A Visual Interactive Framework for Attribute Discretization

Ramesh Subramonian, Ramana Venkata, Joyce Chen
Microcomputer Research Laboratory
Intel Corporation
2200 Mission College Blvd., MS RN6-35
Santa Clara, CA 95052-8119
{subramon, rvenkat1, qjchen}@gomez.sc.intel.com

Abstract

Discretization is the process of dividing a continuous-valued base attribute into discrete intervals, which highlight distinct patterns in the behavior of a related goal attribute. In this paper, we present an integrated *visual* framework in which several discretization strategies can be experimented with, and which visually assists the user in intuitively determining the appropriate number and locations of intervals. In addition to featuring methods based on minimizing classification error or entropy, we introduce (i) an optimal algorithm that minimizes the approximation introduced by discretization and (ii) a novel algorithm that uses an unsupervised learning technique, *clustering*, to identify intervals. We also extend discretization to work with continuous-valued goal attributes.

1 Introduction

Discretization is the process by which the range of a given *base* attribute (or independent variable) is partitioned into mutually exclusive and exhaustive intervals based on the value of a related *goal* attribute (or dependent variable), whose behavior we would like to understand or predict. While the goal is typically discrete/nominal-valued, we extend discretization to continuous goals.

The motivations for discretization are:

Summarization: it provides a high level overview of the behavior of the goal vis-a-vis the base, by partitioning the base into intervals within which the goal behavior is similar.

Decision Tree Splits: when a continuous attribute is used as a decision variable, it needs to be broken into regions, each of which constitutes a path out of a decision node.

Simplicity of learning: several learning algorithms restrict attributes to be discrete. Discretization allows us to convert continuous attributes into nominal-valued ones, and discrete attributes into intervals, albeit with some concomitant loss of information.

We refer the reader to (Dougherty, Kohavi, & Sahami 1995), for an excellent survey of the work on discretization. The following references constitute the most germane comparisons. (Fayyad & Irani 1993) use minimum entropy as a discretization metric and use the minimum-description-length principle to determine the appropriate number of intervals. (Kerber 1991) suggests that intra-interval similarity should be maximized and inter-interval similarity should be minimized. (Maass 1994) provides an efficient algorithm that minimizes classification error.

The input to our discretization algorithms could be either the raw data or a joint probability density function (pdf) of the base and goal derived from the data. Using the data eliminates errors introduced by estimating a pdf. However, using a pdf enables us to experiment with more meaningful error metrics. It allows experienced users to encode their intuitions in a disciplined fashion, by altering the shape and type of kernel used for density estimation. Further, it lends itself to a *client-server* architecture where density estimation could be done on the server (perhaps as a data-blade) and a compact representation shipped to the client to enable interactivity.

1.1 Contributions

We have:

1. built a visual framework (Figure 1) which allows experimentation with different discretization strategies, different numbers of intervals, cut-points other than those recommended by the algorithm, and different metrics to judge goodness of discretization (Section 5). We believe that the single most important contribution of this paper is its demonstration that the visually interactive paradigm of learning, which brings the user and machine into a tight loop, is both viable and useful.

2. introduced a new discretization algorithm based on clustering (Section 3).

3. formalized the notion that discretization is a form of approximation and provided an efficient algorithm that captures this intuition (Section 2).

4. provided a proof (by example) that the algorithm of (Fayyad & Irani 1993) does not find the best multi-interval split whereas our algorithm does (Section 2).

5. extended discretization to work with continuous goal attributes and demonstrated that doing so, rather than imposing artificial categories on the goal, can be advantageous (Section 4).

6. demonstrated that density estimation can be gainfully used for interactive discretization (Section 7).

2 Discretization to minimize difference

Intuition The effect of discretization is to approximate the goal behavior as being the same over an entire base interval. Hence, discretization *flattens* the joint pdf, $p_{G,B}$, of the goal G and the base B, to a constant value within each interval. This suggests that the best discretization is that which minimized the approximation introduced.

Given p and a set of intervals $\mathcal{I} = \{I_i\}$, one can define the approximation $q_{G,B}$ as follows: (i) $b_1, b_2 \in I_i \Rightarrow q_{G,B}(g, b_1) = q_{G,B}(g, b_2)$ and (ii) $\int_{b \in I_i} \sum_g p_{G,B}[g, b]db = \int_{b \in I_i} \sum_g q_{G,B}[g, b]db$. Hence, we want to find the set of intervals that minimizes the distance between p and q. To indicate its derivation from $\mathcal{I}$, we write q as $q(\mathcal{I})$.

Metrics used to measure the distance between the two pdfs, p and q, are:

1. Kulback-Leibler (KL) distance (Cover & Thomas 1991) $= D(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)}$

2. L-1 norm $D(p||q) = \int_b (\sum_g \mid p[g|b] - q[g|b] \mid) \, p[b]db$.

Our implementation offers minimization of **Class Information Entropy** (CIE) (Fayyad & Irani 1993) as an option. CIE $= -\sum_i q(b_i) \sum_j q(g_j|b_i) \log(g_j|b_i)$ where i spans the intervals, and j spans the values the goal can assume. It can be shown that minimizing KL-distance is equivalent to minimizing CIE.

Problem Statement For computational purposes, we evaluate p at n values of B chosen so that the integral of the density between adjacent points is roughly the same. Formally, let $p_{G,B}(g, b)$ be a pdf s.t. $g \in \{g_i\}, b \in \mathcal{B}_0 = \{b_i\}$ where $b_i \in \mathcal{R}, i < j \Rightarrow b_i < b_j, |\mathcal{B}_0| = n$. Given $\mathcal{B}_0$, define $\mathcal{C}_0 = \{\frac{b_i + b_{i+1}}{2}\}$. With any set of cutpoints, $\mathcal{C}$, one can define an associated set of mutually exclusive and exhaustive intervals, denoted as $\mathcal{I}(\mathcal{C})$.

Problem Definition 2.1 *Given a pdf p evaluated at a set of initial points $\mathcal{B}_0$, a candidate set of cutpoints $\mathcal{C}_0$, a desired number of cutpoints C, find $\mathcal{C}_{opt} \subseteq \mathcal{C}_0$ such that (i) $|\mathcal{C}_{opt}| = C$ (ii) $D(p||q(\mathcal{I}(\mathcal{C}_{opt})))$ is minimum.*

Algorithm SP We create a weighted, undirected graph where the nodes are the candidate cutpoints, $\mathcal{C}_0$, augmented by two distinguished vertices, a source S and a destination D representing the left and right extrema of the base. Each node i has a location loc_i such that $i < j \Rightarrow loc_i < loc_j$. We create edges $e_{i,j} = (i, j), i < j$. The weight, $w_{i,j}$ of $e_{i,j}$, is the distance between p and q over the interval spanned by e_{ij}. Recall that this distance is $D(p||q)$ conditioned on $b \in [loc_i, loc_j]$. Finding the optimal cutpoints is equivalent to finding the shortest path from S to D of a specified length, which is $C + 1$ in our case. This is easily done by dynamic programming. Note that $i < k < j \Rightarrow w_{ik} + w_{kj} \leq w_{ij}$. This reflects our intuition that the more the number of cutpoints, the less the discrepancy between p and q.

Theorem 2.1 *Algorithm SP solves Problem 2.1 in $O(n^2(C + 1))$ time using $O(n^2)$ space.*

Proof Sketch: The graph has $O(n^2)$ edges. Let $sp(i, j, l)$ be the shortest path from i to j of length l. Calculating $sp(S, D, k)$ requires $O(n)$ work to find the minimum of $sp(S, i, k-1) + sp(i, D, 1)$. Maintaining $sp(S, i, k)$ for all values of i requires $O(n^2)$ work for each value of k. Creating a single edge weight could require as much as $O(n)$ work, since it might span $O(n)$ intermediate nodes. However, carful organization and re-use of computations and amortizing the cost over all edges, allows us to compute all edge weights in $O(n^2)$ work. To see why this is true, observe that $sp(i+1, j, 1)$ can be easily derived from $sp(i, j, 1)$ by subtracting the influence of $sp(i, i+1, 1)$. The "subtraction" process requires $O(1)$ work but is too detailed for presentation here. □

Sub-optimality of greedy algorithms (Fayyad & Irani 1993)'s algorithm to discretize based on minimizing Class Information Entropy (CIE) is greedy, in that it preserves existing choices of cutpoints when searching for additional cutpoints. A simple example shows that this is a sub-optimal strategy. Let the data initially fall into 4 intervals and let the goal have 2 classes. Let $c_{i,j}$ be number of instances in interval I_i of class j. Let $c_{1,1} = 48, c_{1,2} = 16$, $c_{2,1} = 65, c_{2,2} = 53$, $c_{3,1} = 43, c_{3,2} = 73$, $c_{4,1} = 84, c_{4,2} = 70$. The best *one* cutpoint is between intervals I_1/I_2. The best *two* cutpoints are between intervals I_2/I_3 and I_3/I_4 yielding CIE = 0.67117 . A greedy algorithm, forced to retain its first cutpoint when looking for its second, could not find the optimal *two* cutpoints. Setting cutpoints between intervals I_1/I_2 and I_3/I_4 yields CIE = 0.67168. Setting cutpoints between intervals I_1/I_2 and I_2/I_3 yields CIE = 0.67222.

3 Maximizing adjacent differences

Another approach to discretization is suggested by the rationale that adjacent intervals ought to be as *dissimilar* as possible; else, they should not have been separate intervals at all. Kerber (Kerber 1991) calls the "defining characteristic of a good discretization" as "intra-interval uniformity and inter-interval difference." However, his Chimerge algorithm has the drawbacks that (a) it does not specify what objective function it is trying to optimize (b) the algorithm merges small intervals to achieve large ones. However, for small intervals, the χ^2-test, used to judge whether intervals ought to be merged, is imprecise.

Our approach uses *clustering* to partition the space (defined by the range of the base and goal attributes) into classes so as to minimize intra-class differences and maximize inter-class differences. We then project the class descriptions down onto a single axis, the base, to determine intervals. The advantage is that we leverage the robustness of clustering algorithms. We use a variant of AutoClass, which has performed well in many other domains (Cheeseman & Stutz 1996). We believe that this is an intersting application of unsupervised learning techniques for a supervised learning problem.

Our notion of clustering, explained below, hypothesizes that there exists a parametrized description of the data generation process that needs to be discovered by the clustering algorithm. The data is presented to the clustering algorithm which aims to find the *best* description. This typically involves a computationally expensive search.

3.1 The Finite Mixture Model (FMM)

The FMM postulates that the data is generated in the following fashion. Assume that the number of classes, J, is known. First, class probabilities, p_j are selected. Assume attributes are independent. To each attribute k of each class j is assigned a functional form $F_{j,k}$ with parameters $\{\gamma_{j,k}\}$ (e.g., a Gaussian and corresponding mean and standard deviation $\mu_{j,k}, \sigma_{j,k}$). Typically, the choice of functional forms is restricted to a few simple ones.

Instances are generated as follows. First select a class: class j is selected with probability p_j. The kth attribute of the instance is determined by drawing a random variable from $F_{j,k}$ and parameters $\{\gamma_{j,k}\}$.

3.2 Problem Statement

Given the data, the number of classes, $C+1$, and the types of functional forms permissible, the first problem is to find the best parameters (whether Maximum Likelihood estimates or Maximum A Posteriori estimates, if a Bayesian approach is preferred). Let $v_j(b) = F_j(b : \{\gamma_{j,B}\})$ be the probability that a random variable B drawn from the distribution $F_{j,B}$ with parameters $\{\gamma_{j,B}\}$ has values in $[b, b+db]$. For example, $N(x : \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$ Let $w_j(b)$ be the probability that an instance whose base value is b belongs to class j i.e., $w_j(b) = \frac{p_j v_j(b)}{\sum_j p_j v_j(b)}$. Let

$$K(b) = j' \text{ s.t. } \max_j \; w_j(b) = w_{j'}(b).$$

Problem Definition 3.1 *Given C and a FMM, the problem is to find a set of $C+1$ mutually exclusive and exhaustive intervals $\mathcal{I} = \{I_i\}$ such that $x_1, x_2 \in I_i \Rightarrow K(x_1) = K(x_2)$.*

We need to restrict the problem before we can solve it because:

1. the base attribute is continuous and needs to be discretized (else an infinity of conditions of the form $(\forall x \in I_i, K(x) = j')$ will need to be verified). We do so as in Section 2.

2. the projection of J classes on to a single axis does not always yield J intervals. This is because it may not be possible to satisfy the condition $(\forall x \in I_i, K(x) = j')$ for certain FMMs. For example, consider two equally weighted classes, one being $N(0,1)$ and the other being a bimodal distribution, consisting of $N(-2,1)$ and $N(+2,1)$. In this case, there exists no partitioning of the real number line into 2 intervals such that the condition $(\forall x \in I_i, K(x) = j')$ can be satisfied. Or one class may be subsumed by the other classes i.e. $\exists j \forall x K(x) \neq j$.

Problem Definition 3.2 *Given a FMM consisting of J classes and a candidate set of cutpoints $\mathcal{C}_0$, find the smallest set $\mathcal{C}_{opt} \subseteq \mathcal{C}_0$, (corresponding intervals are $\mathcal{I}(\mathcal{C}_0) = \{I_i\}$), such that $x_1, x_2 \in I_i \Rightarrow K(x_1) = K(x_2)$.*

Theorem 3.1 *Problem 3.2 can be solved in $O(|\mathcal{C}_0|J)$ time.*

Proof Sketch: Classifying a cutpoint into one of J classes using a single attribute (the base) requires $O(J)$ work. We then make a single pass over the $\mathcal{C}_0$ cutpoints, from left to right, accumulating adjacent cutpoints with the same classification into an interval. □

4 Continuous goal attributes

Most existing work on discretization has been confined to discrete goal attributes i.e., where each instance has a nominal valued *class* attribute. A continuous valued goal attribute would have to be discretized (e.g., *return on investment* could be mapped to {*High, Low* }) to fit into existing paradigms. We suggest that there are situations where it is preferable not to coerce an intrinsically continuous goal into discrete classes prior to discretizing the base e.g., discretizing *age* based on *dollar value of insurance claims paid.*

To test this hypothesis, we discretized a synthetic data set created using the FMM (Section 3.1) with 2 attributes (attribute 1 is base and attribute 2 is goal) and 3 classes ($J = 3, K = 2$) and parameters:

$$\forall j\ p_j = 1/J,\ \forall j \forall k\ \sigma_{j,k} = 1$$

$$\mu_{1,1} = -2, \mu_{2,1} = 0, \mu_{3,1} = 2$$

$$\mu_{2,1} = -2, \mu_{2,2} = 2, \mu_{3,2} = 4$$

Discretizing the goal as Positive and Negative and setting the number of intervals to be 2 yielded cut-points of -2.174 and 0.157. However, without discretizing the goal, we got cut-points of -1.0012 and 1.0949. We believe that these cut-points correctly capture the difference between the positive and *more* positive instances which is washed out if the goal is discretized. However, if the goal were discretized as Negative ($x < 0$), Positive ($0 \leq x < 2$) and More-Positive ($2 \leq x$), we get cut-points of -1.425 and 1.405.

While the algorithms of Sections 2 and 3 extend naturally to continuous goals, extending the minimum error discretizer is not as straight-forward. This is because it requires one to quantify the error associated with an instance having a goal value of x when the average goal value in the interval is y. In contrast, measuring the error of classifying an instance as *blue* when it is in fact *red* is simple. None of the error metrics we considered seemed quite right.

5 System Description

We briefly describe some salient features of **Id-Vis**, our interactive, visual data-mining tool. The GUI is built using Tcl/Tk (Ousterhout 1994) and Tix (Lam 1996). Id-Vis currently runs under Windows *NT** 4.0 on a 200 MHz. Pentium(R) Pro Processor.

Some of the key tenets of our design philosophy are:

Multiple Portals Users can interact with the system at their level of expertise and comfort. The system provides the user with information and *manipulation handles* at the level of detail specified. For example, the user can go so far as to alter the pdfs by tweaking the kernel width used for the density estimation, or can let the system choose all parameters, including the discretization type, and the number and locations of the cut-points.

Client-Server Partitioning The database resides on, and operations thereupon take place on, the *server*, while an appropriate *distillation* of the data is sent to the *client*, where all of the visual, GUI-based operations are performed at interactive speeds. When necessary during drill-down, requests are made of the server for more narrowly-focused data.

User Primacy User-chosen values are *always* given priority over system-derived values. The latter, which are optimum by some metric, continue to be available as fall-back options.

Interactivity Immediate feedback is provided during user experimentation, in easy-to-read graphical modes. *Knowledge* gained from this process is always available for immediate recall.

Portability All object creation/display operations are written in Tcl/Tk/Tix, while all speed-critical operations are implemented in ANSI C.

Each analysis tool in **Id-Vis** presents the user with a graphical interface containing a set of interactive controls, display windows which project an immediate feedback to the manipulation of these controls, and a means to propagate the user-chosen values and parameters downstream.

Figure 1 shows the interface to the Discretizer tool. The top pane contains the interactive controls for modifying the type of discretization, the type of *badness* measure (see Section 6 for definition) employed to evaluate the results, the number and locations of the cut-points, and the loss functions associated with the various goal categories, for a discrete goal.

The bottom pane contains the display windows that provide the feedback. In this pane, the top left window displays the results of the discretization process, i.e. the base discretized into intervals, each with similar goal behavior. To start with, the system shows the *optimal* (i.e. with the least badness value) locations for the cut-points, for the *Min. Classification Error* type of discretization and *classification error* as the badness measure. Within each interval are shown the relative proportions of the goal categories, weighted by the loss functions. This display window is interactive, and provides controls for the user to add, delete and move the cut-point locations to satisfy specific user constraints or curiosity. Also, using the controls in the

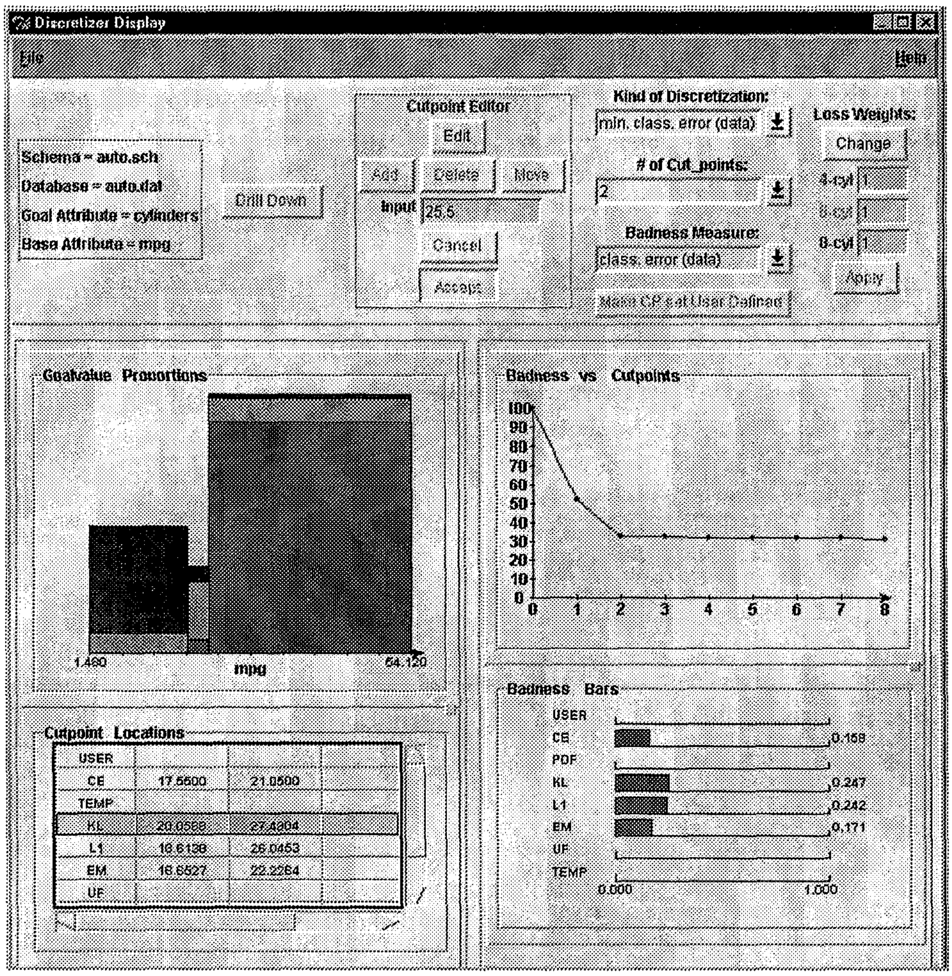

Figure 1: *Id-Vis Discretizer Window*

top pane, the user can experiment with other types of discretization algorithms. The bottom left display shows the locations of the *valid* cut-points for each type of discretization that the user tries out.

The bottom right display shows the chosen badness measure's values, for the current number of cut-points, over all discretization types. Blank fields indicate that the computation is not yet performed. When the user starts experimenting with the number and/or the locations of the cut-points, this window provides a feedback on how the user-chosen values are performing relative to the system-derived optima. This serves as an aid to users in determining the extra cost involved in their choice.

The central philosophy is that the system should place the user within the appropriate context (e.g. in the discretizer window, display system-derived number and locations of the cut-points) and provide the user with the opportunity to *intelligently* modify (e.g. display the impact on accuracy of these modifications) these optima and propagate them downstream.

5.1 "To cut, or not to cut"

The top right display window in the Discretizer interface displays the behavior of the chosen badness measure, for the current type of discretization, as the number of cut-points increases. This is very useful to the user in determining the number of cut-points that best discretize the data. All of our discretization strategies show a marked drop-off in the incremental reduction in the badness measure, when the number of cut-points is increased beyond that which is logical for the data. It is visually apparent to the user in a compelling manner. We will discuss more about this in Section 7.

5.2 Client-Server Interactions

When the server ships the data to the client (either initially, or in response to a change in the discretization type etc.), it executes the following steps: 1. after sorting the data by the base values, it streams this raw data past a *compactor* which keeps track of the goal behavior and assembles large regions tagged with a summary description of the regional contents. e.g. region 3 has 150 instances of red-category goals, 20 of green-category and 15 of blue-category. The explicit locations of these instances *within* a region are *not* tracked; 2. if the client needs the data in a non-pdf format (as at the start), these regions, along with their tags, are directly shipped to the client. Otherwise, the *original* data is streamed past a *density-estimator*, which generates a probability density function; 3. this pdf is evaluated at uniformly spaced points (say, 20000) and the results are sent to a *distiller*. It then re-evaluates the pdf such that the integral of the density between adjacent points (say, 1000) is equi-probable; 4. this distilled pdf is finally shipped to the client.

As mentioned earlier, the top left display in the Discretizer window shows the relative proportions of the goal categories within each interval. If more detailed information is desired, the user can click on the *Drill Down* button in the top pane. The client then asks

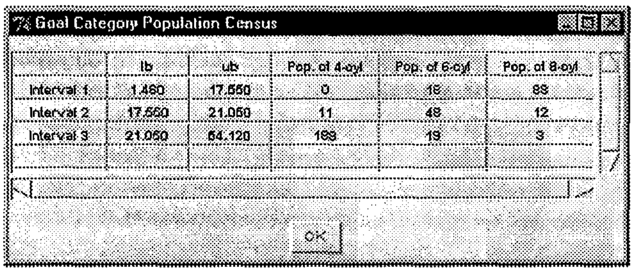

Figure 2: *Drill Down Results*

the user about the accuracy constraints on the request If these can be met using the locally available data, the client generates the necessary population data. Else, the drill-down request, along with the cut-point locations, is transmitted to the server. Depending on the accuracy required, the server computes the population figures within each interval either from the existing regions, or regenerates the regions appropriately. This data is then displayed on the client desk-top in a pop-up window (Figure 2). In future versions, we intend to furnish the user with the estimated costs of meeting the various levels of accuracy, before the user makes the choice.

6 Experiments

We have experimented with several discretization strategies, that differ both in the intuitions behind them as well in their mathematical development. In this section, we report on our experiments using these strategies on different data sets.

		CE	*KL*	*L1*	*CIE*	*EM*
DS1	CP1	17.65	20.24	18.97	20.24	16.68
	CP2	21.05	27.93	26.25	27.93	22.29
DS2	CP1	0.111	0.173	0.125	0.173	0.005
DS3	CP1	80.7	81.8	72.8	81.8	70.1
	CP2	100.9	153.9	146.8	153.9	128.6

Table 1: Cut-Point locations

The following is a brief description of the data-sets upon which the experiments were performed. The **Auto-Mpg (DS1)** (Merz & Murphy 1996) contains 398 instances concerning city-cycle fuel consumption, with each instance describing 6 continuous and 2 discrete attributes. We pruned it to 385 to focus on 4/6/8-cylinder cars. The **Synthetic Data-set 1 (DS2)** contains 1000 instances obtained from a pdf consisting of two overlapping normals, with each instance describing a discrete goal over a continuous base. The **Synthetic Data-set 2 (DS3)** contains 5019 instances concerning the relation between the types of automobiles owned and the family incomes.

Figure 3 shows, for the Auto-Mpg data-set, the behavior of the *badness* measure (i.e., the *approximation* introduced by partitioning the base range into intervals and assuming similar goal behavior throughout each interval) as the number of cut-points increases. For each type of discretization, only the correspondingly appropriate badness measure is shown, scaled such that the zero cut-point value = 100%. Obviously, the badness value decreases as the number of intervals allowed increases, reaching the zero value when the number of intervals equals the number of instances in the data-set. However, in practice, the desired number of intervals (for decision-support etc.) is rarely more than 6. For most data-sets, we can discern the logical number of cut-points from this figure, as the number when the incremental reduction in the badness (with increase in the number of cut-points) sharply falls off.

Table 1 shows the cut-point locations, as returned by each type of discretization for the three data-sets, in each case for the logical number of cut-points.

Columns 3, 4, 5, 6 show the cut-points found by using Classification Error *CE*, KL-distance *KL*, L1-distance *L1*, Class Information Entropy *CIE*, and Maximum Likelihood Estimate *EM*. While *DS2* is best discretized with one cut-point (CP1), both *DS1* and *DS3* are best discretized with two cut-points (CP1, CP2).

7 Discussion

Figure 3 shows quite clearly the rapid drop-off in the badness-measure reduction, with increasing number of

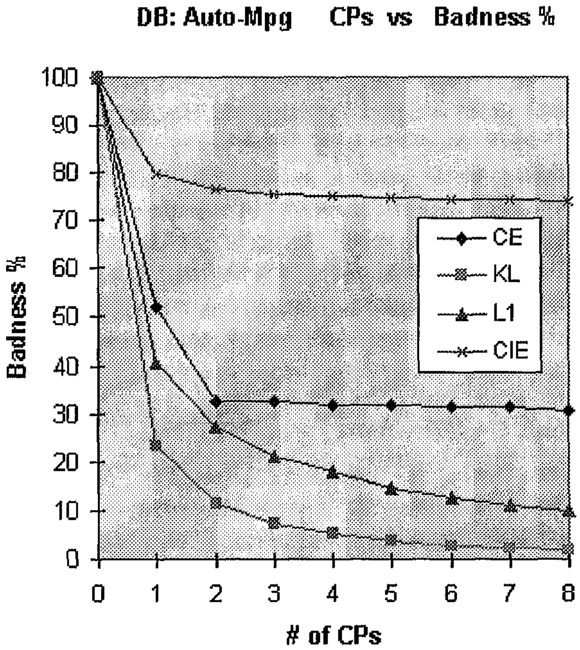

Figure 3: *Badness Measure Behavior*

cut-points, beyond the number logical for the data. We feel that this figure provides a very intuitive and convenient means of assessing the number of cut-points that best discretize the attribute. In all the cases tested, both the *CE* and the *CIE* metrics show a dramatic and very conclusive plateauing of their badness measures, beyond the appropriate number of cut-points. The other measures also show a pronounced decrease in the incremental reduction of the badness measure beyond the appropriate number of cut-points.

All of the discretization strategies show substantive agreement in the locations of the cut-points. The *CE* and *EM* metrics return values that are in close agreement, both being based on the raw data. Note that the cut-point locations returned by *KL, L1* and *CIE*, while being based on the pdfs with all of the concomitant benefits, are also close to the above values for all practical purposes.

Discretization based on clustering *EM* often does not return a valid set of cut-points for cut-point numbers larger than this logical choice. For example, in the Auto-Mpg case, this algorithm returns only up to 2 cut-points. So, unlike the other types of discretization, this method doesn't have a one-to-one correspondence between the number of cut-points asked for and the number returned.

8 Conclusion and Future Work

We have created a novel framework for interactive discretization, which situates the user in an appropriate context and then provides visual controls to modify the *knowledge* gained and obtain graphical feedback of the consequences of doing so. We have incorporated two novel algorithms and have extended discretization to handle continuous goal attributes.

An important area for future work is the partitioning of data and computation between the client and server so as to enable visual interactivity without compromising scalability.

9 Acknowledgments

We would like to thank Richard Wirt, Dave Sprague, Bob Dreyer and the Microcomputer Laboratory at Intel Corporation for providing support for this work. Other trademarks are the property of their respective owners.

References

Cheeseman, P., and Stutz, J. W. 1996. Bayesian classification (autoclass): Theory and results. In et al, U. M. F., ed., *Advances in Knowledge Discovery and Data Mining.* AAAI Press/MIT Press.

Cover, T., and Thomas, J. 1991. *Elements of Information Theory.* Wiley and Sons.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, 194–202.

Fayyad, U. M., and Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1022–1027.

Kerber, R. 1991. Chimerge: Discretizaion of numeric attributes. In *Proceedings of the 10th National Conference on Artificial Intellgence*, 123–128.

Lam, I. K. 1996. *The Tix Programming Guide.* http://www.xpi.com.

Maass, W. 1994. Efficient agnostic pac-learning with simple hypotheses. In *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory*, 67–75.

Merz, C. J., and Murphy, P. 1996. *UCI Repository of learning databases.* http://www.ics.uci.edu/mlearn/MLRepository.html.

Ousterhout, J. K. 1994. *Tcl and the Tk Toolkit.* Addison Wesley.