Branching on Attribute Values in Decision Tree Generation

Usama M. Fayyad
AI Group, M/S 525-3660
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099
Fayyad@aig.jpl.nasa.gov

Abstract

The problem of deciding which subset of values of a categorical-valued attribute to branch on during decision tree generation is addressed. Algorithms such as ID3 and C4 do not address the issue and simply branch on each value of the selected attribute. The GID3* algorithm is presented and evaluated. The GID3* algorithm is a generalized version of Quinlan's ID3 and C4, and is a non-parametric version of the GID3 algorithm presented in an earlier paper. It branches on a subset of individual values of an attribute, while grouping the rest under a single DEFAULT branch. It is empirically demonstrated that GID3* outperforms ID3 (C4) and GID3 for *any* parameter setting of the latter. The empirical tests include both controlled synthetic (randomized) domains as well as real-world data sets. The improvement in tree quality as measured by number of leaves and estimated error rate is significant.

Introduction

Empirical learning algorithms attempt to discover relations between situations expressed in terms of a set of attributes and actions encoded in terms of a fixed set of classes. By examining large sets of pre-classified data, it is hoped that a learning program may discover the proper conditions under which each action (class) is appropriate. Heuristic methods are used to perform guided search through the large space of possible relations between combinations of attribute values and classes. A powerful and popular such heuristic uses the notion of selecting attributes that locally minimize the information entropy of the classes in a data set. This heuristic is used in the ID3 algorithm [11] and its extensions, e.g. GID3 [2], GID3* [4], and C4 [12], in CART [1], in CN2 [3] and others; see [4, 5, 10] for a general discussion of the attribute selection problem.

The attributes in a learning problem may be discrete (categorical), or they may be continuous (numerical). The above mentioned attribute selection process assumes that all attributes are discrete. Continuous-valued attributes must, therefore, be *discretized* prior to attribute selection. This is typically achieved by partitioning the range of the attribute into subranges, i.e., a test is devised that quantizes the range. In this paper, we focus only on the problem of deciding which values of a discrete-valued (or discretized) attribute should be branched on, and which should not. We propose that by avoiding branching on all values (as in ID3), better trees are obtained. We originally developed the GID3 algorithm [2] to address this problem. GID3 is dependent on a user-determined parameter setting (TL) that controls its tendency towards branching on some versus all values of an attribute. We have demonstrated that for certain settings of TL, GID3 produces significantly better trees than ID3 or C4[1]. In this paper we present the GID3* algorithm in which the dependence on a user-specified parameter has been removed. We empirically demonstrate that GID3* produces better trees than GID3 for a wide range of parameter settings.

The Attribute Selection Criterion

Assume we are to select an attribute for branching at a node having a set S of N examples from a set of k classes: $\{C_1, \ldots, C_k\}$. Assuming that some test T on attribute A partitions the set S into the subsets $S_1, \ldots, S_r$. Let $P(C_i, S)$ be the proportion of examples in S that have class C_i. The *class entropy* of a subset S is defined as:

$$\mathrm{Ent}(S) = -\sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S))$$

The resulting class entropy after a set S is partitioned into the r subsets is:

$$E(A, T; S) = -\sum_{i=1}^{r} \frac{|S_i|}{|S|} \mathrm{Ent}(S_i) \qquad (1)$$

The information gain due to the test T, is thus defined to be $Gain(A, T; S) = \mathrm{Ent}(S) - E(A, T; S)$. Having found that the gain measure is biased in favor of tests that induce finer partitions, Quinlan [11] adjusted it by dividing it by the entropy of the test outcomes themselves:

$$\mathrm{IV}(T; S) = \mathrm{Ent}(T; S) = -\sum_{i=1}^{r} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right)$$

[1] In this paper we do not consider pruning. We focus only on generating better trees. Pruning methods can be applied to any tree regardless of how it was generated.

Temp.	selectivity	line width	Si02 flow	Class
100	normal	normal	low	HP
102	low	low	low	LF
105	normal	high	v. high	LP
110	high	high	high	LP
100	high	low	v. low	HP
101	low	medium	normal	LF
115	normal	normal	normal	HP
112	high	high	high	LP

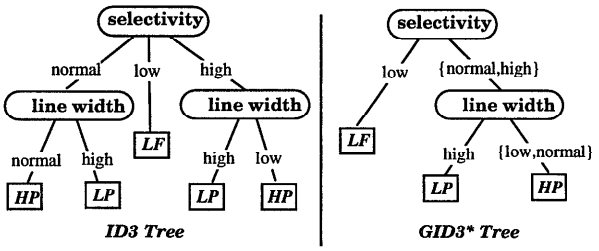

Figure 1: Two Decision Trees for a Simple Training Set.

giving the Gain-ratio measure:

$$\textit{Gain-r}(A,T;S) = \frac{\textit{Gain}(A,T;S)}{\text{IV}(T;S)}.$$

This empirically results in improved behavior. We refer to the ID3 algorithm with the Gain-ratio selection measure as ID3-IV (Quinlan later refers to it as C4 [12]).

Assuming that attribute A is discrete (or previously discretized), the problem we address is how to formulate the test T. In ID3 and C4, the test T is simply a test of the value of A, with a separate branch for each value of A appearing in the data at the local node. This class of tests result in problematic trees since not every value of an attribute may be relevant to the classification task. For example, consider the attribute "color" with values {blue, red, green, yellow, white, black}. It may be the case that only the colors blue and red convey meaningful classification information. The fact that an object's color is, for example, white, conveys no information other than the fact that the color is *neither blue nor red.* branching on each individual value results in excessively partitioning the data. This reduces the quality of subsequent attribute choices, and the problem compounds as subtrees are subsequently grown from over-partitioned nodes [4]. These problems are referred to as the **irrelevant values, reduced data**, and **missing branches** problems. Figure 1 illustrates how an ID3 tree generated from the the given simple training set differs from the corresponding GID3* tree. Note that both trees were generated from the data set, yet the ID3 tree would fail to classify a new example (not in the training set) that has values { selectivity=normal, line width=low}. The GID3* tree can classify this example, hence is more general. We shall later show that on average GID3* trees are also more accurate.

The GID3 Algorithm

In order to overcome overbranching problems, we generalized the ID3 approach by introducing the *attribute phantomization* step prior to attribute selection. An attribute A with r values is transformed into a *phantom attribute* A' with s values, where $s \leq r$. The values of A deemed irrelevant, are mapped to a single branch (value) called *"DEFAULT"*. The other values remain unchanged. In our example above, the values of the phantom attribute *color'* would be {blue, red, DEFAULT}. In GID3, phantomization was achieved by evaluating the Gain of each attribute-value pair separately. The maximum Gain multiplied by the user-specified tolerance level TL, $0 \leq \text{TL} \leq 1$, gives a threshold gain. All pairs whose Gain is not less than the threshold gain are passed. Each phantom attribute is defined to have the values of the original attribute that passed.

As presented, the GID3 algorithm provides the means to generate a family of trees from a given training set. The TL parameter allowed us to control which tree is generated by increasing or decreasing the tendency of the algorithm to branch on some, rather than all, values of the selected attribute. ID3 trees are certainly members of the family of trees generated by GID3 (TL= 0 setting). Thus GID3 served as a convenient tool to conceptually and empirically verify that the irrelevant values problem in ID3 can, in principle, be alleviated. Further empirical evidence for this claim is given in [4]. However, to provide a well-defined algorithm we need to remove the reliance on the user-specified parameter TL.

We empirically demonstrated that there are certain settings of the TL parameter that result in decision trees that are superior to the corresponding ID3 and ID3-IV trees[2, 4]. As a matter of fact, this was also our experience in all our experiments involving data from various companies in semiconductor manufacturing [8]. The main problem with the TL parameter, is that its optimal setting varies from domain to domain, as well as across different training sets within a single domain. Actually, the user typically finds it difficult to decide on a TL setting because the parameter has no simple intuitive interpretation.

The GID3* Algorithm

The GID3* algorithm differs from GID3 only in the attribute phantomization stage. It uses an additional measure to overcome GID3's dependence on the parameter TL.

For a set S of N examples and an attribute A with discrete values over S, let a_i be one of the values of A. We define the *Tear* of the attribute-value pair $\langle A, a_i \rangle$ to measure the degree to which the pair $\langle A, a_i \rangle$ "tears the classes in S apart from each other". The pair $\langle A, a_i \rangle$ may be used to partition S into the subset $S_{A=a_i}$ consisting of examples in S that have $A = a_i$ and the subset $S_{A \neq a_i}$ consisting of the rest of the examples in S. $\textit{Tear}(\langle A, a_i \rangle, S)$ measures the degree to which the classes in $S_{A=a_i}$ and $S_{A \neq a_i}$ are disjoint from each other. In other words, $\textit{Tear}(\langle A, a_i \rangle, S)$ is a measure of the amount of "work" that the pair $\langle A, a_i \rangle$ performs towards getting us to the desirable state of "tearing all the classes apart from each other."

Let $C(j, S)$ be the number of examples in S that have class C_j. The quantity

$$\frac{|C(j, S_{A=a_i}) - C(j, S_{A \neq a_i})|}{C(j, S_{A=a_i}) + C(j, S_{A \neq a_i})}$$

measures the degree to which the partition induced by $\langle A, a_i \rangle$ separates examples of the same class, C_j, from each other. It is maximum when examples of C_j remain together either in $S_{A=a_i}$ or $S_{A \neq a_i}$. Taking the weighted average *intra-class cohesion* (non-separation) over all classes in S we get the measure $Tear1(\langle A, a_i \rangle, S)$

$$= \sum_{j=1}^{k} P(C_j, S) \frac{|C(j, S_{A=a_i}) - C(j, S_{A \neq a_i})|}{C(j, S_{A=a_i}) + C(j, S_{A \neq a_i})}$$

$$= \frac{1}{N} \sum_{j=1}^{k} |C(j, S_{A=a_i}) - C(j, S_{A \neq a_i})|.$$

Note that $Tear1$ reduces to the normalized vector difference (the L_1-norm) between the two class vectors of the sets $S_{A=a_i}$ and $S_{A \neq a_i}$. It can easily be shown that $0 \leq Tear1(\langle A, a_i \rangle, S) \leq 1$. The measure is maximized when the set of classes represented in $S_{A=a_i}$ is disjoint with the set of classes represented in $S_{A \neq a_i}$. It is minimized when the class vectors of the two subsets $S_{A=a_i}$ and $S_{A \neq a_i}$ are identical. See [4, 5] for a discussion on weaknesses of the entropy measure. $Tear1$ is biased in favor of pairs $\langle A, a_i \rangle$ which have very few or very many of S's examples. Let $P(\langle A, a_i \rangle, S)$ be the proportion of examples in S having values a_i for attribute A: $P(\langle A, a_i \rangle, S) = \frac{|S_{A=a_i}|}{|S|}$. Then if $P(\langle A, a_i \rangle, S) \approx 0$, or if $P(\langle A, a_i \rangle, S) \approx 1$, the $Tear1$ measure goes close to its maximum regardless of the disjointness of classes in $S_{A=a_i}$ and $S_{A \neq a_i}$. We can correct for this by multiplying $Tear1$ by the proportion $P(\langle A, a_i \rangle, S)$, however this is not a symmetric correction since it favors large $|S_{A=ai}|$. The proper symmetric correction factor is one that is symmetric about $P(\langle A, a_i \rangle, S) = 0.5$ and is 0 at $P(\langle A, a_i \rangle, S) = 0$ and $P(\langle A, a_i \rangle, S) = 1$. Consider the weighting factor given by:

$$W(\langle A, a_i \rangle, S) = P(\langle A, a_i \rangle, S) \left(1 - P(\langle A, a_i \rangle, S)\right) = \frac{|S_{A=a_i}| \cdot |S_{A \neq a_i}|}{|S|^2}$$

Clearly, $0 \leq W(\langle A, a_i \rangle, S) \leq \frac{1}{4}$, and $W(\langle A, a_i \rangle, S)$ is symmetric about its maximum value of 0.25 at $P(\langle A, a_i \rangle, S) = 0.5$. In general, it is desirable to have the weighting factor be uniform in the middle and drop off to zero very quickly (c.f. an ideal band-pass filter). We can achieve this by using the following function:

$$WF(\langle A, a_i \rangle, S) = \min\{1.0\ ,\ \beta \cdot W(\langle A, a_i \rangle, S)\} = \min\{1.0\ ,\ \beta \cdot P(1-P)\}$$

where P denotes $P(\langle A, a_i \rangle, S)$, and $\beta \geq 4.0$ is a parameter that determines where and how fast the filter WF starts dropping off from its maximum value of 1.0. For example, if $\beta = 25$ the filter is uniformly 1.0 for the interval $0.042 \leq P(\langle A, a_i \rangle, S) \leq 0.958$. In general, the beginning and end of the 1.0 plateau range may be derived by solving for P in the equation: $\beta \left(P(1-P)\right) = 1.0$,whose solution is:

$$P = \frac{1}{2} \pm \sqrt{\frac{1}{4} - \frac{1}{\beta}}.$$

Note that WF rises from 0.0 to 1.0 at a slope of approximately β for large β. We fix β at 25, a fairly large value.

We now define the desired *Tear* measure for an attribute-value pair $\langle A, a_i \rangle$

$$Tear(\langle A, a_i \rangle, S) = WF(\langle A, a_i \rangle, S) \cdot Tear1(\langle A, a_i \rangle, S).$$

Note that $0 \leq Tear(\langle A, a_i \rangle, S) \leq 1$. Also note that the tear measure is minimum either when the class vectors of $S_{A=a_i}$ and $S_{A \neq a_i}$ are identical, or when one of the two subsets of S is empty.

We now describe how attribute A with values $\{a_1, a_2, \ldots, a_r\}$ is *phantomized* to produce the corresponding phantom attribute A'. We first evaluate the information entropy of each of the attribute-value pairs $\langle A, a_i \rangle$, $1 \leq i \leq r$ as described in the GID3 algorithm to get $E(\langle A, a_i \rangle, S)$. We then choose the pair with the minimum entropy. Without loss of generality, let this value be a_1. a_1 will be one of the values of the phantom attribute A'. What is left to be decided at this point is which of the remaining values $\{a_2, \ldots, a_r\}$ are to be admitted as values of A' and which to group together under the value *DEFAULT*.

Let T_1 be the *Tear* value of the best entropy attribute-value pair $\langle A, a_1 \rangle$. Let this tear be $T_1 = Tear(\langle A, a_1 \rangle, S)$, and let S' be the subset of S consisting of examples that have a value other than a_1 for attribute A: $S' = S_{A \neq a_1} = S \sim S_{A=a_1}$. For each of the remaining values of A, $\{a_2, \ldots, a_r\}$, compute the *Tear* of the attribute-value pair with respect to the set S'. That is $T_i = Tear(\langle A, a_i \rangle, S')$. Every value a_i whose tear value T_i is less than T_1, is considered irrelevant and is grouped with others under the *DEFAULT* value. Other values are considered relevant and are considered as proper values of the phantom attribute A'. Once the phantom attribute has been generated for each of the attributes, GID3* proceeds exactly as GID3. It evaluates *Gain-r* of each phantom attribute and selects the best phantom attribute for branching.

from a measure other than entropy. Information entropy still plays the primary selection role since the best entropy value among the values of each attribute is selected, and the phantom attribute with the best information gain is selected. In GID3*, every phantom attribute is guaranteed at least two values: its best entropy value and *DEFAULT*. Note that in GID3 some attributes may not have corresponding phantom attributes.

Note that in other systems, different phantomization (or subsetting) schemes have been proposed. In CART [1], a *binary partition* of an attribute's values is selected. In general this is exponential (except in case of 2-class problems) and is thus not feasible. ASSISTANT [9] also uses a partition scheme requiring extensive search. Partitions provide for a richer language than the extension that GID3* provides (restricted partitions of singletons and a default subset). In the case of GID3*, the complexity of determining the partition is linear in the number of attribute values.

Empirical Evaluation of GID3 and GID3*

In this section, we present empirical evidence that verifies our hypothesis that GID3* produces better trees than GID3.

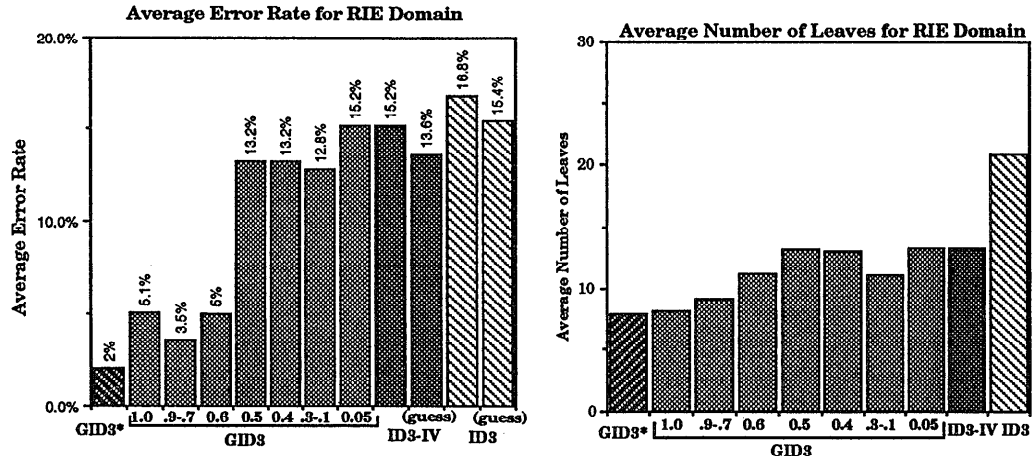

Figure 2: Results (number of leaves and error rates) for the RIE Domain.

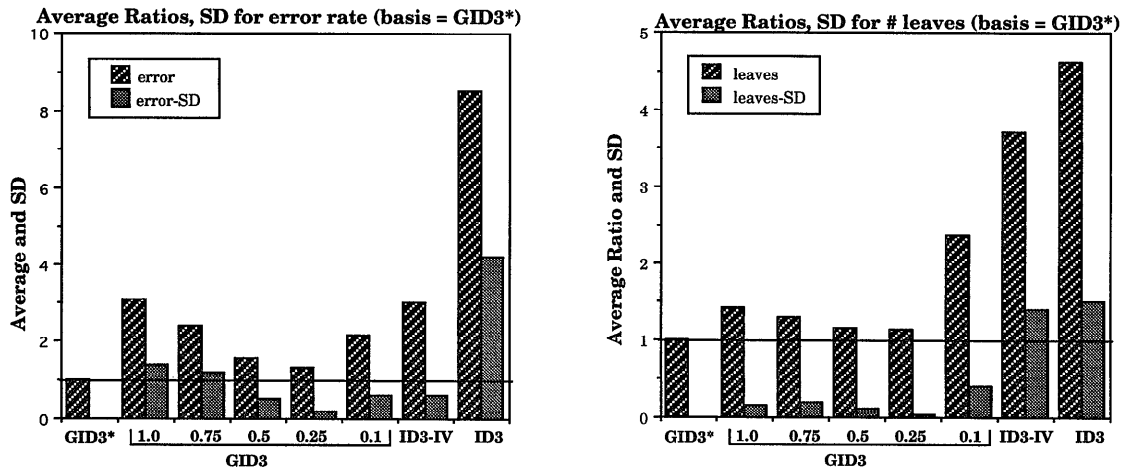

Figure 3: Average Ratios and SD for Random Domains.

Since GID3 has a user-specified parameter TL, we have to compare the tree obtained from GID3* with the "best" tree obtained from GID3 with various settings of TL.

Experiments on Synthetic RIE Data

The first set of experiments were conducted on the data set from the RIE (Reactive Ion Etching in semiconductor manufacturing) domain described in [2]. The results obtained are shown in Figure 2 Also note that although both GID3 (with TL=1.0) and GID3* found the trees with the minimum number of leaves (eight), the two trees were actually different (as evidenced by the different error rates). In general, GID3* did as well as or better than the best GID3 threshold both on compactness and accuracy. Since ID3 suffers from the missing branches problem, on some examples it may fail to make a prediction. The columns labelled (guess) for ID3 and ID3-IV show what happens if the algorithm makes an extra guess at random whenever it gets stuck with a missing branch. This is to demonstrate that GID3 and GID3* generalizations are not essentially making only "random guesses," but more meaningful predictions.

Experiments on Random Synthetic Domains

The next set of controlled experiments were conducted on synthetic randomly generated data extracted from randomly generated domains. The general procedure is to randomly generate a domain by selecting a random number of attributes, values for attributes, and classes. A decision tree is then constructed using an algorithm that randomly selects an attribute and a subset of its values to branch on. The algorithm flips a coin to determine whether or not to deem a node a leaf, and if so, randomly labels it with a class. This results in a decision tree that represents the rules governing examples in the given domain. Examples of each leaf node are then generated randomly as in the synthetic RIE domain. On average, about 25 examples per leaf were generated. The number of attributes varied between 8 and 20 with each attribute's range varying between 2 and 15 values. The number of classes varied between 4 and 10.

As an example, one randomly generated domain had 9

Table 1: Details of the Data Sets in Empirical Evaluation.

Data Set	examples		attributes		classes
	train	test	disc.	cont.	
HARR90	280	220	25	—	72
SOYBEAN	290	340	35	—	15
AUTO	102	103	9	15	6
MUSHRM	50-350	7000	22	—	2

attributes, 6 classes, and a tree with 14 leaves (rules). A training set consisting of 25 examples per leaf and a test set of 50 examples per leaf were generated independently. The process was repeated 10 times. The averaged results for the various algorithms were recorded for that domain. The above procedure was repeated and results were collected for 100 randomly generated domains. Since it does not make sense to average performance measures over different domains, we resorted to collecting the ratios of improvement for each domain, and then averaging the ratios over different domains. Figure 3 shows the average ratio of number of leaves and error rates. The basis of performance (denominator of ratio) is the corresponding GID3* result. Hence, GID3* takes the value 1.0 for the ratio in these figures. The second column shows the standard deviation for each of the average ratios given.

Experiments on Real-World Discrete Data

Finally, we conduct experiments to compare the performance of GID3, GID3*, and ID3 on some real-world domains. One issue needs to be clarified at this point. GID3 and GID3* are designed to overcome the irrelevant values problem. If all attributes for a given data set are binary-valued, then the performance of GID3*, GID3, and ID3-IV on this data will be identical. Furthermore, if attributes are continuous-valued, then the "standard" method for handling them is to discretize them into a binary-valued attribute. We present a method for multiple-interval discretization and its benefits in [4, 7]. Data sets with only binary-valued discrete attributes, or continuous-valued attributes will not be useful for comparison purposes. For comparisons on data with continuous-valued attributes see [4, 7].

The data sets we used are are described in Table 1. HARR90 is a Semiconductor manufacturing data set obtained from Harris Corp. as part of an effort to construct an expert system for diagnosing transistor manufacturing defects. The other three were obtained from the U.C. Irvine ML data repository. Although the MUSHRM data has about 8000 examples, it is an extremely "easy" learning task. With about 500 examples, all algorithms achieve error rates lower than 0.11%. For this reason we kept the training sets small (50-350). The results reported are averaged over 10 such runs. Also, the SOYBEAN data contains some undefined attribute values. The algorithm used by GID3 and GID3* to handle undefined attribute values is described in [4]. For the SOYBEAN data, we used the standard training and test sets provided in the database. For the others, we randomly sampled the data sets to obtain training and test sets. The results are averaged over 10 runs. The results, shown in terms of average error rate and number of leaves for each of the four domains in Figure 4, confirm that GID3* produces better trees than GID3, ID3-IV, and ID3.

Concluding Remarks

Using synthetic randomized domains and real-world data, we have empirically demonstrated that GID3 and GID3* outperform ID3. The trees generated using GID3* are superior to trees in the family of trees generated by GID3. The GID3* trees are more compact, more reliable, and more general decision trees than their GID3 and ID3 counterparts. The improvement was achieved with a negligible increase in computational cost. Of course, this gives no guarantee that GID3* is always better than GID3 (for some optimal TL setting). However, empirical evidence, coupled with the fact that the TL parameter in GID3 complicates its application to industrial data, are enough reasons to tip the balance in favor of a parameter-free algorithm: GID3*.

Further note that we have used data sets which contain only discrete-valued attributes. This is to isolate effects due to the discretization stage for continuous-valued attributes. Also, for binary-valued discrete attributes, the performance of ID3, GID3, and GID3* is identical. This limit the data sets useful for strict comparisons. Powerful results have been obtained on extensive data sets using GID3* (see [7]). For a real-world large scale application see [6]. However, in those papers it is difficult to isolate the improvement due to various factors. Finally, the interested reader is referred to [4, 5] for a more detailed discussion of the branching problem along with alternative (and more powerful) solutions.

The goal of this paper is to present the GID3* algorithm as a possible solution to the branching decision problem. GID3 is not considered a solution since it is not fully specified and is dependent on a user-specified parameter. It is indeed remarkable that the GID3* solution has proven empirically superior to GID3 at all of the TL settings evaluated. We have no formal explanation for this experimental observation.

Acknowledgments

This work was conducted in part while at The University of Michigan, Ann Arbor (1990-91). GID3 was originally developed in collaboration with Prof. K.B. Irani, J. Cheng, and Z. Qian. GID3* work was funded in part by a Hughes Microelectronics Center Unrestricted Grant (Fayyad and Irani). The work described in this paper was carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees.* Monterey, CA: Wadsworth & Brooks.

[2] Cheng, J., Fayyad, U.M., Irani, K.B., and Qian, Z. (1988). "Improved decision trees: A generalized version of ID3." *Proc. of the Fifth International Conference on Machine Learning* (pp. 100-108). San Mateo, CA: Morgan Kaufmann.

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

[4] Fayyad, U.M. (1991). *On the Induction of Decision Trees for Multiple Concept Learning.* PhD dissertation, EECS Dept., The University of Michigan.

[5] Fayyad, U.M. and Irani, K.B. (1992). "The attribute selection problem in decision tree generation" *Proc. of the Tenth National Conference on Artificial Intelligence AAAI-90* (pp. 104-110). Cambridge, MA: MIT Press.

[6] Fayyad, U.M., Weir, N. and Djorgovski, S. (1993) "SKICAT: A machine learning system for automated cataloging of large-scale sky surveys." *Proceedings of the 10th Int. Conf. on Machine Learning*, Morgan Kauffman.

[7] Fayyad, U.M. and Irani, K.B. (1993). "Multi-interval discretization of continuous-valued attributes for classification learning", *Proc. of IJCAI-93.*

[8] Irani, K.B., Cheng, J., Fayyad, U.M., and Qian, Z. (1990). "Applications of Machine Learning Techniques in Semiconductor Manufacturing." *Proc. of The S.P.I.E. Conference on Applications of Artificial Intelligence VIII* (pp. 956-965). Bellingham, WA: SPIE.

[9] Kononenko, I., Bratko, I., and Roskar, E. (1984) "Experiments in automatic learning of medical diagnostic rules." *Technical Report*, Ljubljana, Yugoslavia: Josef Stefan Institute.

[10] Lewis, P.M. (1962). "The characteristic selection problem in recognition systems." *IRE Transactions on Information Theory, IT-8*, 171-178.

[11] Quinlan, J.R. (1986). "Induction of decision trees." *Machine Learning 1*, 81-106.

[12] Quinlan, J.R. (1990). "Probabilistic decision trees." In *Machine Learning: An Artificial Intelligence Approach, Volume III*, Y. Kodratoff & R. Michalski (Eds.) San Mateo, CA: Morgan Kaufmann.

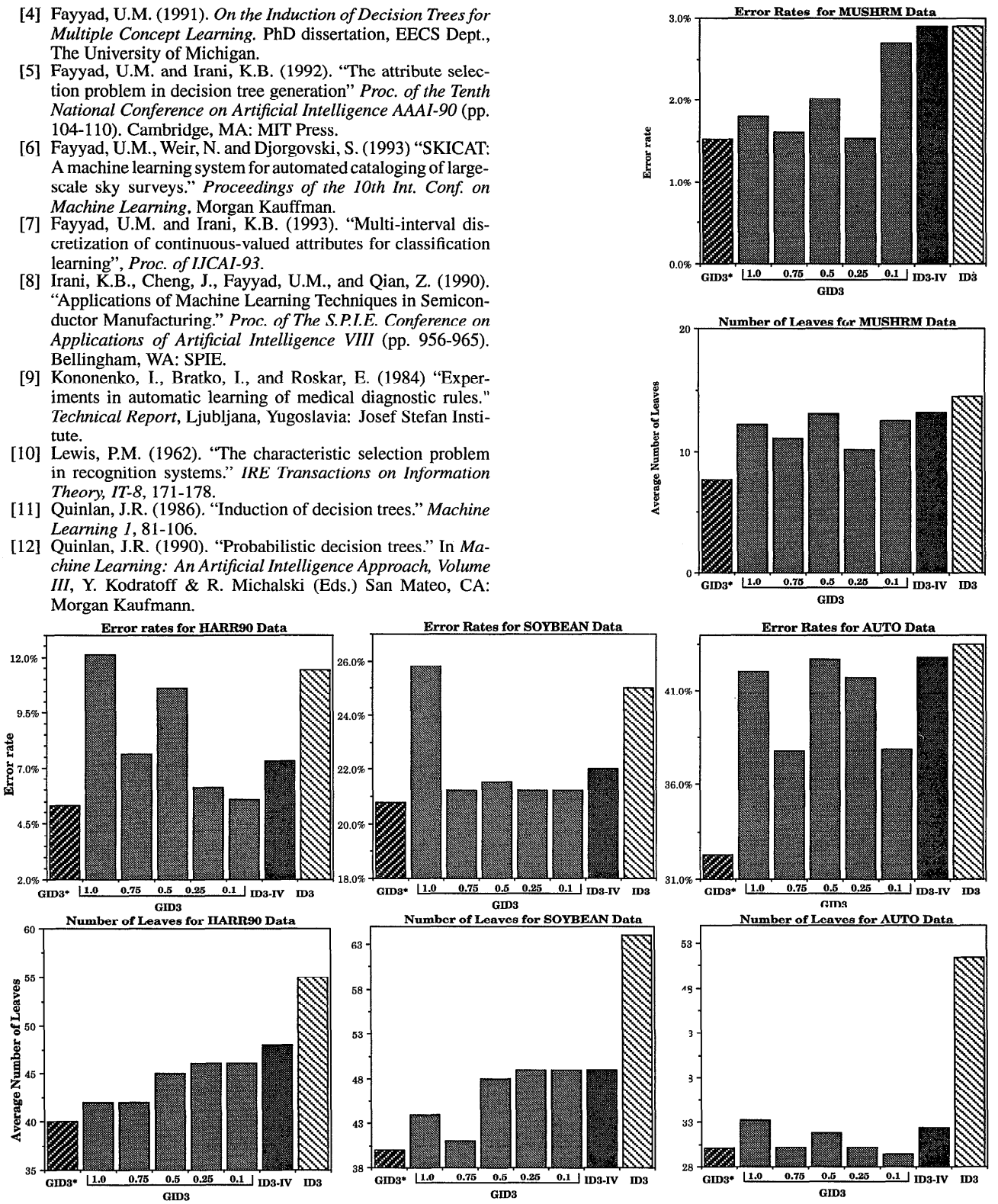

Figure 4: Results for the Four Domains of Table 1.