Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A DISTRIBUTION SENSITIVE CLUSTERING METHOD FOR NUMERICAL VALUES

W. W. Chu
K. Chiang

March 1993
CSD-930006

A Distribution Sensitive Clustering Method for Numerical Values *

Wesley W. Chu and Kuorong Chiang †
Computer Science Department
University of California, Los Angeles

Abstract

In this paper we propose a distribution sensitive clustering method (DISC) for numerical values. The purpose of DISC is to discover interesting and relevant high level concepts hidden in the underlying data. These concepts may be organized into an *abstraction hierarchy*, which can then be used for query modification in cooperative query answering. The clustering problem is formulated as a search for optimal clustering among all possible clusterings. To guide the search for optimal solutions , DISC uses *relaxation error* as a heuristic utility measure that quantifies the "goodness" of clustering. Experiments have been performed to demonstrate the effectiveness of DISC. In particular, comparison to the Maximum Entropy clustering method shows that DISC is more effective in high level concepts discovery with same degree of computation complexity.

1 Introduction

Conventional query processing in database management systems answers a query by listing all the tuples that satisfy the query conditions. If no tuples satisfy the query conditions, a null answer is returned. The null answer may be interpreted by the user in several different ways. For example, the user may think that the query is correctly posted, but there are no tuples in the database that satisfy the query conditions. Or the user may think that the query is not correctly posted based on some erroneous presuppositions[11]. A database system can be made to be much more user-friendly, or more "cooperative," if extra information is provided in case a null answer is returned. For example, the system may inform the user that the query is not correctly posted due to violation of some integrity constraints.

Another problem with the conventional database systems is that the user is required to define a query in the exact terms of the underlying data schema. Thus, the user must understand the data schema in order to satisfactorily solve a problem. Due to normalization of schema, this understanding is difficult.

Recently, several approaches have been introduced to deal with the above problems. In particular, a knowledge-based approach that provides *cooperative query answering* was used to develop CoBase, a cooperative database system [3, 2]. Cooperative query answering is capable of providing general, neighborhood, and associative information that is relevant to queries. This capability is facilitated by a knowledge structure called *type abstraction hierarchy*. The hierarchy organizes a set of *types* [1] where a type at a higher position is said to be more generalized than a type at a lower position, and the latter is said to be more specialized than the former. Operations are provided for generalization (moving up the hierarchy), specialization (moving down the hierarchy), and association (moving between hierarchies).

Specifically, to eliminate the ambiguity of null answers, CoBase returns approximate answers when exact answers are not available. CoBase derives approximate answers by relaxing (generalizing) query conditions according to the type abstraction hierarchy. To relieve the user from having to understand detailed schema and data semantics, CoBase provides conceptual querying capability, making high level problem solving much easier.

*This work supported in part by DARPA contract N00174-91-C-0107

†The authors may be reached at {wwc,kuorong}@cs.ucla.edu

[1]These types are usually called *concepts* or *categories* in the area of conceptual clustering. We shall use these terms interchangeably when the context permits.

One of the key issues in the development of CoBase is to generate type abstraction hierarchies for various domains. For small domains, abstraction hierarchies can be generated manually. For large databases, however, automatic generation of type abstraction hierarchies is necessary. Therefore, a clustering method is proposed in this paper that can automatically generate type abstraction hierarchy for a single numerical domain.

Basically, clustering of numerical values can be done in two ways. First, we start from one cluster, then we find "cuts" to derive more clusters. We repeat this process until a pre-specified criteria is satisfied. Second, we start with n clusters, with n being the number of distinct values, then we "fuse" clusters together until, again, a criteria is met. In this paper, we shall propose a methodology based on the first top-down cutting approach.

The problem of clustering numerical values is formulated as a search for a certain number of cuts such that the resulting clustering is "optimal" in some sense. There are two basic issues: (1) determine the number of cuts, and (2) determine the cuts. In this paper, we assume that the number of cuts is given (either by the user or some other methodology). We develop the notion of *relaxation error* as a measure to quantify the "goodness" of clusterings. The optimal clustering is the one that minimizes relaxation error.

Common heuristics based either on value distribution or frequency distribution (but not both) are not adequate. We shall use a biggest gap (BG) method and a maximum entropy (ME) method to illustrate this [15, 1]. BG method is based on the value distribution of data, and will always find the cuts at the largest gaps. ME method is based on the frequency distribution of data, and the cuts maximize entropy. The inadequacy of both methods can be shown by using the following set of numbers as an example.

(7,9,13,14,17,30,37,40,45,47,50,75,75,100,100)

Suppose we want to find two cuts. Using the BG method, the cuts will be between 50,75 and between 75,100. In both cuts, the gaps are the biggest (25). As can be seen, the numbers of values in each cluster are 11, 2, and 2 respectively. This causes the loss of much information due to the skewed frequency distribution – the less evenly distributed, the more loss of information. On the other hand, by ME method, the cuts will be between 17,30 and between 47,50. Each resulting cluster contains exactly the same number of values. However, the result can be improved if the cut 47,50 is shifted to 50,75. This shows that ME method is not sensitive to the value distribution, and may perform poorly for skewed distributions.

Ideally, we would like the clustering scheme to have both the advantages of the BG method and the ME method. That is, the resulting clusters have more or less an even number of values in each cluster (such that the information loss due to clustering is minimal), and at the same time locate cuts at bigger gaps. Therefore, a new measure *relaxation error* which simultaneously considers both frequency and value distribution is introduced in this paper. This will be used to develop a new method DISC to cluster attribute values which minimizes the relaxation error.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 introduces the notion of relaxation error and its evaluation. Section 4 presents the algorithm of DISC. Section 5 compares relaxation error of DISC with that of ME. Section 6 develops algorithms for determining optimal cuts with minimum relaxation error. And section 7 presents empirical results of DISC based on a large transportation database. Finally, section 8 gives our conclusion and plans for future work.

2 Related Works

Currently, most applications of numerical value clustering are in the areas of statistical pattern recognition and empirical machine learning.

In statistical pattern recognition, three commonly used classification techniques are the Bayesian classifier, linear discriminants, and nearest neighbor methods [14]. Bayesian classifiers aim at minimizing the overall error rate, thus building an optimal classifier. A new evidence (instance) e is classified to a class C_i such that for all $j \neq i$, $p(C_i|e) > p(C_j|e)$. Although theoretically optimal, Bayesian classifiers are not practical since the direct application of the above rule requires an enormous sample size to cover all possible combinations of evidence. Linear discriminants use a linear combination of the evidence to separate among the classes and to select a class for a new instance. They are quite simple in structure, but their error rate is higher than those of the Bayesian classfiers. For them to obtain best results, it is required that the data distribution be known (e.g., normal distribution). In contrast, nearest neighbor methods need not know the data distribution. For

a new instance, its distances to every other instances are calculated, and the nearest neighbor is determined by comparing the distances. The class containing the nearest neighbor is selected for the new instance. To calculate distance, three commonly used measures of closeness are (1) absolute distance, (2) Euclidean distance, and (3) various normalized distances.

Empirical machine learning and classification systems learn from pre-classified examples to derive rules that can be used for classifying future instances. The rules are usually organized in a decision tree [12]. To construct the decision tree, an attribute is selected at each node. The criteria for attribute selection is based on minimization of some heuristic measures, e.g., information entropy. For this purpose, an attribute with numerical values must be clustered (discretized) before it participates in selection process [4].

Both the statistical pattern recognition and empirical machine learning methods described above are *supervised*: given the category membership of examples, the learning systems derive rules that summarize the commonality among members of the same category and differences among competing ones. These rules can then be used to classify future instances. In contrast, the clustering method proposed in this paper, DISC, is an *unsupervised* learning method. Category membership information is not available, and DISC discovers meaningful categories in the data.

DISC is a special form of *conceptual clustering* where only a single numerical attribute is used to describe concepts. In most current conceptual clustering systems, concepts are described by a set of attributes (features) [5, 6, 7, 9, 13]. In these systems, objects that are similar to one another are clustered into the same category. Similarity among objects is usually defined based on inter-relationships among attributes that describe the objects. The meaning of similarity becomes unclear if we specialize its definition to a single attribute. In addition, these systems treat numerical values as categorical – order between numerical values is usually ignored. This is unacceptable for cooperative query answering, especially for deriving approximate answers. Therefore, these methods are not suitable for generation of type abstraction hierarchies.

An approach related to DISC for clustering similar attribute values was proposed in [10]. In this approach, if-then rules are generated from database for *pattern-based knowledge induction* (PKI). However, PKI only works well for discrete non-numerical domains. Therefore, continuous numerical domains will have to be pre-clustered in order to apply PKI algorithm.

3 Relaxation Error

In this section, we shall introduce a new notion called *relaxation error* to measure clustering quality. To explain relaxation error, let us consider the following scenario. Given a cluster C = $(x_1, x_2, ..., x_n)$, and a query with a condition "$X = x_i$" where x_i is a value in C. If there is no answer for this query, then CoBase relaxes the query condition from "$X = x_i$" to "$x_1 \leq X \leq x_n$" to derive approximate answers. The "goodness" of the approximate answers is measured by the difference between the exact value x_i and the approximate values x_j that are returned by CoBase. The sum of these differences weighted by the probability of the occurrence of each value is called the relaxation error of x_i, RE(x_i). The smaller the relaxation error, the better the approximate answer is. Formally, let f_j denote the frequency of occurring for x_j, then the probability of returning x_j by CoBase is $p_j = \frac{f_j}{N}$ where $N = \sum_{j=1}^{n} f_j$. In database, f_j is the number of tuples that have $X = x_j$, and N is the total number of tuples in a table or view. Thus,

$$RE(x_i) = \frac{1}{N} \sum_{j=1}^{n} f_j \cdot |x_i - x_j| \tag{1}$$

We can extend the above definition for the entire cluster. The relaxation error of a cluster C, RE(C), is defined as the mean relaxation error for all the values in C. That is,

$$RE(C) = \sum_{i=1}^{n} q_i RE(x_i) \tag{2}$$

where q_i is the probability of x_i being queried. q_i can be approximated by $\frac{f_i}{N}$ or provided by domain experts. In this paper, we assume the probability of x_i being returned, p_i, and the probability of x_i being queried, q_i, are the same and equal to f_i/N.

If the cluster C is partitioned into s sub-clusters, we can define the relaxation error of a clustering $K_s = \{C_1, C_2, ..., C_s\}$ with s sub-clusters as

$$RE(K_s) = \frac{1}{N} \sum_{i=1}^{s} f_{C_i} RE(C_i) \tag{3}$$

where $f_{C_i} = \sum_{i \in C_i} f_i$. Substituting (1) into (2), we have

$$RE(C_i) = \sum_{x_j \in C_i} \frac{f_j}{N} \sum_{x_k \in C_i} \frac{f_k}{f_{C_i}} |x_j - x_k| \tag{4}$$

Substituting (4) into (3), we have

$$RE(K_s) = \frac{1}{N^2} \sum_{i=1}^{s} \sum_{x_j \in C_i} \sum_{x_k \in C_i} f_j f_k |x_j - x_k|. \tag{5}$$

Notice that $RE(K_s)$ is the mean relaxation error for all values in C, with the constraint that values in different clusters do not contribute to each other's relaxation error. This implies that approximate answers are not allowed to "cross the cluster boundaries." Thus, $RE(K_s)$ decreases as the number of sub-clusters, s, increases. Thus, the more clusters K has, the smaller the mean relaxation error $RE(K_s)$ is. Therefore, to compare relaxation error for different clusterings, we need to use the same number of sub-clusters s.

4 The DISC Algorithm

For a given number of cuts k, DISC uses (5) to evaluate all the possible k cuts, and selects the one with the minimum relaxation error. Obtaining this global optimal solution takes $O(n^k)$ time to complete. This is not practical for a large k. Therefore, a greedy strategy is used to derive sub-optimal solutions where the best cut is determined first, then the second best cut is determined while keeping the best one fixed, and so on. Using this strategy, DISC is presented as follows.[2]

Algorithm DISC(C,k)
DISC finds k cuts to cluster C into k+1 sub-clusters
    NumCut = 0
    repeat
        Call **BestCut** to find the best cut c
        Partition C at c /* remove c from legitimate cuts */
        NumCut = NumCut + 1
    until NumCut = k

5 Relaxation Error of DISC and ME

BG and ME are methods that use simple heuristics for clustering. Since in general BG yields a larger relaxation error than that of ME, we shall only compare DISC with ME.

For ease of presentation, let the (probability) distribution of data be continuous. To further simplify the argument, we shall only consider the behavior of DISC and ME for a single cut. First we present the continuous versions of the definition of relaxation error. Consider figure 1, where $C = \{x | a \leq x \leq b\}$ is partitioned at c into two sub-clusters $C_1 = \{x | a \leq x \leq c\}$ and $C_2 = \{x | c \leq x \leq b\}$. The relaxation error of C is

$$RE(C) = \int_a^b \int_a^b p(x_1) p(x_2) |x_1 - x_2| dx_1 dx_2. \tag{6}$$

[2]The discussion of an efficient implementation of the procedure **BestCut** is postponed until section 6.

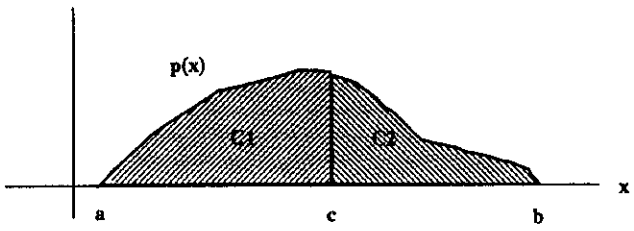

Figure 1: A distribution partitioned at c

Notice that in (6) the difference of every pair of values $|x_1 - x_2|$ is added to the summation *twice*. Therefore, equation (6) can be rewritten as

$$RE(C) = 2\int_a^b \int_{x_1}^b p(x_1)p(x_2)(x_2 - x_1)dx_2dx_1. \tag{7}$$

Apply integration by parts first to x_2, then to x_1, we have

$$RE(C) = 2\int_a^b F(x)[1 - F(x)]dx \tag{8}$$

where $F(x) = \int_a^x p(x)dx$. (We may interpret F(x)[1-F(x)] as the contribution by x to the overall relaxation error. And the relaxation error is the sum over all these contributions.) For the sub-clusters C_1 and C_2, the relaxation errors are

$$RE(C_1) = 2\int_a^c F(x)[F(c) - F(x)]dx \tag{9}$$

$$RE(C_2) = 2\int_c^b [F(x) - F(c)][1 - F(x)]dx \tag{10}$$

respectively. Therefore, the relaxation error of the clustering $K_2 = \{C_1, C_2\}$ is

$$RE(K_2) = 2\int_a^c F(x)[F(c) - F(x)]dx + 2\int_c^b [F(x) - F(c)][1 - F(x)]dx. \tag{11}$$

The goodness of K_2 can be measured by the reduction of relaxation error $RE(C) - RE(K_2)$, which is denoted as R(c). Using (8) and (11), we have

$$R(c) = 2(1 - F(c))\int_a^c F(x)dx + 2F(c)\int_c^b (1 - F(x))dx. \tag{12}$$

Differentiating (12) with respect to c, we obtain

$$R'(c) = 2p(c)[\int_c^b (1 - F(x))dx - \int_a^c F(x)dx]. \tag{13}$$

The optimal binary cut c where R(c) is maximum can be obtained by solving $R'(c) = 0$. Using this property, we shall now compare the relaxation error of DISC and ME in the following.

Theorem 1. Let D and M be the optimal binary cuts by DISC and ME respectively, then $R(D) \geq R(M)$.

Proof. The theorem can be proved if we can show that D is uniquely determined by solving $R'(D) = 0$, and $R''(D) < 0$ (such that R(D) is maximum). From equation (13), $R'(c) = 0$ implies $\int_c^b (1 - F(x))dx - \int_a^c F(x)dx = 0$. For increasing c, $\int_c^b (1 - F(x))dx$ is strictly decreasing and $\int_a^c F(x)dx$ is strictly increasing. Thus, $\int_c^b (1 - F(x))dx - \int_a^c F(x)dx$ is strictly decreasing. Since $\int_c^b (1 - F(x))dx - \int_a^c F(x)dx$ is greater than 0 for c=a, and less than 0 for c=b, it must be 0 at exactly one point c between a and b. Therefore, D is uniquely determined by solving $R'(D) = 0$.

To show $R''(D) < 0$, we first differentiate (13) to obtain $R''(D) = 2p'(D)[\int_D^b (1-F(x))dx - \int_a^D F(x)dx] - 2p(D)$. Since $\int_D^b (1-F(x))dx - \int_a^D F(x)dx = 0$ from $R'(D) = 0$, we have $R''(D) = -2p(D) < 0$. Thus, we have shown that D is uniquely determined by solving $R'(D) = 0$ and $R''(D) < 0$. This implies that R(D) is maximum, so $R(D) \geq R(M)$. □

Theorem 2. Let D and M be the optimal binary cuts by DISC and ME respectively. If the distribution of data is symmetrical at the median, then D = M (i.e., the cuts determined by DISC and ME are the same).

Proof. First let us determine M. For a single cut c, entropy is computed by the following

$$E = F(c)logF(c) + (1 - F(c)log(1 - F(c)).$$

Note E has the maximum value at the median m where $F(m) = 1/2$. So we have M = m.

Now if we can show that the reduction of relaxation error is maximum at M, i.e., $R'(M) = 0$, then the proof is complete. Setting $c = M$ in $R'(c)$, we have

$$R'(M) = 2f(M)[\int_M^b (1 - F(x))dx - \int_a^M F(x)dx.$$

Since the distribution is symmetrical at the median, we have $M - a = b - M$ and $F(x) = 1 - F(2M - x)$ for $a \leq x \leq M$. Substituting these into $R'(M)$, it can be shown that $R'(M) = 0$. Thus, the theorem is proved. □

Notice that symmertry at the median is a sufficient, but not necessary, condition for D=M. D=M is still possible for asymmetrical distributions. However, our experience with empirical results reveals such condition is unlikely to occur. In fact, we shall define "skewness" for a data distribution in section 7, and show that in general as the skewness of the distribution increases, the relaxation error improvement of DISC over ME increases.

6 Computation Complexity

Let us now compare the computation complexity of DISC and ME for clustering numerical values.

6.1 Computation Complexity of DISC

Since computing relaxation error needs to consider all pairs of values, it takes $O(n^2)$ time to complete in the worst case where n is the number of distinct values of data. To determine the best cut, it is necessary to evaluate relaxation error for n-1 possible cuts. Therefore, the time complexity for finding the best cut is $O(n^3)$. This can be improved by using a new representation of relaxation error. The new representation is obtained from equation (2) by the following manipulation which is discussed in [8]. First we get rid of the absolute difference between values by

$$\sum_{i=1}^{n}\sum_{j=1}^{n} |x_i - x_j| = 2\sum_{i \geq j}(x_i - x_j)$$

Replacing the difference

$$x_i - x_j = (x_i - x_{i-1}) + (x_{i-1} - x_{i-2}) + ... + (x_{j+1} - x_j),$$

we have

$$\sum_{i \geq j}(x_i - x_j) = \sum_{h=1}^{n-1} K_h(x_{h+1} - x_h)$$

where K_h is the number of terms of type $(x_i - x_j)$ in $\sum_{i \geq j}$ containing $x_{h+1} - x_h$. K_h can be computed by multiplying the number of j's less than or equal to h by the number of i's greater than or equal to $h+1$.

Thus we have $K_h = (\sum_{j=1}^{h} f_j)(\sum_{i=h+1}^{n} f_i) = F(h)(N - F(h))$, where F is the cumulated frequency and N is the total number of values $\sum_{k=1}^{n} f_k$. Therefore

$$RE(C) = \frac{2}{N^2} \sum_{h=1}^{n-1} F(h)(N - F(h))(x_{h+1} - x_h). \tag{14}$$

The above definition eliminates the double summation in equation (2), thus improving the efficiency of calculating relaxation error of a cluster from $O(n^2)$ to $O(n)$. [3] The complexity for determining a single best cut is now reduced to $O(n^2)$.

When determining the best cut, each possible cut is evaluated sequentially in a loop. For each iteration, only a single value changes its memship from one cluster to another. More specifically, a cluster $C = (x_{1:n}, f_{1:n})$ may be changed to $C_1 = (x_{1:n+1}, f_{1:n+1})$ where a new value x_{n+1} larger than any other values is added. C may also be changed to $C_2 = (x_{2:n}, f_{2:n})$ where an old value x_1 smaller than any other values is deleted. Since the clusterings resulting from these changes are very similar to each other, some of the computations may be re-used for greater efficiency. Motivated by this observation, we shall derive two incremental formulas for calculating relaxation error of a cluster, one for C_1 and another for C_2 given the relaxation error of C. In what follows, $N = \sum_{i=1}^{n} f_i$ is the total number of values in C.

Based on equation (14), we have

$$RE(C_1) = \frac{2}{[N + f_{n+1}]^2} \sum_{h=1}^{n} F(h)[N + f_{n+1} - F(h)](x_{h+1} - x_h)$$

After simple manipulation, we derive

$$RE(C_1) = \frac{2}{[N + f_{n+1}]^2} \{S + f_{n+1}[T + N(x_{n+1} - x_n)]\} \tag{15}$$

where $S = \sum_{h=1}^{n-1} F(h)[N - F(h)](x_{h+1} - x_h)$ and $T = \sum_{h=1}^{n-1} F(h)(x_{h+1} - x_h)$. A similar result can be derived for C_2 when a value is deleted from C:

$$RE(C_2) = \frac{2}{[N - f_1]^2} [S - f_1 W] \tag{16}$$

where $W = \sum_{h=1}^{n-1} [N - F(h)](x_{h+1} - x_h)$.

Based on (15) and (16), the algorithm for finding the best cut is shown below.

Procedure BestCut(C)

The input is $C = (x_{1:n}, f_{1:n})$. $N = \sum_{i=1}^{n} f_i$.

The procedure returns the best cut with minimum relaxation error.

Initialization:

$T = N_1 = S_1 = 0, N_2 = N, Min = x_n - x_1$

$W = \sum_{h=1}^{n-1} [N - F(h)](x_{h+1} - x_h), S_2 = \sum_{h=1}^{n-1} F(h)[N - F(h)](x_{h+1} - x_h)$

for h=1 to n-1 do /* evaluate each possible cut */

$T = T + N_1(x_h - x_{h-1})$

$S_1 = S_1 + f_h T$

$N_1 = N_1 + f_h$

$RE(C_1) = 2S_1/N_1^2$

$S_2 = S_2 - f_h W$

$N_2 = N_2 - f_h$

$W = W - N_2(x_{h+1} - x_h)$

$RE(C_2) = 2S_2/N_2^2$

$RE = \frac{N_1}{N} RE(C_1) + \frac{N_2}{N} RE(C_2)$

[3] From this new definition of relaxation error, we can clearly see that both frequency and value distributions are considered. Essentially, the relaxation error is the sum over all gaps weighted by accumulated frequency distribution on both sides of the gap. Compare equation (8) in section 4.

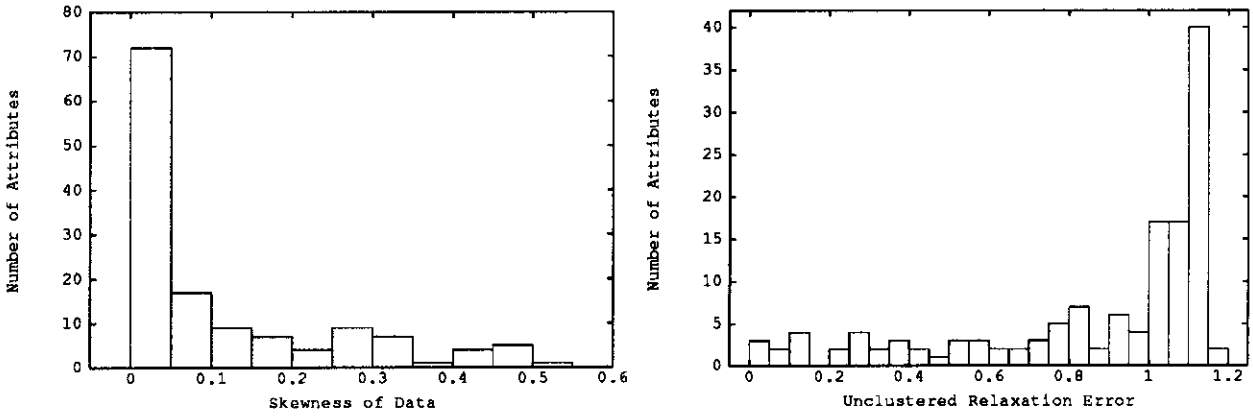

Figure 2: Data skewness and unclustered relaxation error of all attributes

if RE < Min then
Min = RE, Cut = h
Return Cut as the best cut.

In the above procedure, the initialization of variables take $O(n)$ to complete. For each iteration of the loop, we can now calculate the relaxation error of the tentative cut in constant time. Since there are n-1 tentative cuts to be examined, the efficiency of procedure **BestCut(C)** is $O(n)$. Consequently, the efficiency of determining the best cut is reduced from $O(n^3)$ to $O(n)$. For multiple cuts, DISC uses a greedy strategy which determines k cuts in $O(kn)$ time.

6.2 Comparison of DISC and ME

Let us now consider the efficiency of the ME method [15, 1]. ME finds the best cuts by first dividing the total number of data by the specified number of clusters. This provides us with the initial cuts, which may be tentative because same values may be clustered into different clusters. Therefore, local perturbation (a value is put into each of the adjacent clusters) is used to fine tune the clustering, and the computed entropy is used to determine the best clustering. The most time consuming operation is the summation of the frequency of each data value, which is linear. Since both finding the initial cuts and local perturbation can be done in constant time, ME takes linear time to complete.

We have shown that DISC finds the k best cuts in $O(kn)$ time. Since k is a constant given by the user, DISC essentially takes linear time to complete. Therefore, DISC and ME have similar execution time complexity.[4]

7 Empirical Evaluation

We have implemented a greedy DISC and a simple ME method as described in the previous section. In this section, we shall evaluate DISC empirically from a transportation database that is used for planning military missions. It consists of characteristics about ships, aircrafts, tanks, trucks, information about cargo and units, and information about airports and seaports. The database is organized into 104 relations, with

[4]Note that in the above discussion, we assume the input to both ME and DISC is given as a sorted series of data. This assumption may not hold in practical applications. Often the input data is given unsorted, and it requires to be sorted before either DISC or ME can be applied. The cost of sorting turns out to be dominating for both ME and DISC methods, because sorting the input takes at least $O(n \log n)$ time to finish. Taking this into account, DISC has the same computation complexity as that of ME.

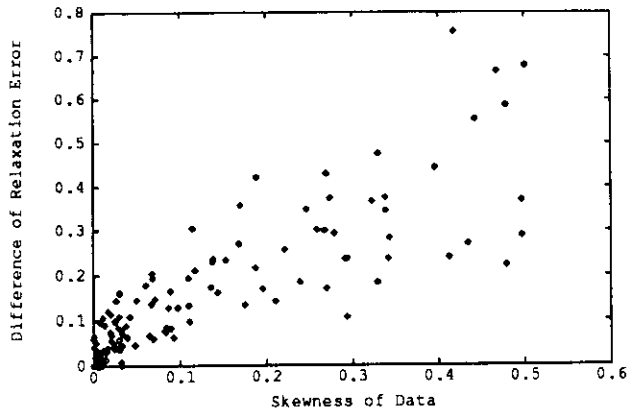

Figure 3: Relaxation error difference between DISC and ME as a function of data skewness. Each point in the chart represents an attribute in the transportation database.

the number of tuples in each relation ranging from 0 to 195,598. There are 376 numerical attributes, 136 among which have more than 10 distinct values [5].

In the following, we shall compare the relaxation error of the resulting clusterings of DISC with that of ME. We shall also investigate the effect of asymmetrical data distribution on DISC performance.

The skewness of a data distribution is defined as the difference between the relaxation error at the mean μ and at the median m, normalized by the unclustered relaxation error; that is,

$$Skewness = \frac{|RE(\mu) - RE(m)|}{RE(unclustered)}.$$

Clearly, if the distribution is symmetrical, then $\mu = m$, and skewness = 0. The skewness and unclustered relaxation error for each of the 136 attributes in the transportation database are computed and the results are plotted in figure 2, with their means and standard deviations shown in table 1.

	Skewness	Relaxation Error
mean	0.114	0.863
std dev	0.142	0.328

Table 1. Average skewness and unclustered relaxation error of all the 136 attributes in the transportation database.

Based on the data values and frequency of the attribute, we compute the skewness of each attribute by equations (1) and (2), and relaxation error by equation (5) for DISC and ME respectively. Figure 3 displays the difference of single-cut relaxation error for DISC and ME with varying data skewness for all 136 attributes. The difference is computed by the relaxation error difference of ME and DISC divided by the unclustered relaxation error, i.e., $[RE(ME) - RE(DISC)]/RE(Unclustered)$. The figure clearly shows that in general the improvements of relaxation error of DISC to ME increases as the skewness of the data distribution increases. The dispersing nature of the trend is due to the fact that the data are taken from various domains with different distributions.

For each of the 136 attributes, the best single cut is determined and its relaxation error is computed using DISC and ME, respectively. Figure 4(a) shows the distribution of relaxation error for DISC, and figure 4(b) shows that of ME, with their difference shown in figure 5. Figure 5 shows that for a single cut, the relaxation error of DISC is always smaller than that of ME, as shown in theorem 1. The means and standard deviations of the single-cut relaxation errors are shown in table 2.

[5] These include key-like attributes, which are essentially uniform. Since the keys are roughly symmetrical at the median, the performance of DISC and ME will not differ much. For better comparison, we normalize all data such that they have the same mean and standard deviation ($\mu = 0, \sigma = 1$).

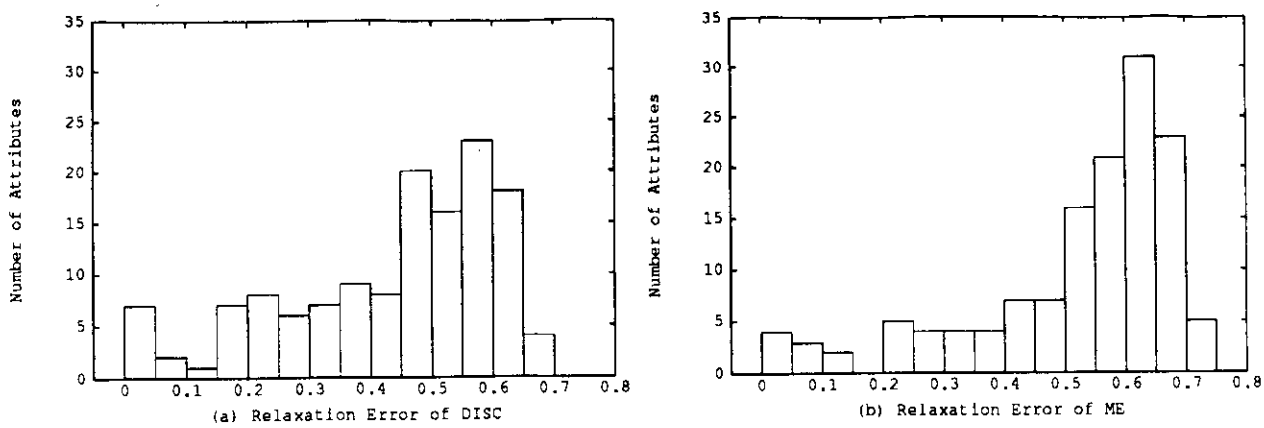

Figure 4: Single-cut relaxation error distribution for all 136 attributes

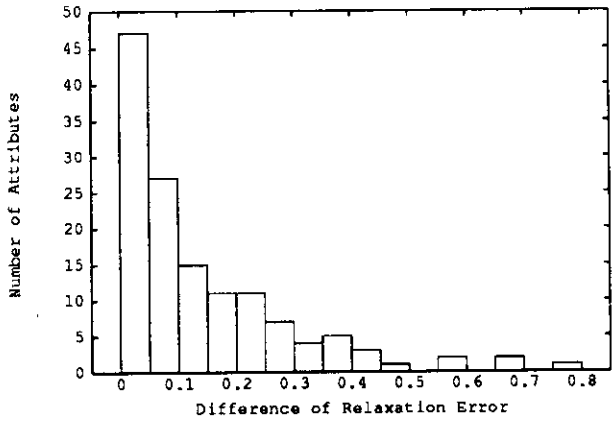

Figure 5: Difference of single-cut relaxation error between DISC and ME for all 136 attributes.

To show the effect of multiple cuts on relaxation error, the relaxation errors of DISC and ME with selected number of cuts for the attribute "UNIT_CHARACTERISTICS.OVERSIZE_CARGO_MTONS" are shown in figure 6. This attribute is selected because of its large size (12,740 tuples). Further, its skewness (0.239) and unclustered relaxation error (0.359) are approximately at the center of the possible ranges. In general, the relaxation error decreases as the number of cuts increases, while the rate of decreasing reduces as the number of cuts increases. Figure 6 shows that DISC performs better than ME for all cuts. Note that the relaxation error of DISC with 4 cuts is lower than that of ME with any number of cuts. Thus, DISC reduces relaxation error more efficiently than ME. Also notice that for this attribute, the DISC curve has a knee around 5 cuts. This information may be useful in determining the optimal number of cuts for the attribute if it is not specified by the user.

To evaluate DISC for multiple cuts, DISC is run repeatedly for each attribute to cluster the data into successive smaller sub-clusters. This process is terminated when a sub-cluster contains less than 10 distinct values. If an attribute has k cuts, then it has a set of clusterings with the number of cuts in each clustering ranging from 1 to k. For each of the k clusterings, we compute its relaxation error. We repeat this process for all the attributes, and the resulting relaxation error distribution is shown in figure 7(a). The total number of cuts for all the attributes using this approach is 1173. The average number of cuts per attribute is 8.6. Similarly, we compute the relaxation error of ME, and the resulting relaxation error distribution is shown in figure 7(b). The difference of relaxation error between DISC and ME is shown in figure 8. The means and standard deviations of the relaxation errors due to multiple cuts are shown in table 2.

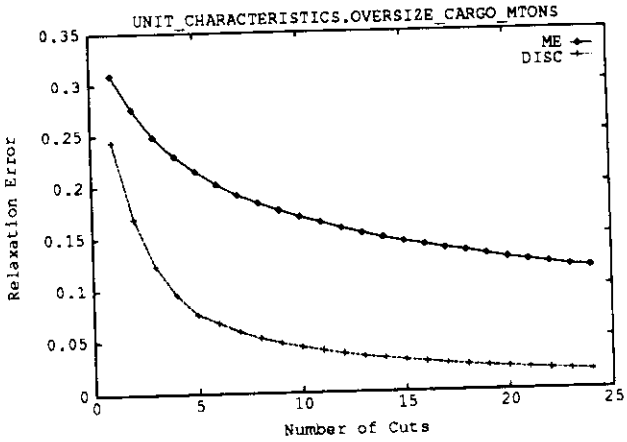

Figure 6: Effect of multiple cuts on relaxation error

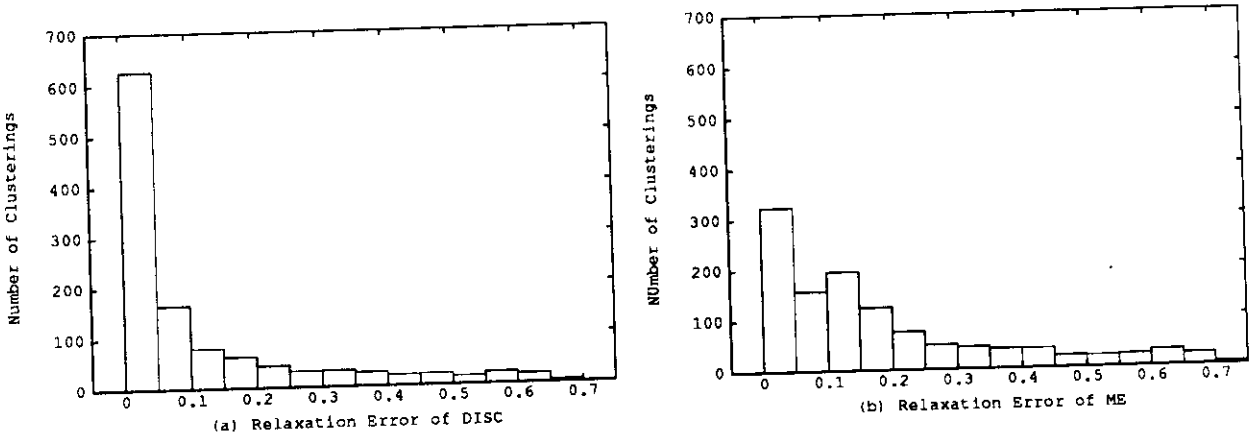

Figure 7: Relaxation error distribution for all 136 attributes with a total of 1173 clusterings

	Single-cut (136 clusterings)			Multiple-cut (1173 clusterings)		
	DISC	ME	DIFF	DISC	ME	DIFF
mean	0.434	0.520	0.141	0.119	0.184	0.150
std dev	0.174	0.171	0.153	0.157	0.177	0.172

Table 2. Comparison of the relaxation errors of DISC and ME.

From figure 8, we note that the relaxation error of DISC is always less than that of ME for a given number of cuts. Thus, DISC also performs better than ME for multiple cuts. The average difference of relaxation error between DISC and ME is greater for multiple cuts (0.150 vs 0.141) as shown in table 2.

Example. We use the attribute LENGTH in table SHIPS as an example to show how DISC can be used for discovering interesting sub-concepts. The table SHIPS has 153 tuples, and the attribute LENGTH has 33 distinct values, ranging from 273 to 947. DISC and ME are used to cluster LENGTH into three sub-concepts: SHORT, MEDIUM, and LONG. The results are shown in figure 9, where vertical broken lines are used for separating concepts. The cuts found by DISC are between 636,652 and 756,791, and those of ME are between 540,560 and 681,685. The average gap of DISC, 25.5, is much bigger than that of ME, 12. In particular, the cut between 681,685 determined by ME is very bad, because it is in the middle of a dense region. To see the effectiveness of the greedy strategy used by DISC, we also determine the optimal cuts by exhaustive search. They are between 605,635 and 756,791, with an average gap of 32.5. The greedy strategy is able to find one of the optimal cuts, while the other cut is only slightly off from the optimal one.

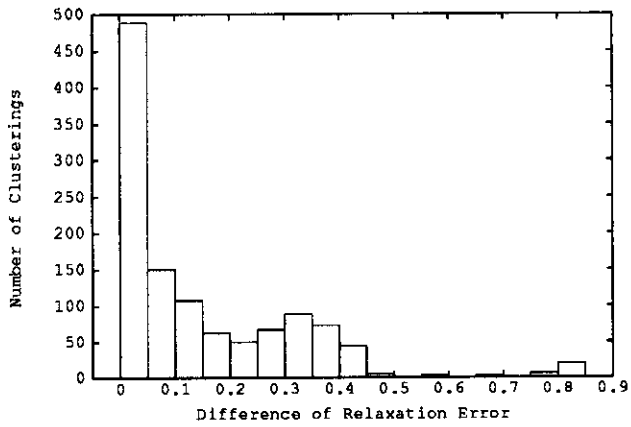

Figure 8: Difference of multiple-cut relaxation error between DISC and ME for all 136 attributes with a total of 1173 clusterings

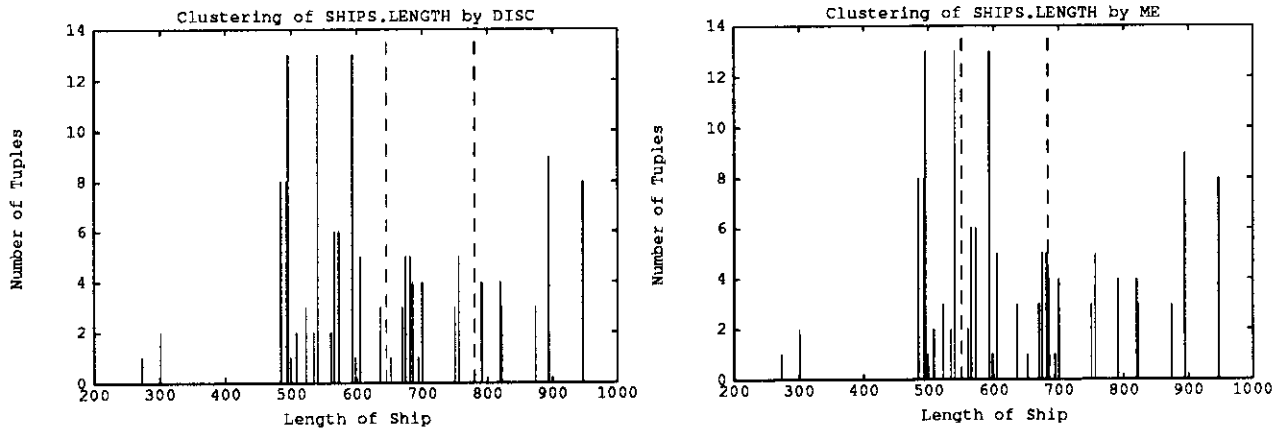

Figure 9: Clustering of SHIPS.LENGTH by DISC and ME

This example shows that DISC is more effective than ME in discovering relevant concepts embedded in the underlying data.

8 Conclusion

In this paper, we present a distribution sensitive clustering method (DISC) for numerical values (integer or real numbers). The goal of DISC is to discover interesting and relevant high level concepts hidden in the underlying data. These concepts may be organized as an abstraction hierarchy which is used for relaxing query conditions to derive approximate answers when exact answers are not available, or used for conceptual queries.

Clustering numerical values is formulated as a search for a given number of optimal cuts. To guide the search for optimal solutions, we introduce a notion of *relaxation error* as a measure for "goodness" of the clustering. Relaxation error takes into account both value and frequency distributions of data. For generation of type abstraction hierarchies, therefore, relaxation error is more suitable than information entropy, which only considers frequency distribution of data. An efficient implementation of DISC with $O(kn)$ time complexity is presented, where n is the number of distinct values of data and k is the number of cuts specified by the user. Empirical evaluation shows that DISC always performs better than the entropy based clustering method ME. Further, the improvements increase as the skewness of data distribution increases.

DISC is implemented as a static top down method since it preprocesses data to discover sub-concepts from a more general concept. we can extend it to bottom up and dynamic where the cutting is determined by the query. An interesting future work would be to explore and compare these different implementations.

References

[1] David K. Y. Chiu, Andrew K. C. Wong, and Benny Cheung. Information discovery through hierarchical maximum entropy discretization and synthesis. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI Press/The MIT Press, 1991.

[2] Wesley W. Chu, Qiming Chen, and Rei chi Lee. Cooperative query answering via type abstraction hierarchy. In *Proceedings of the International Working Conference on Cooperating Knowledge Based SystemsData*, March 1990.

[3] Wesley W. Chu, Rei chi Lee, and Qiming Chen. Using type inference and induced rules to provide intensional answers. In *Proceedings of the 7th International Conference on Data Engineering*, April 1991.

[4] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(2):87–102, 1992.

[5] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[6] Stephen Jose Hanson. Conceptual clustering, categorization, and polymorphy. In *Machine Learning*, volume 3. Margan Kaufmann Publishers, Inc., 1989.

[7] Yannis E. Ioannidis, Tomas Saulys, and Andrew J. Whitsitt. Conceptual learning in database design. *ACM Transactions on Information Systems*, 10(3):265–293, 1992.

[8] Maurice G. Kendall and Alan Stuart. *The Advanced Theory of Statistics*, volume 1, pp. 49-50. Hafner Publishing Company, 1969.

[9] M. Lebowitz. Experiments with incremental conceptual formation. *Machine Learning*, 2(2):103–138, 1987.

[10] Matthew Merzbacher and Wesley W. Chu. Instance-based clustering for databases. In *Proceedings of the 3th Asis SIG/CR Classification Research Workshop*, October 1992.

[11] A. Motro. Seave: A mechanism for verifying user presuppositions in query systems. *ACM Transactions on Office Information Systems*, 4(4):49–50, 1986.

[12] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[13] R. E. Stepp and R. S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. In *Machine Learning*, volume 2. Margan Kaufmann Publishers, Inc., 1987.

[14] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn.* Margan Kaufmann Publishers, Inc., 1991.

[15] Andrew K. C. Wong and David K. Y. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):796–805, 1987.