2009 Ninth International Conference on Intelligent Systems Design and Applications

EACImpute: An Evolutionary Algorithm for Clustering-Based Imputation

Jonathan de Andrade Silva and Eduardo R. Hruschka
Computer Science Department
University of São Paulo (USP) at São Carlos, Brazil
{*jandrade,erh*}*@icmc.usp.br*

***Abstract*—We describe an imputation method (EACImpute) that is based on an evolutionary algorithm for clustering. This method relies on the assumption that clusters of (partially unknown) data can provide useful information for imputation purposes. Experimental results obtained in 5 data sets illustrate different scenarios in which EACImpute performs similarly to widely used imputation methods, thus becoming eligible to join a pool of methods to be used in practical applications. In particular, imputation methods have been traditionally only assessed by some measures of their prediction capability. Although this evaluation is useful, we here also discuss the influence of imputed values in the classification task. Finally, our empirical results suggest that better prediction results do not necessarily imply in less classification bias.**

***Keywords*-Missing values; classification; imputation;**

I. Introduction

Missing values are common in real-world data sets and they can occur for a number of reasons like, for instance, malfunctioning measurement equipment. Such missing data are usually problematic. Therefore, several approaches have been proposed to deal with them [1], [2]. A simple approach to deal with missing values involves ignoring instances and/or attributes containing missing values, but the waste of data may be considerable and incomplete data sets may lead to biased statistical analyses. Alternatively, some approaches for data analysis can be tolerant to missing values. Finally, a significant number of data mining methods only work with complete data sets. For these methods, approaches aimed at filling in missing values are particularly relevant.

The task of filling in missing data is often referred to as missing values substitution or imputation and it can be performed in a number of ways like, for instance, by the widely used mean/mode imputation. However, this approach considerably underestimates the population variance and does not take into account the between-attribute relationships, which are usually relevant to the process of missing values replacement. Moreover, data mining methods usually explore relationships between attributes and, thus, it is critical to preserve them, as far as possible, when replacing missing values [1]. In this sense, imputation is aimed at carefully substituting missing values, trying to avoid the insertion of bias in the data set. If imputation is performed in a suitable way, higher quality data becomes available, and the data mining outcomes can be improved.

Recently, a number of algorithms capable of dealing with missing values have been developed, including nearest-neighbors based imputation algorithms like those described in [3]–[5] and that have been shown very useful in bioinformatics applications. Despite the encouraging results achieved by such imputation algorithms, most of the experimental settings reported in the literature only assess their prediction capabilities, obtained from the simulation of missing entries for some attributes whose values are actually known. From this standpoint, artificially generated missing values are substituted and then compared to their corresponding known values. Although this approach is valid and widely adopted, the prediction results are not the most important issue to be analyzed, mainly in classification problems [6]. In reality, the substitution process should generate values that least distorts the original characteristics of the data set for the classification process. In this context, the main contributions of this paper are twofold. First, we provide a detailed description of an imputation algorithm (named EACImpute, from Evolutionary Algorithm for Clustering-based Imputation), which was briefly introduced in [7]. Second, we present experimental results that suggest that EACImpute performs similarly to five imputation methods (KNN [3], SKNN [4], IKNN [5], KMI [8], and Majority Method [9]) thus becoming eligible to join a pool of methods to be used in practical applications. As a complementary contribution of our work, we report experimental results that suggest that better prediction results may not necessarily lead to less classification bias.

The remainder of this paper is organized as follows. The next section describes EACImpute. Section III briefly reviews a methodology [6] to estimate the bias inserted by imputation methods in the context of classification problems. Section IV reports experimental results obtained in five data sets. Finally, Section V concludes this paper.

II. EACImpute

EACImpute relies on the assumption that clusters of (partially unknown) data can provide useful information for imputation purposes. In particular, data clusters can be viewed as information granules [10] that summarize the spatial distribution of data. Such information granules can provide a workable estimate to fulfill missing values that least distorts the values that are actually present in the

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.86

data set. Having this purpose in mind, several clustering algorithms can be adapted for imputation. For instance, the popular k-means would be a spontaneous choice for being simple and scalable. However, it is sensitive to initialization of prototypes and requires that the number of clusters k be specified in advance. This can be restrictive in practice, leading us to alternatively consider the Evolutionary Algorithm for Clustering (EAC) [11] - which is capable of automatically estimating k from data - as part of our framework. In a nutshell, EAC [11] has been designed to evolve data partitions with variable k by eliminating, splitting, and merging clusters that are systematically refined by k-means. We refer the reader interested in further details of EAC to [11]. In the following, we detail the main features of EACImpute.

Let us first present an overview of the Evolutionary Algorithm for Clustering-based Imputation. EACImpute has a simple encoding scheme. In order to explain it, let us consider a data set composed of N instances. A partition is encoded as an integer string (genotype) of N positions. Each string position corresponds to a data set instance, i.e., the i-th position represents the i-th instance of the data set. Thus, each string component has a value over the possible cluster labels $\{1, 2, 3, ..., k\}$. For example, consider Fig 1, which illustrates a population formed by 6 genotypes that encode different data partitions of a data set composed by 12 instances (ID=0,1,...,11) described by 4 attributes (x_1, x_2, x_3, x_4). The sixth genotype (from top to bottom) encodes 2 clusters (for convenience the number of clusters, k, is shown in the last position of the string). For every string, an objective function value, which quantitatively captures the goodness of the encoded data partition, is computed. Such a value is actually a measure of a relative clustering validity criterion (the simplified silhouette [11] is here used). According to this clustering validity criterion, the sixth genotype encodes the best clustering solution of this population (i.e., it has the higher fitness function value - 1.86) and, for this reason, will be used for imputation purposes. Imputations are performed by means of the widely known nearest neighbor principle used by other imputation algorithms ([3]–[5]), considering all instances that have known values for the attribute that has a missing value in a given instance. To do so, we only take into account instances of the same cluster. For the sake of illustration, consider the substitution of the missing value for the second attribute (x_2) of the second instance (ID=1) of our pedagogical data set. This missing value will be substituted by taking into account only instances belonging to cluster 1 (i.e., instances whose IDs are 0, 2, 3, 4 and 5). This approach leads to an automatic determination of the number of neighbors (in this case, 5 neighbors), whose setting may be hard to be accomplished in practice for other imputation tools. After imputation, the evolutionary process (selection, mutation, etc.) is repeated. Now that we have provided a rather informal overview of

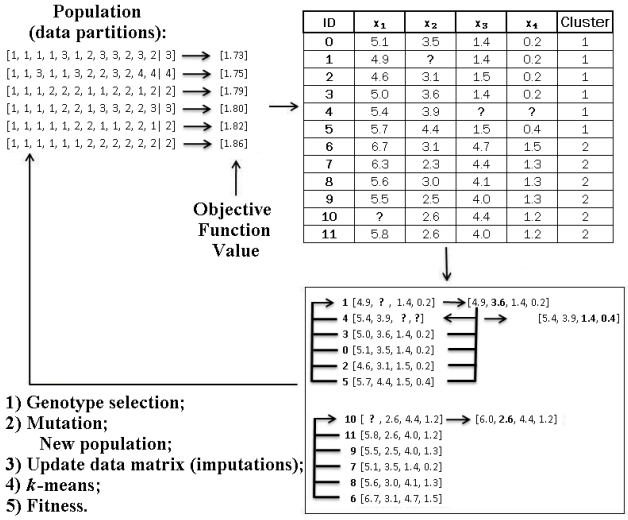


Figure 1. Illustrative process of imputation.

EACImpute, let us focus on its algorithmic details.

More formally, the underlying idea behind EACImpute is to repeatedly performing a clustering step followed by an imputation step. More precisely, let each instance i of a given data set be described by both a vector of m attribute values $\mathbf{x}^i = [x_1^i, x_2^i, ..., x_m^i]$ and its corresponding class c_i, which can take any value from a set of values $C = \{c_1, c_2, ..., c_c\}$. A data set can be represented by a matrix $\mathbf{X}_D$ formed by a set of vectors $\mathbf{x}^i$ $(i = 1, ..., N)$, each one with an associated $c_i \in C$. This matrix is formed by the values of the attributes a_l $(l = 1, ..., m)$ for each instance i and its respective class. Thus, x_l^i is the value of the l-th attribute of the i-th instance in $\mathbf{X}_D$. In general, $\mathbf{X}_D$ is formed by both complete instances (without any missing value) and by instances with at least one missing value. Let $\mathbf{X}_C$ be the subset of instances of $\mathbf{X}_D$ that do not have any missing value, and $\mathbf{X}_M$ be the subset of instances of $\mathbf{X}_D$ with at least one missing value, i.e., $\mathbf{X}_D = \mathbf{X}_C \cup \mathbf{X}_M$. In this context, imputation methods fill in the missing values of $\mathbf{X}_M$, originating a filled matrix $\mathbf{X}_F$. Now we can focus on the main steps of EACImpute summarized in Fig. 2. In classification problems, EACImpute can be adapted for supervised imputation, which involves applying it for the instances of each class separately, as done in this work. A number of EACImpute steps deserve further attention:

a) In step 1, as well as in Step 2.1 for $t = 1$, the computation of the (Euclidean) dissimilarities between instances takes into account missing values. In brief, only attributes a_l $(l = 1, ..., m)$ without missing values are used for computing dissimilarities between two instances (or between an instance and a centroid). Attributes a_o $(o = 1, ..., m)$ containing missing values in any instance for which a particular pair wise dissimilarity computation is

1. Initialize a set (population) of randomly generated data partitions (represented by genotypes) using $\mathbf{X}_D$. ${\mathbf{X}^{(0)}}_D \leftarrow \mathbf{X}_D$;
2. For $t = 1, ..., G$ do: // *G is the number of generations* //
   2.1. Apply k-means to each genotype - using ${\mathbf{X}^{(t-1)}}_D$;
   2.2. Assess genotypes (data partitions) and estimate values to be imputed according to the best available genotype; Store the estimated values in ${\mathbf{X}^{(t)}}_F$ for future reference;
   2.3. Select genotypes using the proportional selection;
   2.4. In 50% of the selected genotypes, apply the mutation operator 1, which eliminates some randomly selected clusters, placing its instances into the nearest remaining clusters. In the remaining genotypes, apply the mutation operator 2, which splits some randomly selected clusters, each of which into two new clusters.
   2.5. Replace the old genotypes by those just formed in step 2.4;
   2.6. Update the data matrix: ${\mathbf{X}^{(t)}}_D \leftarrow (\mathbf{X}_C \cup {\mathbf{X}^{(t)}}_F)$;
3. ${\mathbf{X}'}_D \leftarrow {\mathbf{X}^{(G)}}_D$;

Figure 2. Main Steps of EACImpute (further details in the text).

being performed are omitted in the distance function. To avoid falsely low distances for instances with a lot of missing values, we average each computed dissimilarity by the number of attributes considered in the computation.

b) In Step 2.2, a filled matrix ${\mathbf{X}^{(t)}}_F$ is derived from estimating values for fulfilling missing entries of $\mathbf{X}_M$. We shall refer to such values as estimates (or partial imputations) because they are likely to vary for $t = 1, ..., G$. Thus, we consider that values have been actually imputed when $t = G$. In what concerns such partial imputations, they are performed by considering only the instances of the cluster for which a given instance belongs to (at a given t). In particular, a traditional k-NN imputation method is applied by considering only a subset of ${\mathbf{X}^{(t)}}_D$ given by instances of a specific cluster. Similarly to KNN [3], the computation of the values to be imputed are based on a weighted (Euclidean) distance function for which the closer the instance the more relevant it is for imputation purposes.

c) Differently from the algorithm described in [7], we here do not restrict EACImpute to use only complete instances for imputation purposes. Instead, every instance belonging to a given cluster is employed in the imputation process. This approach allows using EACImpute here described even for data sets in which every instance has missing value(s). Such data sets are not uncommon in real-world applications (e.g., in bioinformatics).

III. Bias in Classification

Recall from Section II that imputation methods fill in the missing values of $\mathbf{X}_M$, originating a filled matrix $\mathbf{X}_F$. We assume that the *class value* is known for every instance. In an ideal situation, the imputation method fills in the missing values, originating filled values, without inserting any bias in the data set. In a more realistic view, imputation methods are aimed at decreasing the amount of inserted bias to acceptable levels, in such a way that a data set ${\mathbf{X}'}_D = \mathbf{X}_C \cup \mathbf{X}_F$, probably containing more information than $\mathbf{X}_C$, can be used for data mining (e.g., considering issues such as attribute selection, combining multiple models, and so on). From this standpoint, it is particularly important to emphasize that we are assuming that the known values in $\mathbf{X}_M$ may contain important information for the modeling process. This information would be (partially) lost if the instances and/or attributes with missing values were ignored.

Two general approaches have been used in the literature to evaluate the bias inserted by imputations. We shall refer to them as *prediction* and *modeling* approaches. In a prediction approach, missing values are simulated, i.e., some known values are removed and then imputed. For instance, some known values from $\mathbf{X}_C$ could be artificially eliminated, simulating missing entries. In this way, it is possible to evaluate how similar the imputed values are to the real, a priori known values. The underlying assumption behind this approach is that the more similar the imputed value is to the real value, the better the imputation method is. Although the prediction approach is valid and widely adopted, the prediction results are not the most important issue to be analyzed as discussed, for instance, in [6]. In brief, the prediction approach does not allow estimating the inserted bias from a modeling perspective. More precisely, the substitution process must generate values that least distorts the original characteristics of $\mathbf{X}_D$, which can be assumed to be the between-attribute relationships, for the modeling process. These relationships are often explored by classification algorithms. For the sake of argument, let us consider the pedagogical example depicted in Fig 3. This example is inspired in the widely known Iris data set, which contains instances formed by 4 attributes (SL, SW, PL, and PW) and the class label. Consider that the instance whose ID is 151 has a missing value for attribute PW. This instance is identical to instance 44, except for the missing value. In other words, instance 151 could be viewed as a result of a procedure for missing value simulation widely used in the literature. In this context, an imputation method should estimate a value as close as possible to 0.6, which is the known value artificially excluded from the instance 44. Let us now assume that 2 imputation methods (A and B) are available to substitute such a missing value. Also, consider that method A substitutes the missing value by 0.2, whereas method B substitutes it by 0.601. Clearly method B is better

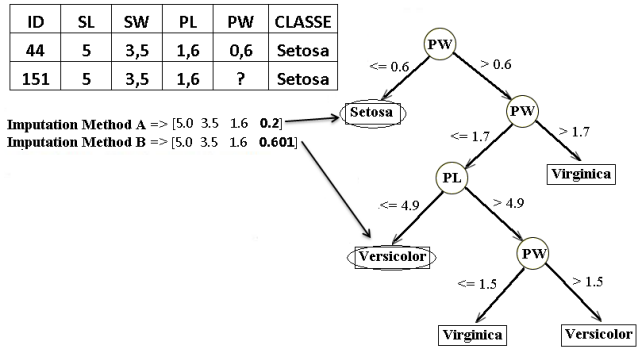


Figure 3. Pedagogical example of insertion bias on classification.

than method A from the prediction point of view. However, consider now that the tree depicted in Fig 3 is a perfect classifier. According to this classifier, imputation method B would make instance 151 to be incorrectly classified, whereas imputation method A, which is not as good as B from the prediction point of view, would lead to the correct classification of the considered instance.

Several authors (e.g., see [6] for a review) have also argued that it is more important to take into account the influence of imputed values in the modeling process (e.g., preserving the relationships between attributes) than to get more accurate predictions. Roughly speaking, although the imputed values are predictions, it is not the accuracy of these predictions that is of most importance when replacing missing values. It is more important that such predictions produce a workable estimate that least distorts the values that are actually present in the data set. In other words, the main purpose of imputation is not to use the values themselves, but to make available to the modeling tools the information contained in the other variables' values that are present. For all these reasons, we have focused on the inserted biases in terms of classification results, which somehow allow evaluating to what extent the relationships between attributes are being maintained after imputation. Finally, one must acknowledge that in real-world applications the imputed values cannot be compared with any value.

The bias inserted by imputation can be defined as [6] "the magnitude of the change in the between-attribute relationships caused by patterns introduced by an imputation process". The problem is that the relationships between attributes are hardly known a priori (before data mining is performed). Therefore, usually the inserted bias cannot be directly measured, but it can be estimated. In classification problems, the underlying assumption is that between-attribute relationships are induced by a particular classifier. Consequently, the quality of such discovered relationships can be indirectly estimated by classification measures like the Average Correct Classification Rate (ACCR). In this sense, we adopt a methodology to estimate the inserted bias

1) Evaluate the classifier's ACCR by cross-validation in $\mathbf{X}_C$, obtaining ACCRC;
2) Evaluate the classifier's ACCR in $\mathbf{X}_F$ (here viewed as a test set) considering that $\mathbf{X}_C$ is the training set.
In other words, this step involves building the classifier in $\mathbf{X}_C$ and then testing it in the instances of $\mathbf{X}_F$,
thus obtaining ACCRF.
3) The bias (b) inserted by the performed imputations is estimated from the difference between the results achieved in steps 2) and 1): $\widehat{b}$ =ACCRF− ACCRC.

Figure 4. Estimating the inserted bias on classification.

detailed in [6] and addressed in the sequel.

In data mining applications, different classifiers are often assessed for a given data set, in such a way that the best available classifier is then chosen according to some criterion of model quality (e.g., the ACCR). Our underlying assumption is that the best classifier (BC) - in relation to $\mathbf{X}_C$ and to the available classifiers - provides a suitable model for classifying instances after imputations have been performed. Thus, it is important to assess if the imputed values adjust themselves to the BC model. It is a common practice to evaluate classifier performance in a test set. The same concept can be adapted to evaluate imputations, considering $\mathbf{X}_C$ as the training set and $\mathbf{X}_F$ as the test set. Then, inserted bias can be estimated by means of the procedure in Fig. 4. According to this procedure, a positive bias is achieved when the ACCR in $\mathbf{X}_F$ (step 2) is greater than in the cross-validation process in $\mathbf{X}_C$ (step 1). In this case, the imputed values are likely to improve the classifier's ACCR in $\mathbf{X'}_D$. Accordingly, a negative bias is inserted when the imputed values are likely to worsen the classifier's ACCR in $\mathbf{X'}_D$. Finally, no bias is likely inserted when the classifier's accuracies in $\mathbf{X}_F$ and in the cross-validation process in $\mathbf{X}_C$ are equal. Assuming that the imputation process should not introduce artificial patterns into the data, this is the ideal situation. Indeed, these artificial patterns, not present in the known values, may be later discovered during the data mining process in $\mathbf{X'}_D$. Therefore, the inclusion of such artificial patterns, which are simply an artifact of the imputation process, should be avoided. According to our elaboration, not only a negative bias but also a positive bias is not desirable, as both imply that artificial patterns have been likely incorporated into the data set.

IV. Illustrative Experimental Results

In our experiments, we used 5 data sets from the UCI[1] Machine Learning Repository, namely: Iris, Glass, Yeast, Segmentation, and Pen-digits, whose main features are summarized in Table I. Following the Rubin's typology, we simulated missing values according to the distribution of missingness known as "missing completely at random"

[1]A. Asuncion and D. Newman, 2007. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

Table I
DESCRIPTION OF DATA SETS USED.

Data sets	# Instances	# Attributes	# Classes
Iris	150	4	3
Glass	214	9	6
Yeast	1484	8	10
Segmentation	2310	19	7
Pen-digits	10992	16	10

Table II
AVERAGE CORRECT CLASSIFICATION RATES (ACCR) FOR THE EMPLOYED CLASSIFIERS - BEST RESULTS IN BOLD.

Data set	J48	KNN	MLP	NB
Iris	94.73%	95.2%	**96.93**%	95.53%
Glass	67.63%	**70.02**%	67.32%	49.45%
Yeast	56.42%	54.35%	**58.86**%	57.99%
Segmentation	96.41%	**99.35**%	94.64%	85.79%
Pen-digits	**96.82**%	96.10%	96.05%	80.31%

(MCAR). In particular, in each simulation we removed values from the real attributes according to different rates, viz. 10%, 30%, 50%, and 70% in a supervised way (i.e., conditioned on the class values). For each of these missing rates, 30 imputation simulations were performed for different quantities of attributes (1, 2,..., $m/2$), where m is the number of attributes (see third column of Table I), resulting in 720, 2880, 4800, 7560 and 9600 data sets with missing values respectively to Iris, Glass, Yeast, Segmentation and Pen-digits. Since we are using Euclidean distance to compute dissimilarities, only the real attributes were considered for imputation.

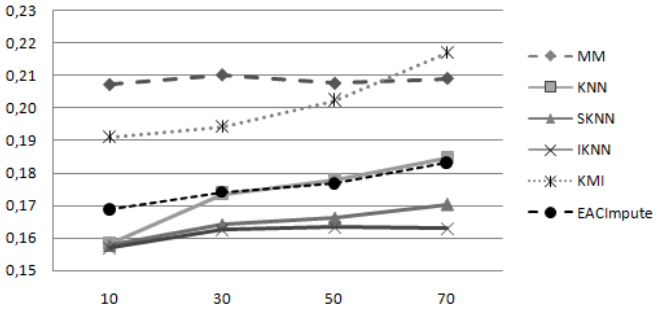
For EACImpute, we arbitrarily adopted the following parameters: populations formed by 5 genotypes and $G = 20$. Sensitivity analysis on these parameters can be the subject of future work, but we shall note that the population size and the number of generations are inversely interdependent on each other, since increasing the size of the evolutionary population typically reduces the number of generations needed for convergence and vice-versa. In what concerns the number of iterations of k-means, empirical evidence suggests that 5 or less repetitions ordinarily will suffice [12]. So, EACImpute runs 5 iterations of k-means for each genotype. We believe that such parameters are not critical if computational efficiency is not of paramount importance.

The results obtained by EACImpute were compared to those achieved by 5 algorithms for imputation. Three of them are considered state of the art algorithms, namely: KNN [3], SKNN [4], IKNN [5]. We run these algorithms by setting their parameters as suggested by the original references. In particular, for KNN, SKNN, and IKNN, the number of neighbors was set to 10. In addition, the number of iterations was set to 2 for IKNN. Besides comparing the performance of EACImpute to those state of the art algorithms, we also report the results achieved by the widely known Majority Method (MM) [9], which substitutes missing entries by the mean (or mode) of the known values for a given attribute, conditioned on the class. Finally, we also included in our experimental setting the k-means imputation (KMI) algorithm [8], for that, similarly to EACImpute, it also performs clustering based imputation. For KMI, we set the number of clusters and neighbors to 2 and 1, respectively.

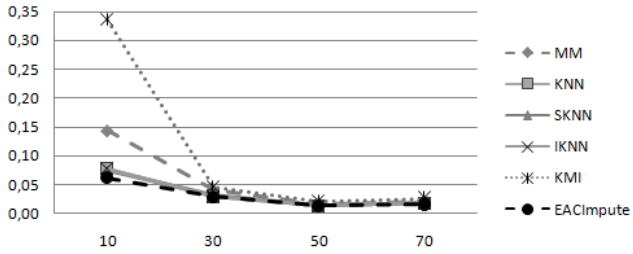
For illustration purposes, let us initially consider the results obtained from a prediction point of view. The accuracy of predictions was evaluated by calculating the error between actual (known) values and the respective imputed values. To do so, we employed the widely known *Normalized Root Mean Squared Error* (NRMSE). The obtained results (averaged over 30 simulations for each pair of <missing rate, number of attributes with missing values>) are illustrated in Fig. 5. One can observe from this Fig that IKNN showed the best results in most of the performed experiments.

Let us now concentrate on the inserted bias on classification - by means of the procedure detailed in Section III. To do so, it would be desirable to employ a classifier as accurate as possible for each of the data sets in hand. Therefore, in order to estimate the bias inserted by the performed imputations, 4 classifiers that are popular in the data mining community were used: Decision tree J4.8, K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), and Naïve Bayes (NB). These classifiers make part of the WEKA System [13], which was used to perform our experiments using its default parameters. For each data set, the best classifier is used for estimating the classification bias inserted from imputations. Classifiers were previously assessed in a 10-fold cross validation process, and ranked according to their ACCR's (See Table II). For instance, J.48 is used to estimate the classification bias for Pen-digits. Due to space restrictions and aimed at facilitating the illustration of the achieved results, in Fig. 6 we report only the average absolute values of the classification bias. KNN and IKNN showed the best results in most of the experiments.

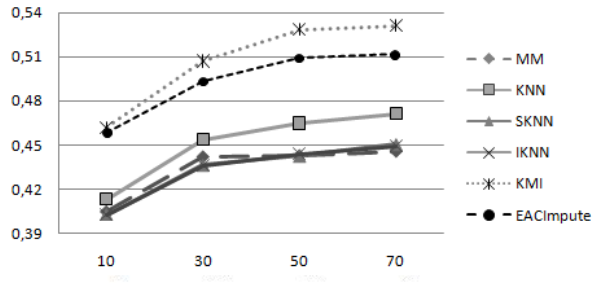
In order to provide some reassurance about the validity and non-randomness of the obtained results, we present the results of statistical tests by following the approach proposed by Demšar [14]. In brief, this approach is aimed at comparing multiple algorithms on multiple data sets, and it is based on the use of the well known Friedman test with a corresponding post-hoc test. The Friedman test is a non-parametric statistic test equivalent to the repeated-measures ANOVA. If the null hypothesis, which states that the algorithms under study have similar performances, is rejected, then we proceed with the Nemenyi post-hoc test for pair-wise comparisons between algorithms. Table III shows the average ranks obtained for performing the Friedman test (the lower the rank the better the algorithm), summarizing the achieved results for both prediction (P) and classification bias (B). It is interesting to observe from Table III that better prediction results do not necessarily lead to less inserted



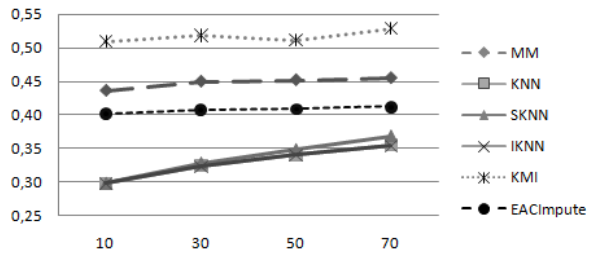
(a) NRMSE-Iris.



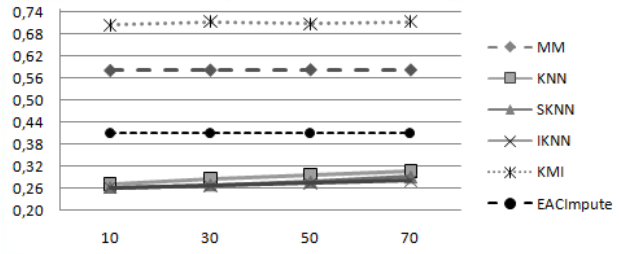
(b) NRMSE-Glass.



(c) NRMSE-Yeast.

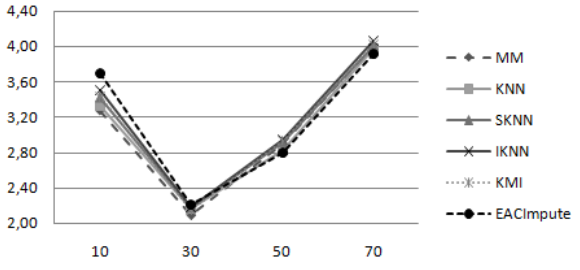


(d) NRMSE-Segmentation.

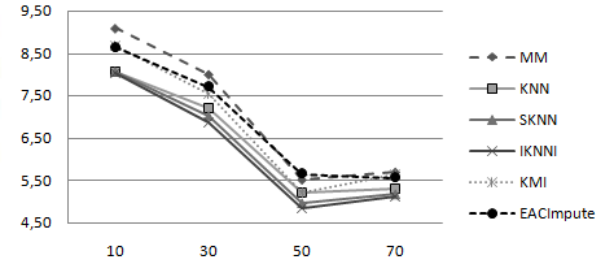


(e) NRMSE-Pen-digits.

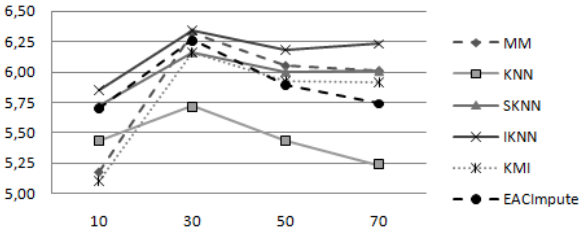
Figure 5. Results of predictions with NRMSE error on 10%, 30%, 50% and 70% of missing values.



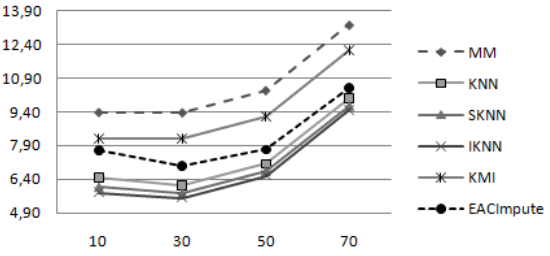
(a) BIAS - Iris.



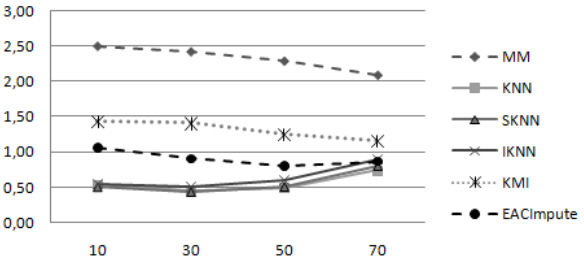
(b) BIAS - Glass.



(c) BIAS - Yeast.



(d) BIAS - Segmentation.



(e) BIAS - Pen-digits.

Figure 6. Results of classification bias on 10%, 30%, 50% and 70% of missing values.

Table III
STATISTICAL ANALYSIS - PREDICTION (P) AND CLASSIFICATION BIAS (B) FOR DIFFERENT MISSING VALUES (MV) RATES.

Average ranks used to perform the Friedman test							
MV Rate		MM	KNN	SKNN	IKNN	KMI	EACImpute
10%	P	4.8	3.0	2.2	1.6	5.8	3.6
	B	4.2	2.6	2.6	3.2	4.0	4.4
30%	P	4.8	3.0	2.6	1.0	5.8	3.8
	B	4.8	2.4	2.4	3.2	3.6	4.6
50%	P	4.6	2.6	2.4	1.8	5.8	3.8
	B	5.0	2.0	2.8	3.4	4.4	3.4
70%	P	4.2	3.2	2.8	1.4	6.0	3.4
	B	5.4	2.0	2.6	3.6	4.6	2.8

bias. The statistical procedure used here ($\alpha = 0.05$) suggests that we cannot reject the null hypothesis of equal prediction capabilities for most of assessed algorithms. Actually, significant differences in pair-wise comparisons were observed between the following pairs of algorithms: <SKNN, KMI> for all missing rates, <IKNN, KMI> for all missing rates, <IKNN, MM> for 10% and 30% of missing entries, and <KNN, KMI> for 50% of missing entries. Considering the classification bias, the null hypothesis of equal biases was only rejected for data sets with 70% of missing entries. In particular, significant differences in pair-wise comparisons were observed between KNN and MM.

V. CONCLUSION

We described an imputation method (EACImpute) that is based on an evolutionary algorithm for clustering. EACImpute was empirically assessed by means of a significant number of experiments performed in 5 different data sets. In particular, missing values substitution has been traditionally assessed by some measures of the prediction capability of imputation methods. Although this evaluation is useful, it does not allow inferring the influence of imputed values in the ultimate modeling task (e.g., in classification as discussed in this paper). In this sense, alternative approaches to the so called prediction capability evaluation are needed. Therefore, we here also assessed the influence of imputed values in the classification task. The results achieved by EACImpute were compared to those obtained by 5 imputation algorithms (Majority Method [9], KNN [3], SKNN [4], IKNN [5], and KMI [8]). In order to provide some reassurance about the validity and non-randomness of the obtained results, we presented the results of statistical tests, which suggest that EACImpute performs similarly to widely used imputation methods, thus becoming eligible to join a pool of methods to be used in practical applications. Finally, our empirical results suggest that better prediction results do not necessarily imply in less classification bias. Our future work will concentrate on performing a comparative study of imputation methods with respect to another well known distribution of missingness, namely: MAR (missing at random).

ACKNOWLEDGMENTS

The authors acknowledge the Brazilian Research Agencies CAPES, CNPq, and FAPESP for their financial support.

REFERENCES

[1] D. Pyle, *Data Preparation for Data Mining (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1999.

[2] J. L. Schafer, *Analysis of Incomplete Multivariate Data*. CRC, 2000.

[3] O. G. Troyanskaya, M. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays." *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[4] K.-Y. Kim, B.-J. Kim, and G.-S. Yi, "Reuse of imputed data in microarray analysis increases imputation efficiency," *BMC Bioinformatics*, vol. 5, p. 160, 2004.

[5] L. P. Brás and J. C. Menezes, "Improving cluster-based missing value estimation of dna microarray data," *Biomolecular Engineering*, vol. 24, pp. 273–282, 2007.

[6] E. R. Hruschka, A. J. T. Garcia, E. R. H. Jr., and N. F. F. Ebecken, "On the influence of imputation in classification: practical issues," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 21, pp. 43–58, 2009.

[7] J. de Andrade Silva and E. R. Hruschka, "An evolutionary algorithm for missing values substitution in classification tasks," in *4th International Conference on Hybrid Artificial Intelligence Systems (HAIS-09)*, ser. Lecture Notes in Artificial Intelligence, vol. 5572. Berlin: Springer-Verlag, 2009, pp. 237–247.

[8] E. R. Hruschka, E. R. H. Junior, and N. F. F. Ebecken, "Towards efficient imputation by nearest-neighbors: A clustering based approach," in *17th Australian Joint Conference on Artificial Intelligence (AI'04)*, ser. Lecture Notes in Artificial Intelligence, vol. 3339. Berlin: Springer-Verlag, 2004, pp. 513–525.

[9] I. Kononenko and I. B. E. Roskar, "Experiments in automatic learning of medical diagnostic rules." Jozef Stefan Institute, Tech. Rep., 1984.

[10] W. Pedrycz, *Knowledge-Based Clustering: From Data to Information Granules*. Wiley-Interscience, 2005.

[11] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro, "Evolving clusters in gene-expression data," *Information Sciences*, vol. 176, no. 13, pp. 1898–1927, 2006.

[12] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic Press, Inc., 1973.

[13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[14] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal Machine Learning Research*, vol. 7, pp. 1–30, 2006.