Preserving Patterns for Categorical Encoding using Soft Computing

ANGEL FERNANDO KURI-MORALES
Department of Computation
Instituto Tecnológico Autónomo de México
Río Hondo No. 1
MÉXICO
akuri@itam.mx   http://www.itam.mx/

Abstract: - In mixed databases MD (those involving both numerical and categorical attributes) it is not possible to apply metric clustering algorithms. In order to do this we must replace categorical instances (i.e. the values in a category) by numerical valid codes (resulting in a purely numerical database ND). We discuss an algorithm (called CENG) which allows us to preserve the patterns in MD when categorical instances are replaced by numeric codes for every instance (say *t*) of every category (say *c*). We must be able to preserve the structure present in all the relations between the now numerical attributes. We consider this to be fulfilled if we identify the set of codes SC which, when inserted in place of the instances of the categories, allows the expression $f_i(\boldsymbol{n-1})$ of attribute *i* as a function of the remaining *n-1* for *i=1,...,n*. We look for one SC (the "perfect" code set or PCS) which achieves this goal with an approximation error $e_i=0$ for all *i*. In practice we consider the problem to be solved when $e_k=max(e_i)$ is minimized. However, we must be able to first find the f_i's. This is done by training a three-layered perceptron network (NN) which takes the role of $f_i(\boldsymbol{n-1})$. Since it is, in general, computationally too costly to try out all possible SC's we appeal to a genetic algorithm (GA). Every individual of the GA will be a set of *ct* binary numbers. For every one such individual the fitness corresponds to e_k and the GA will minimize it. The evolutionary optimization process yields an accurate approximation of the *ct* numbers which minimize e_k for all tuples in the dataset, i.e. those which best preserve the patterns present in the data. Since both the GA's and the NN's start with pseudo-random configurations, two runs of CENG with different seeds for the random number generator will yield different SC's (SC_1 and SC_2). This allows us to test the resilience of CENG since the resulting SC_1 and SC_2 should lead to the same clustering. We show that this is, indeed, the case.

Key-Words: - Data mining; categorical data; multi-layered perceptron networks: genetic algorithms: unsupervised learning.

1 Introduction

Databases may be, basically, classified as purely numerical, purely categorical or mixed. A structured mixed database (*DB1*) is illustrated in Table 1.

Age	Place	Studies	Race	Sex	Salary
35	South	24	Indian	F	100,702.22
37	South	4	White	F	77,770.95
24	South	10	White	M	2,120.13
44	South	20	White	M	9,187.65
59	North	24	Indian	F	12,834.32
57	North	5	Amerindian	M	6,927.62
43	North	8	Amerindian	F	5,174.10
28	Center	8	White	F	2,877.14
62	North	15	White	M	12,118.82
68	Center	23	Amerindian	M	11,180.07
50	North	8	Other	M	31,366.48
28	Center	20	Indian	F	2,996.56

Table 1. Mixed Database

When mining such structured databases one of the most basic questions consists of identifying groups of related data which we call "clusters". Clustering can be considered the most important unsupervised learning problem.

It deals with finding the structures in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. We illustrate this with the graphical example of Fig. 1. Three sets are represented by spheres in 3D Euclidean space. These spheres define three centers C_1, C_2, C_3. The membership of a tuple to the *i-th* cluster is determined by its distance to C_i. This is an example of the so-called *distance-based clustering*.

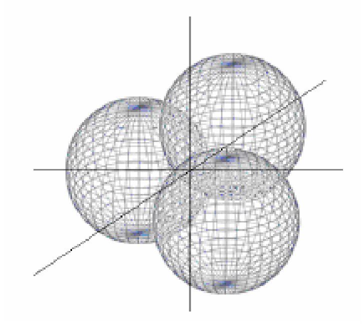


Fig. 1. Three Overlapping 3D Clusters in Euclidean Space

Another kind of clustering is *conceptual clustering* where two or more objects belong to the same cluster if this one defines a concept common to all those objects. Objects are grouped according to their fit to descriptive concepts, not according to basic similarity measures. The method reported here (CENG => Categorical Encoding with Neural Networks and Genetic Algorithms) will allow us, by replacing categorical instances with numerical codes which preserve the patterns in the database, to generalize the application to numerical methods to the analysis of mixed databases. The concept of pattern preservation is a key issue which we make precise in section 4. The rest of the paper is organized as follows. In Section 2 we discuss alternatives to our approach (direct categorical analysis and previous coding approaches). In section 3 we describe the theoretical underpinning which allows us to formalize CENG. In section 4 we describe how CENG results from the combination of a learning algorithm and an optimization one which leads to the solution of the encoding problem. In section 5 we present the experimental results which validate CENG's application. In section 6 we present our conclusions and discuss possible future lines of investigation.

2 Previous Approaches to Mixed Data Analysis

The problem of clustering mixed databases has been approached in the past in, basically, two ways:

a) Attempt the analysis directly.
b) Encode the categorical attributes to make them prone to numerical analysis.

Our method belongs to the second approach.
In 2.1 we make a very brief overview of methods attempting not to assign numerical codes to categorical attributes. Methods which do attempt such assignment are covered in 2.2.

2.1 Non-Metric Clustering Algorithms

Several algorithms have been proposed in recent years for clustering categorical data which do not attempt to assign a numerical value for the categorical instances. In what follows we make a very brief account of some of the latest such attempts.

In [1] a framework to learn a context-based distance for categorical attributes is proposed. The key intuition is that the distance between two values of a categorical attribute A_i can be determined by the way in which the values of the other attributes A_j are distributed in the dataset objects: if they are similarly distributed in the groups of objects in correspondence of the distinct values of A_i a low value of distance is obtained. A solution to the critical point of the choice of the attributes A_j is also proposed. The approach is validated by embedding a distance learning framework in a hierarchical clustering algorithm.

In [2] ROCAT (Relevant Overlapping Subspace Clusters on Categorical Data) is proposed. This is a technique based on the idea of data compression. Following the Minimum Description Length principle, ROCAT detects the most relevant subspace clusters without any input parameter. The relevance of each cluster is validated by its contribution to compress the data. Optimizing the trade-off between goodness-of-fit and model complexity, ROCAT determines a meaningful number of clusters to represent the data. ROCAT is designed to detect subspace clusters on categorical data.

In [3] divisive hierarchical clustering algorithm for categorical data, named DHCC is proposed. The task of clustering categorical data is seen from an optimization perspective, and procedures to initialize and refine the splitting of clusters are also proposed. The initialization of the splitting is based on multiple correspondence analysis (MCA). A strategy for deciding when to terminate the splitting process is also implemented.

In [4] an automated technique, called SpectralCAT, for unsupervised clustering of high-dimensional data that contains numerical or nominal or mix of attributes is proposed. High-dimensional input data is transformed into categorical values. This is done by discovering the optimal transformation according to the Calinski-Harabasz index for each feature and attribute in the dataset. Then, a method for spectral clustering via dimensionality reduction of the transformed data is applied. This is achieved by automatic non-linear transformations, which identify geometric patterns in the data, and find the connections among them while projecting them onto low-dimensional spaces.

In [5] a mean gain ratio (MGR) information theory based hierarchical divisive clustering algorithm for categorical data is proposed. MGR implements clustering from the attributes viewpoint which includes selecting a clustering attribute using mean gain ratio and selecting an equivalence class on the clustering attribute using entropy of clusters.

In [6] a kernel-density-based definition using a Bayes-type probability estimator is proposed. Then, an algorithm called k-centers is proposed for central clustering of categorical data, incorporating a weighting scheme by which each attribute is automatically assigned with a weight measuring its individual contribution for the clusters.

In [7] a general clustering framework based on the concept of object-cluster similarity is advanced which gives a unified similarity metric which can be simply applied to the data with categorical, numerical, and mixed attributes. From it, an iterative clustering algorithm is developed. To circumvent the selection problem of cluster number, it is further developed a penalized competitive learning algorithm within the proposed clustering framework.

In [8] an Incremental Learning based Multiobjective Fuzzy Clustering for Categorical Data is proposed. A multiobjective modified differential evolution based fuzzy clustering algorithm is developed. It integrates with the well-known supervised classifier, called random forest, using incremental learning to propose the aforementioned technique. Here, the multiobjective algorithm produces a set of optimal clustering solutions, known as Pareto-optimal solutions, by optimizing two conflicting objectives simultaneously. Subsequently, through incremental learning using random forest classifier final solution is evolved from the ensemble Pareto-optimal solutions.

Something that is overlooked in all the previously reviewed works is that there is an inherent methodological problem when trying to assign artificial distances to categories. By definition, a category is set by a concept and no two objects within a category may be said to be closer (or farther apart) to one another in the mathematical sense of a metric. It makes no sense to assign more or less similarity to, say, people originating in a region than to people originating in a different one. There may be, of course, more apparent similarity traits between people from neighboring regions than those from geographically distant ones. But the similarities, if any, do not have to do with the regions *per se*. Ultimately, the geographic boundaries are arbitrarily set and attempting to measure the difference between subjects according to their nearness at the moment of birth is equivalent to claiming that some political boundaries are somehow "better" than others. When one includes a place of birth as a defining attribute what is really happening is that we accept that the attribute "place of birth" is somehow easier to manipulate than the ensemble of variables which compose it. By the same token, it also makes no sense to try to assign a distance to the concept "color of eyes". No mathematical meaning may be attached to the hypothesis that blue-eyed colored persons are more "similar" to green-eyed color persons than to black-eyed ones. As in the previous example, the color of the eyes is a fortuitous manifestation (of the pigment of the iris, in this case) of a physical extraneous circumstance. And the fact that light colored iris vs. dark colored ones may be indicative of dissimilar races does not have to do with the color of the eyes in itself but, rather, to a

possible common ancestry. As before, in the concept "color of the eyes" there is an ensemble of variables whose values we find simpler to assume than to quantify. These considerations or similar ones apply, in general, to all categorical attributes.
Furthermore, since the final goal of incorporating categorical attributes to the analysis of the data bases is to find adequate clusters, some of the previously reviewed algorithms are based on the attempted correct determination of the number of clusters. Here they are failing to realize that there is no absolute "optimum" number of clusters. This is due to the fact that the distances between the objects of the data base and the centers of the clusters depend on the metric being used.
This issue is intimately related with the quality of the clusters. As discussed in [9] there are several ways to attempt the assessment of how adequate the groups resulting from a clustering process are. However, it is again the case that the purported quality depends on the selected metric. Indeed, an alternative is to find the clusters which optimize a given validity index as described in [10]. The natural course of action is to pre-process the categorical variables to make them numerical and then apply metric clustering algorithms such as, for example K-Means [11], Self-Organizing Maps [12], Fuzzy C-Means [13], Fuzzy learning vector quantization [14] or Clustering Based on Entropy [15].

2.2 Metric Categorical Analysis

From the previous considerations it is clear that categorical variables cannot be entered into metric algorithms just as they are and need to be recoded into a series of variables which can then be entered into the model. There is a variety of coding systems that can be used when coding categorical variables. Ideally, one would choose a coding system that reflects the comparisons that one wants to make. But this is, in general, unknown. A partial list [16] of different types of previously attempted numerical coding is the following.

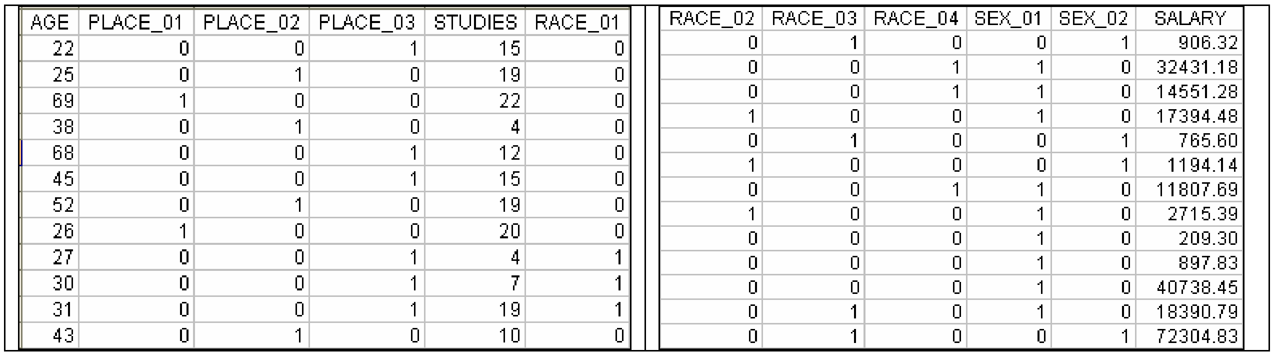
1. Pseudo-binary variables – Every categorical variable is replaced by a set of binary variables, each corresponding to the instances of the category.
2. Simple Coding - Compares each level of a variable to the reference level
3. Forward Difference Coding - Adjacent levels of a variable (each level minus the next level)
4. Backward Difference Coding - Adjacent levels of a variable (each level minus the prior level)
5. Helmert Coding - Compare levels of a variable with the mean of the subsequent levels of the variable
6. Reverse Helmert Coding - Compares levels of a variable with the mean of the previous levels of the variable
7. Deviation Coding - Compares deviations from the grand mean
8. Orthogonal Polynomial Coding
9. Orthogonal polynomial contrasts
10. User-Defined Coding
11. User-defined Contrast

2.3 Limitations of Simple Metric Encoding

All of the coding schemes listed in 2.2 suffer from the following limitations:

a) The categorical attributes in the original mixed database *MD* are replaced by numerical ones where every categorical variable is replaced by a set of *t* numerical codes. *MD* (with *c* categories and *n* numerical variables) will be replaced by a numerical database *ND* with *n-c+ct* variables. The cardinality of *ND* will be larger that of *MD*. In many cases this leads to unwieldy databases which are more difficult to store and handle. A pseudo-binary encoded version of *DB1* is illustrated in Table 2.

b) The type of coding system selected implies a subjective choice since all pseudo-binary variables may be assigned any two values (typically “0” denotes “absence”; “1” denotes “presence”). In Table 2 the attribute “Place” has been replaced by the variables “Place_01”, “Place_02” and “Place_03”. The values of the instances “South”, “North” and “Center” have been replaced by “1” or “0” as appropriate. However, the choice of 0/1 is subjective. Any two different values are possible. The mathematical properties of *ND* will vary with the different choices, thus leading to clusters which depend of the way in which “presence” or “absence” is encoded.



AGE	PLACE_01	PLACE_02	PLACE_03	STUDIES	RACE_01
22	0	0	1	15	0
25	0	1	0	19	0
69	1	0	0	22	0
38	0	1	0	4	0
68	0	0	1	12	0
45	0	0	1	15	0
52	0	1	0	19	0
26	1	0	0	20	0
27	0	0	1	4	1
30	0	0	1	7	1
31	0	0	1	19	1
43	0	1	0	10	0

RACE_02	RACE_03	RACE_04	SEX_01	SEX_02	SALARY
0	1	0	0	1	906.32
0	0	1	1	0	32431.18
0	0	1	1	0	14551.28
1	0	0	1	0	17394.48
0	1	0	0	1	765.60
1	0	0	0	1	1194.14
0	0	1	1	0	11807.69
1	0	0	1	0	2715.39
0	0	0	1	0	209.30
0	0	0	1	0	897.83
0	0	0	1	0	40738.45
0	1	0	1	0	18390.79
0	1	0	0	1	72304.83

Table 2. Pseudo-binary encoding

c) Finally, and most importantly, with this sort of scheme the pseudo-binary variables do no longer reflect the essence of the idea conveyed by the category. The variable Place_01, for instance, reflects the way the tuple is "affected" by belonging, say, to the categorical instance "Center", which is correct. But now the original issue "How does the behavior of the individuals change according to their place of birth?" is replaced by "How does the behavior of the individuals change when their place of birth is 'Center'?" The two questions are not interchangeable.

2.4 Pattern Preserving Metric Encoding

Our method relies on the hypothesis that there exists a *perfect code set (PCS)* consisting of a set of numerical codes (one for every instance of every category) such that all the relationships that exist in MD will be preserved in the resulting ND where the categorical instances have been replaced by the corresponding numerical codes. We look for a coding scheme which allows us to replace the instances of the categories by numerical codes while preserving the original relationships determined by the original values of the categorical variables. PCS may or may not exist. Therefore, we replace the demand of absolute preservation of the patterns by the option which preserves these with the highest likelihood. At any rate, we point out that this is essentially different from trying to adopt a set of codes which reflect a distance between the instances of the categories.

2.4.1 Preserving n-th Degree Patterns

It is a standard practice to mathematically seek for relationships which may exist between variables. For example, the coefficient of determination (R^2) indicates how well data fits a model. It provides a measure of how well observed outcomes are replicated by the model, as the proportion of total variation of outcomes explained by the model [18]. It is assumed that a multiple variable model $f(v_1,v_2,...,v_q)$ exists. If q independent variables are included, R^2 is the square of the coefficient of multiple correlation. A data set has p values (y_i), each associated with a modeled value f_i. Let us denote the total sum of squares by SS_{tot} and note that $SS_{tot} = \sum_i (y_i - \bar{y})^2$, for $\bar{y} = avg(y_i)$. The residual sum of squares is given by $SS_{res} = \sum_i (y_i - f_i)^2$ and $R^2 = 1-(SS_{res}/SS_{tot})$. R^2 is a statistical measure of how well the regression model approximates the real data points. In mixed databases the numerical values corresponding to the instances of the categorical attributes are unknown. Our method is based on the corresponding inverse consideration of identifying the codes which result in $SS_{res} \approx 0$ and, therefore, our aim is to identify the numerical values which would force $SS_{res} \rightarrow 0$. Such mapping of instances to numbers would manifest a perfect relationship between y_i and f_i. We stress, however, that no conceptual meaning is assigned to the numerical values.

2.4.2 Finding the Model

To make this aim practical, we must be able to find the model $f(v_1,v_2,...,v_q)$ which approximates the data as closely as possible. Multilayer perceptron networks (MLP) are very popular in practice (as discussed, for example, in [19]) and were mathematically proven to embody such model. In [20] Cybenko proved the so-called Universal Approximation Theorem (UAT) which may be enunciated as follows.

Theorem 1. Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_{mO} denote the m_O-dimensional unit hypercube [0,1]. The space of continuous functions on I_{m_O} is denoted by

$C(I_{mO})$. Then, given any function $f \in C(I_{mO})$ and $\varepsilon > 0$, there exist an integer M and sets of real constants α_i, b_i and w_{ij}, where $i=1,2,\dots,m_I$ and $j=1,2,\dots,m_O$ such that we may define:

$$F(x_1,\dots,x_{m_0}) = \alpha_0 + \sum_{i=1}^{m_I}\left[\alpha_i \cdot \varphi\left(\sum_{j=1}^{m_0} w_{ij}x_j + b_i\right)\right] \tag{1}$$

as an approximate realization of the function $f()$, that is,

$$|F(x_1,\dots,x_{mO})-f(x_1,\dots,x_{mO})|<\varepsilon \tag{2}$$

for all $x_1,\dots,x_{m_O}$ in the input space.

The UAT states that *a single hidden layer is sufficient for a multilayer perceptron to compute a uniform* ε *approximation to a given training set represented by the set of inputs* $x_1,\dots,x_{mO}$ *and a desired (target) output* $f(x_1,\dots,x_{mO})$.

Equation (1) represents the output of a MLP described as follows: a) The network has m_O input nodes and a single hidden layer consisting of m_I neurons; the inputs are denoted by $x_1,\dots,x_{mO}$, b) Hidden neuron i has synaptic weights $w_{i1}, w_{i2}, \dots, w_{imO}$, c) The network output is a linear combination of the outputs of the hidden neurons, with $\alpha_1,\dots,\alpha_{mI}$ defining the synaptic weights of the output layer. A schematic representation of an MLP corresponding to the UAT is shown in Fig. 2.

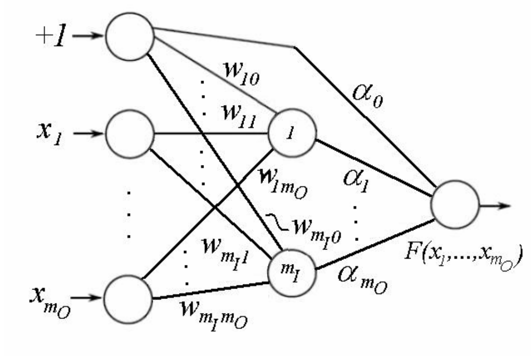


Fig. 2. A 3-Layered Perceptron Network

The UAT tells us that, for a given problem, an adequately designed MLP will approach an arbitrary function as closely as desired. To do so one must adequately pre-process the data, calculate m_I [21] and count with an effective training algorithm [22]. Thereafter, if one is given the correct set of values corresponding to the instances of the categorical variables, the model will result in $R^2 \approx 1$.

2.4.3 PCS Preserves n-th Degree Patterns for All Variables

Notice that (1) refers to strictly numerical data. Therefore, it is not enough to replace the values of the instances of one categorical variables with their numerical counterparts; rather, one has to encode *all* instances of *all* categorical variables to find the multivariate model. We exemplify with the MD illustrated in Fig. 3 which consists of 4 variables: two numerical (N1, N2) and two categorical (C1, C2). C1 has two possible instances (I1, I2) as does C2 (I3,I4).

N1	N2	C1	C2
0.5527	0.3207	I1	I3
0.6955	0.4653	I2	I3
0.7495	0.4611	I1	I4
0.1077	0.5317	I2	I4
0.1493	0.5728	I2	I4
0.5146	0.9077	I1	I3
0.6456	0.8610	I1	I3
0.4420	0.8476	I1	I3

Fig. 3. A Mixed Database

Assume PCS consists of the following codes, corresponding to I1,I2, I3, I4:

0.3411	0.6026	0.9958	0.6897

and that they are inserted into MD yielding a ND as illustrated in Fig. 4.

N1	N2	C1	C2
0.5527	0.3207	0.3411	0.9958
0.6955	0.4653	0.6026	0.6897
0.7495	0.4611	0.3411	0.9958
0.1077	0.5317	0.6026	0.6897
0.1493	0.5728	0.3411	0.9958
0.5146	0.9077	0.6026	0.9958
0.6456	0.8610	0.3411	0.9958
0.4420	0.8476	0.6026	0.6897

Fig. 4. A Numerical Resulting Database

Now we train the NN which is illustrated in Fig. 5.

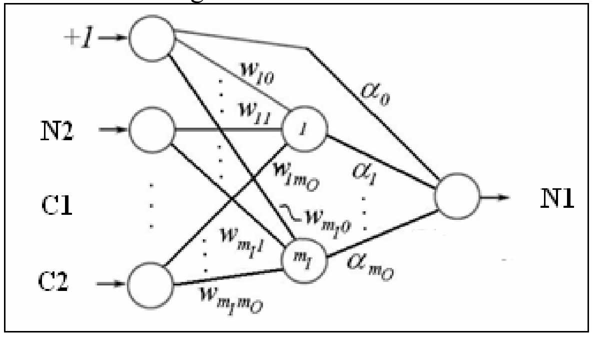


Fig. 5. N1 as a function of N2, C1 and C2.

Denoting the coefficient of determination of the *i-th* variable as R_i^2, from the UAT, we know that N1 will make $R_1^2 \approx 1$. The training process is repeated now making every variable a function of the others as illustrated in Fig.6.

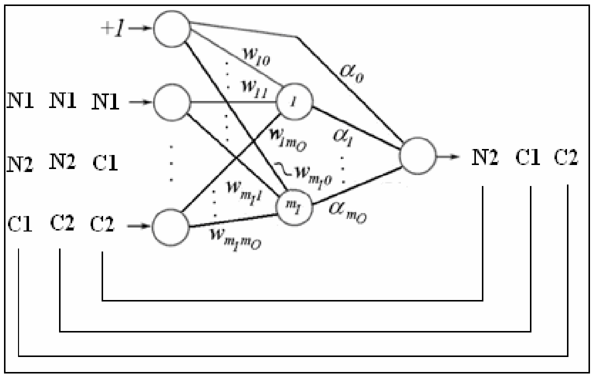


Fig. 6. Different NNs for all Variables

Hence, PCS may be defined as follows

$$PCS \rightarrow R_i^2 = 1 \ \forall i \tag{3}$$

In other words, PCS preserves all *n-th* degree relations for all the attributes in any database. The problem is, of course, that the hypothesized values of PCS are unknown and we must identify them. Our goal is, therefore, to identify PCS as per (3) from all the possible combinations. In order to make this statement concrete we have to consider the fact that a candidate PCS (which we will call CPCS) will have to cope with the fact that identifying the CPCS which attains $R_I^2 \approx 1$ for all i involves the examination of a potentially very large solution landscape.

3 Categorical Encoding from NNs and GAs

In the database of Fig. 3 and 4 we illustrated PCS with 4 decimal places in which case the number of choices (for 4 categorical instances) is 10^{16}. The hypothesis on which our work is based may be stated as follows: "There is a set of numerical codes, one for each instance of all categorical variables, such that the patterns present in the original *MD* will be preserved in the corresponding *ND*". Clearly, an exhaustive enumeration and analysis of all possible solutions is impractical and we choose to resort to a GA.
Before attempting the solution of the problem several pre-processing steps must be taken as discussed in what follows.

3.1 Implementation

Several decisions regarding the parameters of the detailed algorithm were taken. These are incorporated in CENG.

3.1.1 Effective Size of the Data

To establish the number of hidden neurons of the MLPs one must, first, determine the effective size of the data: the one corresponding to the non-redundant representation of the database. This was done applying the lossless compression algorithm PPMZ2 [23] and assuming that the compressed database sets a lower limit on the number of bits needed to represent it. This is the effective size of the database (*N*).

3.1.2 Mapping the data into [0,1)

We know that in order to take advantage of the UAT data must lie in the [0,1) interval. Therefore all numerical (original) data is thusly mapped. This implies that the codes to be found must lie in the same interval.

3.1.3 Determining the number of Hidden Neurons

The architecture of the MLP is determined by m_O, m_I and $\varphi(.)$. In *DB1*, for example, $m_O=6$. Previous experience suggests that $\varphi(.)$, for the hidden layer, be tanh(.) and for the output layer be linear. The triples (m_O, N, m_I) were scaled into the interval [0,1). The corresponding scaled values are denoted with (m_O^*, N^*, m_I^*). In [21] it was shown that a lower bound on m_I^* may be found from:

$$\begin{aligned} m_I^* = {} & c_{01}N^* + c_{12}(m_O^*)(N^*)^2 + c_{21}(m_O^*)^2 N^* + \\ & c_{23}(m_O^*)^2(N^*)^3 + c_{34}(m_O^*)^3(N^*)^4 + \\ & c_{45}(m_O^*)^4(N^*)^5 + c_{56}(m_O^*)^5(N^*)^6 \end{aligned} \tag{4}$$

and $C_{01}\approx+0.94658$, $C_{12}\approx-12.68272$, $C_{21}\approx-0.06392$, $C_{23}\approx$ 64.10765, $C_{34}\approx-141.17719$, $C_{45}\approx139.24502$, $C_{56}\approx-50.39474$. In our program m_I is calculated from (4).

3.1.4 Training the MLPs

The MLPs were trained using back propagation with momentum. As stated, the activation function for the hidden layer was selected to be *tanh()*. The one at the output layer is linear. The learning rates for the hidden and output layers were set to 0.05 and 0.01 respectively. The momentum parameters were set to 0.3 and 0.5. The MLPs were trained for 1000 epochs.

3.1.5 Encoding the Solution

Once the data has been preprocessed as described we may proceed to explore the space of possible solutions with a GA. GAs were shown to converge to a global optimum in [24] and its use is widely spread (for applications of which see, for instance [25], [26]). One particular breed called *EGA (Eclectic Genetic Algorithm)* [27] was shown to be the most efficient among several choices. The solution to the problem consists of a set of codes; one for every instance of every category. In total we need *ct* codes. For the *EGA* to work these codes were encoded in binary. The set of codes corresponding to attribute *i* will be denoted as $Code_i$ while the codes of the set corresponding to instance *k* will be denoted as $code_k$. The binary representation of $code_k$ will be interpreted as a weighted left-justified decimal number *X* with $0 \leq X < 1$. We decided that a precision of, approximately, two in a million (2^{-22}) was enough and, therefore, $|code_k| = 22$, $|Code_i| = 22t$ and the length of *EGA*'s chromosome *L* is 22*ct*. In *DB1* $c=3$, $t=3$ and $L=198$. Therefore, the task of *EGA*, for *DB1*, say, is to correctly identify one code out of $2^{198} \approx$ 40e+58 combinations. The parameters of *EGA* are $G=500$ (number of generations), $Pc=1$ (probability of crossover), $Pm=0.05$ (mutation probability) and $PS=70$ (population size).

4 CENG Algorithm

The steps of CENG are as follows.

```
- Specify the mixed database $MD$.
- $MD$ is analyzed to determine $c$ and $t$. The size of the genome (the number of bits needed to
encode the solution) is $L=22ct$. $EGA$ randomly generates a set of $PS$ strings of size $L$. This is
population $P_0$. Every one of the $PS$ strings is an individual of $EGA$.
for $i=1$ to $G$
  for $j=1$ to $PS$
        From individual $j$ $ct$ numerical codes are extracted and $ND_{ij}$ is generated replacing the
        categorical instances by their numerical counterparts. Numerical variables are left
        undisturbed.
        $MLP_{ij}$'s architecture (corresponding to individual $j$) is determined as described above.
       for $k=1$ to $n-1$
         $MLP_{ij}$ is fed with a data matrix in which the
         $k$-$th$ attribute of $ND$ is taken as a variable
         dependent on the remaining $n-1$. $MLP_{ij}$ is
         trained and the maximum error $e_{ijk}$ (i.e. the
         one resulting by feeding the already trained
         $MLP_{ij}$ with all tuples vs. the dependent
         variable) is calculated.
       endfor
       $Fitness(j) = max(e_{ijk})$
  endfor
  if $Fitness(j)<0.01$ $EGA$ ends. Otherwise the $PS$
  individuals of $P_i$ are selected, crossed over and
  mutated. This is the new $P_i$.
endfor
```

EGA yields the codes for every one of the *ct* instances of the categorical variables and *ND*: a version of *MD* in which all categorical instances are replaced by the proper numerical codes.

4.1 Pattern Preservation

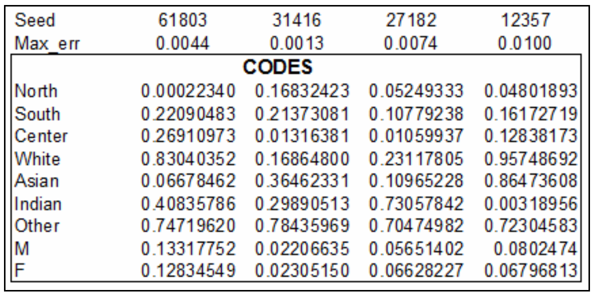
Notice that the algorithm finds the codes which lead to the smallest learning error for the *i-th* variable as a function of the remaining *n-1* and this is simultaneously true for all *n* variables. This is equivalent to finding the code which allows the mathematical representation of all variables as a function of the remaining ones for all functional combinations. Since the best mathematical representation is guaranteed (from the UAT) this implies that all models are simultaneously best for all variables. Finding this ensemble of models is what we mean by "preserving the patterns". In general this is a multi-objective optimization problem since optimally fitting variable *k* will, in general, induce conflicting goals when trying to optimize it along with the remaining ones. We have solved this problem by characterizing a set of codes with the worst case value rather than the best, and then minimizing it. The net result is that the global fitness (which we defined as $e_k \leq 0.01$) is the one corresponding to the worst functional expression of all the *n* expressions implicated. In our method, therefore, the final set of codes corresponds to the one in which all variables are expressed as a function of the rest of the ensemble with a maximum error smaller than 0.01, thus ensuring that the underlying information structure of the database (i.e. the embedded patterns) is preserved.

5 Experimental Validation

To validate the fact that, indeed, the underlying patters in the database are preserved, we note that the whole modeling process is stochastic in nature. Both *EGA* and the *MLP*'s weights are derived from processes taking advantage of random phenomena. In *EGA* mutations are random, whereas in the *MLP*s the back propagation algorithm relies on an initial random setting of the weights of the connections. This implies that, given *MD* and, say, two different seeds for the random number generator the resulting sets of codes (and the associated *ND*s) will be different. However, under the assumption of pattern preservation, both *ND*s should be equally valid. And this allows us to test the validity of the encoding scheme by observing the behavior of the cluster labels assigned to the tuples. When clustering two *ND*s derived from the same *MD* with the same clustering algorithm, we should observe the same cluster labels for all tuples in the database. This is the experiment we conducted, as will be described in what follows.

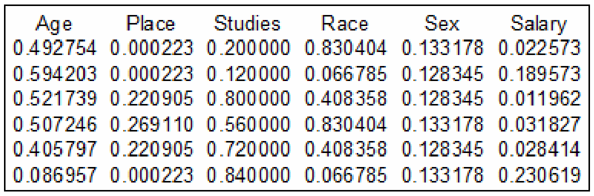
5.1 Encoding with Different Seeds of the Pseudo-Random Number Generator

We applied our method using different seeds (31416, 27182, 61803 and 12357). Table 3 shows the corresponding encodings. No apparent relation is evident from these codes. They were put into *DB1* to yield *ND1*, *ND2*, *ND3* and *ND4* respectively. Remember that all original numerical codes were, first, mapped into [0,1). Small fragments of *ND1* and *ND2* are shown in Tables 4 and 5.



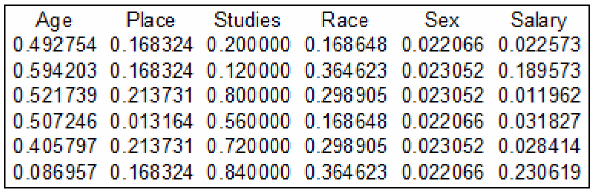
Seed	61803	31416	27182	12357
Max_err	0.0044	0.0013	0.0074	0.0100
	CODES			
North	0.00022340	0.16832423	0.05249333	0.04801893
South	0.22090483	0.21373081	0.10779238	0.16172719
Center	0.26910973	0.01316381	0.01059937	0.12838173
White	0.83040352	0.16864800	0.23117805	0.95748692
Asian	0.06678462	0.36462331	0.10965228	0.86473608
Indian	0.40835786	0.29890513	0.73057842	0.00318956
Other	0.74719620	0.78435969	0.70474982	0.72304583
M	0.13317752	0.02206635	0.05651402	0.0802474
F	0.12834549	0.02305150	0.06628227	0.06796813

Table 3. Codes for different seeds



Age	Place	Studies	Race	Sex	Salary
0.492754	0.000223	0.200000	0.830404	0.133178	0.022573
0.594203	0.000223	0.120000	0.066785	0.128345	0.189573
0.521739	0.220905	0.800000	0.408358	0.128345	0.011962
0.507246	0.269110	0.560000	0.830404	0.133178	0.031827
0.405797	0.220905	0.720000	0.408358	0.128345	0.028414
0.086957	0.000223	0.840000	0.066785	0.133178	0.230619

Table 4. Partial View of *ND1* (*DB1* encoded with Codes Derived from *Seed=61803*).



Age	Place	Studies	Race	Sex	Salary
0.492754	0.168324	0.200000	0.168648	0.022066	0.022573
0.594203	0.168324	0.120000	0.364623	0.023052	0.189573
0.521739	0.213731	0.800000	0.298905	0.023052	0.011962
0.507246	0.013164	0.560000	0.168648	0.022066	0.031827
0.405797	0.213731	0.720000	0.298905	0.023052	0.028414
0.086957	0.168324	0.840000	0.364623	0.022066	0.230619

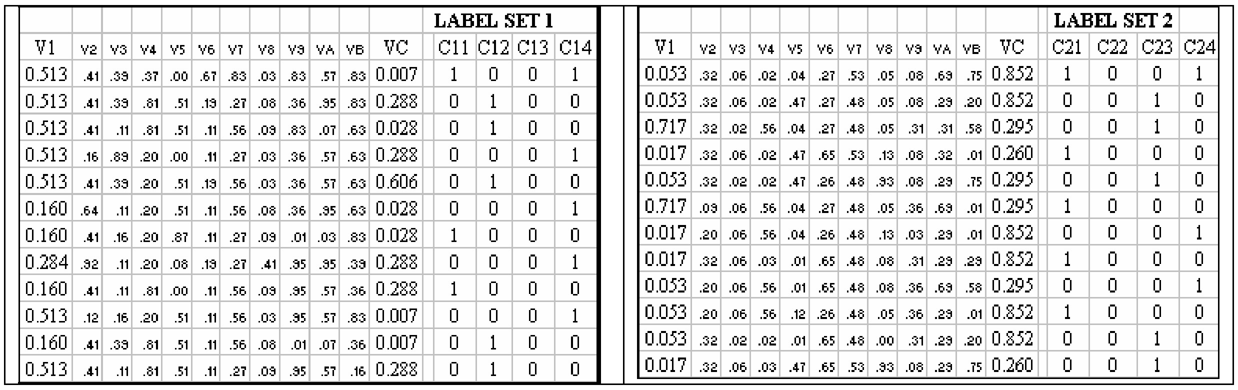
Table 5. Partial View of *ND2* (*DB1* encoded with Codes Derived from *Seed=31416*).

5.2 Similitude of Classification with Different Codes

The correct identification of analogous clusters is compulsory if, as intended, we are to determine the classification equivalence. To test the purported cluster similarity poses the technical problem we describe and solve in what follows.

5.2.1 The problem of cluster matching

Assume that tables T1 and T2 consisting of attributes V1,...,VC have been classified into 4 clusters and labeled as illustrated in Fig. 6.

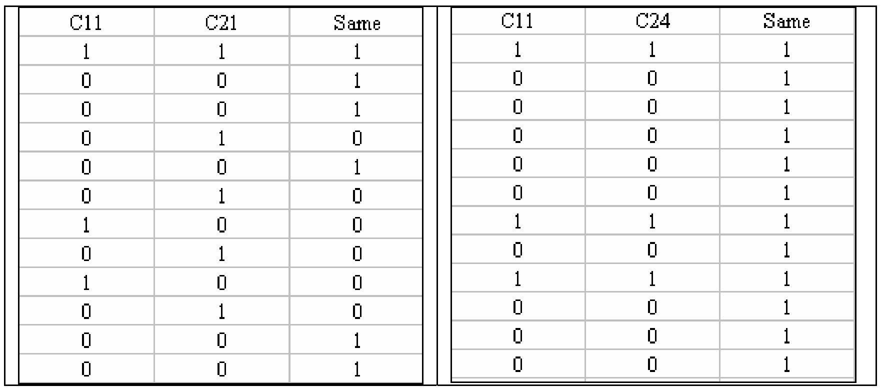


												LABEL SET 1			
V1	V2	V3	V4	V5	V6	V7	V8	V9	VA	VB	VC	C11	C12	C13	C14
0.513	.41	.39	.37	.00	.67	.83	.03	.83	.57	.83	0.007	1	0	0	1
0.513	.41	.39	.81	.51	.19	.27	.08	.36	.95	.83	0.288	0	1	0	0
0.513	.41	.11	.81	.51	.11	.56	.09	.83	.07	.63	0.028	0	1	0	0
0.513	.16	.89	.20	.00	.11	.27	.03	.36	.57	.63	0.288	0	0	0	1
0.513	.41	.39	.20	.51	.19	.56	.03	.36	.57	.63	0.606	0	1	0	0
0.160	.64	.11	.20	.51	.11	.56	.08	.36	.95	.63	0.028	0	0	0	1
0.160	.41	.16	.20	.87	.11	.27	.09	.01	.03	.83	0.028	1	0	0	0
0.284	.92	.11	.20	.08	.19	.27	.41	.95	.95	.39	0.288	0	0	0	1
0.160	.41	.11	.81	.00	.11	.56	.09	.95	.57	.36	0.288	1	0	0	0
0.513	.12	.16	.20	.51	.11	.56	.03	.95	.57	.83	0.007	0	0	0	1
0.160	.41	.39	.81	.51	.11	.56	.08	.01	.07	.36	0.007	0	1	0	0
0.513	.41	.11	.81	.51	.11	.27	.09	.95	.57	.16	0.288	0	1	0	0

												LABEL SET 2			
V1	V2	V3	V4	V5	V6	V7	V8	V9	VA	VB	VC	C21	C22	C23	C24
0.053	.32	.06	.02	.04	.27	.53	.05	.08	.69	.75	0.852	1	0	0	1
0.053	.32	.06	.02	.47	.27	.48	.05	.08	.29	.20	0.852	0	0	1	0
0.717	.32	.02	.56	.04	.27	.48	.05	.31	.31	.58	0.295	0	0	1	0
0.017	.32	.06	.02	.47	.65	.53	.13	.08	.32	.01	0.260	1	0	0	0
0.053	.32	.02	.02	.47	.26	.48	.93	.08	.29	.75	0.295	0	0	1	0
0.717	.09	.06	.56	.04	.27	.48	.05	.36	.69	.01	0.295	1	0	0	0
0.017	.20	.06	.56	.04	.26	.48	.13	.03	.29	.01	0.852	0	0	0	1
0.017	.32	.06	.03	.01	.65	.48	.08	.31	.29	.29	0.852	1	0	0	0
0.053	.20	.06	.56	.01	.65	.48	.08	.36	.69	.58	0.295	0	0	0	1
0.053	.20	.06	.56	.12	.26	.48	.05	.36	.29	.01	0.852	1	0	0	0
0.053	.32	.02	.02	.01	.65	.48	.00	.31	.29	.20	0.852	0	0	1	0
0.017	.32	.06	.03	.47	.65	.53	.93	.08	.29	.75	0.260	0	0	1	0

Fig. 6. Segments of labeled Data Base T1 and T2.

This labeling convention is convenient since one may easily count the number of matches between two clusters. However, clustering algorithms do not necessarily yield the same order of the label columns. For example, in Fig. 7 we have compared column C11 to C21, on the one hand and C11 to C24 on the other. The first choice yields a count of 6 matches leading us to the conclusion that sets C11 and C21 do not match and that, therefore, T1 and T2 do not share the same clusters. The second choice, however, yields the full 12 matches. Therefore, in this instance one must conclude that column C11 (from set 1) actually corresponds to column C24 (from set 2). Accordingly, we should also conclude that T1 and T2 correspond to the same set for cluster 1. The correct pairing has to be achieved in similar fashion for all clusters.



C11	C21	Same
1	1	1
0	0	1
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0
0	1	0
1	0	0
0	1	0
0	0	1
0	0	1

C11	C24	Same
1	1	1
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
1	1	1
0	0	1
1	1	1
0	0	1
0	0	1
0	0	1

Fig. 7. Similarity for choices 1 and 2.

If there are m clusters and p tuples there are m^p possible combinations of valid labeling sets. We need to investigate which of these does actually correspond to the proper matching of the m clusters in T1 with those of T2. Only then we may compare T1 and T2 and determine their similarity. To achieve this identification we applied the following algorithm.

5.2.2 Algorithm to optimize cluster matching

```
1. Create a matching table "MT" of dimensions $m \times m$ .
   Make MT($i,j$) $\leftarrow 0$ for all $i, j$.
2. For $i \leftarrow 1$ to $m$
     For $j \leftarrow 1$ to $m$
       If column $i$ = column $j$
         MT($i,j$) $\leftarrow$ MT($i,j$) +1
       endif
     endfor
   endfor
```

MT(i,j) will contain the number of matches between cluster i of table T1 and cluster j of table T2.

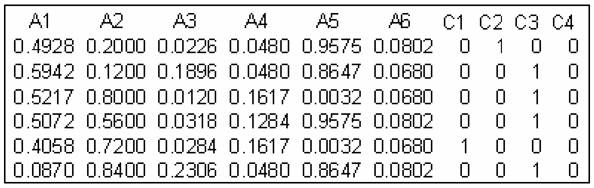
3. Create a table "Scores" of dimension Q.

```
4. For $i \leftarrow 1$ to $Q$   ($Q$ >>0)
   4.1. Set a random valid sequence $S_i$ of $m$ possible matching sequences between the clusters of T1 and those of T2.
   4.2. Find the number of matches $M_i$ between T1 and T2 from table MT as per $S_i$.
   4.3. Make Scores($i$) $\leftarrow M_i$.
  endfor
5. $I \leftarrow$ index of max[Scores($i$)] for all $i$.
6. Select $S_I$. This is the matching set which maximizes the number of coincidences between the clusters of T1 and T2.
```
□

The core of the algorithm lies in step 4.1 where the valid matching sequences are determined. This algorithm will find the sequence which maximizes the number of matches between the clusters of T1 and T2 with high probability provided Q is large enough. In our experiments we made Q =1000. Given the large number of possible pairings between the clusters of T1 and T2 the algorithm is a practical way to select which cluster of T1 should be paired with which cluster of T2.

5.2.3 Comparison of Codes from Different Seeds

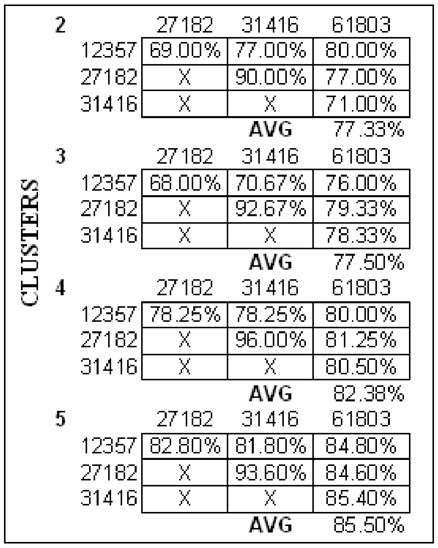
We clustered *ND1*, *ND2*, *ND3* and *ND4* using the Fuzzy C-Means (FCM) clustering algorithm. It requires the user to specify *NC* (the number of clusters). We did not know the actual value of *NC* and, therefore, we trained FCM for *NC=2,3,4,5*. The result of every run of FCM is a set of coordinates defining the center of the clusters. These coordinates consist of one *n*-dimensional vector for each cluster. Next we assigned membership values to all $T = |DB1|$ = 100 tuples. We did this by calculating the closest center to every tuple and assigning a label "1" to the column corresponding to the cluster and "0" to all other columns. A simple example of *MD1* for *NC=4* after labeling is shown in Table 6.



A1	A2	A3	A4	A5	A6	C1	C2	C3	C4
0.4928	0.2000	0.0226	0.0480	0.9575	0.0802	0	1	0	0
0.5942	0.1200	0.1896	0.0480	0.8647	0.0680	0	0	1	0
0.5217	0.8000	0.0120	0.1617	0.0032	0.0680	0	0	1	0
0.5072	0.5600	0.0318	0.1284	0.9575	0.0802	0	0	1	0
0.4058	0.7200	0.0284	0.1617	0.0032	0.0680	1	0	0	0
0.0870	0.8400	0.2306	0.0480	0.8647	0.0802	0	0	1	0

Table 6. An example of *ND1* with labels.

This was done for all *NDi* and for *NC*=2,3,4,5. Having done this we counted, for all 6 combinations of the *NDi*, the coincidences in the clustering columns. The result of these comparisons is shown in Table7.



CLUSTERS		27182	31416	61803
2	12357	69.00%	77.00%	80.00%
	27182	X	90.00%	77.00%
	31416	X	X	71.00%
			AVG	77.33%
3	12357	68.00%	70.67%	76.00%
	27182	X	92.67%	79.33%
	31416	X	X	78.33%
			AVG	77.50%
4	12357	78.25%	78.25%	80.00%
	27182	X	96.00%	81.25%
	31416	X	X	80.50%
			AVG	82.38%
5	12357	82.80%	81.80%	84.80%
	27182	X	93.60%	84.60%
	31416	X	X	85.40%
			AVG	85.50%

Table 7. Similitude Matrix for Different Codes.

The headings correspond to the roots of the pseudo-random number generator. From Table7 we can see, for instance, that, given 4 clusters, the labels of the code resulting from 27182 and those from 31416 match in 96% of the cases. Likewise, if we attempt clustering with *NC*=5 and focus on row 12357 and column 31416 the labels match in 81.80% of the cases. Finally, with **AVG** we denote the matching average for all 4 codes for *NC*=2,3,4,5.
How encouraging is this result? From Table 8 we take the worst case result (68% of the clusters matched), which corresponds to one where 3 clusters were considered for the pair of roots (12357, 27182). The number of possible labeling configurations for 3 clusters and 100 tuples is 3^{100}. The probability of correctly picking the right labeling 68% of the time is $\alpha = [1/3]^{68} \approx 3.6e\text{-}33$. On the other hand, there are $\beta = C(100,68) = 100!/(32!68!) \approx 1.4e+26$ ways to attempt such 68 correct guesses. Hence, the probability of correctly achieving a score of 68% by chance alone is $\alpha \times \beta \approx 5e\text{-}7$. In the best case result, where 96% of the clusters matched, a similar calculation yields a probability of 3.6e-39 of this event to happen by chance. These calculations give strong credence to our hypothesis and we must conclude that CENG is able to preserve the patterns embedded in the data.

6 Conclusions

We have described a method to transform mixed databases (those where numerical and categorical variables are found) into strictly numerical databases by applying machine learning techniques. We look for a "perfect" code set (PCS) which preserves all patterns embedded in the data when each one of the categorical instances is replaced by a numerical code. PCS may or may not exist and, therefore, we approximate PCS via a two-fold strategy. On the one hand, patterns are extracted from the database through a learning process that finds the minmax global approximation error for all attributes. On the other, the best practical set of codes is extracted by minimizing the largest error via a genetic algorithm. We gave the mathematical foundations which allow us to ascertain that the codes thusly found will be optimal, in a practical sense. We described the method in detail. We argued that the solutions are not unique since the learning machine tools we used are stochastically based and, hence, different starting seeds of the pseudo random number generator lead to different final codes. Given this, we tested the practical validity of the method by comparing various sets of codes and comparing the clustering results. Our thesis is that different pattern preserving codes will yield similar cluster assignments to the tuples on the database. To this effect we trained the same mixed database with four different seeds. Then we obtained a similitude matrix in which the worst case number of matches is approximately 70%. We calculated the probability of achieving such high score by chance alone and found that it is $O(10^{-39})$. The conclusion is that CENG leads to a valid numerical assignment of the categorical instances with very high probability.
CENG, thus, allows us to apply numerical clustering algorithms to mixed databases. This frees us from subjective a priori considerations present in all alternative non-metric approaches. There are two major disadvantages: a) The codes are dependent on the data. That is, every data base has to be trained and the codes thusly found are, in general, not applicable to other databases. b) The method is computationally intensive. Both disadvantages are minor when compared to the advantages obtained.
One observation obtained from the similitude matrix is that the best approximation error is smaller as *NC* is larger. Since the probability of better coincidence ratios (say *CR*) is smaller as *NC* increases this may be explained by noticing that *CR* will be higher if the actual value of *NC* coincides with the one in the table. Therefore, an interesting point to investigate is whether our method yields the actual number of clusters (usually unknown) as *CR* reaches a maximum.
An interesting application which arises as a corollary of CENG is that plain text (and other kinds as well) are now susceptible of being clustered directly. For example, if we categorize a text in the English language by the different lexical components (verbs, nouns, pronouns, etc.) the various words become instances in a category and susceptible to numerical analysis after applying CENG. Thereafter, mining text may be tackled via the classical clustering numerical techniques.
These suggested applications will only be proved valid if the method is tested fully. Here we have only presented experimentally basic evidence. We will proceed to test our method in mixed databases with much larger number of tuples and attributes. Theoretically we can claim that our method is applicable in general. However, until those tests are performed we can make no claim to practical generality. We expect to publish in this regard soon.

ACKNOWLEDGEMENT
This paper was partially developed under the auspice of Fundación Mexicana de Cultura A.C.

References:

[1] Ienco, D., Pensa, R. G., & Meo, R. (2012). From context to distance: Learning dissimilarity for categorical data clustering. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(1), 1.

[2] He, X., Feng, J., Konte, B., Mai, S. T., & Plant, C. (2014, August). Relevant overlapping subspace clusters on categorical data. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 213-222). ACM.

[3] Xiong, T., Wang, S., Mayers, A., & Monga, E. (2012). DHCC: Divisive hierarchical clustering of categorical data. Data Mining and Knowledge Discovery, 24(1), 103-135.

[4] David, G., & Averbuch, A. (2012). SpectralCAT: Categorical spectral clustering of numerical and nominal data. Pattern Recognition, 45(1), 416-433.

[5] Qin, H., Ma, X., Herawan, T., & Zain, J. M. (2014). MGR: An information theory based hierarchical divisive clustering algorithm for categorical data. Knowledge-Based Systems, 67, 401-411.

[6] Chen, L., & Wang, S. (2013, August). Central clustering of categorical data with automated feature weighting. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (pp. 1260-1266). AAAI Press.

[7] Cheung, Y. M., & Jia, H. (2013). Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. Pattern Recognition, 46(8), 2228-2238.

[8] Saha, I., & Maulik, U. (2014). Incremental learning based multiobjective fuzzy clustering for categorical data. Information Sciences, 267, 35-57.

[9] Kovacs, F., & Ivancsy, R. (2006). A novel cluster validity index: variance of the nearest neighbor distance. WSEAS Transactions on Computers, 5(3), 477-483.

[10] Kuri-Morales, A., & Aldana-Bobadilla, E. Finding Irregularly Shaped Clusters Based on Entropy, ICDM.

[11] Peterson, Leif E. "K-nearest neighbor." Scholarpedia 4.2 (2009): 1883.

[12] Kohonen, Teuvo. Self-organizing maps. Vol. 30. Springer Science & Business Media, 2001.

[13] Bezdek, James C., Robert Ehrlich, and William Full. "FCM: The fuzzy c-means clustering algorithm." Computers & Geosciences 10.2 (1984): 191-203.

[14] Chung, Fu-Lai, and Tong Lee. "Fuzzy learning vector quantization." Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on. Vol. 3. IEEE, 1993.

[15] Aldana-Bobadilla, Edwin, and Angel Kuri-Morales. "A Clustering Method Based on the Maximum Entropy Principle." Entropy 17.1 (2015): 151-180.

[16] Allison, Paul D. Logistic regression using SAS: Theory and application. SAS Institute, 2012.

[17] Kuri-Morales, A., & Aldana-Bobadilla, E. (2010). Finding irregularly shaped clusters based on entropy. In Advances in Data Mining. Applications and Theoretical Aspects (pp. 57-70). Springer Berlin Heidelberg.

[18] Glantz, S. A., & Slinker, B. K. (1990). Primer of applied regression and analysis of variance. McGraw-Hill, Health Professions Division.

[19] Tirian, G. O., Rusu-Anghel, S., Panoiu, M., & Bretotean, C. P. (2009, July). Control of the continuous casting process using neural networks. In N. E. Mastorakis, V. Mladenov, Z. Bojkovic, S. Kartalopoulos, A. Varonides, M. Jha, & D. Simian (Eds.), WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering (No. 13). WSEAS.

[20] Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of control, signals and systems 2.4 (1989): 303-314.

[21] Kuri-Morales, A., "The Best Neural Network Architecture", In MICAI 2014, Springer, 72-84, 23/11/2014.

[22] Widrow, Bernard, and Michael A. Lehr. "30 years of adaptive neural networks: perceptron, madaline, and backpropagation." Proceedings of the IEEE 78.9 (1990): 1415-1442.

[23] Bloom, C. "PPMZ2: high-compression Markov predictive coder, 1999."

[24] Rudolph, Günter. "Convergence analysis of canonical genetic algorithms." Neural Networks, IEEE Transactions on 5.1 (1994): 96-101.

[25] Little, P., & Rylander, B. (2007). Problem partitioning in hybrid genetic algorithms. WSEAS Transactions on Systems, 6(2), 395.

[26] Kuri-Morales, A. F., (2003). Solution of simultaneous non-linear equations using genetic algorithms. WSEAS Transactions on Systems, (1), 44-51.

[27] Kuri-Morales, Angel Fernando, Edwin Aldana-Bobadilla. "The best genetic algorithm I." Advances in Soft Computing and Its Applications. Springer Berlin Heidelberg, 2013. 16-29.