of ID3-type decision trees. A drawback of the algorithm is that all or part of a decision tree will be discarded whenever it is determined that the test attribute should be replaced with a better attribute. The effect of such replacements is that training effort is lost. Schlimmer and Fisher indicate that the decision tree eventually stabilizes. More training is needed, because the benefit of previous training can be lost, but the expense of training is considered reduced when compared to ID3. It would be preferrable if an incremental algorithm did not suffer setbacks from loss of training information.

4. ID5

ID5 builds on the idea in ID4 that one can maintain positive and negative instance counts of every attribute that could be a test attribute for the decision tree or a subtree. ID5 differs from ID4 in its method for replacing the test attribute. Instead of discarding the subtrees below the old test attribute, ID5 reshapes the tree by pulling the test attribute up from below. This adds to the space requirement of the algorithm because the instances that correspond to each subtree must be retained so that the pull-up is possible.

The algorithm for handling an instance is:

1. For each potential test attribute at a node, update the count of positive and negative instances for that attribute, and the positive and negative count for the observed value of that attribute.

2. If the lowest INFO measure for a non-test attribute is lower than that of the current test attribute, then reshape the tree by pulling A_{best} up from below.

3. If $P(A_{best}) = 0$ or $N(A_{best}) = 0$, then store the rest of the training instance. Stop.

4. Recursively update the decision tree below the value of A_{best} in the instance description.

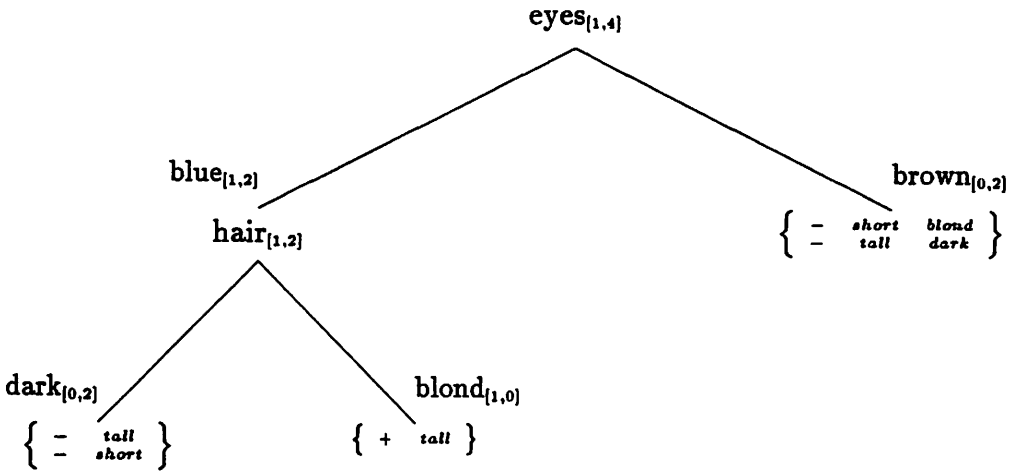
Pulling an attribute up to be the new test attribute requires a process that restructures the tree without losing training information. This requires that the portion of each instance description not implicit in the decision tree be stored at each leaf so that a pull-up is possible. The algorithm for pulling up an attribute is:

1. Recursively pull the attribute to the root of each immediate subtree. (Expand any set of instances at a subtree to form a proper decision tree as necessary.)

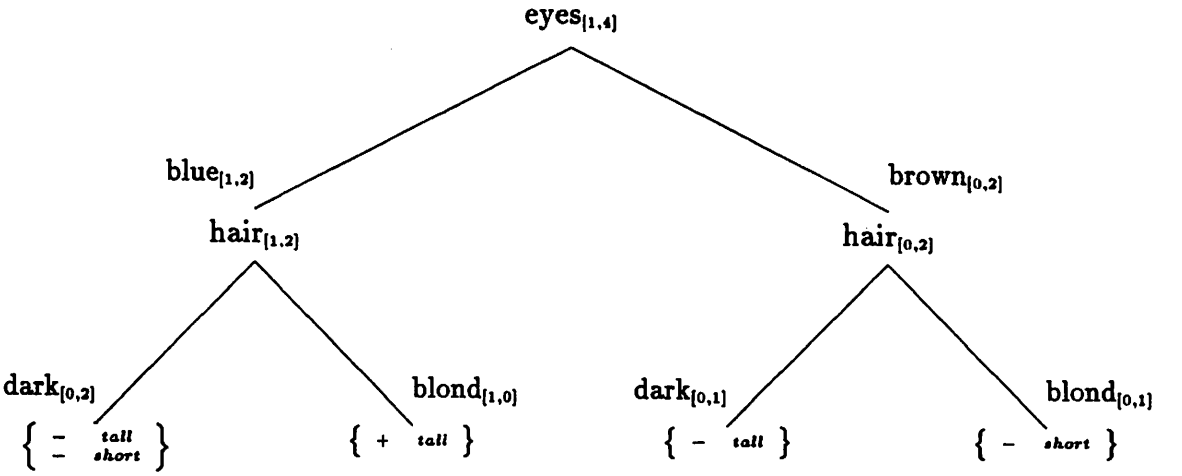
2. For each value branch of each subtree, construct a new decision tree by pushing the old test attribute below the new test attribute. This results in a set of subtrees, each with the desired test attribute at the root.

3. Merge the subtrees, so that the desired test attribute is at the root of a single decision tree.

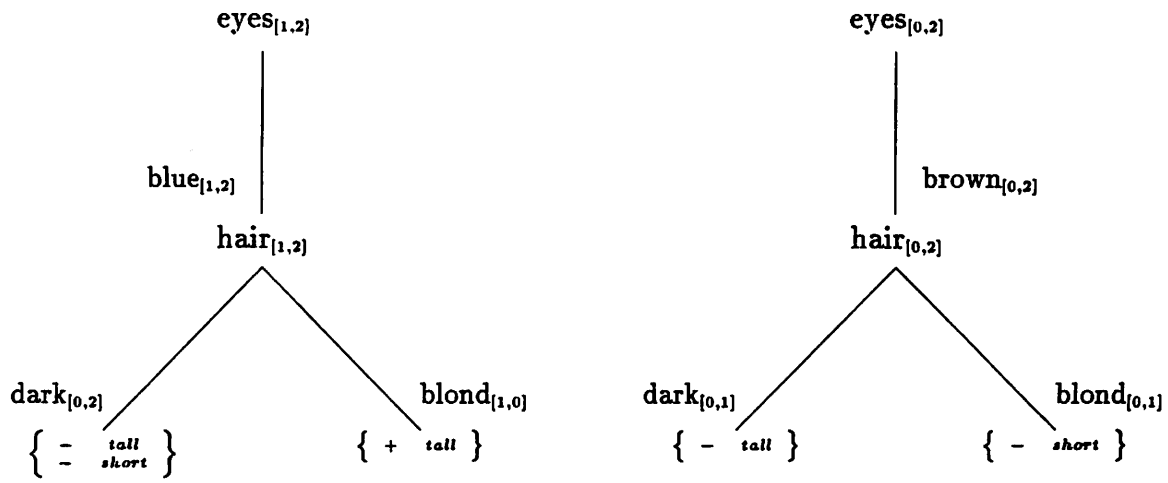
Consider an illustration of the pull-up process. Assume the current decision tree is:



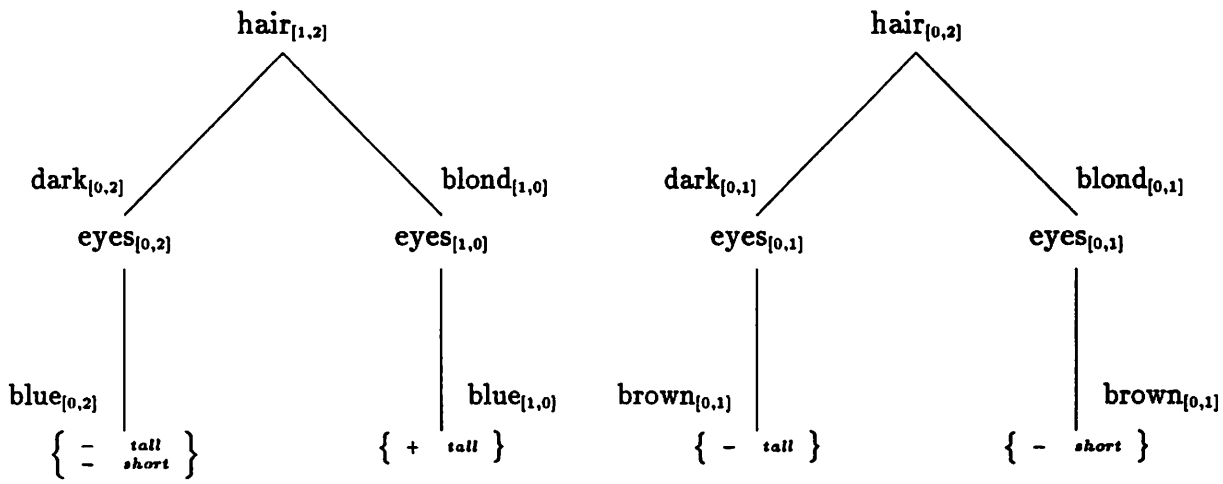
Though the tree is well formed, "eyes" does not have the lowest $INFO$ value and is therefore not the best test attribute. Instead, "hair" should be the test attribute. Hence, "hair" is to be pulled to the root of the tree. The tree below value "brown" is a set of instances. It is expanded into a proper tree, with "hair" at the root. This produces:



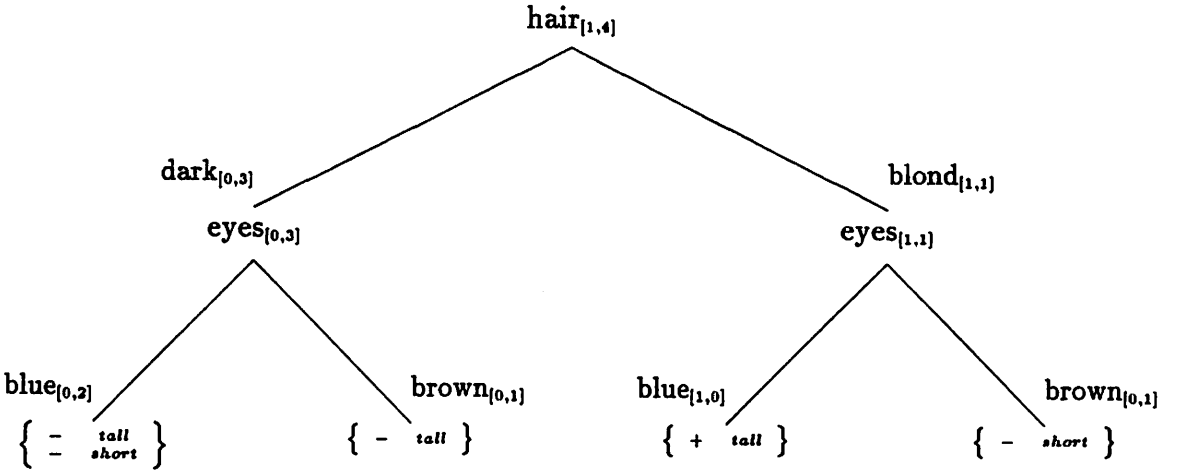
Step 1 of the pull-up process does nothing in this case, because "hair" is already the test attribute of every subtree of "eyes". Step 2 splits the tree into a set of trees, one for each subtree of "eyes", producing:



Step 2 then proceeds to push the test attribute "eyes" down each branch of its single subtree, producing:

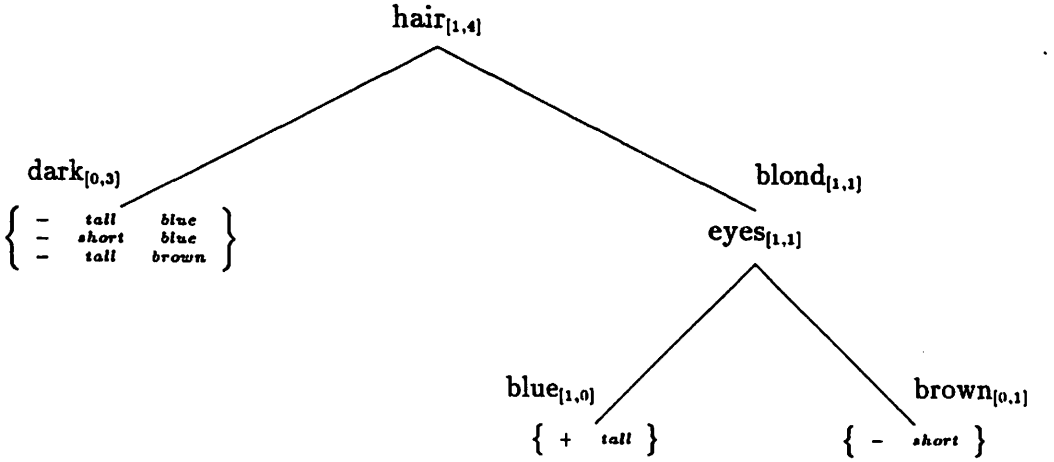


Step 3 merges the set of trees, all of which have "hair" at the root, making:



Note that the subtrees below the new test attribute are not guaranteed to have the best test attribute at the root. This is because the former test attribute, which is now the test attribute of each subtree, was not selected by choosing the best test attribute according to the $INFO$ function. Rather, the test attribute was determined as a byproduct of the pull-up step. To assure that each subtree has the best test attribute at its root, it is necessary to re-estabblish the best test attribute for each subtree recursively. This is simple enough to do, but it is not done in ID5 because it is expensive to do on a regular basis. Furthermore, it is not critical, unless one needs an ID3-equivalent tree immediately, because subsequent training that causes a subtree to be visited will bring the best test attribute to the root of that subtree.

A second point to note is that the subtree below value "dark" can be contracted to a set of instances. This is a simple operation, but there is no need to incur the expense unless space is in demand. If this were done in the above example, the result would be:



Now observe how ID5 performs on the 8 instance example. After the first instance

(−,short,blond,brown), the decision tree is:

$$\{ - \quad short \quad blond \quad brown \}$$

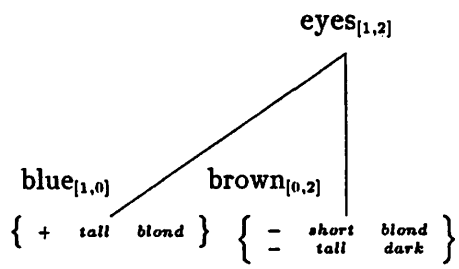
Note that this is not yet a tree. ID5 only builds a tree at nodes for which an attribute test is needed. When all instances have the same classification, a test attribute is not needed, and the decision is indicated by the instance classification, in this case "−".

After the second instance (−,tall,dark,brown), the decision tree is:

$$\left\{ \begin{array}{cccc} - & short & blond & brown \\ - & tall & dark & brown \end{array} \right\}$$

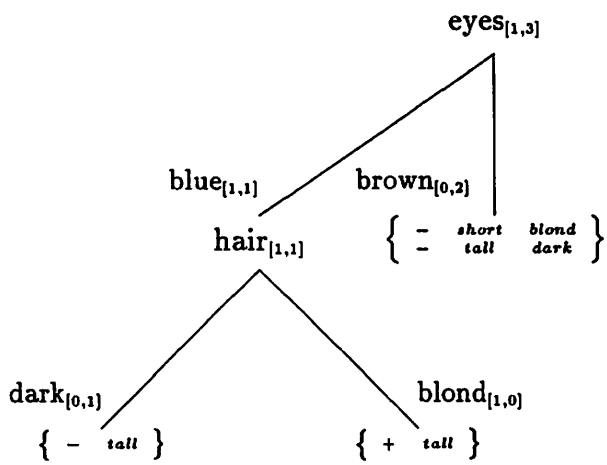
There was still no need for a test attribute, so the instance is added to the list of instances at the root of the tree.

The third instance (+,tall,blond,blue) is positive, so a tree is formed:



This is the same tree that ID4 had at this point.

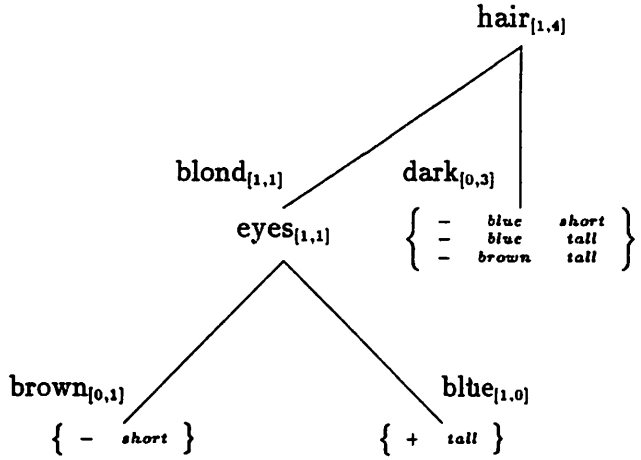
The fourth instance (−,tall,dark,blue) leads to:



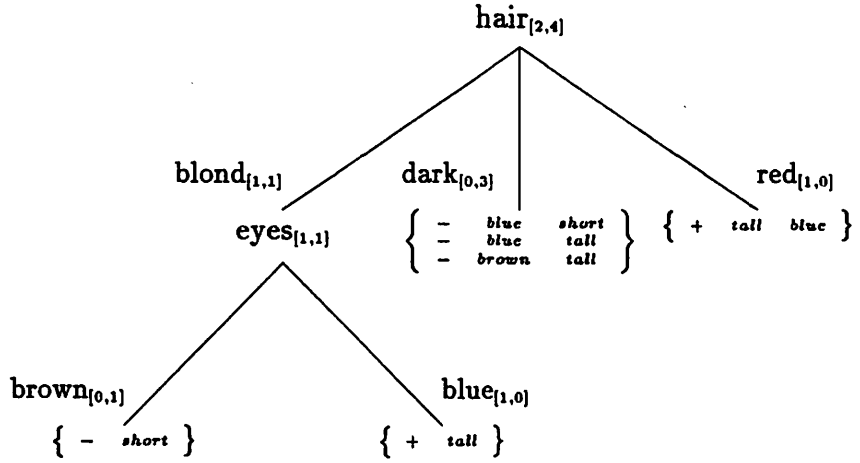
Note that the tree has "hair" as the test attribute below "blue". Because ID5 retained the

instances for this subtree, it was able to determine that "hair" is the best test attribute.

The fifth instance (−,short,dark,blue) causes the tree to be restructured,[4] producing:



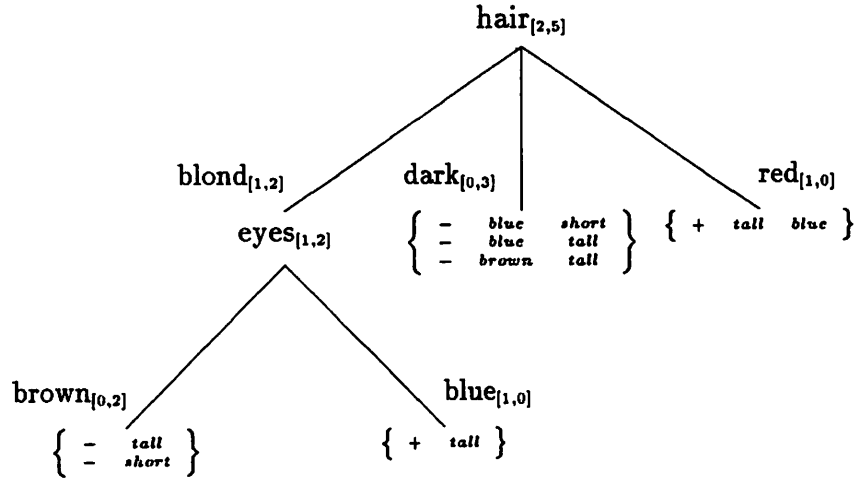
The sixth instance (+,tall,red,blue) updates the tree to:



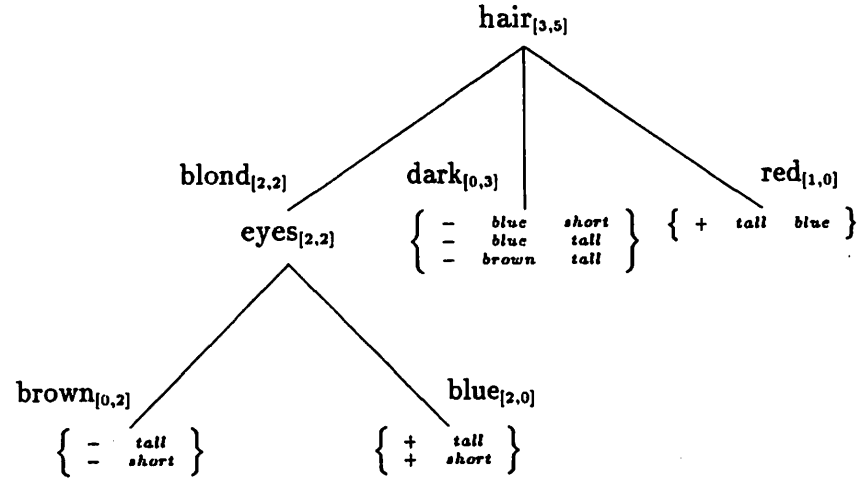
At this point, the decision tree happens to classify all eight instances correctly.

Continuing with the seventh instance (−,tall,blond,brown) updates the tree to:

[4]This is the case that was used above to illustrate the pull-up process



The final instance (+,short,blond,blue) updates the tree to:



For this example, ID5 built the same decision tree that ID3 built. To guarantee that ID3 and ID5 will always build the same tree, two changes would be required. First, as mentioned above, it would be necessary to re-establish the best test attribute for each subtree below a test attribute that was just pulled up. Second, it would be necessary to insist that ties for equally good test attributes be resolved identically. In practice, it would be a waste of time to make such an imposition, because the ID3 algorithm does not require a tie-breaking algorithm.

5. Analysis

As with any induction algorithm, one needs to consider how much training is required to achieve a specific level of performance, and how costly that training is. One measurement of training effort is the total number of instances that must be examined or re-examined. Coupled with the number of instances examined is the cost of examining each instance. These are considerations for all of ID3, ID4, and ID5. Further considerations for ID4, and

ID5 are the expected frequency of revising a test attribute and the associated cost of such a revision, both in terms of execution cost and in terms of impact on the need for further training.

5.1 ID3

For ID3, unless all instances are of the same classification, all $|A|$ attributes of all $|I|$ instances are examined. For each subtree y of A_{best}, $|I_{best,y}|$ instances are examined. Because the subtrees are derived from a partition of I, as many as $|I|$ are examined on the second level and each successive level of the tree. Less than $|I|$ instances are examined when a subtree terminates with a "+" or a "−". Hence, the total number of instances examined is:

$$O(|I| \cdot |A|)$$

For each instance on level one, $|A|$ attributes are examined. For each instance on level two, $|A| - 1$ attributes are examined. Thus, the total number of attribute examinations is:

$$O(|I| \cdot |A|^2)$$

If ID3 is used to build a new decision tree after each observed training instance,[5] the total number of attribute examinations is:

$$\sum_{i=1}^{|I|} O(i \cdot |A|^2) = O(|I|^2 \cdot |A|^2)$$

5.2 ID4

For ID4, each instance is examined once at the root, and once at each lower level of the tree. As with ID3, lower levels will exist only where an attribute test is necessary. Still, the number of times the instance is examined is $O(|A|)$. To examine all $|I|$ instances, the total number of instance examinations is:

$$O(|I| \cdot |A|)$$

As with ID3, the number of attributes examined for each instance is $|A|$ on level 1, $|A| - 1$ on level 2, down to 1 on level $|A|$. Thus, for each instance examined, $O(|A|^2)$ attributes are examined. For all $|I|$ instances, the total number of attribute examinations is:

$$O(|I| \cdot |A|^2)$$

A single pass over the instances does not necessarily cause an ID3 equivalent tree to be constructed. One needs to ask how many total instances, or passes over the given set of instances, are needed to achieve the same result as ID3. Schlimmer and Fischer indicate that each pass over the instances should stabilize the next level of the tree. Thus, one could need as many as $|A|$ passes over the instances. The total number of instances examined would be:

$$O(|I| \cdot |A|^2)$$

[5] i.e. a crude incremental algorithm

and the total number of attributes examined would be:

$$O(|I| \cdot |A|^3)$$

This is usually better than building a new tree with ID3 after each instance, but usually worse than using ID3 to build a tree from all $|I|$ instances at once.

It is important to consider the expense of discarding subtrees when ID4 chooses a new test attribute. How much training effort is lost with such a strategy? Consider revision of the test attribute at the root of the decision tree. All subtrees would be discarded. This is considered acceptable because the first pass through the instances will result in the same test attribute that ID3 would pick. The same argument would apply at each subtree. However, there is an assumption here that the test attribute will remain the best choice after every instance in I that ID4 will see. If that is not the case, then ID4 will not stabilize because it will repeatedly revise the test attribute. For example, ID4 will not stabilize on the following set of instances:

class	a	b	c
+	0	0	0
-	0	0	1
-	0	1	0
+	0	1	1
-	1	0	0
+	1	0	1
+	1	1	0
-	1	1	1

5.3 ID5

For ID5, the number of times an instance is examined is $O(|A|)$, as with ID4. Similarly, to examine all $|I|$ instances, the total number of instance examinations is:

$$O(|I| \cdot |A|)$$

As with ID3 and ID4, for each instance examined, $O(|A|)$ attributes are examined. For all $|I|$ instances, the total number of attribute examinations is:

$$O(|I| \cdot |A|^2)$$

One pass over the instances is sufficient to build a tree that correctly classifies all the instances. However, the cost of revising a test attribute and the expected frequency of revision must be taken into account. The expense of revision is the sum of the cost of the revisions of each subtree[6] plus the cost of the pull-up step. The pull-up step itself does not depend on the depth of the tree. As the tree is restructured, there are m_{old} values

[6] to get the test attribute to the root of each subtree

of the old test attribute and m_{new} values of the new test attribute. Hence, there will be $m_{old} \cdot m_{new}$ merges of subtrees to compute the revised tree. Each merge causes at most $|A|$ attribute merges. Thus, the expected number of attributes examined for a test attribute revision is $O(m_{old} \cdot m_{new} \cdot |A|)$. Let r be the expected number of revisions. Then the overall expected number of attributes examined is:

$$O(|I| \cdot |A|^2 + r \cdot m_{old} \cdot m_{new} \cdot |A|)$$

This estimate does not take into account the effect of depth on the cost of a revision. This is because experience has shown that the attribute to be pulled up tends to be close to the root already, and because the pull-up step itself does not depend on the depth of the tree.

It is difficult to say how often a revision can be expected. This depends on how strongly the attributes are associated with the classification. It should be noted that ID5 and ID4 revise a test attribute in the exact same cases. The difference is on how each algorithm carries out the revision. The cost of tree revision for ID4 is small in the short-term, because discarding subtrees is easy, but large in the long-term because so much additional training effort is needed. For ID5, the cost of tree revision can be small or large in the short-term (depending on the number of values the new and old test attributes can take on), but small in the long-term because training effort need not be repeated.

6. Experiments

The three algorithms were implemented and performance was compared empirically. For ID3, a crude incremental version was used; after each training instance, build a new decision tree from the set of all instances observed thus far. Two experiments were performed, one on a multiplexor task, the other on a preference predicate task. As in Schlimmer & Fisher, versions in which only necessary training[7] is performed were also tested. Here, these versions are called $\widehat{ID3}$, $\widehat{ID4}$, and $\widehat{ID5}$. The objective was to measure expense and performance of each algorithm so that they could be compared. Each experiment was conducted automatically according to the following algorithm:

1. Set *Time* to 0. Set *Events* to 0. Set *Dtree* to "?". Set *Proportion* to 0%.
2. If *Proportion* is 100% or *Events* $= |I| \cdot |A|$ then stop.
3. Increment *Events*.
4. Set decision tree *Dtree* to the result of training on the next instance from I. The instances I are considered to be a circular list. The first instance in the list I follows the last instance.
5. Add to *Time* the cpu seconds taken in step 4.
6. Set *Proportion* to the proportion of instance set I correctly classified by current decision tree *Dtree*.

[7] i.e. only if the current decision tree misclassifies the training instance

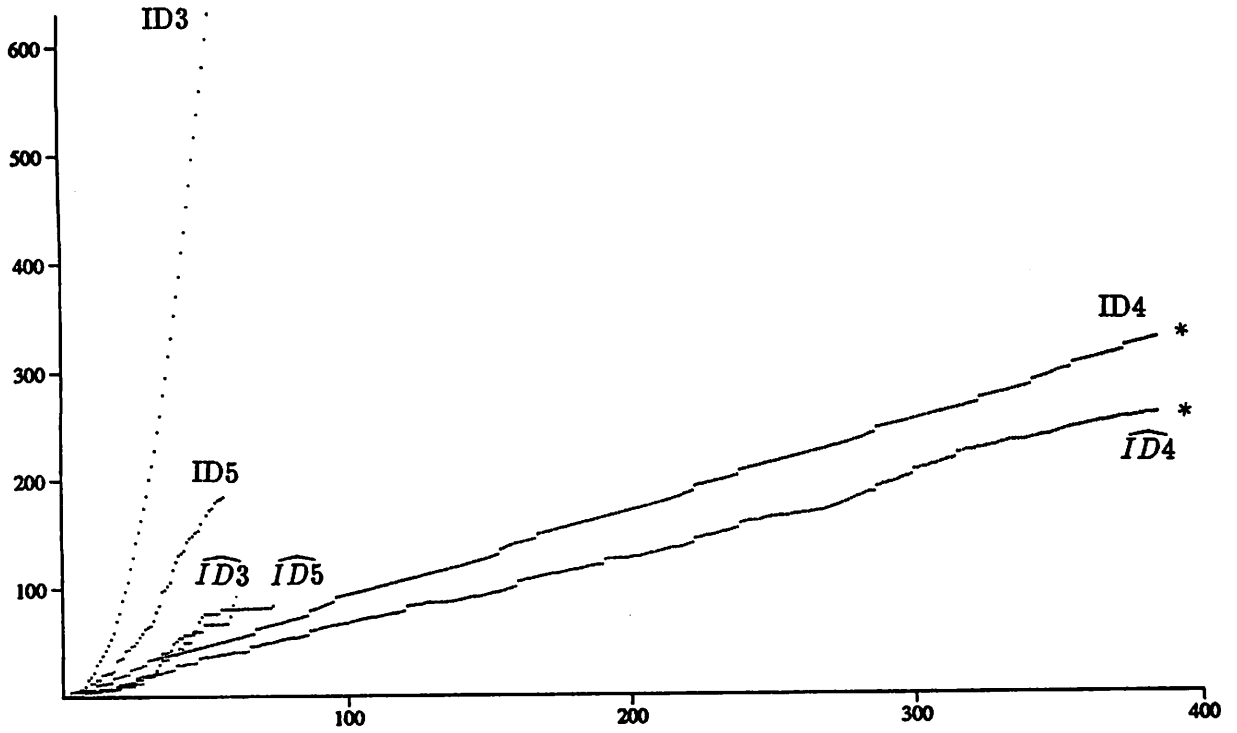


Figure 1: Cost (y axis) vs. Events (x axis)

7. Update each of three x-y plots.

   (a) *Time* vs. *Events*

   (b) *Proportion* vs. *Events*

   (c) *Proportion* vs. *Time*

8. goto 2

6.1 Multiplexor

For this experiment, the database I consisted of 64 instances from a commonly studied multiplexor task. Each instance is described by 6 binary-valued attributes. Two of the attributes, the address bits, select one of the four remaining attributes, the data bits. The classification of the instance is equal to the value of the selected attribute. This task is a difficult one for learning programs because no single attribute accounts for the classification.[8]

Figure 1 shows the cumulative training cost (cpu time) versus the number of training instances (recall that the list of 64 instances is treated as a circular list) for all six algorithms. The final values for *Events*, *Time*, and *Proportion* for the six algorithms were:

[8]Indeed, the ID3 algorithm finds a suboptimal decision tree for this learning task.

Algorithm	Events	Time	Proportion
ID3	53	630.4	100
$\widehat{ID3}$	61	92.3	100
ID4	384	327.7	63 *
$\widehat{ID4}$	384	258.1	50 *
ID5	57	184.3	100
$\widehat{ID5}$	74	83.5	100

ID5 and $\widehat{ID5}$ required more training instances than ID3 and $\widehat{ID3}$ respectively because ID5 does not re-establish the best test attribute for each subtree below a new test attribute.[9] Instead, the algorithm waits for subsequent instances to cause subtrees to be visited, whereupon the best test will be re-estabblished.[10]

Figures 2 through 7 show the proportion of the 64 instances classified correctly versus the number of training events for each algorithm. In terms of proportion of instances classified correctly, ID3 and $\widehat{ID3}$ achieve the best performance earliest. This is because ID5 and $\widehat{ID5}$ lag in being ID3 equivalent. Note that ID4 and $\widehat{ID4}$ do not stabilize. This is indicated by a "*" in the figure and in the table above. Figure 4 shows a repeating pattern in classification performance, indicative of the repeating versions of the decision tree that occurred.

Figures 8 through 13 show the proportion versus the cumulative training cost. This is the same data in the y-axis as figures 2 through 7. The only difference is that the x-axis shows cumulative training cost instead of cumulative number of training events. The figures give an indication of classification performance per cpu second spent.

6.2 Preference Predicate

For this experiment, the database I contained 100 instances of pairs of feature vectors describing pairs of Othello boards. The 7 features from each pair combine to make 14 integer-valued attributes. Most of the attributes can take on approximately 30 different values. An instance is labelled positive when the first board is provably better than the second board and negative otherwise (Utgoff and Saxena, 1987; Utgoff and Heitman, 1988).

Figure 14 shows the cumulative training cost (cpu time) versus the number of training instances for all six algorithms. The final values for *Events*, *Time*, and *Proportion* for the six algorithms were:

[9]This is known to be the case because, for the experiment, ties for best test attribute were broken identically for ID3, $\widehat{ID3}$, ID5, and $\widehat{ID5}$.

[10]In the implementation, recursive re-establishment of best test attribute at each subtree following a pull-up is toggled by a switch, which is left turned off. It is cheaper to make the call only when an ID3 equivalent tree must be guaranteed.

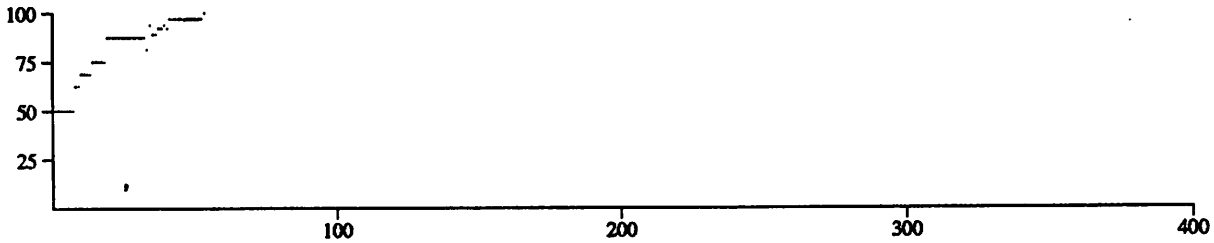


Figure 2: Proportion (y axis) vs. Events (x axis) for ID3

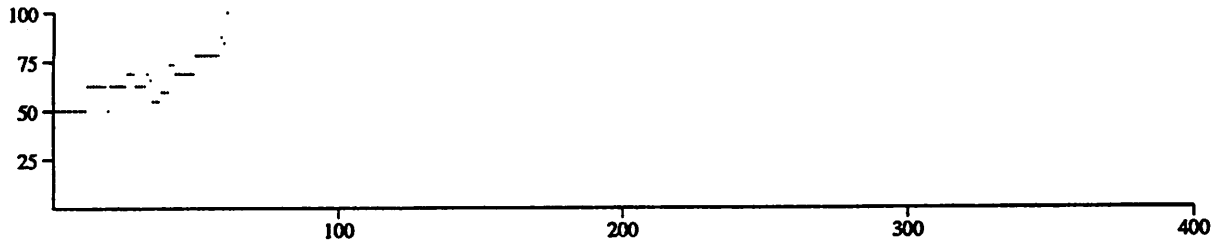


Figure 3: Proportion (y axis) vs. Events (x axis) for $\widehat{ID3}$

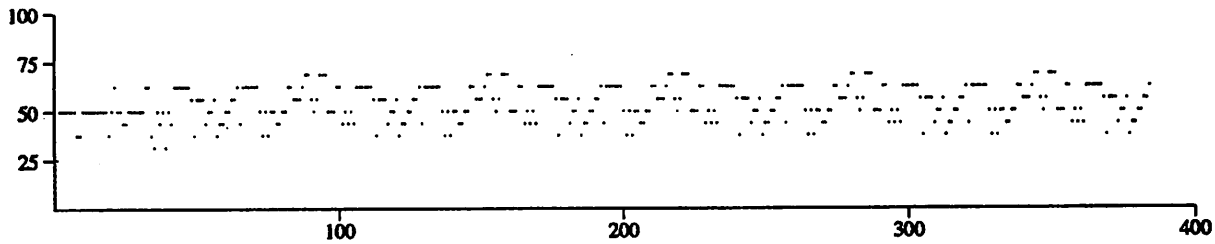


Figure 4: Proportion (y axis) vs. Events (x axis) for ID4

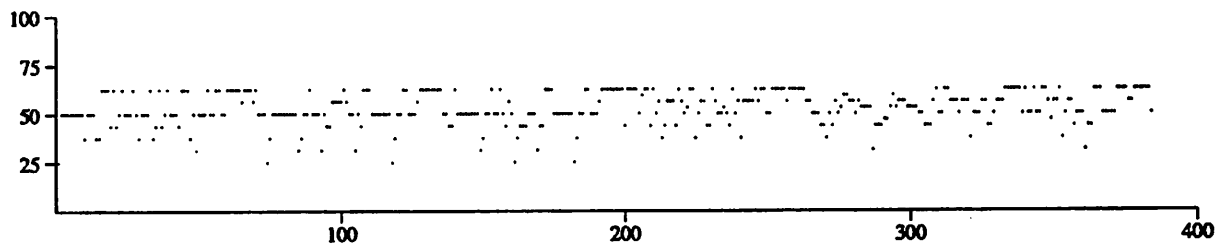


Figure 5: Proportion (y axis) vs. Events (x axis) for $\widehat{ID4}$

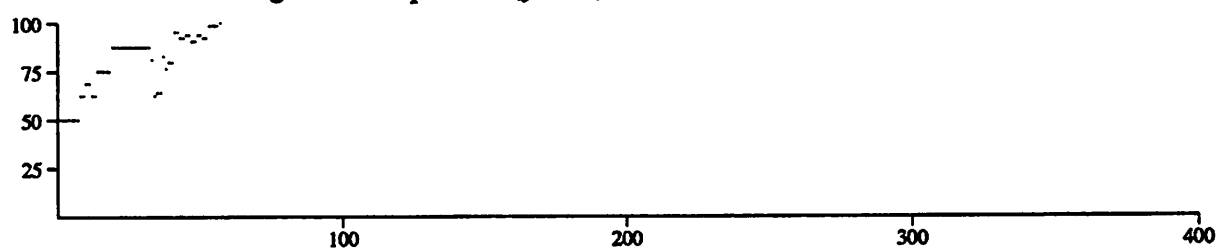


Figure 6: Proportion (y axis) vs. Events (x axis) for ID5

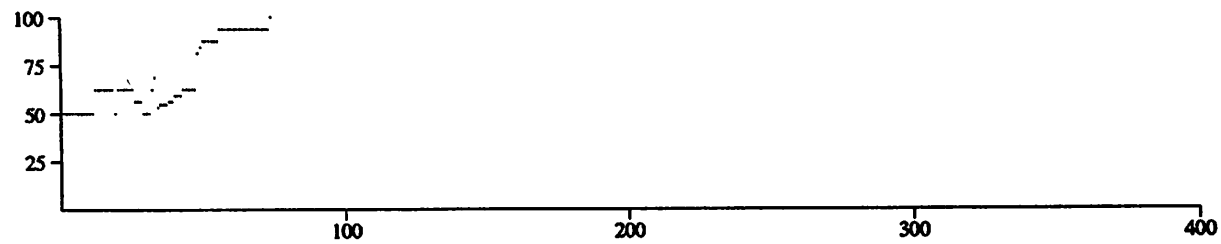


Figure 7: Proportion (y axis) vs. Events (x axis) for $\widehat{ID5}$

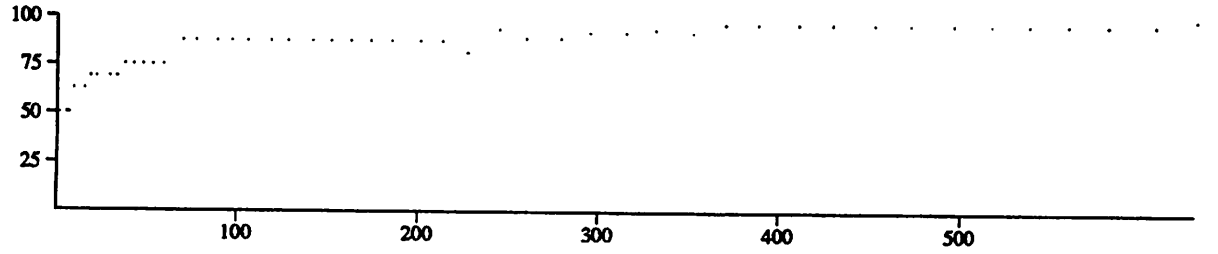


Figure 8: Proportion (y axis) vs. Cost (x axis) for ID3

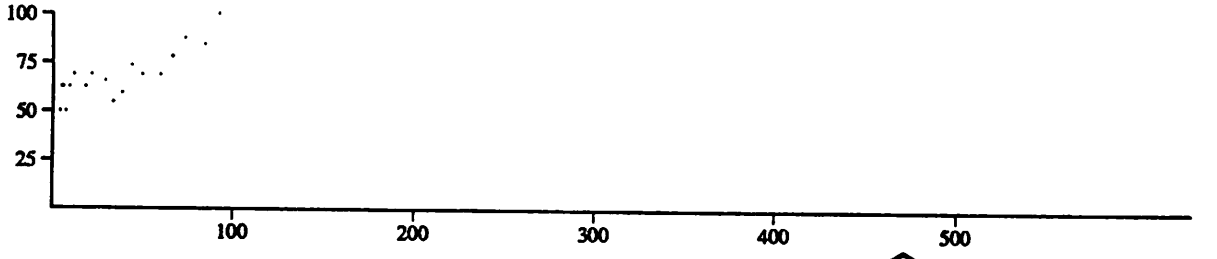


Figure 9: Proportion (y axis) vs. Cost (x axis) for $\widehat{ID3}$

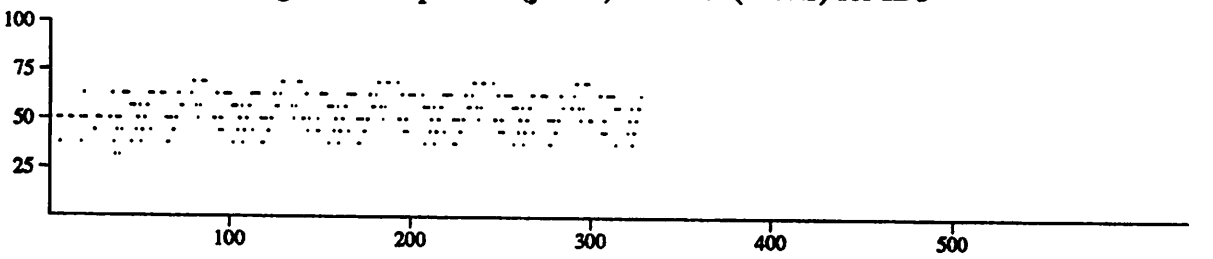


Figure 10: Proportion (y axis) vs. Cost (x axis) for ID4

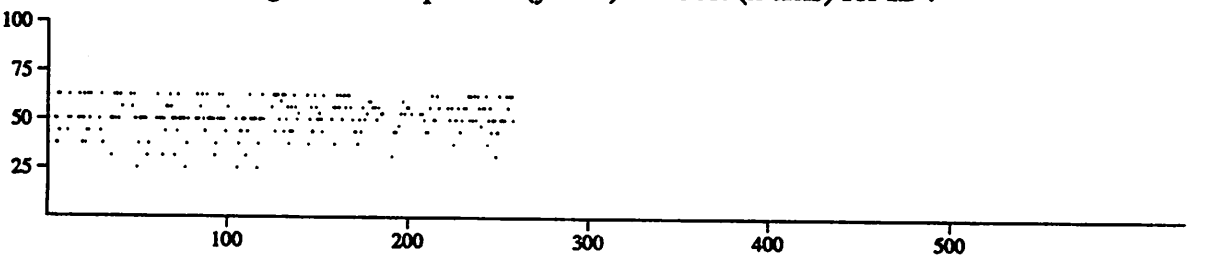


Figure 11: Proportion (y axis) vs. Cost (x axis) for $\widehat{ID4}$

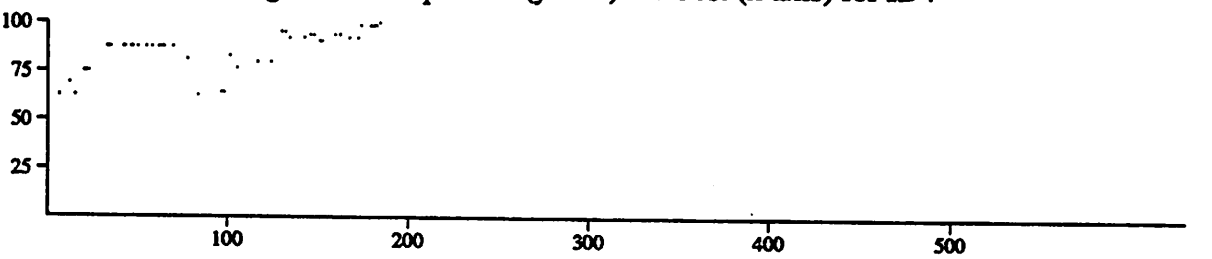


Figure 12: Proportion (y axis) vs. Cost (x axis) for ID5

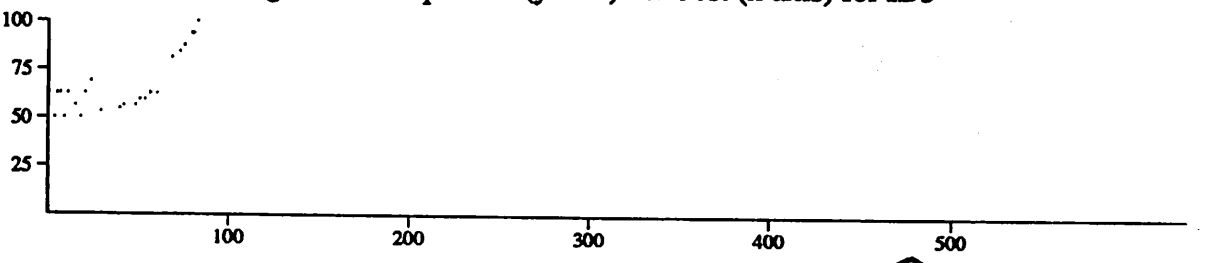


Figure 13: Proportion (y axis) vs. Cost (x axis) for $\widehat{ID5}$

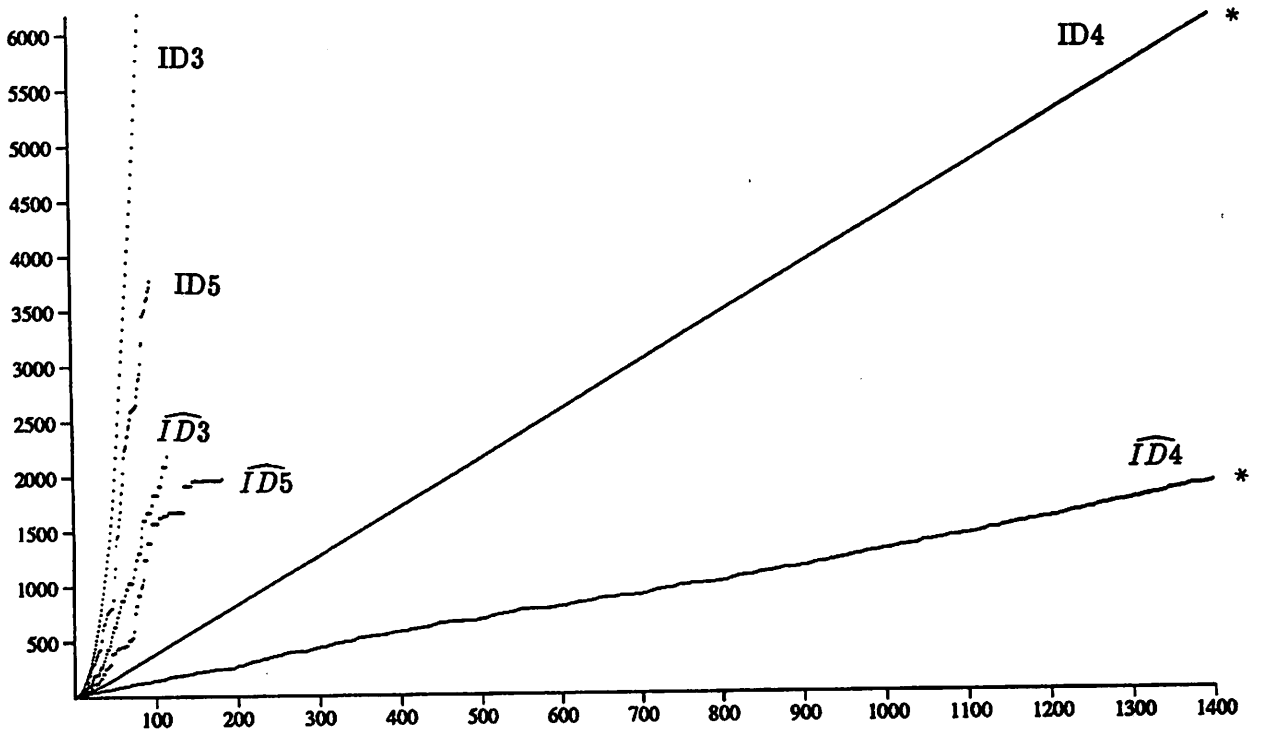


Figure 14: Cost (y axis) vs. Events (x axis)

Algorithm	Events	Time	Proportion
ID3	88	6178.4	100
$\widehat{ID3}$	117	2180.3	100
ID4	1400	6084.8	74 *
$\widehat{ID4}$	1400	1874.3	62 *
ID5	98	3769.3	100
$\widehat{ID5}$	184	1974.4	100

Figures 15 through 20 show the proportion of the 100 instances classified correctly versus the number of training events for each algorithm. As with the multiplexor task, in terms of proportion of instances classified correctly, ID3 and $\widehat{ID3}$ achieve the best performance earliest. Note that ID4 and $\widehat{ID4}$ do not stabilize.

Figures 21 through 26 show the proportion versus the cumulative training cost.

7. Conclusion

ID5 provides an incremental method for building ID3 type decision trees. The cost of using ID5 is less than the cost of using ID3 to build a new decision tree after each training instance, unless tree revisions become too frequent or the attributes being exchanged have a large number of values. ID5 offers an alternative to ID3 and ID4, and the analysis gives guidance on which one to pick for a given learning task.

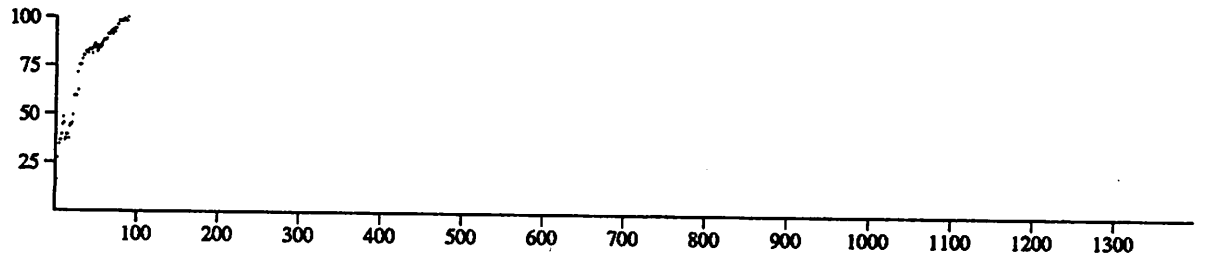


Figure 15: Proportion (y axis) vs. Events (x axis) for ID3

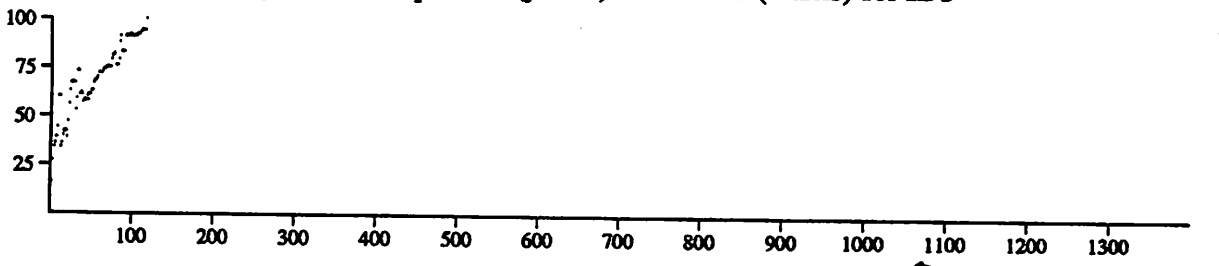


Figure 16: Proportion (y axis) vs. Events (x axis) for $\widehat{ID3}$

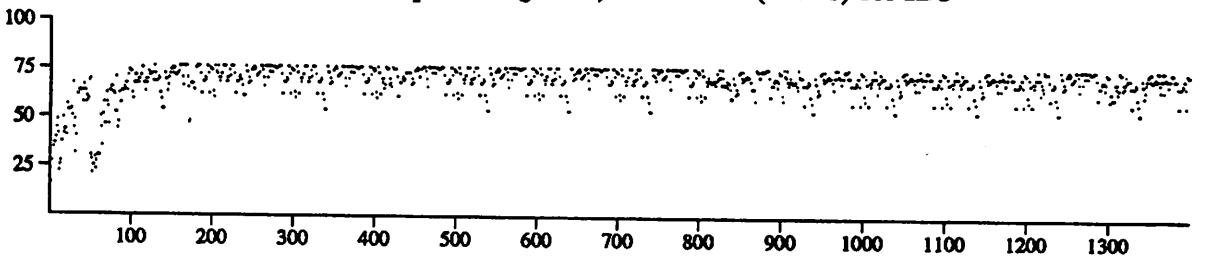


Figure 17: Proportion (y axis) vs. Events (x axis) for ID4

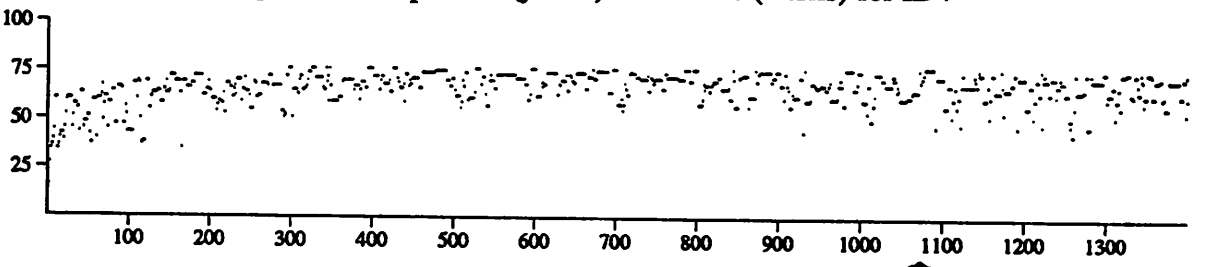


Figure 18: Proportion (y axis) vs. Events (x axis) for $\widehat{ID4}$

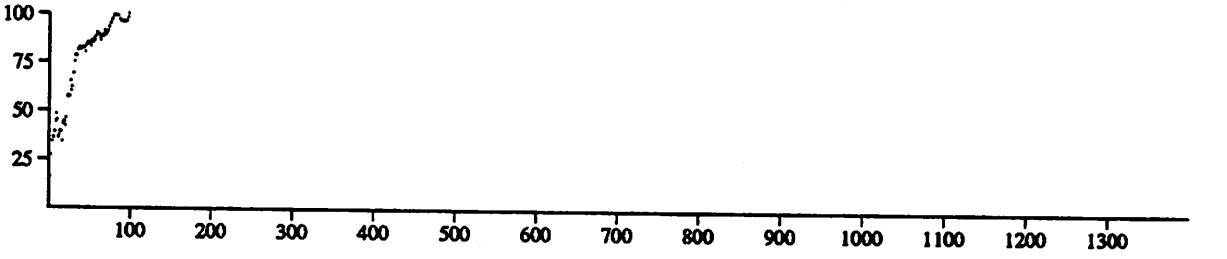


Figure 19: Proportion (y axis) vs. Events (x axis) for ID5

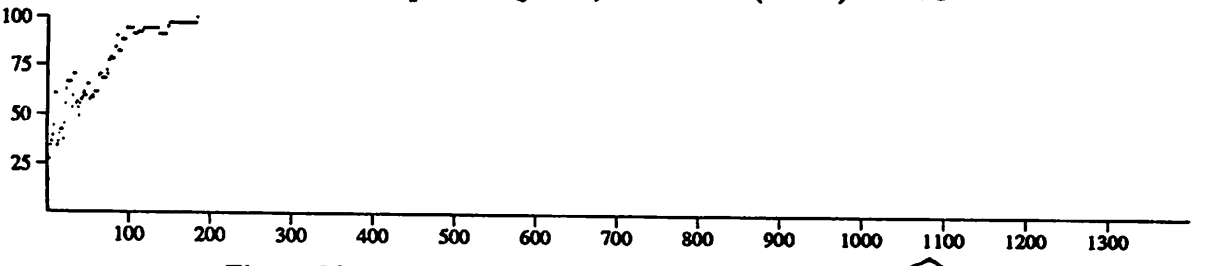


Figure 20: Proportion (y axis) vs. Events (x axis) for $\widehat{ID5}$

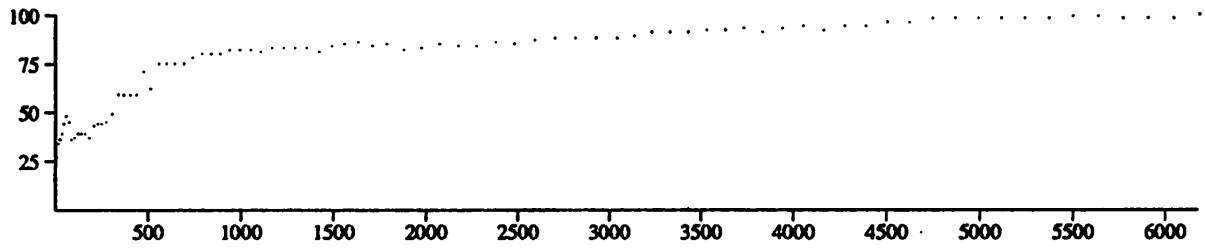


Figure 21: Proportion (y axis) vs. Cost (x axis) for ID3

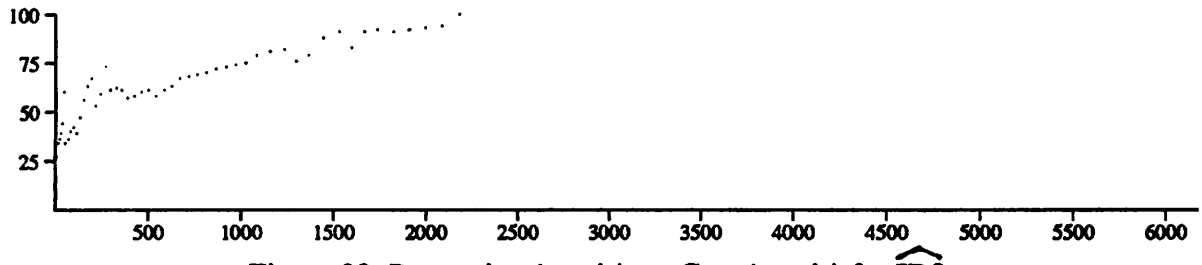


Figure 22: Proportion (y axis) vs. Cost (x axis) for $\widehat{\text{ID3}}$

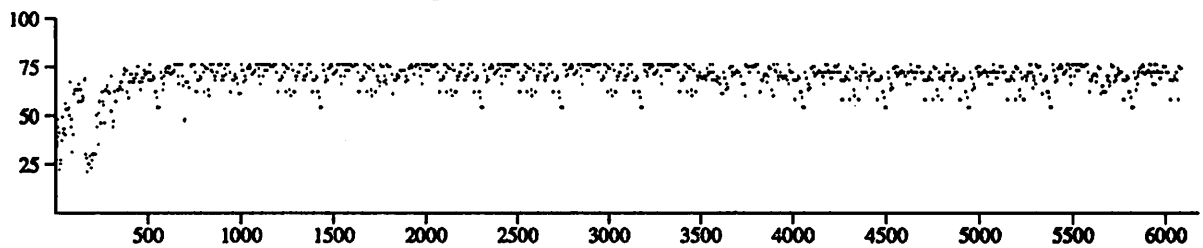


Figure 23: Proportion (y axis) vs. Cost (x axis) for ID4

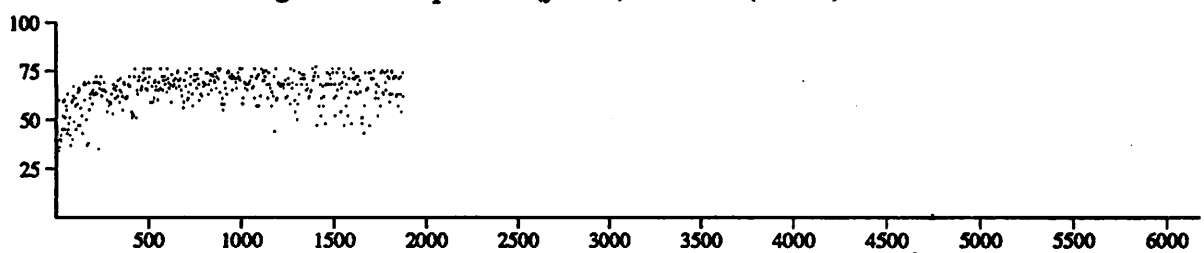


Figure 24: Proportion (y axis) vs. Cost (x axis) for $\widehat{\text{ID4}}$

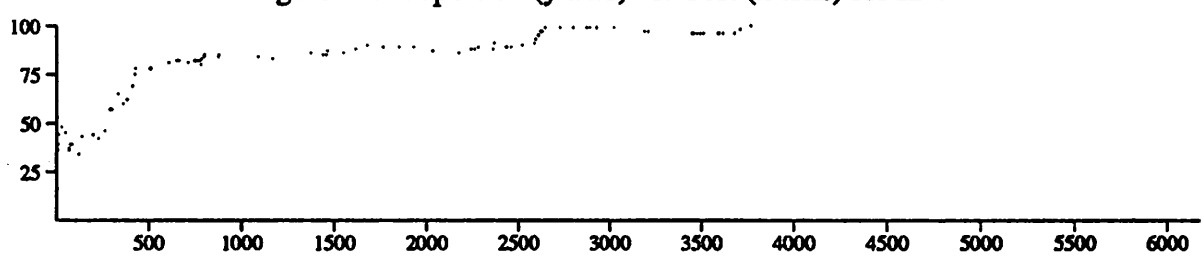


Figure 25: Proportion (y axis) vs. Cost (x axis) for ID5

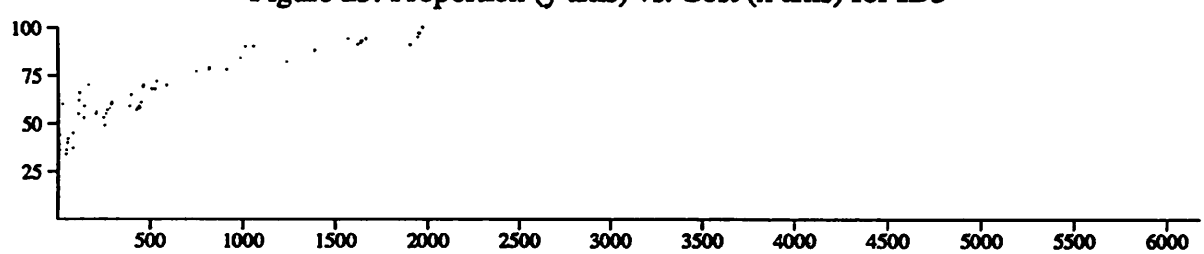


Figure 26: Proportion (y axis) vs. Cost (x axis) for $\widehat{\text{ID5}}$

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IRI-8619107 and by a General Electric Faculty Fellowship. The several discussions with Jeff Schlimmer on ID3 and ID4 were very helpful. The presentation benefitted from helpful comments and criticisms from Sharad Saxena, Margie Connell, Jamie Callan, Peter Heitman, Kishore Swaminathan, and Dan Suthers.

References

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA.

Buchanan, B. G. and Mitchell, T. M. (1978) Model-directed learning of production rules. *Pattern-Directed Inference Systems*, Waterman, D. A. and Hayes-Roth, F. (Eds.), Academic Press, New York.

Michalski, R. S. and Chilausky, R. L. (1980), Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Policy Analysis and Information Systems 4(2).* (Special issue on knowledge acquisition and induction).

Mitchell, T. M. (1978) Version spaces: an approach to concept learning, Ph.D. dissertation, Stanford University, (also Stanford CS report STAN-CS-78-711, HPP-79-2).

Mitchell, T. M., Utgoff, P. E. and Banerji, R. B. (1983) Learning by experimentation: acquiring and refining problem-solving heuristics. *Machine Learning*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.), Morgan-Kaufman.

Moret, B. M. E. (1982). Decision trees and diagrams. *Computing Surveys, 14*, 593-623.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In Michalski, Carbonell, & Mitchell (Eds.) *Machine Learning: An artificial intelligence approach*, Morgan Kaufmann, 463-482.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning 1(1)*, 81-106, Kluwer.

Schlimmer, J. C. & Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496-501). Morgan Kaufmann.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379-423.

Utgoff, P. E. and Saxena, S. (1987). Learning a preference predicate. *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufman, pp. 115-121.

Utgoff, P. E. and Heitman, P. S. (1988) Learning and generalizing move selection preferences, *Proceedings of the AAAI Symposium on Game-Playing*, Morgan Kaufman. (to appear)

Vere, S. A. (1980), Multilevel counterfactuals for generalizations of relational concepts and productions, *Artificial Intelligence 14(2)*, pp. 138-164.

Winston, P. H. (1975) Learning structural descriptions from examples. *The Psychology of Computer Vision*, Winston, P. H. (Ed.), McGraw-Hill, New York.

Appendix: The INFO Function

The INFO function is defined (refer to notation on page 2):

$$INFO(A_x) = \sum_{i=1}^{m_x} \frac{P(V_{x,i}) + N(V_{x,i})}{P(A_x) + N(A_x)} M(V_{x,i})$$

The M function is defined:

$$M(V_{x,y}) = \left[\begin{array}{l} 0 \; if \; P(V_{x,y}) = 0 \\ 0 \; if \; N(V_{x,y}) = 0 \\ -u \log_2 u - v \log_2 v \; otherwise, \; with \; \begin{array}{l} u = \frac{P(V_{x,y})}{P(V_{x,y}) + N(V_{x,y})} \\ v = \frac{N(V_{x,y})}{P(V_{x,y}) + N(V_{x,y})} \end{array} \end{array} \right]$$