Missing Values Imputation for a Clustering Genetic Algorithm

Eduardo R. Hruschka[1], Estevam R. Hruschka Jr.[2], and Nelson F.F. Ebecken[3]

[1] Catholic University of Santos (UniSantos), R. Carvalho de Mendonça, 144,
11.070-906, Santos, SP, Brazil
[2] Federal University of São Carlos, CP 676, 13.565-905, São Carlos, SP, Brazil
[3] COPPE / Federal University of Rio de Janeiro, Bloco B, Sala 100,
CP 68.506, 21.945-970, Rio de Janeiro, RJ, Brazil
erh@unisantos.br, estevam@dc.ufscar.br, nelson@ntt.ufrj.br

Abstract. The *substitution* of missing values, also called *imputation*, is an important data preparation task for data mining applications. This paper describes a nearest-neighbor method to impute missing values, showing that it can be useful for a clustering genetic algorithm. The proposed nearest-neighbor method is assessed by means of simulations performed in two datasets that are benchmarks for data mining methods: Wisconsin Breast Cancer and Congressional Voting Records. The efficacy of the proposed approach is evaluated both in prediction and clustering scenarios. Empirical results show that the employed imputation method is a suitable data preparation tool.

1 Introduction

Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [1]. Although the terms KDD and Data Mining (DM) are sometimes employed interchangeably, DM is usually considered as a step in the KDD process that centers on the automated discovery of patterns in data. In this context, data preparation is a step in the KDD process that involves the selection, preprocessing, and transformation of data to be mined. When data preparation is performed in a suitable way, higher quality data are produced, and the outcomes of the KDD process can be improved. In spite of its importance, the data preparation step became an effervescent research area only in the last few years. An important problem to be tackled in this step concerns about missing values. The absence of values is common in real-world datasets and it can occur for a number of reasons like, for instance [2]: malfunctioning measurement equipment, changes in experimental design during data collection, collation of several similar but not identical datasets, refusing of some respondents to answer certain questions in surveys. Missing values resulting from such situations may generate bias in the data, affecting the quality of the KDD process.

Many approaches have been proposed to deal with the missing values problem - e.g. see [3,4,5]. A simple solution involves ignoring instances and/or attributes containing missing values, but the waste of data may be considerable and incomplete datasets may lead to biased statistical analyses. Another alternative is to substitute the

L. Wang, K. Chen, and Y.S. Ong (Eds.): ICNC 2005, LNCS 3612, pp. 245 – 254, 2005.

missing values by a constant. However, it assumes that all missing values represent the same value, leading to considerable distortions. The substitution by the mean/mode value is common and sometimes can even lead to reasonable results. However, this approach does not take into account the between-attribute relationships, which are usually explored by data mining methods. Therefore, a more interesting approach involves trying to fill missing values by preserving such relationships.

The task of fulfilling missing values is often referred to as either *missing values substitution* or *missing values imputation*. Imputation methods can be helpful for a variety of data mining tasks, such as classification, extraction of association rules and clustering. In this work, we focus on clustering tasks, in which one seeks to identify a finite set of categories (clusters) to describe the data. More specifically, we describe and evaluate a Nearest-Neighbor Method (NNM) to substitute missing values in datasets to be partitioned by the Clustering Genetic Algorithm (CGA) [6], which can find (according to a numeric criterion) the optimal number of clusters. Similar NNMs for imputation have been proposed in the literature – e.g. see [7,8] for classification problems and [9,10] for clustering tasks. NNMs usually do not generate a model to describe the data and, when used for imputation, they basically search for the best instance(s) of the dataset to be used for substituting missing values. This characteristic may produce a high computational cost. On the other hand, as the *learning process* is specific to each *query*, it may be more accurate. Under this perspective, we believe that a NNM can be a suitable data preparation tool for the CGA.

The remainder of this paper is organized as follows. The next section presents our proposed method to substitute missing values. Section 3 reviews the Clustering Genetic Algorithm (CGA) [6]. The employed NNM is evaluated in two datasets that are benchmarks for data mining methods, and the obtained results are described in Section 4. Finally, Section 5 concludes our work.

2 Nearest-Neighbor Method (NNM)

The Nearest-Neighbor Method (NNM) substitutes missing values by the corresponding attribute value of the most similar complete instance, i.e. it is a K-nearest-neighbor method [11] for K=1. Let us consider that each instance is described by ρ attributes. Thus, each instance can be represented by a vector $\mathbf{y}=[y_1,y_2,...,y_\rho]$. The distance between two vectors (instances) $\mathbf{u}$ and $\mathbf{y}$ will be here called $d(\mathbf{u},\mathbf{y})$. Also, let us suppose that the *i-th* attribute value ($1 \leq i \leq \rho$) of vector $\mathbf{u}$ is missing. The NNM calculates distances $d(\mathbf{u},\mathbf{y})$, for all $\mathbf{y} \neq \mathbf{u}$, $\mathbf{y}$ representing a complete instance, and use these distances to compute the value to be imputed in u_i. The Euclidean metric – expression (1) – is used to compute distances between continuous/ordinal instances, whereas the simple matching approach – expression (2) – is employed to compute distances between instances formed by nominal/binary attributes.

$$d(\mathbf{u},\mathbf{y})_E = \sqrt{(u_1 - y_1)^2 + ... + (u_{i-1} - y_{i-1})^2 + (u_{i+1} - y_{i+1})^2 + ... + (u_\rho - y_\rho)^2} \,. \quad (1)$$

$$d(\mathbf{u},\mathbf{y})_{SM} = \sum_{j=1, j\neq i}^{j=\rho} s_j \;;\; s_j=0 \text{ if } u_j=y_j; \;\; s_j=1 \text{ otherwise}\,. \quad (2)$$

In the above expressions, the i-*th* attribute is not considered, because it is missing in **u**. After computing the distances d(**u**,**y**) for all **y**≠**u**, **y** representing a complete instance, the more similar instance (the neighbor of **u**) is employed to complete u_i. The nearest neighbor of **u** is here called **s**. This way, d(**u**,**s**)=min d(**u**,**y**) for all **y**≠**u**, and u_i is substituted by s_i. For a set of instances whose distances d(**u**,**y**) are equal, the substituted value comes from the first instance of this set. Although expressions (1) and (2) just consider one missing value (in the *i-th* attribute), they can be easily generalized for instances with more missing values.

The imputation by the K-Nearest Neighbor (KNN) method is simple, but it has provided encouraging results [7,8,9,10]. In clustering problems, this approach is particularly interesting, because the imputation is based on distances between instances, as well as the clustering process is. In other words, the inductive biases of clustering and imputation methods are equal.

3 Review of the Clustering Genetic Algorithm (CGA)

Clustering is a task in which one seeks to identify a finite set of categories (clusters) to describe a given dataset, both maximizing homogeneity within each cluster and heterogeneity among different clusters [12]. In other words, instances that belong to the same cluster should be more similar to each other than instances that belong to different clusters. Thus, it is necessary to devise means of evaluating the similarities between instances. This problem is usually tackled indirectly, i.e. distance measures are used to quantify the dissimilarity between instances. Several dissimilarity measures can be employed for clustering tasks, such as the Euclidean distance – expression (1) – or the simple matching approach – expression (2). In both cases, the CGA uses all the available information (attribute values) to calculate such dissimilarities.

The CGA assumes that clustering involves the partitioning of a set **X** of instances into a collection of mutually disjoint subsets $\boldsymbol{C}_i$ of **X**. Formally, let us consider a set of N instances $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_N\}$ to be clustered, where each $\mathbf{x}_i \in \Re^\rho$ is an attribute vector consisting of ρ measurements. The instances must be clustered into non-overlapping groups $\mathbf{C}=\{\mathbf{C}_1,\mathbf{C}_2,...,\mathbf{C}_k\}$ where k is the number of clusters, such that:

$$\mathbf{C}_1 \cup \mathbf{C}_2 \cup ... \cup \mathbf{C}_k = \mathbf{X}\,, \quad \mathbf{C}_i \neq \emptyset, \quad \text{and} \quad \mathbf{C}_i \cap \mathbf{C}_j = \emptyset \text{ for } i \neq j. \tag{3}$$

The problem of finding an optimal solution to the partition of N data into k clusters is NP-complete [13] and, provided that the number of distinct partitions of N instances into k clusters increases approximately as $k^N/k!$, attempting to find a globally optimum solution is usually not computationally feasible [12]. This difficulty has stimulated the search for efficient approximate algorithms. Evolutionary algorithms [14,15] are widely believed to be effective on NP-complete global optimization problems and they can provide good sub-optimal solutions in reasonable time [13]. Under this perspective, a genetic algorithm specially designed for clustering problems was introduced in [6] and it is here reviewed. Figure 1 provides an overview of the CGA, whose main features are described in the sequel.

```
1) Initialize a population of genotypes;
2) Evaluate each genotype in the population;
3) Apply a linear normalization;
4) Select genotypes by proportional selection;
5) Apply crossover and mutation;
6) Replace the old genotypes by the ones formed in step 5);
7) If the convergence criterion is attained, stop; if not, go to step 2).
```

Fig. 1. Clustering Genetic Algorithm (CGA)

3.1 Encoding Scheme

The CGA [6] is based on a simple encoding scheme. Let us consider a dataset formed by N instances. Then, a genotype is an integer vector of (N+1) positions. Each position corresponds to an instance, i.e., the i-th position (gene) represents the i-th instance, whereas the last gene represents the number of clusters (k). Thus, each gene has a value over the alphabet $\{1,2,3,...,k\}$. For example, in a dataset composed of 20 instances, a possible genotype is: 223451234533214545525. In this case, 5 instances {1,2,7,13,20} form the cluster whose label is 2. The cluster whose label is 1 has 2 instances {6,14}, and so on. Finally, the last gene represents the number of clusters.

Standard genetic operators may not be suitable for clustering problems for several reasons [6,16]. First, the encoding scheme presented above is naturally redundant. In fact, there are $k!$ different genotypes that represent the same solution. Thus, the size of the search space is much larger than the original space of solutions. This augmented space may reduce the efficiency of the genetic algorithm. In addition, the redundant encoding also causes the undesirable effect of casting context-dependent information out of context under the standard crossover, i.e., equal parents may originate different offspring. Mainly for these reasons, the development of genetic operators specially designed for clustering problems has been investigated [6,16]. In this context, the CGA operators are of particular interest since they operate on constant length genotypes.

3.2 Crossover and Mutation Operators

The crossover operator combines partitions codified in different genotypes. It works in the following way. First, two genotypes (**A** and **B**) are selected. Then, assuming that **A** represents k_1 clusters, the CGA randomly chooses $c \in \{1,2,...,k_1\}$ clusters to copy into **B**. The unchanged clusters of **B** are maintained and the changed ones have their instances allocated to the corresponding nearest clusters (according to their centroids). In this way, an offspring **C** is obtained. The same procedure is employed to get an offspring **D**, but now considering that the changed clusters of **B** are copied into **A**. Thus, the crossover operator produces offspring usually formed by a number of clusters that are neither smaller nor larger than the number of clusters of their parents.

Two operators for mutation are used in the CGA. The first operator works only on genotypes that encode more than 2 clusters. It eliminates a randomly chosen cluster, placing its instances to the nearest remaining clusters (according to their centroids). The second operator divides a randomly selected cluster into 2 new ones. The first cluster is formed by the instances closer to the original centroid, whereas the other cluster is formed by those instances closer to the farthest instance from the centroid.

3.3 Objective Function

The objective function is based on the silhouette [17]. To explain it, let us consider an instance *i* belonging to cluster **A**. The average dissimilarity of *i* to all other instances of **A** is denoted by $a(i)$, whereas the average dissimilarity of *i* to all instances of a different cluster **C** will be called $d(i,\mathbf{C})$. After computing $d(i,\mathbf{C})$ for all clusters $\mathbf{C} \neq \mathbf{A}$, the smallest one is selected, i.e. $b(i) = \min d(i,\mathbf{C})$, $\mathbf{C} \neq \mathbf{A}$. This value represents the dissimilarity of *i* to its neighboring cluster, and the silhouette $s(i)$ is given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{\ a(i), b(i)\}} \tag{4}$$

It is easy to verify that $-1 \leq s(i) \leq 1$. Thus, the higher $s(i)$ the better the assignment of instance *i* to a given cluster. In addition, if $s(i)$ is equal to zero, then it is not clear whether the instance should have been assigned to its current cluster or to a neighboring one [18]. Finally, if cluster **A** is a singleton, then $s(i)$ is not defined and the most neutral choice is to set $s(i) = 0$ [17]. The objective function is the average of $s(i)$ over $i = 1,2,...,N$ and the best clustering is achieved when its value is maximized.

3.4 Selection, Settings and Initial Population

The genotypes corresponding to each generation are selected according to the roulette wheel strategy [19], which does not admit negative objective function values. For this reason, a constant equal to one is summed up to the objective function before the selection procedure takes place. In addition, the best (highest fitness) genotype is always copied into the succeeding generation.

The CGA does not employ crossover and mutation probabilities; that is, the designed operators are necessarily applied to some selected genotypes after the roulette wheel selection procedure is performed. Particularly, 50% of the selected genotypes are crossed-over, 25% are mutated by Operator 1 and 25% are mutated by Operator 2.

In this work, we have employed the methodology developed in [17] to set up the initial population. The process is based on the selection of representative instances. The first selected instance is the most centrally located in the set of instances. Subsequently, other instances are selected. Basically, the chance of selecting an instance increases when it is far from the previously selected ones and when there are many instances next to it. After selecting the representative instances, the initial population is formed considering that the non-selected instances must be clustered according to their proximity to the representative ones. Considering *k* representative instances, the first genotype represents 2 clusters, the second genotype represents 3 clusters,..., and the last one represents *k* clusters. Thus, we have employed initial populations formed by (*k*-1) genotypes, each one representing a different data partition.

4 Simulation Results

Imputation methods can be evaluated as *prediction tools*. To do so, known values can be *artificially* excluded from a dataset (missing values simulation), with the goal of predicting them by a particular imputation method. Thus, the *predicted* value can be compared with the real, known value *artificially eliminated* from the dataset. Considering this scenario, the more similar the imputed value is in relation to the real one, the better the imputation method is. In this work, we compare the prediction results obtained by the NNM with those achieved by the mean/mode imputation. Although the prediction results provide an efficient way to compare different imputation methods, requiring few computations after imputation, they do not provide any guarantee that the imputed values will be suitable for the ultimate data mining task – e.g. the clustering process. In summary, the *prediction* results are not the only important issue to be analyzed. Data mining methods usually explore relationships between attributes and, thus, it is critical to preserve them, as far as possible, when replacing missing values [3]. This aspect has motivated us to propose the NNM as an imputation tool for the CGA. Indeed, since both methods (NNM and CGA) are based on distance measures, which can somehow reflect the between-attribute relationships, the patterns inserted by the NNM tend to be consistent with the clustering process performed by the CGA. To assess this aspect, we compare the partitions obtained in the original datasets with those obtained in the imputed datasets. The next section describes the procedure employed to generate datasets formed by imputed values.

4.1 Missing Values Simulation and Imputation

Our simulations consider that there is just one missing value at a time. Let us consider a dataset formed by N instances $\mathbf{x}^i$=[$x_1^i, x_2^i, ..., x_\rho^i$]. First, we simulate that x_1^1 is missing and it is consequently substituted. Second, x_2^1 is missing and it is consequently substituted. This process is repeated until x_ρ^1 is substituted. After that, we simulate that x_1^2 is missing and it is consequently substituted. In summary, this procedure is repeated for all x_j^i (i=1,...,N; j=1,...,ρ). This way, simulations can be easily reproduced, i.e. they are not influenced by the choice of random samples. After the imputation process, we obtain a substituted dataset, which is formed only by imputed values (same number of instances and attributes of the original dataset). Thus, it is possible to compare the *imputed* values with the *original* ones, as well as the partitions obtained in the original dataset can be compared with those achieved in the imputed datasets.

4.2 Employed Datasets

The assessment of clustering results usually requires datasets for which the clusters are *a priori* known. In this sense, clustering algorithms can be evaluated by means of classification datasets. To do so, the clustering algorithm is applied in the classification dataset (without the class labels) in order to verify whether it finds the correct clusters (according to the known classes) or not. Our simulations were performed in two classification datasets that are benchmarks for data mining methods: Wisconsin Breast Cancer and Congressional Voting Records. These datasets are available at the UCI Machine Learning Repository [20]. These datasets were chosen because they are formed by ordinal and binary attributes, showing the applicability of our method, which can also be employed for continuous and nominal attributes, e.g. using expressions (1) and (2) to compute distances respectively. In this sense, we extend our previous work [21], in which only ordinal attributes were considered.

In the Wisconsin Breast Cancer dataset, each instance has 9 ordinal attributes (A_1,…, A_9) and an associated class (benign or malignant). The attribute values belong to the set {1,2,…,9}. There are 699 instances, of which 16 have missing values. We removed those instances (to allow evaluating the prediction results) and used the remaining ones to simulate imputations. The Congressional Voting Records dataset includes votes for each of the U.S. House of Representatives Congressmen on 16 key votes (attributes A_1,…,A_{16}). There are 435 instances, of which 203 have missing values. These instances were removed (to make the prediction evaluation possible) and the proposed method was employed in the remaining ones.

4.3 Evaluating the NNM as a Prediction Tool

In this section, we compare the imputed values with the original ones (*artificially excluded* from the dataset). This is performed by reporting the average prediction error for each attribute. For the ordinal attributes of the Wisconsin dataset, we calculate the average absolute differences between substituted and original values for each attribute – considering all substitutions. In this case, the NNM imputations are compared with those achieved by the mean value, and the obtained results (average prediction error) are depicted in Figure 2. For the binary attributes of the Congress dataset, the NNM average error rate is compared with the results obtained by the mode imputation (Figure 3). The NNM provided better results than the substitution by the mean in all attributes of Wisconsin, whereas in Congressional the NNM provided better results in 14 out of 16 attributes. The mean/mode imputation was also performed according to the methodology described in Section 4.1.

4.4 Evaluating the Influence of NNM Imputation in a Clustering Task

In this section, we report results that allow estimating the suitability of the NNM in the context of the partitions found by the CGA. As previously mentioned, imputed values should preserve the between-attribute relationships observed in the clean (original) dataset. In a clustering process, it means that the correct clusters should be preserved, i.e. it is expected that the imputed values do not change the *classification* of each particular instance. To evaluate this aspect, it is assumed that the correct clusters are given by the classes. Thus, it is possible to verify to what extent the CGA is capable of finding the correct clusters, which are given by the instances of each class. In this sense, we compare the A*verage Correct Classification Rates (ACCRs)* obtained by CGA in the original dataset with those obtained in the substituted datasets.

The CGA was applied in the original dataset and in the dataset formed only by substituted values, using populations formed by 20 genotypes that, in turn, implies in using 21 clusters at most (see Section 3.4 for details). We simulated the clustering process 11 times (this number is convenient to perform the Wilcoxon/Mann-Whitney test [22]) for each dataset and the maximum number of generations was set to 100. In all simulations, the CGA has found the correct number of clusters. Table 1 shows the obtained results in terms of the Average Correct Classification Rates (ACCRs).

Table 1. ACCRs (%): average *(μ);* standard deviation *(σ)*

Dataset	CGA (Original)	CGA (Imputed by NNM)
Wisconsin Breast Cancer	μ=95.45; σ=0.30	μ=95.27; σ=0.30
Congressional Voting	μ=86.41; σ=0.22	μ=88.36; σ=0.00

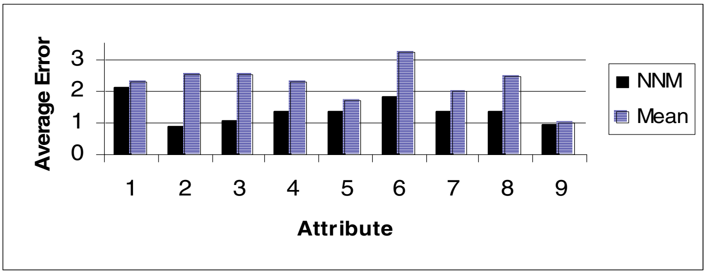

Fig. 2. Average prediction error - Wisconsin Breast Cancer

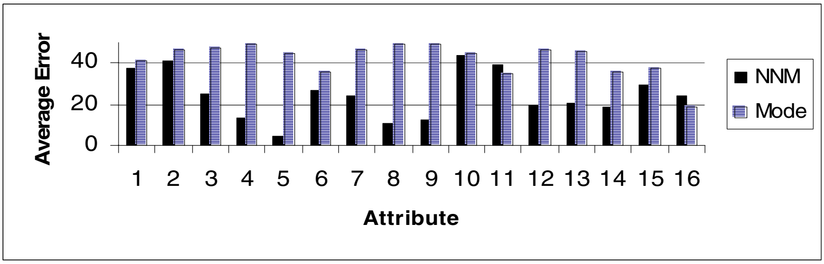

Fig. 3. Average prediction error (%) - Congressional Voting Records

The CGA has provided similar ACCRs in both datasets (original and imputed by NNM). This aspect was statistically evaluated by means of the Wilcoxon/Mann-Whitney test [22]. In the Wisconsin Breast Cancer, it was performed supposing that the ACCR values in the original dataset are equal to those obtained in the substituted dataset, and we concluded that the results are statistically significant at the 5% significance level. In the Congressional Voting Records, there is sample evidence (α=5%) suggesting that the results in the imputed dataset are slightly better than in the original dataset. These results suggest that the proposed method is a suitable estimator for missing values, preserving (Wisconsin) or slightly improving (Congress) the relationships between attributes in the clustering process. Finally, due to the methodology employed in our simulations (Section 4.1), most of the values imputed by the mean/mode are equal across all instances. In this sense, instances in the substituted dataset form only one cluster. In this case, it does not make sense to evaluate the clustering results achieved by the mean/mode imputation. However, the methodology described in Section 4.1 is particularly interesting to evaluate clustering results in the datasets imputed by NNM, because these datasets do not contain any original values and, thus, the corresponding CGA´s results are not *positively* biased by them.

5 Conclusions

Missing values are a critical problem in data mining applications. In this work, we presented a Nearest-Neighbor Method (NNM) to substitute missing values and showed that it can be useful for a Clustering Genetic Algorithm (CGA). In the NNM, each instance containing missing values is compared with the complete instances, using a distance metric, and the most similar complete instance is used to assign the missing value for a particular attribute.

The proposed method was assessed by means of simulations performed in two datasets that are benchmarks for data mining: Wisconsin Breast Cancer and Congressional Voting Records. Our simulations were designed to evaluate the NNM both in prediction and in clustering tasks. In the prediction task, we compared the results obtained by the NNM with those achieved by the mean/mode imputation. In the clustering task, we compared the partitions obtained in original datasets with those achieved in imputed datasets. The prediction results showed that the NNM provided better results than the mean/mode imputation. Although the prediction results are relevant, they are not the only important issue to be analyzed. In fact, imputation methods must generate values that least distort the original characteristics of the original sample, preserving the between-attribute relationships. In our work, we evaluated this aspect in the CGA context, performing clustering simulations and comparing the results obtained in the original datasets with the substituted ones. These results indicated that the proposed method is a suitable estimator for missing values.

Considering our future work, there are many aspects that can be further investigated. One important issue involves evaluating the best number of neighbors (K) in the K-nearest-neighbor method. Finally, we are also going to assess the efficacy of the proposed method in real-world datasets, comparing the NNM results with those obtained by other imputation methods.

Acknowledgements. The authors acknowledge CNPq, FAPESP and FAPERJ for the financial support.

References

1. Fayyad, U. M., Shapiro, G. P., Smyth, P. From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining, Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., Editors, MIT Press, pp. 1-37, 1996.
2. Witten, I. H., Frank, E., Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, USA, 2000.

3. Pyle, D., Data Preparation for Data Mining, Academic Press, 1999.
4. Little, R., Rubin, D. B., Statistical Analysis with Missing Data, John Wiley & Sons, New York, 1987.
5. Rubin, D. B., Multiple Imputation for non Responses in Surveys, New York, John Wiley & Sons, 1987.
6. Hruschka, E. R., Ebecken, N.F.F. A genetic algorithm for cluster analysis, Intelligent Data Analysis (IDA), Netherlands, v.7, n.1, 2003.
7. Batista, G. E. A. P. & Monard, M. C., An Analysis of Four Missing Data Treatment Methods for Supervised Learning, Applied Artificial Intelligence, v.17, n.5-6, 519-534, 2003.
8. Hruschka, E. R., Hruschka Júnior, E.R., Ebecken, N.F.F, Towards Efficient Imputation by Nearest-Neighbors: A Clustering Based Approach, Proc. of the 17th Australian Joint Conference on Artificial Intelligence , LNAI 3339, pp. 513-525, Springer, 2004.
9. Hruschka, E. R., Hruschka Junior, E. R., Ebecken, N. F. F. Evaluating a Nearest-Neighbor Method to Substitute Continuous Missing Values In: The 16th Australian Joint Conference on Artificial Intelligence, LNAI 2903, pp. 723-734, Springer-Verlag, 2003.
10. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B., Missing value estimation methods for DNA microarrays, Bioinformatics, 17(6), 520-525, 2001.
11. Mitchell, T. M., Machine Learning, McGraw-Hill, 1997.
12. Arabie, P., Hubert, L. J., An Overview of Combinatorial Data Analysis (Chapter 1). Clustering and Classification, ed. P. Arabie, L.J. Hubert, G. DeSoete, World Scientific, 1999.
13. Park, Y., Song, M., A Genetic Algorithm for Clustering Problems, Proceedings of the Genetic Programming Conference, University of Wisconsin, July, 1998.
14. Yao, X., Evolutionary Computation: Theory and Applications, World Scientific, Singapore, 1999.
15. Tan, K. C., Lim, M. H., Yao, X., Wang, L., Recent Advances in Simulated Evolution and Learning, World Scientific, Singapore, 2004.
16. Falkenauer, E., Genetic Algorithms and Grouping Problems, John Wiley & Sons, 1998.
17. Kaufman, L., Rousseeuw, P. J., Finding Groups in Data – An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics, 1990.
18. Everitt, B.S., Landau, S., Leese, M., Cluster Analysis, Arnold Publishers, London, 2001.
19. Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley Longmann, 1989.
20. Merz, C.J., Murphy, P.M., UCI Repository of Machine Learning Databases, http://www.ics.uci.edu, University of California, Irvine, CA.
21. Hruschka, E.R, Hruschka Júnior, E.R., Ebecken, N.F.F., A Nearest-Neighbor Method as a Data Preparation Tool for a Clustering Genetic Algorithm, Proceedings of the 18^{th} Brazilian Symposium on Databases, pp. 319-327, Manaus, Brazil, 2003.
22. Triola, M. F., Elementary Statistics, 7th Edition, Addison Wesley Longman Inc., 1999.