Compression-Based Discretization of Continuous Attributes

Bernhard Pfahringer[1]

Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna
Austria

E-mail: bernhard@ai.univie.ac.at
Phone: (+43 1) 533-6112
Fax: (+43 1) 532-0652

Keywords: Inductive Learning, Discretization, Minimum Description Length Principle, Noise

Abstract

Discretization of continuous attributes into ordered discrete attributes can be beneficial even for propositional induction algorithms that are capable of handling continuous attributes directly. Benefits include possibly large improvements in induction time, smaller sizes of induced trees or rule sets, and even improved predictive accuracy. We define a global evaluation measure for discretizations based on the so-called Minimum Description Length (MDL) principle from information theory. Furthermore we describe the efficient algorithmic usage of this measure in the MDL-Disc algorithm. The new method solves some problems of alternative local measures used for discretization. Empirical results in a few natural domains and extensive experiments in an artificial domain show that MDL-Disc scales up well to large learning problems involving noise.

[1]Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Research. I would like to thank Gerhard Widmer for constructive discussion and help with this paper.

1 Introduction

Discretization groups continuous numeric values into discrete intervals. Originally the motivation for discretization was the inability of handling continuous values exhibited by early propositional learning algorithms. Even though most propositional algorithms nowadays can handle continuous values, discretization can still be beneficial for various reasons:

1. Efficiency: handling (lots of) continuous values tends to slow down induction considerably.

2. Intelligibility: especially when faced with a large number of noisy training examples, decision trees involving continuous attributes tend to be rather complex.

3. Accuracy: in the presence of noise good discretizations can sometimes improve predictive accuracy.

Simple discretization methods are *class-blind*, i.e. they group training examples into intervals without taking into account the respective classes of the training examples. [Chiu et al. 91] compares three such methods: *equal-width discretization*, *equal-frequency discretization* and *maximum marginal entropy discretization*. Even though these methods seem to produce reasonable abstractions for many learning cases, there are situations where they perform poorly due to their class-blindness.[2]

To address these problems, a few new discretization methods using class information have recently been described in the literature [Catlett 91, Kerber 92, Lee & Shin 94, Richeldi & Rossotto 95]. They make use of either statistical or information-theoretical measures. These algorithms either work bottom-up, repeatedly merging intervals or work top-down, recursively splitting intervals until some stopping criterion is satisfied. All of those methods suffer from at least two problems (as most of their authors explicitly note):

1. They lack global evaluation. All used measures are only local in the sense that they estimate the value of merging two neighboring intervals taking into account no other ([Kerber 92, Lee & Shin 94]) – or only a small number of ([Richeldi & Rossotto 95]) – surrounding intervals, or that they estimate the value of splitting a single interval into exactly two intervals ([Catlett 91]). This restricted local analysis, which is motivated by efficiency reasons, sometimes misses the opportunity of forming larger uniform intervals. Consequently these algorithms may produce superfluous intervals, which may have a detrimental influence on both efficiency and accuracy.

[2] See [Kerber 92] or [Lee & Shin 94] for both convincing examples and some comparisons of class-blind methods to more sophisticated methods for discretization.

2. They also cannot distinguish truly uncorrelated (irrelevant) attributes from what [Kerber 92] calls *second-order* correlated attributes, i.e. attributes correlating only in the presence of some other condition.

In this paper we will introduce a new *global* information-theoretical measure for judging discretizations as a whole. The measure is an application of the *Minimum Description Length* (MDL) principle [Rissanen 78] as an objective function for evaluating the *global goodness* of a specific discretization. This measure will be described in the next section. Section 3 will outline efficient use of the new measure for discretizing even large sets of examples. Section 4 will report on extensive experiments in one artificial domain and in a few natural domains. Section 5 discusses open problems and further research directions.

2 An MDL Measure for Discretized Attributes

The basic assumption used here is as follows. If some discretization was synthesized taking class information into account, then such a discretization of a single attribute can be viewed as a – rather crude and approximate – set of rules classifying examples. To classify new examples determine the interval that covers the example's respective attribute value and just return that interval's majority class.

Adopting this interpretation we can straightforwardly use the so-called *Minimum Description Length* (MDL) principle to assign a numerical measure of quality to any single discretization. We only need to adapt one of the known MDL schemas applicable to e.g. decision trees [Quinlan & Rivest 89, Wallace & Patrick 93, Forsyth 93] or propositional rule sets [Pfahringer 95].

To motivate the formulas introduced below, a short introduction to the generic MDL principle seems appropriate. The MDL principle tries to measures both the simplicity and the accuracy of a particular theory in a common currency, namely in terms of the number of bits needed for encoding theory and data. A very good introduction to MDL and also its close relation to Bayesian theory can be found in [Cheeseman 90].

We define the cost (bit size) of a discretization as follows (for the sake of simplicity we restrict ourselves to two-class problems, but all formulas can be straightforwardly generalized to multi-class problems, or alternatively multi-class problems can also be cast as two-class problems [Dietterich & Bakiri 95]):

$$cost(Disc) = cost(DiscDef) + cost(Examples) \quad (1)$$

$$cost(DiscDef) = cost(Split) + cost(MajorityClasses) \quad (2)$$

$$cost(Split) = cost(choose(UsedSplits, PossibleSplits)) \quad (3)$$

$$cost(MajorityClasses) = cost(choose(Positive, Intervals)) \quad (4)$$

$$cost(Examples) = \Sigma cost(Examples/Interval_i) \quad (5)$$

$$cost(Examples/Interval_i) = cost(choose(Ex_{Majority_i}, Ex_{Int_i})) \quad (6)$$

$$cost(choose(E,N)) = N * entropy(E,N) \quad (7)$$

$$entropy(E,N) = -(plog(E/N) + plog(1-E/N)) \quad (8)$$

$$plog(P) = P * log(P) \quad (9)$$

So total cost (1) is the sum of the cost for defining the discretization (the *model*) and of the cost for encoding example classes with the use of the discretization (encoding the *data* in terms of the *model*).

A discretization (2) is defined by the list of split-points and the majority class for each interval. The cost for defining the split-points (3) can be estimated as the cost of encoding which split-points are actually used out of all possible split-points. Analogously we can estimate the cost for encoding majority classes (4) of each interval by judging the cost of encoding which of all the intervals have *positive* (remember we assume two-class problems) as their majority class.

The cost for encoding classification of examples (5) is the sum over all intervals of the cost of encoding examples with respect to their interval's majority class. For each interval we encode which examples agree with the classification assigned by the interval (6).

For estimating the cost of encoding the selection of E elements out of N possible elements two formulas are available from information theory. One is the so-called *Huffman coding* using variable-length strings for encoding messages of differing prior probabilities. The other one is the so-called *arithmetic* coding (as used e.g. in [Quinlan & Rivest 89]), which yields slightly smaller (i.e. better) estimates. Still we use Huffman coding here for practical reasons: we found no significant differences between the two possibilities with respect to size or accuracy of induced theories. But Huffman coding estimates can be computed much more efficiently (only 2 logarithms need to be computed versus $2 * E$ logarithms). Formulas 7 to 9 define the cost for the selection of E elements (split-points, positive classifications, or examples agreeing with the majority class of their respective interval) out of N possible elements based on Huffman coding.

Note that this formula for computing bitcost for a discretization is perfectly symmetric (exchanging "+" and "–" classes yields exactly the same formula) and that this formula can be generalized to multi-class problems in a straightforward way.

Now according to the MDL principle the one attribute discretization which minimizes the above cost-function, i.e. the one with the smallest bitcost (also-called the most *compressive* theory) is the most probable theory given the class distribution of the training examples.

3 Algorithmic Usage: MDL-Disc

Discretization of continuous attributes can be a very costly endeavor. In the worst case, if there is always one positive example between two negative examples and vice versa, N training examples can define $N-1$ split-points. As any subset of the set of possible split-points forms a valid discretization, there exist 2^{N-1} different discretizations in principle. So enumerating all discretizations and just choosing the one with minimal MDL cost is clearly out of the question.

Searching the space of all split-point combinations with either hill-climbing or best-first search proved also impractical in initial experiments when dealing with lots of examples involving hundreds or thousands of possible split-points. With regard to efficiency nothing is gained if the preprocessing step itself takes more time than the induction algorithm directly applied to the raw data. But luckily most of the possible split-points are *bad* split-points anyway that lead to inferior discretizations. So one can choose a heuristic method for selecting a subset of *promising* split-points. This subset then can be searched more thoroughly without incurring too much runtime cost.

Any of the above mentioned class-driven discretization algorithms could be chosen for such a heuristic pre-selection of split-points, especially as they all tend to produce too fine a discretization. For efficiency reasons we employ a simplified version of D-2 [Catlett 91], as it is the only top-down method. The other three methods basically all merge intervals bottom-up, starting from the smallest possible intervals. Therefore their complexity is $O(N^2)$ for N being the total number of possible split-points. Contrary, if we limit D-2 to a fixed depth D, its complexity is only $O(N)$.[3]

The original D-2 recursively splits the training examples using the info-gain heuristic of ID3 (or its successor C4.5 [Quinlan 93]) until one of a set of predefined stopping criteria is fulfilled. Our simplification replaces this set of criteria by a simple depth limit. For all experiments reported here we used $D = 5$ which leads to the selection of at most 31 split-points ($2^5 - 1$). The conjecture that this is a reasonable fine-grained limit is supported by the experimental results reported below. The original D-2 had 7 as the approximate limit for the number of split-points as one of its stopping criteria.

The set of promising split-points returned by D-2 then defines the space for a best-first search that tries to find the best discretization according to the MDL estimate defined in the last section. Typically this search results in a discretization using only very few split-points which are most of the time in accordance with the *correct* discretization where such a discretization is known at all.

The use of the MDL estimate effectively solves problem 1 of class-sensitive discretization methods: due to the global nature of the MDL estimate most co-

[3] We neglect the effort of sorting the attribute values first, which is of course $O(N * log(N))$ for N being the total number of training examples.

incidental split-points, that local methods would keep, are discarded. Problem 2 is also partially solved. Whereas local methods cannot even *recognize* such possibly problematic attributes, the MDL measure reliably identifies attributes that are either *unrelated* or *second-order related*: the result of discretization is one large interval covering all training examples! This means that their is no significant correlation between this single attribute and the classification. Unfortunately there is no simple way to decide between the two cases, only costly search involving other attributes could uncover second-order correlations. Obviously such a search would to some degree duplicate the efforts of the induction algorithm proper. There are at least three different alternative ways of handling such attributes:

1. Drop the attribute: This would be sensible if the attribute is truly irrelevant. But it is also unsafe, as we might drop an essential second-order related attribute.

2. Keep the raw attribute: This is safe but inefficient. When dealing with lots of training examples, even one non-discretized attribute can double induction runtimes.

3. Use a class-blind method: As class-membership does not seem to convey useful information for discretizing the respective attribute, using a class-blind method will produce at least a reasonable discretization.

As our aims are both accuracy and efficiency, we choose alternative 3 for our discretization method MDL-Disc. Specifically we use a slight modification of the *equal-frequency method*. The number of intervals is automatically chosen by the algorithm depending on the total number of possible split-points with an upper limit of 30. This is a kind of compromise between efficiency considerations (have as few intervals as necessary) and provisions for reasonable fine-grainedness (have enough intervals so that the induction algorithm is able to detect potential second-order correlations at all).

To summarize, MDL-Disc is a two-step discretization method. First a subset of promising split-points is heuristically determined by an efficient top-down procedure. Second this set is searched thoroughly by a best-first search for the *most compressive* discretization possible according to the MDL estimate defined in the previous section. Special provisions (escape to a class-blind method) are made for the degenerate case of a resulting discretization of just a single interval.

4 Experiments and Empirical Results

For empirical testing of MDL-Disc we have coupled it with C4.5 [Quinlan 93], a well-known induction algorithm for decision trees. According to our aims for

MDL-DISC (efficiency, accuracy, and intelligibility) we report for all experiments average C4.5 runtimes, average error rates on unseen examples, and the average number of nodes of the resulting decision trees. For comparison we give figures for C4.5 using the raw data, for C4.5 using data discretized by our simplified version of D-2[4] , and for C4.5 using data discretized by MDL-DISC. All results reported are averages of ten runs each using different random samples for learning; we give the mean value and the standard deviation. Regarding the cost of the preprocessing step itself, we note that MDL-DISC's runtimes are typically less than twice the runtimes of D-2. For large training sets the total runtime of MDL-DISC combined with C4.5 can be more than 10 times better than the runtime of C4.5 alone.

All the domains used in this empirical evaluation are 2-class problems (but see above comments regarding N-class problems). The main test-bed is a simple artificial domain introduced by [Catlett 91]. Its artificial nature allows for extensive experiments with various sizes of training sets and levels of class noise. Additionally we also performed some experiments with so-called *natural* domains mostly available from the UC Irvine Machine Learning Repository [Murphy & Aha 94].[5] These experiments also support our claims, but to a lesser degree, as these databases tend to be rather small.

4.1 An Artificial Domain

[Catlett 91] introduces the artificial DEMON domain as a kind of worst case scenario. Every example has three continuous attributes A1, A2, and A3 drawn randomly from the interval $[0, 1)$. A1 acts as a kind of *switch* (and as such A1 is not directly correlated to an example's class). An example belongs to class "+", if the following expression holds:

$$(A1 < 0.5 \wedge A2 < 0.8 \wedge A3 > 0.2) \vee (A1 > 0.5 \wedge A2 > 0.2 \wedge A3 < 0.8)$$

We have done experiments for training set sizes between 100 and 50000 and class noise levels of 0%, 10% and 20%. N% class noise means that the class-label of N% of the training examples is switched from "+" to "–" or vice versa. Tables 1, 2, and 3 give average runtimes, average predictive error, and average decision tree sizes measured in number of nodes for all experiments respectively.

[4]Such a comparison might seem unfair, as the original D-2 used a more sophisticated set of explicit stopping criteria: don't split small example sets, have a tight limit for the maximal number of split-points, stop if all split-points yield equal gain, and don't split pure intervals. But some of the conditions are implicitly present in our simplified version anyway (e.g. splits must yield non-zero gain, therefore pure intervals will not be split, etc.). The only significant difference seems to be the larger upper limit of 31 instead of only 7 for the total number of split-points.

[5]Exceptions are the THALLIUM SCAN domain which comes from a local clinics, and the HEART domain which is available from the StatLog project (URL = ftp://ftp.ncc.up.pt/pub/statlog/).

We can see that MDL-Disc clearly outperforms C4.5 in terms of runtime, the ratio varies between 1.5 and more than 30. C4.5's runtime increases sharply for large numbers of noisy examples. But this higher runtime does not lead to any better results: the predictive error rate of MDL-Disc is never significantly (according to a t-test) worse than that of C4.5 on the raw data. For small and medium sized noisy training sets MDL-Disc can even be significantly better. Also the average tree size clearly favors the usage of MDL-discretized data over raw data. In the extreme case we get 20 versus almost 700 nodes on average. It is also interesting to see that the number of nodes remains *constant* (even slightly decreases) when going from medium to large training sets for MDL-Disc.

The bad results of D-2 in this domain are not surprising. Even for the original D-2 reported results are significantly worse compared to using just raw data [Catlett 91]. Allowing for a more fine-grained discretization does not seem to solve the basic problem: discretizations for the switch attribute A1 tend to be poor. The search is misled by small localized random perturbations of an ideally equal class distribution. As MDL-Disc is able to identify such switch attributes reliably even in the presence of noise, and as it has a heuristic means for dealing with such attributes, MDL-Disc can perform considerably better than D-2 in this domain.

4.2 Natural Domains

The natural domains we have chosen either use only continuous attributes or some combination of nominal and continuous attributes. All these databases consist of only a few hundred examples. Table 4 summarizes relevant characteristics of these databases. Average results (runtime, error, and size) of ten test runs randomly using two thirds of the examples for training and the remaining third for testing are given in table 5.

We see that the results are not as pronounced as for the artificial domain (a fact that we attribute to the rather small number of examples in these domains), but we can notice the same trends when comparing raw C4.5 to MDL-Disc: for the CREDIT-APP runtime is 50% better and trees are half as big; in the DIABETES domain runtime and tree size are reduced to a half, furthermore the error rate is significantly (t-test level 95%) better; for both HEART and LIVER DISORDER the tree size is significantly smaller; and for IONOSPHERE the runtime is reduced to 50%. Only for THALLIUM SCAN there is no significant difference in any measure. This might be explainable by the abundance of attributes in this smallest of all domains, which leads to a higher probability of incidental correlations.

When comparing MDL-Disc to simple D-2 in these natural domains, we notice equal runtimes, slightly better predictive error rates and significantly smaller trees resulting from MDL-Disc. We conclude that MDL-Disc is able to improve discretizations computed by D-2 for two reasons. The global nature of the MDL evaluation allows for the formation of large intervals, which sometimes even

cover several neighboring subtrees of D-2's original search tree. The MDL evaluation also allows for the reliable identification of possibly second-order related attributes. Such attributes are discretized in a class-blind manner by MDL-Disc.

To summarize, MDL-Disc empirically performs at least as well as C4.5 applied directly to the raw data in terms of predictive error. Especially in the presence of noise MDL-Disc can outperform C4.5 significantly. This is a direct consequence of the MDL principle that effectly distinguishes between true and chance regularities given enough data. Up to tenfold improvement can be found for both runtime and final tree size. Once again improvements are especially strong for large and noisy training sets.

5 Conclusions and Further Research

We have defined an MDL measure for globally evaluating discretizations of continuous attributes. This new measure is information-theoretically plausible in the way it encodes discretizations and the training examples in terms of the discretizations. The MDL-Disc algorithm using the new measure yields good results in the experiments reported above.

Further research will have to compare MDL-Disc to bottom-up methods discussed in the introduction in terms of efficiency, accuracy, and intelligibility. Preliminary experiments suggest these methods will be inferior regarding both runtime and intelligibility.

A further open question is whether resorting to class-blind methods is really the best solution for attributes that are not directly correlated to classification outcome. Maybe feature subset selection algorithms could help determining truly irrelevant attributes.

Furthermore it might be possible to adapt the defined MDL measure for clustering nominal attributes with a large number of possible values into a few useful subsets. MDL-Disc could also be applied recursively to single intervals/subsets of the global discretization. This would yield a hierarchy of interval/subset approximations for a given attribute. Such hierarchies could be used by induction methods that handle hierarchically defined attributes. It might prove valuable to add this capability to decision tree inducers like C4.5: large training sets could probably be handled much more efficiently with hopefully even improved error rates.

References

[Catlett 91] Catlett J.: On Changing Continuous Attributes into Ordered Discrete Attributes, in Kodratoff Y.(ed.), EWSL-91, Springer, Berlin, 1991.

[Cheeseman 90] Cheeseman P.: On Finding the Most Probable Model, in Shrager J., Langley P.(eds.): *Computational Models of Discovery and Theory Formation*, Morgan Kaufmann, Los Altos, CA, 1990.

[Chiu et al. 91] Chiu D., Wong A., Cheung B.: Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis, in Piatetsky-Shapiro G. & Matheus C.J., Knowledge Discovery in Databases:, MIT Press, 1991.

[Dietterich & Bakiri 95] Dietterich T.G., Bakiri G.: Solving Multiclass Learning Problems via Error- Correcting Output Codes, Journal of AI Research, Vol.2, pp. 263-286, 1995.

[Forsyth 93] Forsyth R.S.: Overfitting Revisited: An Information-Theoretic Approach to Simplifying Discrimination Trees, in JETAI 6(3), 1994.

[Kerber 92] Kerber R.: ChiMerge: Discretization of Numeric Attributes, in Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Menlo Park, pp.123-128, 1992.

[Lee & Shin 94] Lee C., Shin D.-G.: A Context-Sensitive Discretization of Numeric Attributes for Classification Learning, in Cohn A.G.(ed.), Proceedings of the 11th European Conference on Artificial Intelligence (ECAI94), John Wiley & Sons, Chichester, pp.428-432, 1994.

[Murphy & Aha 94] Murphy P.M., AhA D.W.: UCI repository of machine learning databases. For information contact ml-repository@ics.uci.edu.

[Pfahringer 95] Pfahringer B.: A New MDL Measure for Robust Rule Induction (Extended Abstract), 8th European Conference on Machine Learning (ECML95), 1995.

[Quinlan & Rivest 89] Quinlan J.R, Rivest R.L.: Inferring Decision Trees using the Minimum Description Length Principle, in Information and Computation, 80:227-248, 1989.

[Quinlan 93] Quinlan J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.

[Richeldi & Rossotto 95] Richeldi M., Rossotto M.: Class-Driven Statistical Discretization of Continuous Attributes (Extended Abstract), ECML95, Iraklion, Greece, (in press), 1995.

[Rissanen 78] Rissanen J.: Modeling by Shortest Data Description, in Automatica, 14:465-471, 1978.

[Wallace & Patrick 93] Wallace C.S., Patrick J.D.: Coding Decision Trees, Machine Learning, 11(1), 1993.

Size	Method	0% Noise	10% Noise	20% Noise
100	C4.5	1.6±0.4	1.5±0.0	1.5±0.1
	D-2	0.9±0.0	0.9±0.1	0.9±0.0
	MDL	0.9±0.0	0.9±0.0	0.9±0.0
250	C4.5	1.7±0.1	1.8±0.1	1.9±0.1
	D-2	1.0±0.0	1.0±0.0	1.0±0.0
	MDL	0.9±0.0	0.9±0.0	1.0±0.0
500	C4.5	2.0±0.1	2.4±0.3	2.5±0.1
	D-2	1.1±0.1	1.1±0.2	1.1±0.0
	MDL	1.1±0.2	1.1±0.1	1.2±0.1
1000	C4.5	2.7±0.2	3.7±0.1	4.3±0.3
	D-2	1.3±0.1	1.5±0.1	1.6±0.3
	MDL	1.2±0.2	1.5±0.0	1.5±0.1
2500	C4.5	4.3±0.2	8.3±0.3	10.3±0.5
	D-2	1.8±0.1	2.2±0.2	2.3±0.3
	MDL	1.6±0.1	2.6±0.4	2.6±0.1
5000	C4.5	7.9±0.2	20.5±1.1	26.1±1.2
	D-2	3.1±0.2	4.1±0.7	4.0±0.4
	MDL	2.5±0.2	4.5±0.1	4.8±0.2
10000	C4.5	15.4±1.0	62.0±0.5	75.9±0.5
	D-2	5.5±0.9	8.2±0.8	8.7±0.2
	MDL	4.1±0.2	8.8±0.2	8.7±0.2
25000	C4.5	38.6±1.2	405.4±7.3	488.9±22.7
	D-2	17.0±2.7	20.3±2.8	20.4±5.8
	MDL	11.3±0.9	30.1±4.1	26.0±0.3
50000	C4.5	75.9±5.7	1507.6±42.1	2063.7±100.9
	D-2	31.2±0.9	45.3±8.1	40.9±2.3
	MDL	24.2±2.8	58.8±1.5	59.4±0.6

Table 1: Demon: Average runtimes (in seconds) and standard deviations C4.5 using the raw data, using data discretized by D-2 and using data discretized by MDL-Disc, for various training set sizes and levels of class noise.

Size	Method	0% Noise	10% Noise	20% Noise
100	C4.5	11.4±3.8	21.1±5.3	27.7±6.1
	D-2	17.8±5.9	25.6±6.7	30.9±8.0
	MDL	14.6±4.5	21.8±4.3	25.6±4.8
250	C4.5	4.4±2.1	10.9±4.3	19.8±4.1
	D-2	20.0±7.7	22.3±6.6	21.7±5.8
	MDL	8.3±7.8	11.1±2.9	15.4±5.5
500	C4.5	1.8±0.9	4.7±1.8	17.5±3.9
	D-2	12.6±11.1	16.5±10.5	19.7±7.9
	MDL	2.3±1.4	5.4±2.4	10.4±4.0
1000	C4.5	1.1±0.4	3.7±2.0	11.6±3.0
	D-2	13.5±8.1	15.8±9.9	19.5±8.9
	MDL	1.7±0.7	1.8±0.8	4.4±2.8
2500	C4.5	0.3±0.2	1.0±0.5	8.0±1.0
	D-2	12.7±11.1	17.0±11.1	12.7±8.6
	MDL	0.9±0.5	0.8±0.4	1.6±0.6
5000	C4.5	0.1±0.1	0.7±0.3	7.2±2.0
	D-2	20.2±4.7	10.6±7.2	15.7±7.2
	MDL	0.5±0.3	0.8±0.3	1.0±0.4
10000	C4.5	0.1±0.1	0.3±0.2	4.1±0.8
	D-2	13.0±9.8	12.8±8.3	15.8±10.7
	MDL	0.2±0.0	0.7±0.5	0.5±0.2
25000	C4.5	0.1±0.1	0.1±0.0	1.8±1.0
	D-2	17.3±10.5	11.5±8.6	23.3±6.2
	MDL	0.2±0.2	0.5±0.1	0.2±0.2
50000	C4.5	0.0±0.0	0.1±0.1	1.0±0.4
	D-2	19.0±4.3	14.1±2.1	21.5±3.5
	MDL	0.1±0.0	0.1±0.1	0.1±0.1

Table 2: Demon: Average predictive errors (percentages) and standard deviations for C4.5, D-2, and MDL-Disc, for various training set sizes and levels of class noise.

Size	Method	0% Noise	10% Noise	20% Noise
100	C4.5	17±2	20±3	28±7
	D-2	18±3	18±3	16±5
	MDL	16±2	16±4	18±8
250	C4.5	21±2	33±10	47±14
	D-2	28±5	26±6	24±6
	MDL	18±3	20±3	26±8
500	C4.5	23±5	31±8	87±24
	D-2	25±6	28±7	32±7
	MDL	21±2	21±2	30±12
1000	C4.5	23±5	48±19	127±29
	D-2	33±12	38±8	42±10
	MDL	20±2	22±3	26±5
2500	C4.5	21±6	38±12	204±50
	D-2	31±10	39±15	38±13
	MDL	20±1	21±2	22±2
5000	C4.5	23±6	47±18	412±126
	D-2	33±8	25±7	34±9
	MDL	21±2	20±1	21±1
10000	C4.5	25±4	38±20	452±35
	D-2	41±26	40±13	37±11
	MDL	21±0	20±1	21±3
25000	C4.5	24±7	39±15	368±179
	D-2	38±17	24±4	40±10
	MDL	20±1	20±1	19±0
50000	C4.5	18±2	51±10	694±152
	D-2	40±8	21±0	34±4
	MDL	20±1	20±1	20±1

Table 3: Demon: Average tree sizes (number of nodes) and standard deviations for C4.5, D-2, and MDL-Disc, for various training set sizes and levels of class noise.

Domain	Examples	Nominal Attrs	Continuous Attrs	Base Acc
Credit App (crx)	690	9	6	55.5
Diabetes	768	0	8	65.1
Heart	270	6	7	55.6
Ionosphere	351	0	34	64.1
Liver disorder	345	0	6	58.0
Thallium scan	160	0	45	65.0

Table 4: Characteristics of the various natural domains used. Baseline accuracy (`Base Acc`) is the percentage of examples belonging to the majority (default) class.

Domain	Method	Runtime	Error	Size
CRX	C4.5	1.6±0.1	13.5±2.3	42±11
	D-2	1.2±0.2	15.1±1.3	39±14
	MDL	1.1±0.1	14.0±1.5	28±11
Diabetes	C4.5	2.1±0.1	27.7±1.8	96±13
	D-2	1.4±0.0	28.0±1.5	81±10
	MDL	1.2±0.1	25.5±2.7	44±16
Heart	C4.5	1.0±0.0	25.5±5.7	34±5
	D-2	0.9±0.0	26.3±5.6	34±4
	MDL	0.8±0.0	23.9±4.1	25±7
Ionosphere	C4.5	3.7±0.2	10.6±1.3	22±3
	D-2	1.4±0.0	11.0±2.4	18±4
	MDL	1.4±0.1	10.9±2.8	20±5
Liver	C4.5	1.0±0.1	37.3±4.1	55±12
	D-2	0.9±0.0	36.3±3.2	49±8
	MDL	0.8±0.0	36.0±4.1	34±8
Scan	C4.5	1.4±0.2	24.6±6.2	19±3
	D-2	1.1±0.0	22.1±6.4	19±3
	MDL	1.0±0.0	22.1±6.5	18±4

Table 5: Results for various natural domains for C4.5, D-2, and MDL-Disc. Runtimes are in seconds, predictive error are percentages, and size is the size if the decision tree in number of nodes.