TO APPEAR IN THE PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. ANN ARBOR, MI (JUNE 1988).

Improved Decision Trees: A Generalized Version of ID3[†]

Jie Cheng
Usama M. Fayyad
Keki B. Irani
Zhaogang Qian
Department of Electrical Engineering and Computer Science
The University of Michigan

Please direct all correspondence to:
Prof. Keki B. Irani
Computer Science and Engineering
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
(313) 747-3820
INTERNET: Irani@um.cc.umich.edu

Abstract

A popular and particularly efficient method for inducing classification rules from examples is Quinlan's ID3 algorithm. This paper examines the problem of overspecialization in ID3. Two causes of overspecialization in ID3 are identified. An algorithm that avoids some of the inherent problems in ID3 is developed. The new algorithm, GID3, is applied to the development of an expert system for automating the Reactive Ion Etching (RIE) process in semiconductor manufacturing. Six performance measures for decision trees are defined. The GID3 algorithm is empirically shown to outperform ID3 on all performance measures considered. The improvement is gained with negligible increase in computational complexity.

Keywords: Concept Learning, Empirical Techniques, Empirical Evaluation.

[†] This work is supported in part by SRC under contract 87-MP-085, and in part by the AFOSR under contract F49620-82-C0089

Improved Decision Trees: A Generalized Version of ID3

Jie Cheng (JC@eecs.umich.edu)
Usama M. Fayyad (Fayyad@ub.cc.umich.edu)
Keki B. Irani (Irani@caen.engin.umich.edu)
Zhaogang Qian (ZQ@eecs.umich.edu)

Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122, U.S.A.

Abstract

A popular and particularly efficient method for inducing classification rules from examples is Quinlan's ID3 algorithm. This paper examines the problem of overspecialization in ID3. Two causes of overspecialization in ID3 are identified. An algorithm that avoids some of the inherent problems in ID3 is developed. The new algorithm, GID3, is applied to the development of an expert system for automating the Reactive Ion Etching (RIE) process in semiconductor manufacturing. Six performance measures for decision trees are defined. The GID3 algorithm is empirically shown to outperform ID3 on all performance measures considered. The improvement is gained with negligible increase in computational complexity.

1 Introduction

Automated knowledge acquisition is one of the primary areas of application for concept learning techniques[Quin86]. Some steady and promising advances towards the direction of autonomous knowledge acquisition have taken place[Mich78,Mich79,Moze86,Quin86]. In this paper, we focus on Quinlan's ID3 approach[Quin83]. Some of the problems of ID3, and their causes, are identified. We present a novel approach designed to avoid these problems without imposing any significant increase in computation. Finally, the new approach is empirically verified to be superior to ID3.

A Program that learns from examples is presented with examples of different concepts (classes). For each concept, the program constructs a description that covers all of its examples and excludes all others. The ideal goal of such a learning program is to construct the minimal set of maximally general concept descriptions. This, in general, is an NP-complete problem[Haus87], which makes a heuristic approach necessary.

2 The ID3 Approach

A particularly efficient method for exploring the space of concept descriptions (expressions) is to generate a decision tree[Hunt66,Quin83,Quin86].

In the case of ID3, each branch in the tree corresponds to an expression of the form: $(A_i = V_{ij})$, where V_{ij} is the j-th value of attribute A_i. Thus for each (non-leaf) node, ID3 chooses an attribute, and creates a branch *for each value* of that attribute appearing in the subset of examples corresponding to the node. ID3 is applied recursively to each non-leaf node. A set of examples along with the decision tree generated by ID3 are shown in figure 1.

An algorithm for decision-tree generation requires the specification of the following:

1. A rule for choosing an attribute to use in partitioning the examples at a node.
2. A rule for choosing a particular partition of the examples.
3. A rule for deciding when to stop partitioning a node, thus deeming it a leaf node.
4. A rule for assigning a leaf to a class.

	A	B	Class
e-1	a1	b1	C1
e-2	a1	b2	C2
e-3	a2	b2	C2
e-4	a2	b3	C1
e-5	a3	b1	C3
e-6	a3	b2	C3

Two test examples not in the training set:
e : (A = a3), (B = b3)
e': (A = a1), (B = b3)

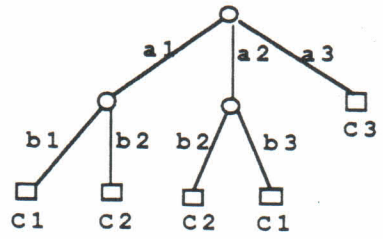

Figure 1. Sample decision tree generated by ID3

In this paper we are concerned with the details of rules 1 and 2. In the case of ID3, rule 2 is realized by creating a subset for each value of the attribute chosen in 1. We now focus our attention on the details of rule 1 in ID3. For a detailed simplified discussion of this algorithm, readers are referred to [Quin86]. Let A be a set of m attributes $\{A_1, A_2, \ldots, A_m\}$, let C be a set of p classes $\{C_1, C_2, \ldots, C_p\}$, and let S be a set of examples at some node. The set of possible values for an attribute A_i is referred to as $Range(A_i)$. Each example in S is an $m+1$-tuple of the form: $\langle V_1, V_2, \ldots V_m, C_k \rangle$, where $V_i \in Range(A_i), i = 1, \ldots, m$, and $C_k \in C$ is the class of the example.

Define P_{S,C_k}, the probability of occurence of each class $C_k \in C$ in a set S of examples, to be the proportion of examples in S that are in class C_k. As a measure of the randomness of example distribution in a set S over the possible classes in C, the *information measure* is defined to be:

$$I(S) = -\sum_{k=1}^{p} P_{S,C_k} \log_2 P_{S,C_k}.$$

ID3 tries to choose a partition of S that results in subsets in which the examples are distributed "less randomly" over the possible classes. To choose the attribute that would best achieve this, for each attribute A_i that takes more than one value for the examples in S, ID3 partitions S into the sets S_j consisting of all examples in S having value V_j for attribute A_i: $S_j = \{e \in S | A_i = V_j \text{ for } e\}$. The information *entropy* of the resulting partition is given by:

$$E(A_i, S) = \sum_{V_j \in Range(A_i)} \frac{|S_j|}{|S|} I(S_j) \tag{1}$$

For further partitioning of a node S in the tree, ID3 chooses the attribute A_i which maximizes:

$$Gain(A_i, S) = I(S) - E(A_i, S) \tag{2}$$

3 Problems with the ID3 Approach

ID3 is essentially employing a heuristic, hill-climbing, non-backtracking search through the space of possible decision trees. Thus, weaknesses in the ID3 algorithm may cause it to "miss" better decision trees for the same data. By "better" we mean "more general". In section 5.1, we quantify this notion by defining measures of generality for decision trees. This section addresses some of the problems *inherent* in the ID3 approach that cause *overspecialization* to occur. In [Quin87], Quinlan deals with the issue of reducing the effect of overspecialization in decision trees by *retrospectively pruning* a tree or its corresponding set of rules. On the other hand, we aim at avoiding overspecialization during the generation of a decision tree by improving the search heuristic.

Perhaps the most pronounced of the problems is the **irrelevant values problem.** When ID3 chooses an attribute for branching out of a node, it creates a branch for each value of that attribute

that appears in the examples. Some of the values of that attribute may be relevant to the classification, yet the rest may not be. The subtrees generated by such irrelevant values will result in overspecialized classification rules—rules that check for unnecessary or irrelevant preconditions.

Another related problem is the **missing branches problem.** The missing branches problem is essentially a reduction in the inductive capacity of the tree. It is due to the fact that some of the reduced subsets at the non-leaf nodes do not necessarily contain examples of every possible value of the branching attribute. Figure 1 shows an example illustrating the missing branches problem. Both e and e' are examples *never seen* by the program. Yet the ID3 tree shown classifies e and fails to classify e'. If a high degree of induction is desirable, this problem should be avoided whenever possible. Section 4.2 illustrates how the proposed approach reduces the likelihood of occurence of missing branches.

Another problem with the ID3 heuristic is that the formula of equation (2) was shown to be heavily biased in favour of attributes with larger ranges of values[Quin86]. In [Quin86] Quinlan introduced a new measure to compensate for this bias. For an attribute $A_i \in A$, and a set S of examples partitioned into the subsets S_j consisting of all examples in S that have value V_j for the attribute A_i,

$$IV(A_i) = -\sum_{V_j \in Range(A_i)} \frac{|S_j|}{|S|} \log_2(\frac{|S_j|}{|S|}).$$

$IV(A_i)$ measures the degree of randomness of the distribution of the examples in S over the values of A_i. Note that it does not take into account what the classes of these examples are. The gain formula of equation (2) is modified to be $Gain(A_i, S) = \frac{I(S)-E(A_i,S)}{IV(A_i)}$. We refer to this version of ID3 as ID3-IV.

4 An Alternate Approach

4.1 The Proposed Approach

In order to avoid the irrelevant values problem, we have modified the ID3 algorithm described above. For a given subset of examples, we choose an attribute to be used in inducing a partition on that set, but we do not necessarily branch on every value of that attribute. We have therefore deviated from ID3 in the specifications of rules 1 and 2 of section 2. For a set S of examples at a node, the algorithm shown on the following page is used to determine which attribute to use for a partition (rule 1), and how to use it to induce the partiton (rule 2).

Essentially, the algorithm is similar to ID3 except that not every value of the attribute is chosen to create a branch. In order to avoid branching on irrelevant values of the attribute, only the values that appear to be relevant, according to the information measure, may potentially be branched on. All other values are lumped together in one default value for the attribute. The tolerance level (TL) is user determined; we have not yet determined how to set it systematically. TL specifies the degree of tolerance for deviation of the entropy measure of an attribute-value pair from the minimal entropy measure over all pairs (see algorithm on next page for the definitions of TL and the set of *phantom attributes* AP). Attribute values that fall outside this tolerance range are treated as a single "default" value. Note that if all values of an attribute fall outside the tolerance range, no corresponding phantom attribute is constructed. Thus $|AP| \leq |A|$. Furthermore, $|Range(AP_i)| \leq |Range(A_i)|$ for all $AP_i \in AP$.

Setting TL= ∞ results in behaviour that exactly matches that of ID3. The other extreme occurs when TL= 1.0, where only the attribute-value pairs whose entropy measure *exactly matches* the minimal entropy measure may potentially be branched on; the rest are grouped together in a "default" branch. In most cases, this setting is expected to result in a binary decision tree—branching on one

value with the rest of the values on the default branch. As TL varies from 1.0 to ∞, the program generates different trees. For a given set of data, we claim the existence of a setting for TL that results in the generation of a "better" decision tree than that of ID3. The systematic determination of the proper setting of TL for a given set of data remains as future work. In section 5 we empirically verify our claim. In subsequent sections of this paper, we refer to our algorithm as the Generalized ID3 (GID3) algorithm.

Begin Algorithm

1. For each attribute-value pair $\langle A_i, V_j \rangle$ appearing in the examples in S[1], create a binary-valued "temporary attribute" $\langle A_i = V_j \rangle$ which takes the value TRUE for examples that have value V_j for attribute A_i, and the value FALSE for all other examples in S.
2. For each binary-valued temporary attribute $\langle A_i = V_j \rangle$ defined in step 1, $E(\langle A_i = V_j \rangle, S)$ as defined in equation (1) is computed. Note that $Range(\langle A_i = V_j \rangle) = \{TRUE, FALSE\}$.
3. Let MIN-E be the minimum of the values computed in step 2 above.
4. Let Thrshld-E = TL $\times$ MIN-E, where TL $\geq$ 1.0 is referred to as the *tolerance level* and is determined by the user.
5. Construct a set of *phantom attributes*, AP, as follows:
   (a) $AP \leftarrow \emptyset$.
   (b) **For** each attribute $A_i \in A$, for which more than one value appears in the examples in S, **Do**
      i. Default $\leftarrow \emptyset$; $R \leftarrow \emptyset$.
      ii. **For** each temporary attribute $\langle A_i = V_j \rangle$ defined in step 1 **Do**
      IF $E(\langle A_i = V_j \rangle, S) \leq$ Thrshld-E
      THEN $R \leftarrow R \cup \{V_j\}$
      ELSE Default $\leftarrow$ Default $\cup \{V_j\}$
      iii. IF $R \neq \emptyset$ THEN $AP \leftarrow AP \cup \{AP_i\}$
      where AP_i is the ***phantom attribute*** corresponding to A_i,
      and $Range(AP_i) = R \cup \{$ Default $\}$.
6. Conduct the ID3-IV algorithm to choose an attribute out of the set of phantom attributes AP constructed in 5.

End Algorithm.

4.2 Overcoming ID3 Problems

The procedure described above is obviously designed to avoid the irrelevant values problem. As a side effect, the procedure will also generate trees that are less likely to suffer from missing branches. Figure 2 illustrates how the missing branches problem may potentially be avoided by the new approach. The problems of the tree of figure 1 are overcome by the tree of figure 2. Note that both e and e' of figure 1 are now classifiable, indicating a higher inductive power for the tree generated by the new approach.

5 Evaluation Criteria and Test Results

5.1 Evaluation Criteria

Little work has been done on evaluating or comparing decision trees. Quinlan[Quin87] used the number of nodes in the tree as a measure of its optimality. He also used the percentage error rate

[1] Values V_j appearing on the path from the root to the node representing S are excluded from consideration.

```
Two test examples not in the training set:

     e  :  (A = a3),    (B = b3)
     e' :  (A = a1),    (B = b3)
```

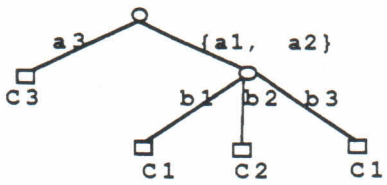

Figure 2. An improved decision tree generated by GID3

when the tree is used to classify examples not in the training set[2]. A decision tree may also be translated into a set of rules that perform an equivalent classification. Some performance measures may be defined on this set of classification rules as well. We list below our measures of performance. The (↑) and (↓) symbols indicate that the measure is to be maximized or minimized respectively.

1. Number of nodes in the tree (↓).
2. Percentage errors on classifying unseen examples (↓).
3. Number of rules in rule-base (leaves in tree) (↓). This measures how close the algorithm gets to achieving the ideal goal of finding the minimal set of rules[Quin87].
4. Total number of preconditions in rule-base (↓). This measures the generality of the entire set of rules. It is a more accurate measure of generality than the average number of preconditions per rule, since the latter is not very indicative if the number of rules is large.
5. Average example support per rule (↑). This is the average number of examples in the test set on which a rule is applicable. It is a measure of the applicability (generality) of a rule.
6. Average number of decisions per example in the test set (↓). This is a data-dependent measure of the efficiency of the generated classification tree.

5.2 Test Domain

The primary motivation for this research is the application of machine learning techniques to the development of expert systems for diagnosing faults in the Reactive Ion Etching (RIE) process in semiconductor manufacturing. The process is at the forefront of the technology and no models for it exist. Our goal is to apply techniques of learning from examples to speed up the process of constructing a knowledge base to be used in regulating the manufacturing process. Examples are readily available, but rules are hard to come by. This makes the domain a prime application area for machine learning. Among other approaches, we attempted the ID3 approach. ID3 produced rules that were overspecialized, or that included irrelevant preconditions and excluded important conditions. This led us to investigate and identify some of the causes for overspecialization. The test results verify that the GID3 algorithm does indeed reduce overspecialization problems significantly.

5.3 Testing Method and Results

In order to verify that the new method is indeed an improvement over ID3, we conducted tests on artificially generated data. This was necessary in order to establish an objective rather than a subjective basis for judging the program's performance. A set of rules was constructed manually for diagnosing a well-understood portion of the RIE process. The rules were verified physically and semantically by the domain experts. This set of rules was used to generate random examples. Each rule specifies the values of only a few of the available attributes. Since attributes not appearing in a rule's precondition are considered irrelevant to the classification task, random values are generated for those attributes in order to obtain examples. The goal of the learning program is then to attempt to

[2]Both ID3 and GID3 guarantee 100% correctness when their generated trees are used to classify the training set.

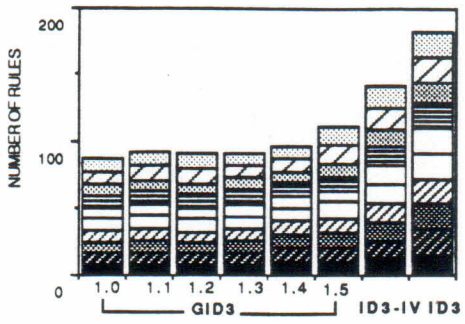

Figure 3. Number of rules/leaves in tree

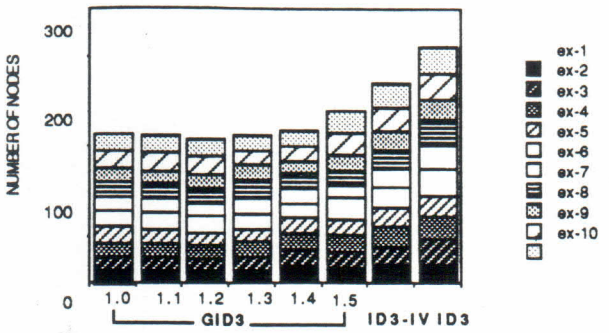

Figure 4. Number of nodes in tree

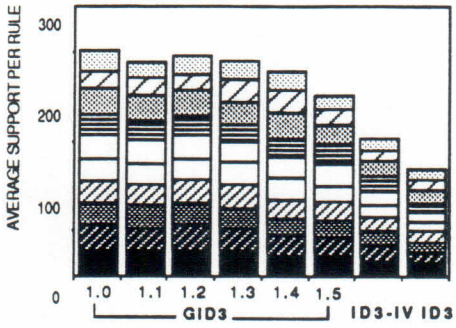

Figure 5. Average support per rule

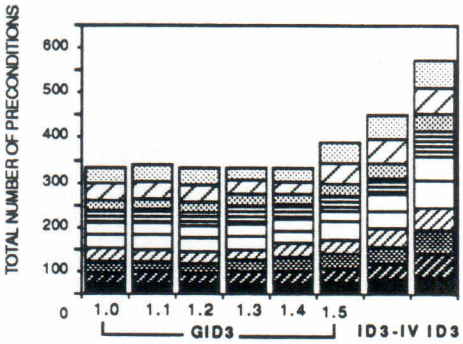

Figure 6. Total preconditions in rule-base

rediscover, or approximate, the original set of rules. This establishes a reference point for comparing the performance of the learning algorithms.

In order to eliminate random variation, 10 independent experiments were conducted on 10 independently generated random data sets[3]. The different shadings that appear in each individual column in the graphs of figures 3-6 correspond to the 10 experiments. Note that the heights of the composite columns are proportional to the corresponding statistical average over the 10 experiments. The performance data for measures 2,5, and 6 were collected by using the decision trees generated to classify examples in a fixed test set of 5000 examples. The graphs of figures 3-6 depict results for performance measures 1-4. Measure 6 was essentially the same for all methods; it is therefore not included because of space restrictions.

It is evident from these graphs, that performance varies with different settings for TL. Furthermore, for some setting of TL, GID3 outperforms ID3 *on all* performance measures. Results for performance measure 2 are shown in figure 7 (averaged over 10 experiments). For this chart, the two extra columns on the far right, labelled ID3(Guess) and ID3-IV(Guess) need to be explained. One of our goals is to overcome overspecialization in ID3 by avoiding the missing branches problem. A missing branch in ID3 corresponds to failure to predict a class for some examples. If we claim that GID3 reduces missing branches, we have to verify that it is not simply replacing a missing branch by some random guess (default branch) of some class. The two columns labelled '(Guess)' show the result of replacing every occurence of a missing branch in ID3 by a branch leading to a leaf node that is randomly assigned to some class in C. The graph verifies that GID3 is doing better than random guessing. This is evidence in support of the fact that GID3 is *actually avoiding* the missing branches problem.

[3] A training set consists of 200 examples on the average

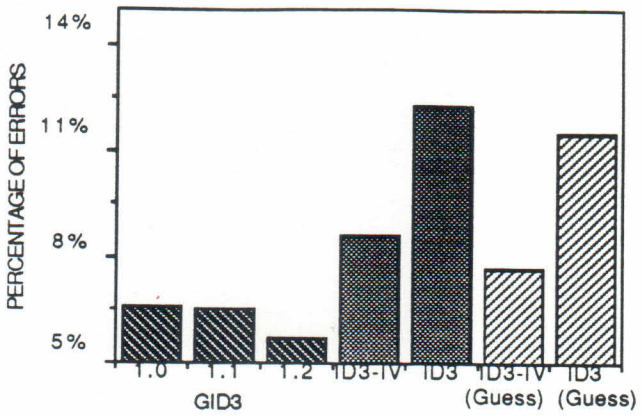

Figure 7. Average percentage of errors

6 Conclusions and Future Work

We have identified some problems in the ID3 approach for generating decision trees and hypothesized their causes. A new algorithm, GID3, targeted at avoiding these problems, is formulated. We have empirically demonstrated that the new method outperforms ID3 *on all* performance measures considered. The trees generated using the new approach are more compact, more reliable, and more general decision trees than their ID3 counterparts. The gains are achieved with a negligible increase in computational cost. We have not yet developed an algorithm for setting the single parameter (TL) in our system; this remains as future work. We have, however, verified the existence of a setting for TL that results in an algorithm which avoids overspecialization in ID3.

7 Acknowledgements

This work is supported in part by SRC under contract 87-MP-085, and in part by the Air Force Office for Scientific Research under contract F49620-82-C0089. Thanks to Clare Congdon and Kristina Van Voorhis for proof reading earlier drafts of this paper.

References

[Haus87] Haussler, D. "Learning conjunctive concepts in structural domains." *Proceedings of the National Conference on Artificial Intelligence, AAAI-87.* pp. 466-470. Seattle, WA (1987).

[Hunt66] Hunt, E.B., Marin, J., and P.J. Stone *Experiments In Induction.* New York: Academic Press, (1966).

[Mich78] Michalski, R.S. and Larson, J.B. "Selection of most representative training examples and incremental generation of VL1 hypotheses: The underlying and the description of programs ESEL and AQ11. Report No. 867. Computer Science Dept., University of Illinois, Urbana, (1978).

[Mich79] Michie, D. (Ed.) *Expert Systems in the Micro Electronic Age.* Edinburgh: Edinburgh University Press (1979).

[Moze86] Mozetic, I. "Knowledge extraction through learning by examples." In Mitchell, T.M., Carbonell, J.G., and Michalski, R.S. *Machine Learning: A Guide to Current Research.* pp. 227-231. Boston: Kluwer Academic Publishers, (1986).

[Quin83] Quinlan, J.R. "Learning efficient classification procedures and their application to chess endgames." In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. *Machine Learning: An Artificial Intelligence Approach.* pp. 463-482. Palo Alto, CA: Tioga Publishing Company, (1983).

[Quin86] Quinlan, J.R. "Induction of decision trees." *Machine Learning 1, No. 1.* pp. 81-106. Boston: Kluwer Academic Publishers, (1986).

[Quin87] Quinlan, J.R. "Generating production rules from decision trees". *IJCAI-87.* pp. 304-307. Milan, Italy (1987).