Rough Classification in Incomplete Databases by Correlation Clustering

László ASZALÓS Tamás MIHÁLYDEÁK

Faculty of Informatics, University of Debrecen, Hungary

Abstract

In the context of data mining, missing data can be handled in several ways. The most common is the artificial construction of missing data, but we can predict it, or transform the whole database in a fuzzy way. In this article we propose a different approach: we extend our rough classification to incomplete databases. This uses the correlation clustering as a tool, which uses a tolerance relation of the similarity between objects of a database. This relation can be generated from the distance between objects and can be sensitized based on missing data. We demonstrate our method in the wine database.

Keywords: Incomplete database, classification, correlation clustering, harmony search, rough clustering

1. Introduction

In the information society the demand continuously grows from the industry to process and use data sets generated and recorded in every part of our life. Therefore, more and more researchers work on the field of Big Data, and offer different tools from various scientific sources. Most of these tools come from statistics, however the Data Mining is its favourite common name nowadays. From this major research area we are going to deal with clustering and classification in this paper.

Clustering (cluster analysis) is commonly used for unsupervised learning. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups. During more than 80-year history of the cluster analysis many clustering methods have been developed and used successfully in different areas of the sciences and real life. For example, clustering can help us to discover the irregular items i.e. the traces of an intrusion on the basis of computer systems, or a poisoned eagle by using signals of GPS tracking devices.

Classification is a supervised learning. Here, given a training set, whose elements are classified into different categories form the base of categorisation for other elements. There are many classification methods, and their application penetrates our lives, for example the labelling of e-mails (junk/not junk). As we can see, in both clustering and classification, similarity and the dissimilarity play very important roles.

In this paper we explore how clustering objects could be achieved based on partial similarity, and how can we use this for classification. To show that this actually works in practice, we use the *wine* database form the UCI repository [1].

The paper organized as follows. The scientific background—correlation clustering, concept of distance, tolerance relation—described in Section 2. A view of handling missing data is given in Section 3. Parts of the classification presented in Section 4. Simulation results are shown in Section 5. We discuss the limits of our methods, and suggest other ones in Section 6. Finally the conclusions are drawn in section 7.

2. Scientific background

The algorithm we have developed and implemented has used a number of uncommon methods. In this section these are shown schematically and are referred to the original sources.

2.1. Correlation clustering

Objects are described by their data, called a *feature vector*. In the following to simplify the text we shall identify objects with their feature vectors.

Most of the clustering methods use the fact that a distance can be defined between objects. As feature vectors can contain real, integer and categorical (yes/no, female/male, A/B/AB/O, values on a Linkert scale) data, the latter ones are usually converted to real values, and scaled somehow, so we get n-dimensional vectors containing real values. Then, based on the distance of vectors (or based on the density of vectors relying on this distance) the clusters of objects are created. To find out whether the results are correct or not, some statistical analysis or visual survey is needed. Of course, in case of high dimensional feature vector, visual representations are difficult.

Correlation clustering [2] has chosen a completely different approach. Let T be a tolerance relation (reflexive, symmetric, not necessarily transitive). The task is to find a *closest* equivalence relation R. Mathematically, for each partition (equivalence relation) we can assign a cost value, which provides the number of pair of objects (x, y), for which either

$(x, y) \in R$ or $(x, y) \in T$, but not both is satisfied. If we treat a partition as a function $P : V \to \mathbb{N}$ (where V is the set of feature vectors), which assigns its cluster's identifier to each object, and a t is a *quasi-characteristic* function of T, i.e.

$$t(x, y) = \begin{cases} 1 & \text{if } (x, y) \in T \\ -1 & \text{if } (x, y) \notin T \end{cases}$$

then we can define this cost value of a partition according to T as

$$c_T(P) = \sum_{t(x,y)=-1} \delta_{P(x),P(y)} + \sum_{t(x,y)=1} \left(1 - \delta_{P(x),P(y)}\right),$$

where $\delta_{i,j}$ is the Kronecker's delta. The formula of the cost value remain the same if we use partial tolerance relation, where its quasi-characteristic function is the following:

$$t(x, y) = \begin{cases} 1 & \text{if } (x, y) \in T \\ -1 & \text{if } (x, y) \notin T \\ 0 & \text{if } T \text{ not defined for } (x, y) \end{cases}$$

Unfortunately—as in [2] it has been demonstrated—solving a correlation clustering problem is NP-hard, and for example in case of more than 15 objects is worth to use some optimization methods to find a near optimal solution.

The authors constructed a framework for optimization methods, and implemented many methods in it (see in [3]). The experiments on these implementations lead them to choose the Harmony search and the Contraction.

2.2. Different concepts of distance

Although we want to use the similarity, in real life the tolerance relation is rarely given, i.e. it needs to be generated. The easiest way to generate it, is from the distance. At first we show the different kinds of distances we have used.

Let us start with the case when *all* elements of the feature vector are real numbers, so objects can be treated as points in an n-dimensional Euclidean space. The Minkowski distance of $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is the value of

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

Some values for parameter p are very common in practice: 1, 2 and ∞. The distances they generate are called Manhattan distance, Euclidean distance and Chebyshev distance, respectively. In the latter, the limit value can be expressed more easily: $d_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{n} |x_i - y_i|$. To simplify the calculations, we redefine d_2 as follows:

$$d_2(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^{n} (x_i - y_i)^2.$$

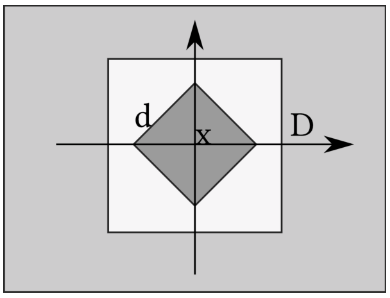

Figure 1: Distances and similarity for $p = 1$ and $q = \infty$.

This simplification is not a big mistake here, because as it turns out from the following section, we are not interested in the exact value of the distance of points, just whether it is smaller or bigger than a pre-given value. If we replace both sides with it's square root, we get back the original distance. But with this simplification we can discuss the different distances in a more uniform way.

2.3. Generated tolerance relation

Correlation clustering requires a tolerance relation, so we have to generate similarities and differences from distances. We believe that the situation is relatively straightforward: if two objects are close, they are similar, while if they are distant, they are different. Therefore we need two threshold parameters: d and D. If the distance of $\mathbf{x}$ and $\mathbf{y}$ is less than d, we treat objects $\mathbf{x}$ and $\mathbf{y}$ similar (central dark grey region on Fig. 1), but if their distance is greater than D, then we treat them different (outer dark grey region). Of course before using these parameters, it is worth to normalize the database, e.g. to fit the data to the standard normal distribution, such that the order of the magnitude of the members of a sum/maximum are the the same. As the discussion will show, it is worth to permit the use of different kinds of distances as Fig. 1 shows. Of course, in case of fixed p and q we have constraints for parameter pair (d, D): it is not allowed that both $d_p(\mathbf{x}, \mathbf{y}) < d$ and $D < d_q(\mathbf{x}, \mathbf{y})$ are satisfied for some $\mathbf{x}$ and $\mathbf{y}$, i.e. $\mathbf{x}$ and $\mathbf{y}$ cannot be similar and dissimilar at the same time. (All the other combinations are allowed: could be just similar, just be dissimilar, or nor similar and dissimilar.)

This allows us to generate the tolerance relation of objects based on their distance:

$$t_{dD}^{pq}(x, y) = \begin{cases} 1 & \text{if } d_p(\mathbf{x}, \mathbf{y}) < d \\ -1 & \text{if } D < d_q(\mathbf{x}, \mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

We left to the reader to prove that the relation of this quasi-characteristic function is really tolerance relation. (Its reflexivity is trivial, the symmetry follows from the definition of the distance.)

3. Distance of incomplete objects

3.1. Coping with missing data

In practice, several methods are known to complement the missing data [4]. For clustering, the average of values in the same dimension could replace the missing data in a continuous case, and the most common value in a categorical case. We do not need to take into consideration all the objects, we can restrict our investigation to a small neighbourhood of the actual object, or to its category at classification. We accept that this works well for practical tasks, but we considered this complementation as fraud.

In our opinion if the data is missing, it needs to be stored in its original form somewhere, and the user has to cope with it.

In the previous section we wrote that two objects are similar if they are close to each other and they are dissimilar if they are far away from each other. In the case of distances d_1 and d_2 if the value of $|x_j - y_j|$ is not known for some j because x_j or y_j is missing, then

- it is trivial, that if $\sum_{i \neq j} |x_i - y_i|^p > D$, then $\sum_i |x_i - y_i|^p = d_p(\mathbf{x}, \mathbf{y}) > D$, i.e. $\mathbf{x}$ and $\mathbf{y}$ are surely dissimilar
- if $\sum_{i \neq j} |x_i - y_i|^p < d$, then any of $d_p(\mathbf{x}, \mathbf{y}) < d$, $d_p(\mathbf{x}, \mathbf{y}) = d$ and $d_p(\mathbf{x}, \mathbf{y}) > d$ can be fulfilled.

We can handle the missing data in three ways:

Optimistic. We do not care about missing values: if you do not see/know something, then it does not even exists for you. It seems a crazy idea at Euclidean space with the standard distance, but this is not the situation in all cases. If we don't know that our flight is cancelled, we go to the airport.

The fact, that we ignore data, has some consequences, e.g. even a dissimilarity can be judged as similarity, therefore we have less and bigger clusters at clustering. We note, that later we will handle the distance d_∞ in an optimistic way: we have no better estimation on the maximal member of a set, that the maximal member among the known ones.

Pessimistic. We assume that the missing values are the most important ones. In the case of Euclidean space, we are sure, that the missing distance is big enough to sum up to D, so similarity cannot occur. This means that we can judge similarity as dissimilarity, hence we have more and smaller clusters at clustering.

Realistic. We can try to extrapolate the exact distance from the known values. In our example we replace the missing distance with an average distance, so calculate instead of the realizable difference ($\sum_{i \neq j} |x_i - y_i|^p$) with an imaginary one ($\frac{n}{n-1} \sum_{i \neq j} |x_i - y_i|^p$ where n is the size of the feature vector). If $\frac{n}{n-1} \sum_{i \neq j} |x_i - y_i|^p < d$, we can treat $\mathbf{x}$ and $\mathbf{y}$ as similar, but if $D < \frac{n}{n-1} \sum_{i \neq j} |x_i - y_i|^p$, then we can treat them as dissimilar. It is obvious, that by using this extrapolation, and replaced any kind of tag with an average one, a dissimilarity can be treated as similarity, and even a similarity can be treated as dissimilarity.

The pessimistic attitude is not acceptable at clustering: as there are no similar objects, the result consist of singletons or sets of unrelated objects.

3.2. Characteristic function of the similarity

In the previous subsection we suggested a *realistic* method to hide a shortage of one data item. In our implementation we generalized this idea: if k differences from n were known in the sum of the Minkowski distance at $p = 1$ or 2, then we multiply it with n/k to get the interpolated sum, which is used at comparisons.

It is well known that the extrapolation could bring significant errors into the calculations. For this reason, the results obtained by extrapolation should be used with criticism. In the previous section we defined a quasi-characteristic function based on distance: $t_{dD}^{pq}(\mathbf{x}, \mathbf{y}) \to \{-1, 0, 1\}$. As parameters p, q, d and D are fixed we omit for the sake of simplicity. In our case intermediate values may appear due to missing data, so we use a more general *characteristic* function $t'(\mathbf{x}, \mathbf{y}) \to [-1, 1]$.

Let us denote S the sum of k known tags from Minkowski distance. In the case of d_∞ S denotes the maximum of known differences. We define this new characteristic function as follows:

- If $k = n$, then $t'(\mathbf{x}, \mathbf{y}) = t(\mathbf{x}, \mathbf{y})$.
- If $D < S$, then objects $t'(\mathbf{x}, \mathbf{y}) = -1$, because the objects are certainly dissimilar.
- If $S < D < \frac{n}{k} \times S$, then $t'(\mathbf{x}, \mathbf{y}) = -k/n$ in the case of distances d_1 and d_2.
- If $S < d$ then $t'(\mathbf{x}, \mathbf{y}) = k/n$ in the case of distance d_∞.
- If $S \times \frac{n}{k} < d$ then $t'(\mathbf{x}, \mathbf{y}) = k/n$ in the case of distances d_1 and d_2.
- Otherwise we cannot say anything, so $t'(\mathbf{x}, \mathbf{y}) = 0$

The function t' is used at contraction steps in Figure 6 to solve a correlation clustering problem as to construct a partition.

One may ask what is the purpose of using fractions instead of the three values 1, 0 and -1. We remark, that using fractions is not a new idea [5, 6], this kind of generalisation of the original idea of correlation clustering is very natural. The structure of the cost function and its calculation are similar to the original case:

$$c'(P) = \sum_{t(x,y)>0} t'(x,y) \times \left(1 - \delta_{P(x),P(y)}\right) - \sum_{t'(x,y)<0} t'(x,y) \times \delta_{P(x),P(y)}$$

4. Clustering and classification

4.1. Contraction

We can treat the similarity and dissimilarity of object as forces, where the similar objects attract each other, while dissimilar objects are repulse each other. We can extend this kind of forces to clusters, too. Let A and B be two clusters, and let

$$F(A,B) = \sum_{\mathbf{x}\in A, \mathbf{y}\in B} t(\mathbf{x},\mathbf{y}).$$

If $F(A,B) > 0$, then two clusters attract each other; if we construct the partition P' in this case by merging A and B in partition P, then $c(P') < c(P)$, i.e. we get a better partition.

Our contraction method—which is a greedy algorithm—merges clusters, where the benefit of merging is the greatest [3]. It easy to check that $F(A,B) + F(C,B) = F(A \cup C, B)$, so we need to sum forces at a contraction step.

Similarly we can define a force F' based on function t', which is an extension of the original force concept. In the following we present this generalized contraction method in action. The rectangles on the following figures denote the clusters, and the weighted edges the (non-neutral) forces between them. At beginning we start from singleton clusters, hence $F'(\{\mathbf{x}\},\{\mathbf{y}\}) = t'(\mathbf{x},\mathbf{y})$, for objects $\mathbf{x}$ and $\mathbf{y}$ as Figure 2 shows.

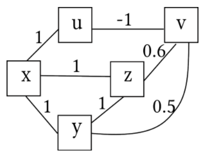

Figure 2: Intitial partition

Initially—as the original distances are from $[-1,1]$—the pair of singleton clusters merge together are those for which value 1 was assigned, i.e. the similar objects are fully known. Let us take the objects on Figure 2. Here $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are completely known and similar pairs of objects, so any two of them could be joined. Let us assume that $\mathbf{x}$ and $\mathbf{y}$ merged as Figure 3 shows.

The attraction between clusters $\{\mathbf{x},\mathbf{y}\}$ and $\{\mathbf{z}\}$, became 2, as we needed to sum $f'(\mathbf{x},\mathbf{z})$ and $f'(\mathbf{y},\mathbf{z})$. Since this value is the biggest in the figure, these clusters merged as Figure 4 shows.

Usually if there are objects (almost) fully known, similar to each other, then at beginning they construct small factions. Next if the algorithm depletes

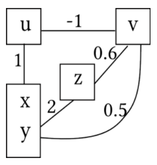

Figure 3: Result of the first contraction

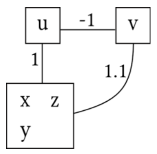

Figure 4: Result of the second contraction

such strong attractions, then it continues with the other objects with smaller attractions. In Figure 4 there is the object $\mathbf{u}$ which is similar only to object $\mathbf{x}$, and the object $\mathbf{v}$ which is partially similar—has missing data—to objects $\mathbf{y}$ and $\mathbf{z}$. But the cluster $\{\mathbf{x},\mathbf{y},\mathbf{z}\}$ attracts object $\mathbf{v}$ more strongly than object $\mathbf{u}$, hence by the next merging we get clusters on Figure 5. The repulsion between objects $\mathbf{v}$ and $\mathbf{u}$ neutralizes the attraction between objects $\mathbf{x}$ and $\mathbf{u}$, hence there is no reason to merge the remaining two clusters.

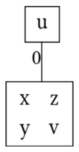

Figure 5: Result of the last contraction

4.2. Harmony search

In the previous subsection we have shown how we can get a near optimal correlation clustering by contraction method. The cost value of partition helps us to select the closest partition to a given tolerance relation. But our aim to classify objects. Therefore we check the resulted clusters, how they are related to the classes of the classification. (We note, that the contraction method knows nothing about classification of objects, it knows only the function t'.) For this we determine the dominant class type of each cluster, and count the objects of minority types. The sum of these numbers give a different type cost value as a function of threshold parameters. This cost value measures deviation of the partition from the homogeneity. Our task is to optimize this function to get the best threshold parameters, and for this optimization we use harmony search.

Like many other optimization methods, harmony search is also derived from modelling a process from a real-life [7]. This method has many advantages including that the continuity of the cost function it is not necessary, and the parameters of the cost function may be discrete or continuous.

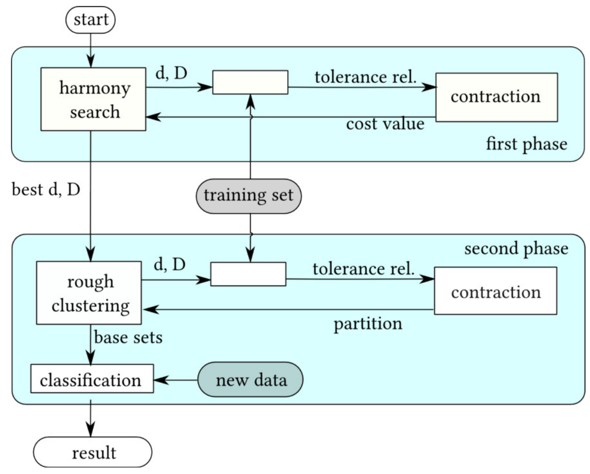

Figure 6: The process of the classification

The whole method is based on a harmony memory, which initially is randomly filled. Then, it executes the given cycle a prescribed number of times. In the core of the cycle the value of each parameter in feature vector is a generated new one or selected from the memory randomly. In the latter case, the chosen value is changed to a small, random amount. Then the parameters are tested: calculating the value of the cost function. If they are better than the worst of the stored ones, they are replaced. At the end, it selects the parameters belonging to the best cost value.

4.3. Classification's workflow

The workflow of our classification process is shown in Figure 6. The first phase is to determine the values of the two threshold parameters, as we described in the previous subsection. For this the harmony search tries a number of pair of values, calculates the deviation from the homogeneity for them, and select the best ones. These best threshold parameters are the outcome of the first phase.

The second phase uses these values, generates the function t', and runs the contraction method on this function several times. As the method is randomized, we can get several different results (partitions) with different cost values. The process selects the best partitions (with minimal $c'(P)$ values). Next it generates the intersections of this partitions. In one intersection the objects can be treated as inseparable elements, hence the whole structure of intersections can be treated as the base sets in the sense of rough set theory [8]. If two objects are always in a common cluster, then they really should be similar to each other. Typically, their classes are the same. Or if not, then we may question the correctness of the training data. To correct such arguable cases we classify the intersections (the base sets in the sense of rough set theory [8]). This can be done in several ways, in our experiments we have chosen the dominance principle, this is same as the English electoral system: the winner takes all. In the case of a tie, we assigned to the set the *other* label, which could be reasonable in medical applications. But in some special cases at tie the random choice of the class of the set or using fuzzy values is acceptable in our opinion.

We are almost ready, only the classification step is left. Since the classification usually includes a small training set and many objects to classify, it is helpful if the classification of one object has no high complexity. For this let us extend the tolerance relation to the new object, then summarize the attractions by base sets. At this summing the attractive and repulsive forces could neutralize each other (like at Figure 5). Finally the class of the most attractive base set will be assigned to the element.

5. Discussion

In the database we have chosen to test our method three classes are separable [9], and some methods perfectly separate these classes. We do not intend to compete with these results, we run our classification method on the original data to get a basis for comparison.

The following figures show the result of running our method using different kinds of distances. To speed up the counting process, we have made some tests to restrict the random values from $\mathbb{R}^+$ to a finite interval, and used this limit at harmony search. This pre-calculation greatly helped, the result of the harmony search gave clusters that were more or less homogeneous. In the 45-item random sample usually 5–6 items are classified incorrectly. The worst level of the error was 9 and the best was 2.

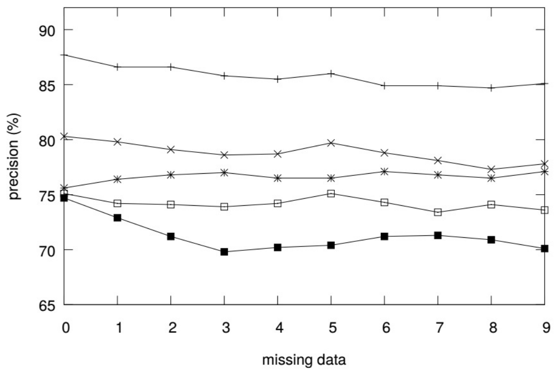

Figure 7: Manhattan-Manhattan distance[1]

Next our method generated the base sets with the best threshold parameters, and classified using the dominant class of its elements. The original database has no missing data. However, we are interested in the effectiveness of our method on incomplete data. Therefore we modified our algorithm to delete one piece of data step-by-step, and repeated

[1]Notation: $+ - -0\%$, $\times - -10\%$, $* - -20\%$, $\square - -30\%$, $\blacksquare - -40\%$

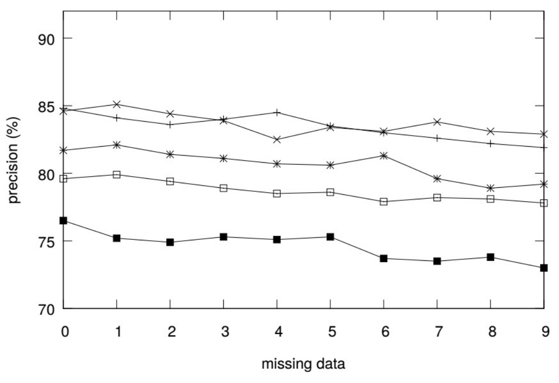

Figure 8: Manhattan-Euclidean distance

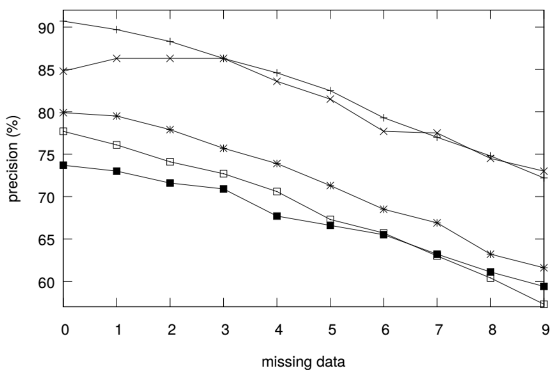

Figure 9: Manhattan-Chebyshev distance

the classification of the new object. At each step, we examined whether classifying an object coincides with its real class, or not. On the scale y the precision is shown. We continued this process until 4 pieces of data remained from the 13, i.e. finally we deleted 70 percent of the data of the object. On the figures the scale x denotes the number of pieces of data deleted.

Then 10 percent of the data has been deleted from the training set, and the whole method was repeated: the remaining training set was clustered and other objects were classified. Then 10 percent of the initial data was again omitted from the training set, and the method repeated. This was continued until 40 percent of the data of the training set were missing. In the worst case 40 percent of the data of the training set, and 70 percent of the data of the object to classify were missing, but the result of our experiment was at least 35 percent in each case. The second worst result in this extreme case was 59 percent, while the best one was 73 percent.

The results presented on figures are collected in Table 1. The numbers denotes percentages here, they show intervals of successful rates of classification in the cases from the minimal to no missing information.

We deleted data randomly from the training set and from the other objects. This helps a lot. Figure 16 shows the difference. On the left from the

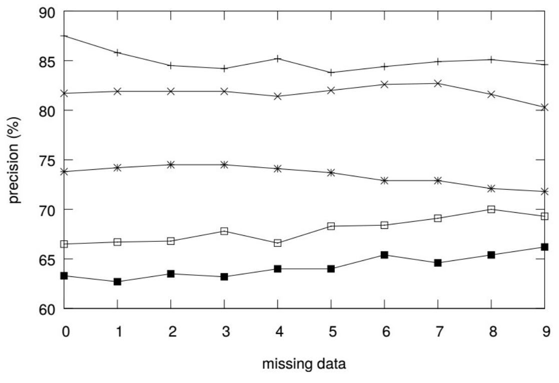

Figure 10: Euclidean-Manhattan distance

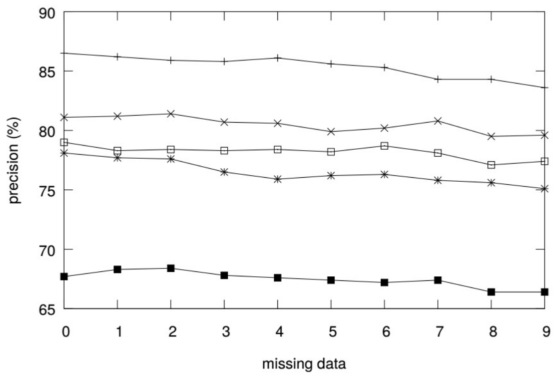

Figure 11: Euclidean-Euclidean distance

same rows and from the same columns are missing the pixels, while on the right this is totally random. We can recognize the standard picture of image processing from the latter one.

In the case if we have two incomparable objects, but they are both similar to other objects, there is a high probability that they are similar. The correlation clustering put them into the same cluster. Similarly if some incomparable objects are similar to two dissimilar objects, there is a little chance, that they are similar, then the correlation clustering probably put each of them into the similar object's cluster, hence separate them. Moreover the correlation clustering have an error correcting property, it can put two "similar" objects into different clusters, if their relation to other object query this relation. By our idea this incomparable nature makes the correlation clustering a conductive tool at coping missing data.

Table 1: Precision of the classification

	$q=1$	$q=2$	$q=\infty$
$p=1$	70.1–87.7	73.0–84.8	59.4–90.7
$p=2$	66.2–87.5	66.4–86.5	61.7–87.6
$p=\infty$	72.6–88.6	68.8–85.4	35.7–78.3

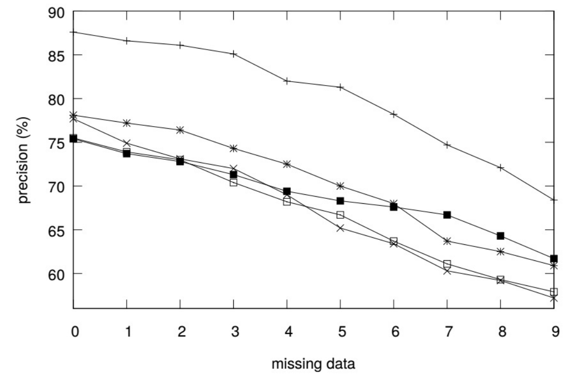

Figure 12: Euclidean-Chebyshev distance

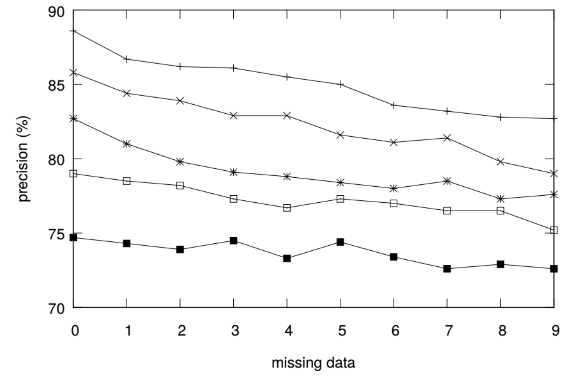

Figure 13: Chebyshev-Manhattan distance

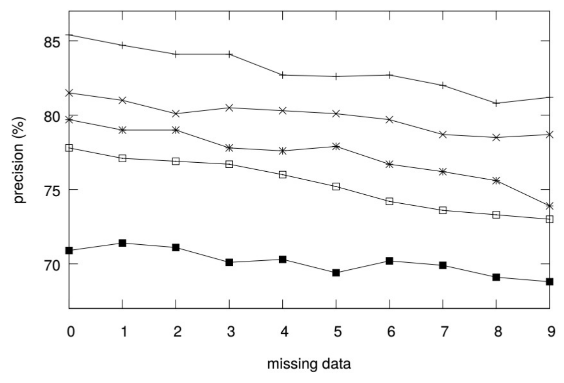

Figure 14: Chebyshev-Euclidean distance

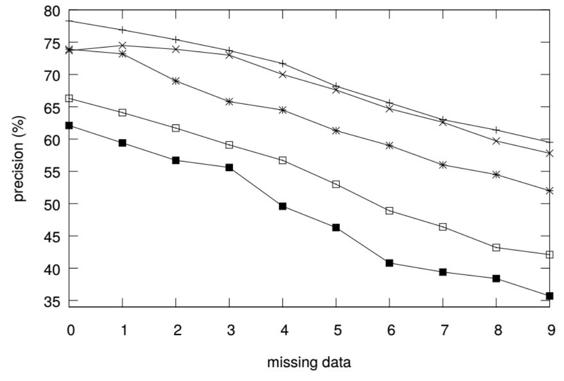

Figure 15: Chebyshev-Chebyshev distance

6. Further work

6.1. Categorical data

As we have seen before, it is possible that for some parameters (d, D) objects $\mathbf{x}$ and $\mathbf{y}$ are similar, and for some other parameters (d', D') the same objects are dissimilar. The situation is quite different if we use categorical data. It is clear that the *not* is not similar to *yes*, the *male* is not similar to the *female*, or the different blood types are not similar. However in case of a nine-level Linkert scale the items next to each other (or even items in the distance of two) will be considered somewhat similar.

We believe that in case of databases with categorical data a human decision cannot be ignored. Determining similarity/dissimilarity is the user's responsibility. A user can be very accurate for determining similarities and dissimilarities for a categorical data. But what happens if they need to compare vectors consisting of about fifty categorical data? Assuming that one can solve problems coordinate-wise, we need to count similarities and dissimilarities in each dimensions. If the number of similarities is bigger than a parameter (let say v_1), the two vectors can be treated as similar, and if the number of dissimilarities is bigger than an other parameter (let say v_2), the vectors are treated as dissimilar. This kind of counting is not a new thing. It is common practice in law: sometimes the jury is directed to reach an unanimous verdict, otherwise some level of majority is needed. (The failure of agreement may lead to retrial, i.e. the partiality appears here, too.) We can extend this case easily: calculate the distance once for all real valued parameters, or separately for each dimension, and sum with the result of categorical data to get the final result.

While a human can decide on similarity of categorical data, this does not works for a continuous one. It can occur, that for parameter d the value of 0.97 is too small, but 1.02 is too big. These parameters cannot be set by experience—even the training set is always different—so some kind of (possibly automatic) feedback is necessary. In [10] we used an optimization method (simulated annealing) to determine the best values for parameters d and D.

It looks useful to apply our approach to database contains categorical data.

6.2. Limit of the methods

In this article we combined some less known methods and some we invented by ourselves. These methods ought to be examined exhaustively, to find the limits of their applicability.

The contraction method is based on a very simple idea, but its effective implementation is a challenge. However during the solution of classification problems, correlation clustering has to be performed

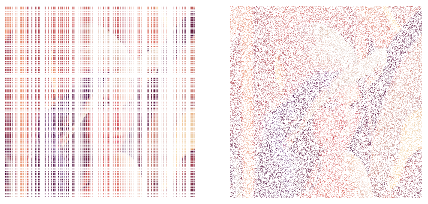
Figure 16: Missing data from a photo

quite often, hence the efficiency and speed of the method is vital. We plan to compare it with other methods on the basis of speed and performance.

Our method—to classify incomplete database—has only been tested on one database; although the results are encouraging. It might be worth to examine what kind of results this method gives in the case when the classes are not separable. On the other hand it would be worthwhile to test the method on a database that is already incomplete, to check that the results of this method are similar to the results of the simulation.

7. Conclusion

The most important lesson of our experiments with the wine database is that the classification of incomplete databases can be performed, and with much better efficiency than our preconception. We do not need to fool ourselves and falsify our databases, but take into account the missing data, and with appropriate caution and carefully selected algorithms we can achieve more reliable results.

References

[1] S. Aeberhard, D. Coomans, and O. de Vel. Comparison of Classifiers in High Dimensional Settings. Technical Report 92-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, 1992.

[2] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[3] László Aszalós and Mária Bakó. Advanced search methods (in Hungarian). http://morse.inf.unideb.hu/~aszalos/diak/fka, 2012.

[4] Kiri Wagstaff. *Clustering with missing values: No imputation required.* Springer, 2004.

[5] Erik D Demaine and Nicole Immorlica. Correlation clustering with partial information. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 1–13. Springer, 2003.

[6] Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 51–60. IEEE, 2011.

[7] Zong Woo Geem, Joong Hoon Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[8] László Aszalós and Tamás Mihálydeák. Rough clustering generated by correlation clustering. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 315–324. Springer Berlin Heidelberg, 2013.

[9] S Aeberhard, D Coomans, and O De Vel. The classification performance of rda. *Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, Tech. Rep*, pages 92–01, 1992.

[10] László Aszalós and Tamás Mihálydeák. Rough classification based on correlation clustering. In *Rough Sets and Knowledge Technology*, pages 399–410. Springer, 2014.