Barceló Rico, F.; Diez, J. (2012). Geometrical codification for clustering mixed categorical and numerical databases. Journal of Intelligent Information Systems. 39(1):167-185. doi:10.1007/s10844-011-0187-y

Noname manuscript No.
(will be inserted by the editor)

Geometrical codification for clustering mixed categorical and numerical databases

Fatima Barcelo-Rico · Jose-Luis Diez

Received: date / Accepted: date

Abstract This paper presents an alternative to cluster mixed databases. The main idea is to propose a general method to cluster mixed data sets, which is not very complex and still can reach similar levels of performance of some good algorithms. The proposed approach is based on codifying the categorical attributes and use a numerical clustering algorithm on the resulting database.

The codification proposed is based on polar or spherical coordinates, it is easy to understand and to apply, the increment in the length of the input matrix is not excessively large, and the codification error can be determined for each case.

The proposed codification combined with the well known k-means algorithm showed a very good performance in different benchmarks and has been compared with both, other codifications and other mixed clustering algorithms, showing a better or comparable performance in all cases.

Keywords Mixed data · Clustering · Data Conversion · k-means · Codification error

Institut d'Automatica i Informatica Industrial
Universitat Politecnica de Valencia
Cami de Vera s/n CP 46022, Valencia, Spain
e-mail: *fabarri@upv.es*
Tel: +34963877007 Ext. 75782

1 Introduction

Nowadays, there is an increasing need for pattern-discovery in many and different areas. Applications range from engineering to medical, and also cover social or economical needs [13]. The mechanisms for extracting useful information embedded in (probably large) databases are being developed in several directions, all of them with the aim of helping the automated search to be performed more accurately. In this context, clustering is one of the data mining techniques that is receiving more attention, probably due to its valuable properties of finding groups of behaviour [11]. The target of clustering is to group data into different sets, called clusters, in which the data belonging to each cluster will be more similar to the elements of the same cluster than to elements of others clusters. Thus, a similarity measure has to be defined, according to which the elements will be clustered.

The applications of clustering techniques are in an increasing trend [8]. Most of the already designed algorithms are either only numerical or only categorical [12]. However, the mixed data applications are becoming very demanding. Medical and bank sets of data are examples of this issue [19]. These databases usually include attributes of both types, viz: age, height, sex, type of work, and so on.

There are two main approaches in order to solve the problem of dealing with mixed (numerical and categorical) databases. The first one is the data conversion; the other is the design of a similarity measure able to deal with both kinds of data. In the first approach, one type of data is converted to the other in order to use an algorithm able to work only with one kind of data. Either the categorical attributes are codified [16], or the numerical ones are discretized to be considered as a category [15]. In the second approach, a common approximation is to use a similarity measure that considers each type of attribute in a different one way [1], [17] .

In this paper, the authors propose a new type of codification for the categorical attributes. The aim of this approach is to be able to use the numerical clustering algorithms which have been already designed and that are known to work well for numerical data. Although this codification could have been used with many numerical algorithms, it has been chosen *k-means*, due to its simplicity and also because it is well known by research community.

This paper is organised as follows. In section 2, critical conversion of attributes' characteristics and some codification approaches are presented. In section 3, the proposed approach and its advantages and drawbacks are described. In section 4, the experiments done using this new codification together with the *k-means* algorithm is presented, also the comparative with other codification and other algorithms. Finally, in section 5, the conclusions are described and some future works are proposed.

2 Review of Mixed Data Clustering Approaches.

Clustering algorithms are applied to a set of objects, X, called **databases** or **data sets**, $X = \{x_1, x_2, ..., x_n\}$, which contain all the information of the set. The data set is composed by n elements called **objects**, x_i where $i = 1, 2, ..., n$. Each object can be represented in the following way $x_i = (x_{i1}, ..., x_{iz})$, where each component from 1 to z will represent an attribute from the attribute space A, $x_{iv} \in A_v$, $v = 1 \dots z$. Attributes may represent characteristics, variable, dimension, field, etc. This *object by attribute* data format corresponds to an $n \times z$ matrix and is used by most of the clustering algorithms.

The purposes of clustering techniques are many. The most common is to find hidden patterns in the data set. Nevertheless, there are many other applications, like to identify input-output models of systems [2] [9], to find linear operating regions of non-linear behaviours [10], etc.

Within the clustering algorithms, different classifications can be found. Two common ones are: 1) To classify them by the way they cluster the objects and, 2) To classify them by the type of attributes they can work with. This work is around this second classification.

2.1 Types of Attributes.

The attributes or characteristics of a database can be classified according to its nature. This classification divide them into many types. *Categorical* and *numerical* are two very important types among them in terms of designing or selecting a clustering algorithm. This is the upper level classification and some subtypes can be found within it.

An attribute is said to be **numerical** if it is represented by a (usually real) number, the attribute domain is $A_v \in R$. An obvious subdivision arises from this definition, given that the characteristic of the magnitude can be *discrete* or *continuous*. The continuous case refers to those situations where the number can take any value in the considered interval. A discrete attribute can only take a finite set of values within the interval. An example of each of them could be *temperature* and *age*, respectively.

On the other hand, an attribute is **categorical** if its magnitude is represented by a category. This is a symbolic representation, $A_v \in A_A, A_B, ..., A_W$, where W is the number of different categories for that attribute. Categories are, by definition, discrete values. Simple examples of categorical attributes are: *job*, *favourite music*, etc. A particular case of categorical attributes is when this can only adopt two different values: **Binary** attributes, $A_v \in A_A, A_B$. Examples of this are the pairs: [Yes, No], [Left, Right], [Up, Down], etc.

Gradable attributes are another subtype of categorical attributes, like: [very good, good, regular, bad, very bad]. As well as **Ordinal** attributes (First, Second, Third, etc.). However, both subtypes, even though categorical, have the inner property of being related to each other. This relation enables them, in clustering terms, to be treated as numerical ones.

The type of data is a key piece when clustering and has to be highly considered. Thus, attending to the type of data than can work with, the clustering algorithm can be classified as *numerical*, *categorical* or *mixed*. The numerical algorithms are the ones that first came up. As said before, clustering algorithms are based on a similarity measure, which quite often measures distances between objects, like equation (1), where x and y are two objects. A common distance measure used is the Euclidean distance ($p = 2$ in (1)). Obviously, this measure can only be used with numbers, and this is why numerical algorithms were the first developed. Nowadays, many other distance measures have been defined, but numerical algorithms are still the most developed type for clustering purposes.

$$\delta(x, y) = \|x - y\|_p \tag{1}$$

Categorical methods came up with the aim of clustering symbolic objects. Now the similarity measure has to be able to treat non numerical attributes. Therefore, another way of measuring the similarity/dissimilarity between object had to be defined for these cases. A very popular one is the one shown in equation (2) [7], where x and y are objects and i refers to the attribute.

$$\delta(x_i, y_i) = \begin{cases} 0 \ (x_i = y_i) \\ 1 \ (x_i \neq y_i) \end{cases} \tag{2}$$

In most real world applications, the databases are usually not only numerical nor only categorical. Usually they are formed by a combination of both types in different proportions. One approach that can be found in the literature to work with mixed sets is the definition of a similarity/dissimilarity measure that differs between the two types of attributes. In that way an option could be to use equation (1) when the attributes are numerical and equation (2) when they are categorical and then add both values weighed accordingly to the application. Yet, the design of a mixed (dis)similarity measure capable of handle both types of attributes and perform well is complex. That is the reason why other approaches are investigated.

Another approach used in literature is the conversion of one type of data to the other. This is the base of the new codification approach offered here and the comparative study done in this work. In the following section it will be described in more detail.

2.2 Pros and Cons of Codification approaches.

In the literature it can be found that the most of mixed algorithms are designed for a particular case or conditions. Besides, when there is dis(similarity) function for each type of attribute, the determination of the appropriate balance (contribution of each type) is an intuitive and difficult task.

Other approaches have been proposed to work with mixed databases. One is the **data codification**, eq. (3), where x_{iv} is the attribute to be converted

and y_{iv} the assigned attribute. The basic idea is to use an algorithm which works well with only one type of data (either categorical or numerical) and to convert the data which is no appropriated through a codification.

$$x_{iv} \leftarrow y_{iv} \; \forall \; x_{iv} \subset x_i \tag{3}$$

If the chosen algorithm is categorical, the numerical attributes can easily be converted by taking them by intervals. Each of interval will be considered as a category and will be treated independently from the others [15]. What happens here is that the converted categories are dependent, one is larger or smaller than the other, and this information is ignored by treating them as independent. Besides, in general terms, numerical algorithms are further developed than categorical ones.

In this line, the approach that usually gives better results is to use a numerical algorithm, and to convert categorical attributes into numerical values. However, numbers have the inner property of being dependent, while categorical attributes are not, unless they are ordinal attributes [5]. This makes the codification of categorical attributes a complicated process.

The codification used to transform categorical into numerical attributes must keep properties of the categorical set in the converted numerical one.

Among the most common codifications are:

- **Ordinal code**. It consists in assigning directly an *integer number* to each of the categories, i.e. if the attribute can adopt four possible categories, they would be codified as {1,2,3,4}. The main advantage of this codification is that the length of the converted attribute is the same as the original one. The main drawback is that this code introduces a grade between the converted attributes that is not present in the symbolic variables.
- **Gray code**. Used in [16]. This consists in codifying each categorical attribute with a vector whose length will be the number of possible categories for that attribute. Then, for each object, this vector will be filled with zeros but the value of its category, which will be considered as one. An example can be seen in table (1). The main advantage of this codification is that the independence between categories is kept for all the objects, being each one completely different from all the others. Nevertheless, this codification technique has also many drawbacks:
  1. The first and main one is that if the number of possible values is large and the number of categorical attributes is big, then the converted input database can have a very large number of inputs, i. e. 5 categorical attributes for 10 categories each will make a vector of 50 inputs which will not be very easily treated by the clustering algorithm.
  2. A second drawback, consequence of the previous one, is the computation time of algorithms when working with converted data sets.
  3. The third one, and worst, is that each converted attribute will have different length and, therefore, they will affect differently in the clustering process.

– **Binary code**. It consist in codifying each category assigning a value in binary code. If the attribute has four possible categories the values would be {00, 01, 10, 11}. The advantage of this codification is that the number of new attributes is not very large, not affecting much to the length of the new dataset and the computing time. The two main drawbacks are:
  1. Converted categories are not completely independent from all the others.
  2. And, attributes with different lengths will affect differently in the clustering process.

All codifications have different good and bad points, but non of them could be described as perfect for the purpose described here. For this is the reason, the aim of this paper is to propose a new codification, in order to improve the existing ones.

Table 1 *Codification with only one 1 for each object (Grey code)*

Jobs	Engineer	Farmer	P.A.	Lawyer
Obj. 1	0	1	0	0
Obj. 2	0	0	0	1
Obj. 3	1	0	0	0
Obj. 4	0	0	1	0

3 Polar and Spherical codification approach.

In this section the new approach will be described in detail, relating its advantages and drawbacks. A previous analysis of this method can be found in [3], but the approach is fully developed and studied in the following

One of the important properties of the categorical attributes is the *independence* among categories, which involves no fix way of ordering them. Thinking of Euclidean distance, one of the most used ones in similarity measures, the proposed spatial points should be as equally distanced as possible.

3.1 Polar codification.

The approach proposed in this work is based on assigning a pair of polar coordinates within the unit circle to each category. Thus, the 2D space will be the conversion domain. Then, the circle will have to be divided into as many regularly distanced points as the number of categories in the attribute. Here, *regularly distanced* means that the distance from one point to next is always constant. The advantages of this new codification are mainly two:

1. The first is that, no matter how large the number of categories is, the codified attribute will be a vector of two elements. Then, all categorical attributes will affect in the same way in the clustering process.

2. The second advantage is that all the categories will have the same characteristics as the others. This means that the converted set of points will be balanced, getting in this way one characteristic needed in categorical values.

The disposition of the points in a circle makes them not be all equally spaced, in terms of Euclidean distance. Yet, all of them have the same properties given that it does not matter the order they are placed, they will be balanced. Looking at Figure (1) it can be seen a graphic example when an attribute can adopt six different categories: the circle has been divided into six points evenly distributed. It is obvious that the distance from one point to another in the circle are not all the same. However, it can be seen that all points have the same characteristics, they have 2 next neighbours, 2 following neighbours and an opposite point. And **they all** follow the **same pattern**. This property of symmetry and balance among the points is a good characteristic for the conversion of categorical values.

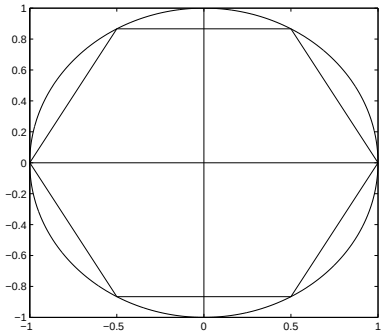

Fig. 1 *Case when 6 categories are possible and the circle is divided into 6 points*

Mathematically, the new values can be represented as shown in equation (4) where N is the number of different categories that the attribute can take, i is related to each category and goes from 1 to N.

$$Cat_i = \left[\cos\left(\frac{(i-1)\cdot 2\cdot\pi}{N}\right) , \sin\left(\frac{(i-1)\cdot 2\cdot\pi}{N}\right) \right] \tag{4}$$

An exception for the conversion of one categorical input into 2 numerical ones is when the categorical input is binary. In that case there is no need for 2 new coordinates but for one, given that the division of the circle into 2 points will have both of them projected onto the same axis and then the other coordinate is in both cases null and can be removed. It can be seen that when equation (4) is applied, all codified attributes will be between the interval [-1,1].

It is clear that the polar codification is not perfect either: it also introduces some error. Yet, it has advantages, specially the fact that converted points are balanced, that could make it perform better than others codifications.

In section (4), it will be seen some examples where a comparison with other codifications is shown and also the good performance of the proposed approach.

3.2 Spherical Coordinates.

The larger the number of points to divide the circle into, the larger the codification error will be too. It is clear that the proposed codification is not very accurate to codify attributes with a large number of categories. For this reason, an option could be to pass from the 2D domain conversion to 3D, i.e. *spherical coordinates*. Thus, the codification error will be reduced.

In order to distribute regularly points in the sphere, the procedure is not as straightforward as in the circle. The regular polyhedra in 3D correspond to the name of *platonic solids* [6] and they and their number of vertices (V) are: the tetrahedron ($V = 4$), the octahedron ($V = 6$), the hexahedron ($V = 8$), the icosahedron ($V = 12$), the dodecahedron ($V = 20$) and the truncated icosahedron ($V = 60$).

The coordinates of the vertices of each figure depend on the polyhedra itself and there is not a general equation to describe all placements. Most of them depend on the *golden ratio* (φ) and the coordinates for each of the vertices have determined values [4].

As the number of vertices can not be any number, for codifying an attribute with N categories it will have to be used the regular polyhedra with the number of vertices equal or larger than this N. Then it can happen that the number of points exceed the number of categories. In this case what should be tried is to place the points in a way that they are as balanced as possible.

The spherical code is a 3D extension of the 2D polar code. Thus even when not all the vertices of the figure are taken for the codification, the codification error from the codification will be smaller. This should help the clustering process and in many cases it does, as will be seen in experiments of section (4).

3.3 Higher order Coordinates.

If more accuracy is desired in the codification, an extension of the polar and spherical coordinates can be applied, i.e. higher order coordinates . Nevertheless, finding the vertices in any space R^N, when $N > 3$ is not quite complex. The regular cases in higher dimensions are called **polytopes**. This is something that goes far away from the aim of this paper and will not be considered here. Besides, in spite of the codification error, it will be shown in the next section how the proposed approach works well.

3.4 Codification error.

It has already been mentioned in the description of the different codifications that all of them but the gray code have some codification error. Table (2) shows how the relative error for the described codifications varies for different number of possible categories. The relative error means the difference between the maximum and the minimum distances scaled by the minimum distance between points: $e_{rel} = (max_d - min_d)/min_d$.

Table 2 *Relative error for several number of categories*

No. cat	**Ordinal**	**Gray**	**Binary**	**Polar**	**Spherical**
2	0	0	0	0	0
3	1	0	0.41	0	0
4	2	0	0.41	0.41	0
5	3	0	0.73	0.62	0.41
6	4	0	0.73	1	0.41
7	5	0	0.73	1.24	0.73
8	6	0	0.73	1.61	0.73
9	7	0	1	1.88	0.9

Table (3) shows how the variation in length of the converted attributes affect to the databse.

Table 3 *Influence in the attribute size for several number of categories*

No. cat	**Ordinal**	**Gray**	**Binary**	**Polar**	**Spherical**
2	1	2	1	1	1
3	1	3	2	2	2
4	1	4	2	2	3
5	1	5	3	2	3
6	1	6	3	2	3
7	1	7	3	2	3
8	1	8	3	2	3
9	1	9	4	2	3

In both tables it can be seen how polar and spherical codification are interesting. One the one hand, because the error, specially of the spherical codification is small for many cases. Error of the polar codification is also small when the number of categories is under six.

On the other hand, both codifications are interesting because they do not contribute by much on the length of the converted matrix and, besides, their contribution is almost constant. These tables show only up to nine categories. However, when the number of them increases, polar and spherical codification still contribute with the same amount of converted attributes, which is very convenient for the clustering process.

3.5 Comparative study of codification approaches.

Many different approaches for codification have been described through this paper. It seems a good idea, therefore, to make a comparison of all of them in terms of the most important characteristics that any codification method should meet. Table (4) shows the main characteristics of the codification methods mentioned before, which are the main methods found in the literature.

Table 4 *Comparative of the several codification methods*

	Ordinal	**Gray**	**Binary**	**Polar**	**Spher**
Conversion size	Small	Large	Small/Medium	Small	Small
Influence	None	Large	Medium	Small	Small
E_{rel}	N-2	0	Small	Medium	Small
Comput load	Low	High	Low	Low	Low
Difficulty	Easy	Easy	Easy	Easy	Med

It can be seen that the approach here suggested has, as the rest of codifications, some advantages and some drawbacks. Nevertheless, it can be seen that it is clearly the codification that offers a compromise between both. Results of section (4) will show the performance of the proposed approach. Analysis shows that this can be considered the best codification approach to use for clustering purposes.

4 Examples.

Experiments have been done to validate the proposed codification approach. As the proposed method is for codifying categorical attributes into numerical values, it is needed to choose a numerical clustering algorithm. Many numerical algorithms can be found in literature. However, it can be said that the **k-means** algorithm is the most known one. It has been chosen for this work because it is one of the simplest and most intuitive. Clustering using several codifications approaches will be performed using this algorithm and results and conclusions will be shown after each experiment.

4.1 Description of k-means algorithm.

This algorithm [14] is by far the most popular clustering tool, used widely in the past and nowadays. The goal in k-means is to produce k clusters from a set of n objects, so that the objects of all clusters minimize the total error from objects to center. What is the same, the aim is to minimize the addition of all distances (Euclidean) from all the objects to the center of the cluster where they have been assigned to, equation (5):

$$J = \sum_{i=1}^{k} \left(\sum_{l=1}^{n} d\left(x_l, c_i\right) \right) \tag{5}$$

where n is the number of objects, k the number of clusters, c_i is center of cluster i, $d(x_l, c_i)$ is the distance of element x_l to the center of the cluster where it has been assigned i.

If this equation is expanded, then it is like equation (6):

$$J_{total} = \sum_{i=1}^{k} \left(\sum_{l=1}^{n} \sqrt{\sum_{v=1}^{z} \left(x_{l,v} - c_{i,v}\right)^2} \right) \tag{6}$$

where z is the number of attributes. So, the total distance will be root squared of the addition of the squares of the difference between each attribute and the center of its cluster.

k-means algorithm has as input parameter k the number of clusters. The algorithm returns (output) the centers or representatives of all clusters, c_i. The algorithm procedure is as follows:

1. Select k objects as initial centers,
2. Assign each data object to the closest center,
3. Recalculate the centers of each clusters, as the mean of all the elements belonging to that cluster,
4. Repeat steps 2 and 3 until centers do not change.

This algorithm works very well if and only if the parameter k is set properly. Here, in the selected benchmarks, the number of clusters in the output is known, so it will not be a problem.

It is also important to see that the cost index shown in equation (6) will consider all attributes with the same importance. For this reason, attributes will have to be normalized to make them all have same importance a priory on the clustering process.

4.2 Data Processing Considerations.

Before doing the experiments, some pre-processing of the data should be done, to make the benchmarks have certain properties needed for clustering. The first thing to do is the normalization of the attributes. There are many possible normalizations (interval, z-score, etc.). In this case it has been preferred to normalize all attributes between the values that the codified categorical values will adopt, which will be the range [-1,1] as the circle or the sphere will have radius $R = 1$.

It is also important to define a way to treat the missing values, given that many benchmarks found are not complete. Here, it has been decided than any object that has one or more missing values will be removed from the data set. Another possibility could be to replace the missing value with the mean of

the attribute [18]. However, this can lead to very unrealistic objects and to inaccurate results. The most accurate option is to omit these values as nobody is sure of them [15].

The similarity measure does not have to be defined, as the algorithm for the clustering is k-means, which uses the Euclidean distance.

Given that the k-means algorithm starts randomly, it does not always offer the same result. If the same experiment is run many times, there is a result which appears more often, this result corresponds to the minimum index, considering this as expressed before, and this is the value that has been considered as the *right* value of the clustering process using this algorithm. For this reason, all experiments have been run a large enough number of times (200 times). The results shown have been taken from the most repeated value and computational time refers to all repetitions together. As the two last databases are quite large, the number of repetitions is just 100.

4.3 Databases experiments.

To check the performance of the new proposed codification several benchmarks have been chosen. The data sets were provided by *UCI Repository of machine learning databases* (http://archive.ics.uci.edu/ml/). The selected data sets were the ones suitable for this work. Most of them, present both types of attributes in different proportions, and two of them only categorical attributes.

In the following subsections the databases will be briefly described, including details about the number of objects, the number of attributes and their types and the number of clusters at the output. Results of the performance with different codifications and with the proposed one will be presented, using k-means in all cases. Finally, the results will be analyzed over the whole comparative.

4.3.1 Vote Database.

The Vote database is fully categorical. It is composed by 435 objects with 16 binary attributes each. Only 232 objects are complete. The output is divided into 2 clusters corresponding to either the Democrat (124) or the Republican (108) voters. The proposed codification will have to be applied to all the attributes, as seen in section (3). Results of all the experiments using the different codifications are shown in table (5).

It can be seen that all experiments give same results in terms of well clustered objects and very similar ones in terms of computation time. This is due to the fact that all attributes are binary and every codification but the gray code assign just two values ({1,2} when ordinal, {0,1} when binary and {-1,1} in polars). Then, after normalization, all codified matrices result in the same normalized values for these three codifications. When the gray code is used the codified values are {[0 1], [1 0]}, this means that the resulting codified matrix

Table 5 *Comparison of some codifications for the Vote data set*

Codific.	Database size	Results	Time
Ordinal	232 × 16	89.66%	2.3s
Binary	232 × 16	89.66%	2.4s
Gray	232 × 32	89.66%	2.8s
Polar	232 × 16	89.66%	2.3s

doubles the size of the original matrix. However, as the database is not large, the computing time is similar to the others'. Results are exactly the same in all cases.

This is a good case to show that all codifications are equivalent for binary attributes and that the proposed one works as wells as the already existing codifications.

4.3.2 Heart Disease Database.

This is a mixed database, where there are a total of 13 attributes, 7 are categorical and the rest, 6, are numerical. That means that the mixture of types of data is quite balanced. There are 303 objects, 296 of which do not have missing values and are good for clustering. Of those, 136 correspond to sick patients (with heart disease) and the rest, 160, to people with no cardiopathies.

Now not all attributes have the same number of categories and them will be codified following equation (4), adopting the proper N for each case. The length of the objects will increase after using the codification and the final length will be 17, as 4 of the 7 categorical attributes have more than 2 possible categories. Table (6) shows the results in this case.

Table 6 *Comparison of some codifications for the Heart Disease data set*

Codific.	Database size	Results	Time
Ordinal	296 × 13	71.2%	2.3s
Binary	296 × 17	80.7%	3.2s
Gray	296 × 25	81.8%	3.4s
Polar	296 × 17	82.4%	3.2s

It can be seen now how the results using the different codification differ from one another more than in the previous case. This makes sense because now, after codified, the size and the values of the matrix are different in each case.

Comparing the four codifications, it can be observed than the performance of the polar approach is slightly better than the other three and it is even faster than the one using the gray code. Here the number of different categories is not large in any attribute and there is no need of using the 3D coordinates. All computing times are small as all converted data sets are not very large. Yet, even with these few instances it can already be seen some differences with

the timing, being the polar codification approach the fastest codification that offers good results.

4.3.3 Credit approval Database.

This database is also a mixed one with a total of 690 objects, where 653 objects were complete. The database has 15 attributes: 9 categorical and 6 numerical. Categorical attributes now can adopt from 2 to 14 categories. This does not affect the length of the converted attributes, which will be two for the polar codification and 3 for the spherical. The output reflects whether the credit card was approved for each person: in 307 cases it was and in 383 it was not. After the codification the length of converted matrix has become 20 instead of the initial 15, as 5 of the 9 categorical attributes can adopt more than 2 categories. Table (7) shows comparison between codifications for this case.

Table 7 *Comparison of some codifications for the Credit Approval data set*

Codific.	**Database size**	**Results**	**Time**
Ordinal	653 × 15	86.4%	3.8s
Binary	653 × 25	54.2%	6.6s
Gray	653 × 46	54.2%	6.4s
Polar	653 × 20	86.4%	4.7s
Spherical	653 × 23	86.4%	5.1s

These results differ very much to the ones of the previous database. Similarities in results are a mere coincidence. Here, in opposition to the other experiment, the ordinal codification is the one that offers the best clustering results. The other two, binary and gray codes, have a very poor performance in this case. The polar codification works as well as the ordinal one, offering, again, the best results of all codifications. Extension to 3D coord. has been done as some attributes have a large number of possible categories. This does not improve the results, indicating that those attributes do not affect much for the clustering process. Again computing time is not large in any of the cases, being the time of polar codification in an intermediate term.

4.3.4 Cylinder Band Database.

This database is also a mixed one with a total of 512 objects. However, only 277 instances are complete. They are not many, but the quantity of attributes is large (40). Some of them have been removed because they do not contribute with any extra information. Finally, the database has 21 numerical attributes and 13 categorical. The number of categories for each categorical attribute differs between 2 and 8, but only one attribute has 8 different categories. The others have between 2 and 5 being then the committed error not very large using polar coordinates for codification. The output can adopt three different

categories, adding this a bit of difficulty to the clustering process. Table (8) shows results for the different codifications.

Table 8 *Comparison of some codifications for the Type of Cylinder data set*

Codific.	Database size	Results	Time
Ordinal	277 × 34	62.82%	3.6s
Binary	277 × 42	60.3%	3.7s
Gray	277 × 63	62.45%	5.6s
Polar	277 × 40	63.18%	3.4s
Spherical	277 × 47	64.26%	4.0s

None of the codification approaches offers a very good result. Nevertheless, all them have a very similar behaviour, in percentage of matching instances and in computation time. Again, as in the previous experiments done, the Polar codification offers a slightly better result than the others and the computation time is not large due to the medium size of the resulting database. In this case the spherical codification does offer a slightly better result than the polar approach. Although it is no very significant the increment of performance, it is noticeable. These results confirm that the the proposed approach works slightly better than the others for this case.

4.3.5 Mushrooms Database.

This database, as the Vote one, is fully categorical. But here, not all attributes are binary. It has only missing values in one of the attributes which has been fully removed. Finally, the database has 20 categorical attributes and the output is divided into two groups: edible and poisonous mushrooms. The number of objects is much larger than in the previous cases, being 8416 the total of objects to be classified. The number of different categories for each of the attributes varies between 2 and 12, having many of them 9 different options. Table (9) shows results of the experiments.

Table 9 *Comparison of some codifications for the Mushrooms data set*

Codific.	Database size	Results	Time
Ordinal	8416 × 20	73%	100s
Binary	8416 × 51	88.91%	160s
Gray	8416 × 111	89.5%	265s
Polar	8416 × 35	88.78%	180s
Spherical	8416 × 49	88.81%	140s

Here the Ordinal codification differs quite a bit from the others, which have all similar results, being the gray code the one which offers best results. Nevertheless, result of polar and spherical codifictions are comparables as they are less than 1% different.

In this case, more differences can be observed in the computational time, which is much larger than in the previous experiments. Here, polar and spherical codes are quite fast and offer quite good results.

Again, the application of the spherical codification offers good results but does not improves much the performance of the polar code. As said before, this is probably due to the low importance of those attributes.

4.3.6 Salary Database.

This is mixed database much larger than the previous. After removing the objects with missing values the total of instances to be clustered is 30161. It has 14 attributes, 6 of which are numerical and 8 are categorical. The number of categories of the several categorical characteristics vary between 2 categories and 41, although there is just one attribute with so many possibilities. The normal numbers of categories are 6 and 14. Obviously, the attribute with 41 categories will have a large error with all codifications but the gray code and large influence for this code for the number of converted attributes. There are two clusters, people who has a salary bigger than 50K dollars per year and people whose salary does not exceed this quantity. Table (10) shows all results of these experiments.

Table 10 *Comparison of some codifications for the Salary limits data set*

Codific.	**Base size**	**Results (%)**	**Time (s)**
Ordinal	30161 × 14	50.05%	284s
Binary	30161 × 33	71.58%	450s
Gray	30161 × 105	71.2%	495s
Polar	30161 × 21	50.05%	290s
Spherical	30161 × 28	75.36%	440s

After a quick look it is seen that none of the databases offers an extraordinary good performance, although some of them reach a relatively good result. The ordinal and the polar approaches offer the worst results of all, differing quite a lot from the others. The binary and the gray codes offer a much better result and very similar to one another. Also, the spherical coordinates offer a better result than any of the other codifications. This is because with this code the error of the polar codification is reduced, and it seems that in this case the attributes with a large number of categories do have a big importance in the clustering process. This is the reason why polar code does not work very well here and spherical coordinates do improve the results.

The computation time is large, due to the size of the database. Differences can be observed in the magnitude of this time, being the fastest approaches also the ones with worst results. The spherical coordinates approach is the codification that offers best results with the smallest time of the group of the codifications with good results.

4.3.7 Analysis of the results.

In all experiments performed it has been showed the results of many codifications approaches. It has been seen how the existing codifications so far work well in several cases, but none of them work well for all the experiments. Nevertheless, results show how the proposed polar and spherical codifications, mainly this second one, work well or comparable to the best performance in all cases, showing certain robustness.

The approach based on polar coordinates works well in all cases but the last one (Salary). It is has been seen how other codifications are comparable to the ones proposed here in each experiment, but it is not always the same codification that offers good results. There is only one case, Salary database, where the polar codification does not work well. This is probably because there are some attributes with large number of categories and the codification error of this code affects to the performance in this case.

On the other hand, the spherical coordinates have a good performance in all experiments. This results point out the fact that this codification offers the best trade off between the error of codification and the influence of the converted attributes in the clustering process. In some cases it has been observed that spherical coordinates do not improve the performance of polar code. This might be due to the small importance of the converted attributes for the description of the clusters. Nevertheless, it is important to remark that results in those cases are always comparable to the ones using polar coordinates.

Then, attending to the results, it can be said that the geometrical approach, 2D and 3D coordinates, offers a very good approach to the codification problem for clustering purposes.

Besides, because the converted matrices do not increase their sizes much, the computing time is one of the smallest ones in all experiments, This is another advantage of the proposed approach, specially for large data sets.

4.4 Comparison with Other algorithms.

In last subsection, the proposed approach was compared with other codifications. However, it is also very important, to compare it to other clustering algorithms for mixed data. In order to perform this comparison, the algorithm described in [1] has been chosen as reference. The reason is that such algorithm is capable of working with mixed data sets and, moreover, provides better results than other mixed data algorithms.

The examples where the comparison will be performed will be some of the databases described before: *Vote*, *Heart Disease* and *Credit Approval*, given that they were also included in the experiments done in [1].

4.4.1 Introduction of weights to k-means.

As has mentioned before, the normalization of attributes tries to avoid some attributes being higher considered than others just by the value of its magnitude. However, in a database it is common that not all attributes have same relevance describing the output.

For this reason, some weights have been introduced to give higher or lower importance to the attributes, according to their contribution of information about the clusters. The purpose of these weighting factors is mainly to get more information included in the data set.

In the expression of the k-means algorithm in equation (6) a weight ω_v is introduced for each attribute v ($v = 1, 2, .., z$), resulting in equation (7).

$$J_{weighed} = \sum_{i=1}^{c} \left(\sum_{k=1}^{n} \sqrt{\sum_{v=1}^{z} \omega_v \cdot (x_{k,i,v} - c_{i,v})^2} \right) \tag{7}$$

This information can be extracted in different ways. One of them is that the embedded information is given by an expert on the subject, who will indicate which of the characteristics of the objects is more relevant for their classification. If an expert is not available, some other options are still possible. For example, if the data set includes information about the classification of the outputs, then this information can be used to see, in the non supervised case, which weights give a better clustering of the data set, compared with the output that is known. This method is the one used in the examples described below, where several weights have been included and finally the combination that offers the best classification has been chosen.

In the results show above, it has been seen how polar codification works well for the three data sets used for this comparison. Therefore, this codification will be the one used together with the weighted k-means algorithm to perform this comparison, as there is no need to use the spherical one in these cases.

4.4.2 Vote Database.

After codifying the database using the polar codification, the search of weights was performed. Table (11) shows the combination of weights that offers the best results, while table (12) shows a comparative of the results applying different algorithms.

Table 11 *Weights for the Vote data set*

Attribute	**A1**	**...**	**A4**	**...**	**A16**
Value	1	...	7	...	1

Table 12 *Comparison with other algorithms for the Vote data set*

Approach	**Results** (%)
K-modes	84%
Hierarchical clustering algorithm	86%
Ahmand et. al. algorithm	86.7%
Polar Cod.	**89.7**%
Polar Cod. + Weights	**97**%

It can be seen how in this case, just using the codification proposed and the normal k-means the results are quite good, being better than those offered by some algorithms. What is more, if weighing factors are introduced, performance can increase and reach very good levels.

4.4.3 Heat Disease Database.

Tables (13) and (14) show respectively the weighing factors and comparison with other algorithms.

Table 13 *Weights for the Heart Disease data set*

Attribute	**A1**	**...**	**A15**	**...**	**A17**
Value	1	...	3	...	1

Table 14 *Comparison with other algorithms for the Heart-Disease data set*

Approach	**Results** (%)
K-prototypes	66%
ECOWEB	74%
SBAC	75%
Polar Cod.	**82.4**%
Ahmand et. al. algorithm	84.8%
Polar Cod. + Weights	**85.15**%

Again, only the use of the polar codification with the basic k-means algorithms has a very good performance, better than some algorithms. When the weighing factors are introduced this performance increases a bit. In both cases a simple approach as this is comparable to the performance of some complex and good mixed data clustering algorithms.

4.4.4 Credit Approval Database.

As in previous cases, tables (15) and (16) shows the best combination of weights and the comparison with other algorithms.

Table 15 *Weights for the Credit Card data set*

Attribute	**A1**	**...**	**A6**	**...**	**A16**	**...**	**A20**
Value	1	...	1.8	...	0.4	...	1

Table 16 *Comparison with other algorithms for the Credit Approval data set*

Approach	**Results** (%)
Modha and Spangler's algorithm	83%
Polar Cod.	**86.4**%
Polar Cod. + Weights	**87.4**%
Ahmand et. al. algorithm	88.3%

As in the previous cases the results are very good using both, only the codification and also when the weights are included, and, although the performance in this experiment is not better than Ahmand's algorithm, the results a very close.

4.4.5 Analysis of the results.

These three experiments and comparatives studies performed showed how the proposed codification approach has a performance comparable to the performance of some complex and good mixed data algorithms. This shows that the alternative of using a codification for clustering mixed databases could be an option instead of developing a new complex algorithm.

Indeed, with a basic algorithm like *k-means* and the introduction of weights, the performance reaches very good levels, better than many mixed algorithms and comparable to those that offer a very good performance.

5 Conclusions.

It has been presented a new approach for codifying categorical attributes in order to be able to use numerical clustering algorithms. This approach is quite easy to understand and to apply and offers a good compromise between codification error and influence of the converted attributes in the database.

Results shown in section (4), are quite significant. Performance of the new approach offers always the best of the results or comparable to the best one. Another advantage is that the computational cost is not very large, making it easier to work with large data sets. The main drawback is that when the number of categories is big, the codification error can be large, both for the 2D and 3D cases. However, in subsections (3.4) and (3.5) has been seen how any of the codifications is suitable for attributes with large number of categories.

A comparative study with other algorithms shows how the codification, even used with a simple algorithm, offers results comparable to those obtained when using complex mixed algorithms. It has also been showed how a small

improvement in the numerical algorithm applied can also lead to an improvement in the results offered by the codification on its own.

So, the proposed codification shows a better performance than other codifications, being this performance comparable to the performance of other algorithms. It has also to be considered that the computation time is not very large and that the codification is easy to apply and understand.

Acknowledgments

The authors acknowledge the partial funding of this work by the National projects DPI2007-66728-C02-01 and DPI2008-06737-C02-01.

References

1. A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.
2. R. Babuska. *Fuzzy Modeling and Identification*. PhD dissertation, Delft University of Technology, Delf, The Netherlands, 1996.
3. F. Barcelo-Rico, and J.L. Diez Comparative Study of Codification Techniques for Clustering Heart Disease Database. *Modeling and Control in Biomedical Systems*, 7(1) 2009.
4. P. Bourke. http://local.wasp.uwa.edu.au/, 1993.
5. R.K. Brouwer. A method for fuzzy clustering with ordinal attributes. *International Journal of Intelligent Systems*, 22:590–620, 2007.
6. H. S. M. Coxeter. *Regular Polytopes*. Methuen, 1948.
7. J. Crossa and J. Franco. Statistical methods for classifying genotypes. *Euphytica*, 137(1):19–37, 2004.
8. J. V. de Oliveira and W. Pedrycz. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Inc. New York, NY, USA, 2007.
9. J. L. Diez, J. L. Navarro, and A. Sala. Algoritmos de agrupamiento en la identificacion de modelos borrosos. *Revista Iberoamericana de Automtica e Informtica Industrial*, 1(2):32–41, 2004. In Spanish.
10. J. L. Diez, A. Sala, and J. L. Navarro. Target-shaped possibilistic clustering applied to local-model identification. *Engineering Applications of Artificial Intelligence*, 19:201–208, 2006.
11. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Ed. John Wiley & Sons, New York, USA, 2000.
12. R. Gelbard, O. Goldman, and I. Spiegler. Investigating diversity of clustering methods: An empirical comparison. *Data & Knowledge Engineering*, 63(1):155–166, 2007.
13. M. Goebel and L. Gruenwald. A survey of data mining and knowledge discovery software tools. *SIGKDD Explor. Newsl.*, 1(1):20–33, 1999.
14. J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *JR Stat. Soc., Ser. C*, 28:100–108, 1979.
15. Z. He, X. Xu, and S. Deng. Scalable Algorithms for Clustering Large Datasets with Mixed Type Attributes. *International Journal of Intelligent Systems*, 20:1077–1089, 2005.
16. C. C. Hsu, C. L. Chen, and Y. W. Su. Hierarchical clustering of mixed data based on distance hierarchy. *Information Sciences*, 177(20):4474–4492, 2007.
17. Z. Huang and M. K. Ng. A fuzzy k-modes algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, 7(4):446–452, August 1999.
18. H Timm and R. Kruse. Fuzzy Cluster Analysis with Missing Values. *Fuzzy Information Processing Society - NAFIPS, 1998 Conference of the North American*, 1, 1998.
19. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for large databases. *Proc. SIGmod*, 96:103–114, 1996.