Neural Networks 23 (2010) 406–418

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A study on the use of imputation methods for experimentation with Radial Basis Function Network classifiers handling missing attribute values: The good synergy between RBFNs and EventCovering method☆

Julián Luengo [a,*], Salvador García [b], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, CITIC-University of Granada, 18071, Granada, Spain

[b] Department of Computer Science, University of Jaen, 23071, Jaen, Spain

ARTICLE INFO

Article history:
Received 8 February 2009
Received in revised form 17 June 2009
Accepted 19 November 2009

Keywords:
Classification
Imputation methods
Missing values
Radial Basis Function Networks

ABSTRACT

The presence of Missing Values in a data set can affect the performance of a classifier constructed using that data set as a training sample. Several methods have been proposed to treat missing data and the one used more frequently is the imputation of the Missing Values of an instance.

In this paper, we analyze the improvement of performance on Radial Basis Function Networks by means of the use of several imputation methods in the classification task with missing values. The study has been conducted using data sets with real Missing Values, and data sets with artificial Missing Values. The results obtained show that EventCovering offers a very good synergy with Radial Basis Function Networks. It allows us to overcome the negative impact of the presence of Missing Values to a certain degree.

1. Introduction

Real data are not usually perfect – they contain wrong data, incomplete or vague (Pyle, 1999). Hence, it is usual to find missing data in most of the information sources used. There are two main reasons why an attribute value is missing: either the value was lost (e.g. it was erased) or the value was not important. The detection of incomplete data is easy in most cases: we can look for null values in the data set. However, this is not always true, since Missing Values (MV) can appear with the form of outliers or even wrong data (i.e. out of boundaries) (Pearson, 2005).

Missing data is a common problem in statistical analysis (Little & Rubin, 1987). Rates of missing data less than 1% are generally considered trivial, 1%–5% manageable. However, a rate of 5%–15% requires sophisticated methods to handle, and more than 15% may have severe impact on any kind of interpretation and harm the model's results.

Missing data treatment should be carefully thought through, otherwise bias might be introduced into the knowledge induced. Depending on the way MVs have been produced, our approach will be different. Several methods have been proposed in the literature to treat missing data (Acuna & Rodriguez, 2004; Batista & Monard, 2003). Missing data can be treated in three different ways (Li, Deogun, Spaulding, & Shuart, 2004):

- The first approach is to discard the examples with missing data in their attributes. Moreover, the case of deleting attributes with elevated levels of missing data is included in this category too.
- Other approach is the use of maximum likelihood procedures, where the parameters of a model for the complete data are estimated, and used later for impute by means of sampling.
- Finally, the imputation of MVs is a class of procedures that aims to fill in the MVs with estimated ones. In most cases, data sets attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined. This is the most commonly used approach.

Missing data has a similar impact on neural networks as it does on other types of classification algorithms, such as K-Nearest Neighbour. These similarities include variance underestimation, distribution distortion, and correlation depression. As Kros, Lin, and Brown (2006) states: "By training the network with cases containing complete data only, the internal weights developed with this type of training set cannot be accurately applied to a test set containing missing values later", and has been deeper studied in Markey, Tourassi, Margolis, and DeLong (2006). So we must impute both training and test data with the same method.

We assume that the MVs are well specified, and that we know where they appear. The study has been conducted using data sets

☆ This work was supported by the Project TIN2008-06681-C06-01. J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science.

* Corresponding author. Tel.: +34 958240598; fax: +34 958243317.
E-mail addresses: julianlm@decsai.ugr.es (J. Luengo), sglopez@ujaen.es (S. García), herrera@decsai.ugr.es (F. Herrera).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.11.014

with real MVs and data sets with artificial MVs. There is a wide family of imputation methods, from mean imputation to those which analyze the relationships between attributes. We analyze the use of different imputation strategies versus deletion case and the total lack of missing data treatment.

We will focus our attention on Radial Basis Function Networks (RBFNs), in the task of classification with MVs and the use of imputation methods for them. To this end, we present a snapshot of the state-of-the-art in the literature about MVs and Artificial Neural Networks. Moreover, a specific analysis of imputation methods for the RBFN models is the main contribution, where the EventCovering method stands out. From the results obtained we can observe that:

- Imputation methods produce significant improvements in the RBFNs results.
- A statistical analysis confirms a good synergy of RBFNs with the EventCovering method (Wong & Chiu, 1987).

The rest of the paper is organised as follows. In Section 2, we describe the imputation methods that we have used in the study and the RBFN models. Section 3 describes the experimental framework, along with the results and their analysis. Finally, in Section 4 we discuss our conclusions.

2. Preliminaries: Missing values, imputation methods and their use in Neural Networks

In this section we briefly introduce the missing data randomness, describe the imputation methods used in the study, their parameters, and present a short review on the use of imputation methods for Neural Networks. We also describe the RBFN models used in this paper.

2.1. Randomness of missing data

Depending on the reason why MVs have been produced, our approach to handle them will be different. Missing data randomness can be divided into three classes, as proposed by Little and Rubin (1987):

- Missing completely at random (*MCAR*). This is the highest level of randomness. It occurs when the probability of an instance (case) having a missing value for an attribute depends neither on the known values nor on the missing data.
- Missing at random (*MAR*). When the probability of an instance having a missing value for an attribute may depend on the known values, but not on the value of the missing data itself.
- Not missing at random (*NMAR*). When the probability of an instance having a missing value for an attribute could depend on the value of that attribute.

It is important to state the randomness of the MVs, since it will allow the use of imputation methods. In particular, we only consider *MCAR* and *MAR* situations (see Section 3.1), which are consistent with imputation techniques.

2.2. Description of imputation methods

In this subsection, we briefly describe the imputation methods that we have used. Imputation methods replace MVs with estimated values based on information available in the data set. There are many options varying from simplistic methods such as the mean imputation, to more robust methods based on relationships among attributes.

A short description of some widely used imputation methods which we have employed follows.

- Do Not Impute (*DNI*). As its name indicates, all the missing data remain un-replaced, so the networks must use their default MVs strategies. We want to verify whether imputation methods allow the Neural Networksto perform better than using the original data sets. As a guideline, we find in Grzymala-Busse and Hu (2000) a previous study of imputation methods. However, no Machine Learning method is used after the imputation process.
- Case deletion or Ignore Missing (*IM*). Using this method, all instances with at least one MV are discarded from the data set.
- Global Most Common Attribute Value for Symbolic Attributes, and Global Average Value for Numerical Attributes (*MC*) (Grzymala-Busse & Goodwin, 2005). This method is very simple: for nominal attributes, the MV is replaced with the most common attribute value; numerical values are replaced with the average of all values of the corresponding attribute.
- Concept Most Common Attribute Value for Symbolic Attributes, and Concept Average Value for Numerical Attributes (*CMC*) (Grzymala-Busse & Goodwin, 2005). As stated in *MC*, we replace the MV by the most repeated one if nominal or the mean value if numerical, but considering only the instances with same class as the reference instance.
- Imputation with K-Nearest Neighbor (*KNNI*) (Batista & Monard, 2003). Using this instance-based algorithm, every time we find a MV in a current instance, we compute the k nearest neighbors and impute a value from them. For nominal values, the most common value among all neighbors is taken, and for numerical values we will use the average value. Indeed, we need to define a proximity measure between instances. We have chosen Euclidean distance (it is a case of a L_p norm distance), which is usually used.
- Weighted imputation with K-Nearest Neighbor (*WKNNI*) (Troyanskaya et al., 2001). The Weighted K-Nearest Neighbor method selects the instances with similar values (in terms of distance) to a considered one, so it can impute as *KNNI* does. However, the estimated value now takes into account the different distances to the neighbors, using a weighted mean or the most repeated value according to the distance.
- K-means Clustering Imputation (*KMI*) (Li et al., 2004). Given a set of objects, the overall objective of clustering is to divide the data set into groups based on similarity of objects, and to minimize the intra-cluster dissimilarity. In K-means clustering, the intra-cluster dissimilarity is measured by the addition of distances among the objects and the centroid of the cluster which they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Once the clusters have converged, the last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken as nearest neighbors of each other, and we apply a nearest neighbor algorithm to replace missing data, in a way similar to that of K-Nearest Neighbor Imputation.
- Imputation with Fuzzy K-means Clustering (*FKMI*) (Acuna & Rodriguez, 2004; Li et al., 2004). In fuzzy clustering, each data object x_i has a membership function which describes the degree which this data object belongs to a certain cluster v_k. In the process of updating membership functions and centroids, we take into account only complete attributes. In this process, we cannot assign the data object to a concrete cluster represented by a cluster centroid (as done in the basic K-mean clustering algorithm), because each data object belongs to all K clusters with different membership degrees. We replace non-reference attributes for each incomplete data object x_i based on the information about membership degrees and the values of cluster centroids.

- Support Vector Machines Imputation (*SVMI*) (Feng, Chen, Yin, Yang, & Chen, 2005) is a SVM regression based algorithm to fill in missing data, i.e. set the decision attributes (output or classes) as the condition attributes (input attributes) and the condition attributes as the decision attributes, so we can use SVM regression to predict the missing condition attribute values. In order to do that, first we select the examples in which there are no missing attribute values. In the next step we set one of the condition attributes (input attribute), some of those values are missing, as the decision attribute (output attribute), and the decision attributes as the condition attributes by contraries. Finally, we use SVM regression to predict the decision attribute values.
- EventCovering (*EC*) (Wong & Chiu, 1987). Based on the work of Wong et al., a mixed-mode probability model is approximated by a discrete one. First, they discretize the continuous components using a minimum loss of information criterion. Treating a mixed-mode feature n-tuple as a discrete-valued one, the authors propose a new statistical approach for synthesis of knowledge based on cluster analysis. This method has the advantage of requiring neither scale normalization nor ordering of discrete values. By synthesis of the data into statistical knowledge, they refer to the following processes: (1) synthesize and detect from data inherent patterns which indicate statistical interdependency; (2) group the given data into inherent clusters based on these detected interdependency; and 3) interpret the underlying patterns for each clusters identified. The method of synthesis is based on author's *event–covering* approach. With the developed inference method, we are able to estimate the MVs in the data.
- Regularized Expectation-Maximization (*EM*) (Schneider, 2001). Missing values are imputed with a regularized expectation maximization (EM) algorithm. In an iteration of the EM algorithm, given estimates of the mean and of the covariance matrix are revised in three steps. First, for each record with missing values, the regression parameters of the variables with missing values on the variables with available values are computed from the estimates of the mean and of the covariance matrix. Second, the missing values in a record are filled in with their conditional expectation values given the available values and the estimates of the mean and of the covariance matrix, the conditional expectation values being the product of the available values and the estimated regression coefficients. Third, the mean and the covariance matrix are re-estimated, the mean as the sample mean of the completed data set and the covariance matrix as the sum of the sample covariance matrix of the completed data set and an estimate of the conditional covariance matrix of the imputation error. The EM algorithm starts with initial estimates of the mean and of the covariance matrix and cycles through these steps until the imputed values and the estimates of the mean and of the covariance matrix stop changing appreciably from one iteration to the next.
- Singular Value Decomposition Imputation (*SVDI*) (Troyanskaya et al., 2001). In this method, we employ singular value decomposition to obtain a set of mutually orthogonal expression patterns that can be linearly combined to approximate the values of all attributes in the data set. In order to do that, first we estimate the MVs with the *EM* algorithm, and then we compute the Singular Value Decomposition and obtain the eigenvalues. Now we can use the eigenvalues to apply a regression over the complete attributes of the instance, to obtain an estimation of the MV itself.
- Bayesian Principal Component Analysis (*BPCA*) (Oba et al., 2003). This method is an estimation method for missing values, which is based on Bayesian principal component analysis. Although the methodology that a probabilistic model and latent variables are estimated simultaneously within the framework of Bayes inference is not new in principle, the actual BPCA implementation that makes it possible to estimate arbitrary missing variables is new in terms of statistical methodology. The missing value estimation method based on BPCA consists of three elementary processes. They are (1) principal component (PC) regression, (2) Bayesian estimation, and (3) an expectation maximization (EM)-like repetitive algorithm.

Table 1
Methods parameters.

Method	Parameter
SVMI	Kernel = RBF $C = 1.0$ Epsilon = 0.001 Shrinking = No
KNNI, WKNNI	$K = 10$
KMI	$K = 10$ Iterations = 100 Error = 100
FKMI	$K = 3$ Iterations = 100 Error = 100 $m = 1.5$
EC	$T = 0.05$
EM	Iterations = 30 Stagnation tolerance = 0.0001 Inflation factor = 1 Regression type = multiple ridge regression
SVDI	Iterations = 30 Stagnation tolerance = 0.005 Inflation factor = 1 Regression type = multiple ridge regression Singular vectors = 10

A more extensive and detailed description of this method can be found in the web page http://sci2s.ugr.es/MVDM, and a PDF file with the original source paper descriptions is present in the web page formerly named "Imputation of Missing Values. Methods' Description".

There are more specialized methods, some derived from Bioinformatics. In the reviews of Farhangfar, Kurgan, and Pedrycz (2004); Grzymala-Busse and Hu (2000); Schafer and Graham (2002) we can find a good compilation of imputation methods that are not considered in this study due to their specialization.

2.3. Parameters used

In Table 1 we show the parameters used by each imputation method which has been used in this work (in the case of the method would use them). The values chosen are recommended by their respective authors.

2.4. A short review on the use of imputation methods for Neural Networks

We can find a study of the influence of MVs on Neural Networks in Ennett, Frize, and Walker (2001), where the MVs were replaced with "normal" values (i.e. replaced by zero) as well. In Yoon and Lee (1999) a specific method for training Neural Networkswith incomplete data was proposed, called Training-Estimation-Training (train with complete instances, impute the MV with the network, and train with the whole data set).

Besides, it is possible to find some work in areas related to Neural Networks. In Lim, Leong, and Kuan (2005) the authors propose a hybrid Neural Networks, in which the missing values are replaced with four Fuzzy C-Means technique based strategies.

After data set completion, the FAM module is applied. Self Organizing Maps (SOMs) are not capable of handling MV as Wang (2003) states. He proposes a SOM-based fuzzy map model for data mining with incomplete data. This model has two key components: translation of observations with missing data into fuzzy observations, and histogram-style fuzzy maps. This is not an imputation scheme, but a method which is capable of handling MVs for itself. With SVMs, Pelckmans, De Brabanterb, Suykensa, and De Moor (2005) contemplates an alternative approach where no attempt is made to reconstruct the values which are missing, but only the impact of the missingness on the outcome and the expected risk of the SVM is modeled explicitly. This is possible only when MVs are MCAR.

Artificial Neural Networks have been used as imputation methods as well in some specific applications. In Sun and Kardia (2008), the authors employ an Artificial Neural Networks based method for imputing the MVs artificially generated over genotype data. Pisoni, Pastor, and Volta (2008) also use Artificial Neural Networks for interpolating missing satellite data. A comparison between Auto-associative Neural Networks with genetic algorithm combination, and a variant of the Expectation-Maximization algorithm can be found in Nelwamondo, Mohamed, and Marwala (2007). Mileva-Boshkoska and Stankovski (2007) employ Radial Basis Function Networks for imputing ozone concentrations and SVMs as well.

2.5. Radial Basis Function Networks: Short outlook and description

The RBF architecture (Musavi, Ahmed, Chan, Faris, & Hummels, 1992) consists of a simple two layered network (a hidden layer and an output layer), each layer is fully connected to the one following.

As we have mentioned, the hidden layer is composed of a number of nodes, RBF nodes, with radial activation functions, which shall be taken, in this analysis, as Gaussian functions. Two parameters are associated with each RBF node, the "centre" and "width". Both of these quantities refer to properties of the Gaussian function. Associated with the hidden to output connections, are conventional signal multipliers: the weights. The final output processing unit merely yields a weighted sum of its inputs.

Their use in the literature is extensive, and its application varies from face recognition (Er, Wu, Lu, & Hock-Lye, 2002) to time series prediction (Harpham & Dawson, 2006). The RBFNs are under continuously research, so we can find abundant literature about extensions and improvements of RBFNs learning and modeling (Billings, Wei, & Balikhin, 2007; Ghodsi & Schuurmans, 2003; Lázaro, Santamaría, & Pantaleón, 2003; Wallace, Tsapatsoulis, & Kollias, 2005; Wei & Amari, 2008). Recently, we can find some work analyzing the behavior of RBFNs (Eickhoff & Ruckert, 2007; Liao, Fang, & Nuttle, 2003; Yeung, Ng, Wang, Tsang, & Wang, 2007) and improving their efficiency (Arenas-Garcia, Gomez-Verdejo, & Figueiras-Vidal, 2007; Schwenker, Kestler, & Palm, 2001). As we can see from the recent and past literature, we can conclude that RBFNs are a widely employed and well-known model which is actually used. Regarding MVs treatment in RBFNs there are some contributions, using the RBFNs to predict the MVs (Uysal, 2007) or obtaining the Radial Basis Function from a Vector Quantization of the data set with MVs (Lendasse, Francois, Wertz, & Verleysen, 2005). Also, the impact of MVs in the RBFNs has been considered in Morris, Boddy, and Wilkins (2001), but only in a case study.

The experimentation in this paper is conducted by using the following models of RBFNs:

- Radial Basis Function Network (*RBFN*) (Broomhead & Lowe, 1988; Buhmann, 2003). It is well suited for function approximation and pattern recognition due to its simple topological structure and its ability to reveal how learning proceeds in an explicit manner. A RBF is a function which has been built into a distance criterion with respect to a centre. Different basis functions like thin-plate spline functions, multiquadratic functions, inverse multiquadratic functions and Gaussian functions have been proposed for the hidden-layer neurons, but normally the selected one is the Gaussian function. Compared with other types of Artificial Neural Networks (ANNs), such as feed-forward networks, the RBFN requires less computation time for learning and also has a more compact topology. RBFs have been applied in the area of ANNs where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in multi-layer perceptrons. The original RBF method has been traditionally used for strict multivariate function interpolation (Powell, 1987) and for this fact, it requires as many RBF neurons as data points. Broomhead and Lowe (1988) removed this strict interpolation restriction and provided a neural network architecture where the number of RBF neurons can be far less than the data points. A RBFN mainly consists of two layers, one hidden-layer and one output layer.

  Each input example x is applied to all hidden neurons. The neuron i computes the function

$$h_i(x) = \exp\left[\frac{(x-u_i)^2}{2\sigma_i^2}\right] \tag{1}$$

  where u_i is the center of the neuron i, and h_i the output of such neuron. The RBFN has only one output (i.e. j has only one value), and it is defined by

$$z(x) = \frac{\sum_i^k h_i(x) w_i}{\sum_i^k h_i(x)}. \tag{2}$$

  The term w_i refers to the neuron weight, and k is the number of neurons in the hidden layer. In order to initialize the neurons in the network, we use a K-means clustering algorithm. The centre of the neuron is set equal to the centroid of the cluster, and its radius equal to the mean of the distance between the given center and the $N = 2$ nearest neurons:

$$\sigma_i = \sum_{j=1}^{N} \frac{d(u_i, u_j)}{N}. \tag{3}$$

  The network is initialized with a K-means clustering algorithm. In a similar fashion to the imputation method KMI, we set the number of clusters equal to the number of neurons. The initial centroids are set to random chosen examples, which are all different. By successive iterations, the neurons are adjusted using the Euclidean distance until the centroids (i.e. the neurons) do not change.

  Once the centers and radius of the neurons has been initialized, the output's weight matrix can be optimized by means of supervised training. For each train example x_i and expected output t_i, we compute the output of the hidden layer's neurons, the vector h. Next, we compute the output of the network y and compare it with the expected output t, and adjust each weight in w to reduce the Mean Square Error (MSE) with the Least Mean Squares algorithm (LMS). This method implies the use of gradient descent (delta rule) and adjusting the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t_j - y_j)h_i \tag{4}$$

  where η is the learning rate ($\eta \ll 1.0$). This process is repeated for each train example, until the max iteration limit is reached. The center and radius of the neurons is adjusted as well to minimize the output error. We use the error derivative with respect to these parameters, in a fashion similar to that of

backpropagation. With $i = 1, \ldots, m$ hidden neurons, $j = 1, \ldots, p$ inputs and $k = 1$ output, we update both parameters simultaneously from iteration n to $n + 1$ with:

$$u_{ij}(n+1) = u_ij(n) + \eta_c \frac{\left[\sum_k (t_k - y_k) w_{ik}\right] h_i (x_j - u_{ij}(n))}{(\sigma_i(n))^2} \quad (5)$$

$$\sigma_i(n+1) = \sigma_i(n) + \eta_\sigma \frac{\left[\sum_k (t_k - y_k) w_{ik}\right] h_i \parallel x - u_i(n) \parallel^2}{(\sigma_i(n))^3}. \quad (6)$$

The number of hidden-layer neurons is defined by the user a priori. In our study, we have fixed the number of neurons at 50. The η is set to 0.3. Both η_c and η_σ are set to $\frac{1}{maxIterations}$. The parameter *maxIterations* denote the maximum number of iterations of the network training algorithm, and is established to 10.

- RBFN Decremental (*RBFND*) (Yingwei, Sundararajan, & Saratchandran, 1997). In the classical approach described above, the number of hidden units is fixed a priori based on the properties of input data. A significant contribution that overcomes these drawbacks was made through the development of an algorithm that adds hidden units to the network based on the novelty of the new data. One drawback of this approach is that once a hidden unit is created, it can never be removed. The authors have proposed an algorithm that adopts the basic idea of a pruning strategy. The pruning strategy removes those hidden neurons which consistently make little contribution to the network output. Pruning becomes imperative for the identification of nonlinear systems with changing dynamics, because failing to prune the network in such cases will result in numerous inactive hidden neurons, being present as the dynamics which cause their creation initially, become nonexistent. If inactive hidden units can be detected and removed while learning proceeds, a more well-suited network topology can be constructed. Also, when the neural networks are employed for control, the problem of over-parametrization should be avoided.

  The network initialization is the same that the simple RBFN model described previously: We use a K-means algorithm to set the initial position of the neurons in the problem space. Next, we apply an initial adjust with the LMS algorithm explained in the RBFN model with 10 iterations. This procedure allow us to obtain a set of initial weights. We compute the mean $\overline{w_j}$ of the weights from all neurons for the output j as

$$\overline{w_j} = \frac{1}{N} \sum_{i=1}^{N} w_{ij}. \quad (7)$$

  Since we are using the RBFN network for classification problems, only one output is considered.

  The pruning will be applied over the neurons with a low weight compared to the computed mean. For every neuron, if its weight w_{ij} is lower than the threshold $\rho \times \overline{w_j}$, the neuron is removed from the network. The threshold ρ should be a low value. A high value will delete almost all neurons of the net, and prevent the algorithm from finish.

  Now we apply gradient descendent on the remaining neurons over u_{ij} and w_{ij}. This will re-adjust the neurons positions and their weights, trying to fill up the gaps produced by the pruning strategy.

  These combined pruning–adjusting steps are repeated over the data set and the network. When the network stays with the same neurons for λ iterations, the network training finishes.

  In our study, we provide this model with 50 initial neurons, a ρ percentage of 0.1 under the average of the weights used to decide whether a neuron must be removed or not, and a learning factor $\alpha = 0.3$ of the Least Mean Square (LMS) algorithm for adjusting the neurons. The limit iteration λ to achieve convergence has been empirically set to 15.

- RBFN Incremental (*RBFNI*) (Plat, 1991). This approach builds a RBFN composed of one hidden layer and one output layer. This topography is similar to non-incremental RBFN's one, but we do not know the number of neurons of the hidden layer. This idea is similar to RBFN Decremental, but in this model we will not set any limit to the hidden layer's neurons number. We use the Resource-Allocating Network (RAN) algorithm, which consists of a network, a strategy for allocating new units, and a learning rule for refining the network. The units on the first layer store a particular region in the input space. When the input moves away from the stored region the response of the unit decreases. As we mentioned in the RBFN model, we employ a Gaussian function to achieve this behaviour. The Eq. (1) is used to obtain the response of a single neuron, and (2) describes how to compute the network output. In Plat (1991) the author employs a default output γ parameter, which is added to the result of Eq. (1), but we do not apply it to the classification problem.

  The network starts with a blank slate: no patterns are yet stored. As patterns are presented to it, the network chooses to store some of them. At any given point the network has a current state, which reflects the patterns that have been stored previously. The allocator identifies a pattern that is not currently well represented by the network and allocates a new unit that memorizes the pattern. The output of the new unit extends to the second layer. After the new unit is allocated, the network output is equal to the desired output y. Let the index of this new unit be n.

  The peak (center) of the response of the newly allocated unit is set to the novel input example x_i:

$$u_n = x_i. \quad (8)$$

  The weight associated to this neuron to the output layer is set to the difference between the output of the network and the novel output,

$$w_n = y - z(x_i). \quad (9)$$

  The width of response (the neuron's radius) of the new unit is proportional to the distance from the nearest stored neuron to the novel input vector,

$$\sigma_n = \kappa \parallel x_j - u_{nearest} \parallel \quad (10)$$

  where κ is an overlap factor. As κ grows larger, the responses of the units overlap more and more. The RAN uses a two-part novelty condition. An input–output pair (x_j, y_j) is considered novel if the input is far away from existing centers,

$$\parallel x_j - u_{nearest} \parallel > \delta(t) \quad (11)$$

  and if the difference between the desired output and the output of the network is large

$$\parallel y_j - z(x_j) \parallel > \epsilon. \quad (12)$$

  Typically ϵ is a desired accuracy of output of the network. Errors larger than ϵ are immediately corrected by the allocation of a new unit, while errors smaller than ϵ gradually minimized using gradient descent. The distance $\delta(t)$ is the scale of resolution that the network is fitting at the tth input presentation. The learning starts with $\delta(t) = \delta_{\max}$; which is the largest length scale of interest, typically the size of the entire input space of non-zero probability density. The distance $\delta(t)$ shrinks until the it reaches $\delta_{\min}$, which is the smallest length scale of interest. The network will average over features that are smaller than $\delta_{\min}$. We used a function:

$$\delta(t) = \max(\delta_{\max} \exp(-t/\tau), \delta_{\min}) \quad (13)$$

Table 2
Data sets used for experimentation.

Data set	Abbreviation	# Ex.	# Atts.	# Classes	% MV	% Ex. with MV
Horse-colic	HOC	368	24	2	21.82	98.10
Bands	BAN	540	40	2	4.63	48.70
Hepatitis	HEP	155	20	2	5.39	48.39
House-Votes-84	HOV	434	17	2	5.30	46.54
Mammographic	MAM	961	6	2	2.81	13.63
Mushroom	MUS	8124	23	2	1.33	30.53
Autos	AUT	205	26	6	1.11	22.44
Crx	CRX	689	16	2	0.61	5.37
Post-operative	POP	90	9	3	0.37	3.33
Breast	BRE	286	10	2	0.31	3.15
Wisconsin	WIS	699	10	2	0.23	2.29
Cleveland	CLE	303	13	5	0.14	1.98
Iris+MV	IRI	150	4	3	10.00	32.67
Pima+MV	PIM	768	9	2	10.00	50.65
Wine+MV	WIN	178	14	3	10.00	70.22
Australian+MV	AUS	690	15	2	10.00	70.58
New-thyroid+MV	NTH	215	6	3	10.00	35.35
Ecoli+MV	ECO	336	8	8	10.00	48.21
Satimage+MV	SAT	6435	37	7	10.00	87.80
German+MV	GER	1000	21	2	10.00	80.00
Magic+MV	MAG	1902	11	2	10.00	58.20
Shuttle+MV	SHU	2175	10	7	10.00	55.95

where τ is a decay constant. At first, the system creates a coarse representation of the function, then refines the representation by allocating units with smaller and smaller widths. Finally, when the system has learned the entire function to the desired accuracy and length scale, it stops allocating new units altogether.

The two-part novelty condition is necessary for creating a compact network. If only condition (11) is used, then the network will allocate units instead of using gradient descent to correct small errors. If only condition (12) is used, then fine-scale units may be allocated in order to represent coarse-scale features, which is wasteful. By allocating new units the RAN eventually represents the desired function ever more closely as the network is trained. Fewer units are needed for a given accuracy if the hidden-layer outputs $h_i(x_j)$, the output-level outcome $z(x_i)$, and the thresholds γ_i are adjusted to decrease the error:

$$\Xi = \| y_j - z(x_j) \|^2 . \tag{14}$$

We use the Widrow-Hoff LMS algorithm to decrease the error whenever a new unit is not allocated, as Yingwei states:

$$\Delta z(x_j) = \alpha(y_j - z(x_j))x_j. \tag{15}$$

In addition, we adjust the centers of the responses of units to decrease error:

$$\Delta u_j = 2\frac{\alpha}{\sigma_j}(x_k - u_j)h_j(x_k)[(y_j - z(x_k)) \cdot w_j]. \tag{16}$$

Eq. (16) is derived from the gradient descent and Eq. (1). Eq. (16) also has an intuitive interpretation. Units whose outputs would cancel the error have their centers pulled towards the input. Units whose outputs would increase the error have their centers pushed away from the input.

This method tries to find the correct number of neurons for a given data set, without an initial limitation such as in the RBFN model (which has its neurons number fixed) and the RBFN Decremental model (which also has a maximum number of neurons fixed a priori). However, if δ is too low, we can find that our network overfits the training data. The model that we have used is set with $\alpha = 0.3$, $\delta_{\max} = 0.5$, $\delta_{\min} = 0.1$ and $\epsilon = 0.1$. κ is set to 1.

The parameters of the RBFN models have been empirically estimated in order to optimize the performance of RBFN in classification problems analyzed, without tuning them individually for each data set.

3. Experimental study: Imputation methods

In this section we describe the experiments we have performed. First, the data sets used and the setup of our experiments are described. Next, the obtained results are shown with an analysis of them. Finally, a specific statistical study on the behavior of EventCovering method is done. We include a graphical study of this EventCovering analysis.

3.1. Experimentation framework

We have selected a group of 22 data sets taken from the UCI repository (Asuncion & Newman, 2007). In Table 2, we summarize their properties. The column labeled as "% MV" indicates the percentage of all values of the data set which are missing. The column labeled as "% Ex. with MV" refers to the percentage of examples in the data set which have at least one MV.

Their origin is described as follows:

- We have selected 12 data sets which have MVs in a "natural" way. The percentage range of MVs varies from 20% to 0.1%. We cannot know anything about the randomness of MVs in the first 12 data sets, so we assume they are distributed in a *MAR* way.
- We have used 10 classical data sets with induced MVs (last ten rows in the Table 2). We have generated a 10% of the data set values as MVs in the training partition "artificially" in a *MCAR* way. The reason for inducing MVs only in the training partition is that we only want to discard information in the training stage, and affect the test task the least. With this configuration we can see how effective is the imputation to the unaffected instances of test.

For the experiments, we have used 10-fold cross validation. Since the training of the RBFNs is not deterministic, we have repeated each experiment 5 times per partition, each time with different seed, also randomly generated too. At the end, we have 50 executions of the model with each data set and imputation method. The imputation method uses the training partition to infer the required knowledge and relations between attributes. Then it is applied in both training and test partitions. Therefore, the test partition in each case is not used to impute the MVs.

Table 3
RBFN test accuracy

	AUT	BRE	CLE	CRX	WIS	BAN
IM	32.78 ± 10.03	**71.92 ± 7.84**	34.14 ± 10.59	66.09 ± 7.15	97.12 ± 2.50	69.41 ± 7.21
EC	41.20 ± 8.33	71.34 ± 6.45	**46.92 ± 13.01**	**84.54 ± 4.54**	**97.43 ± 2.65**	68.11 ± 4.92
KNNI	29.42 ± 8.09	70.72 ± 7.21	33.68 ± 10.68	67.18 ± 7.34	96.97 ± 2.76	71.93 ± 5.14
WKNNI	29.06 ± 7.09	69.74 ± 7.96	33.84 ± 9.63	66.11 ± 6.19	96.91 ± 2.83	71.78 ± 5.00
KMI	29.96 ± 8.87	70.76 ± 6.55	34.07 ± 9.84	65.81 ± 5.89	96.85 ± 2.86	72.26 ± 5.77
FKMI	28.46 ± 7.06	71.12 ± 7.53	33.82 ± 9.55	66.61 ± 6.43	96.97 ± 2.85	73.00 ± 4.16
SVMI	30.11 ± 6.99	71.49 ± 6.95	33.09 ± 10.06	67.30 ± 7.21	96.91 ± 2.80	**73.41 ± 5.70**
EM	**43.32 ± 8.90**	71.39 ± 5.91	34.01 ± 10.13	64.22 ± 6.01	96.82 ± 2.87	70.11 ± 4.64
SVDI	42.07 ± 9.74	71.58 ± 6.87	34.65 ± 10.22	62.10 ± 6.97	97.00 ± 2.70	70.63 ± 5.21
BPCA	40.41 ± 7.29	69.63 ± 4.65	26.32 ± 6.52	51.97 ± 6.33	45.38 ± 3.50	56.96 ± 3.79
MC	30.32 ± 7.67	70.39 ± 6.96	35.64 ± 10.54	66.90 ± 6.68	97.03 ± 2.78	72.41 ± 4.29
CMC	29.29 ± 7.37	71.63 ± 7.24	36.23 ± 10.60	66.71 ± 6.02	96.94 ± 2.83	71.89 ± 4.98
DNI	27.73 ± 7.72	71.06 ± 7.88	34.17 ± 10.51	66.81 ± 6.48	96.85 ± 2.89	71.70 ± 5.57
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	95.45 ± 4.40	79.62 ± 5.53	68.81 ± 11.22	**81.71 ± 8.99**	**99.63 ± 0.39**
EC	**79.59 ± 7.64**	95.15 ± 3.93	**82.42 ± 4.98**	70.00 ± 9.75	78.03 ± 7.57	99.59 ± 0.26
KNNI	59.50 ± 6.10	94.88 ± 3.63	79.36 ± 5.19	70.22 ± 11.43	75.24 ± 7.97	99.54 ± 0.27
WKNNI	59.55 ± 5.69	95.07 ± 3.67	78.88 ± 4.53	69.33 ± 9.57	76.03 ± 7.91	99.45 ± 0.49
KMI	59.13 ± 6.54	95.06 ± 4.04	79.15 ± 4.94	69.78 ± 10.43	77.25 ± 5.22	99.56 ± 0.30
FKMI	59.33 ± 6.10	94.97 ± 3.54	79.76 ± 4.80	68.89 ± 10.66	77.67 ± 4.52	99.57 ± 0.28
SVMI	59.06 ± 6.72	**96.12 ± 3.48**	80.98 ± 4.96	**70.44 ± 9.06**	79.89 ± 6.85	99.53 ± 0.29
EM	60.22 ± 6.90	91.86 ± 5.81	79.76 ± 5.59	68.67 ± 12.11	75.71 ± 5.55	99.44 ± 0.36
SVDI	59.84 ± 5.58	91.77 ± 5.49	79.34 ± 4.78	69.78 ± 10.90	75.13 ± 6.79	99.43 ± 0.34
BPCA	57.71 ± 6.27	49.51 ± 6.01	50.59 ± 5.39	56.44 ± 17.05	79.37 ± 2.25	50.84 ± 0.68
MC	60.17 ± 5.49	95.20 ± 3.72	79.07 ± 4.57	69.11 ± 9.76	77.35 ± 5.49	99.57 ± 0.26
CMC	59.66 ± 6.12	96.02 ± 4.17	81.13 ± 4.77	67.56 ± 11.51	77.08 ± 7.82	99.59 ± 0.21
DNI	59.07 ± 6.05	95.20 ± 3.76	79.92 ± 4.32	68.44 ± 11.19	74.78 ± 6.52	99.44 ± 0.33
	IRI	PIM	WIN	AUS	NTH	ECO
IM	90.27 ± 10.35	71.57 ± 3.69	65.29 ± 10.26	65.88 ± 5.39	87.65 ± 5.36	19.94 ± 3.87
EC	**94.40 ± 6.16**	71.89 ± 4.76	**96.52 ± 4.01**	**85.71 ± 4.39**	**92.13 ± 4.26**	**51.43 ± 13.24**
KNNI	93.87 ± 7.04	**72.49 ± 5.40**	66.69 ± 10.49	66.75 ± 5.66	90.45 ± 5.96	21.73 ± 3.11
WKNNI	92.40 ± 7.54	72.38 ± 4.83	65.65 ± 9.13	66.75 ± 4.85	88.40 ± 7.10	20.78 ± 3.67
KMI	91.73 ± 8.39	71.51 ± 5.39	67.82 ± 8.83	65.91 ± 4.77	89.48 ± 6.48	22.87 ± 2.33
FKMI	86.13 ± 10.65	71.80 ± 4.97	65.13 ± 10.77	66.29 ± 4.41	88.48 ± 6.55	22.02 ± 3.56
SVMI	94.40 ± 5.23	71.57 ± 4.67	69.79 ± 9.19	67.13 ± 4.85	90.82 ± 5.36	22.40 ± 3.38
EM	89.73 ± 8.45	71.24 ± 5.29	65.41 ± 9.33	63.36 ± 4.45	89.68 ± 6.38	25.43 ± 8.58
SVDI	88.53 ± 10.59	71.68 ± 4.43	62.88 ± 8.86	61.19 ± 4.42	89.02 ± 6.45	23.49 ± 5.62
BPCA	33.47 ± 1.63	65.04 ± 4.67	28.10 ± 2.57	51.10 ± 5.54	65.15 ± 3.18	22.15 ± 2.58
MC	94.27 ± 5.81	71.94 ± 5.44	68.12 ± 5.94	66.41 ± 5.29	88.69 ± 5.93	22.23 ± 2.71
CMC	94.40 ± 5.87	71.62 ± 4.68	67.25 ± 7.98	66.55 ± 4.67	90.93 ± 5.74	21.91 ± 2.89
DNI	91.47 ± 7.66	71.85 ± 4.53	67.55 ± 10.20	66.29 ± 5.72	89.52 ± 6.10	20.68 ± 3.80
	SAT	GER	MAG	SHU		
IM	59.97 ± 4.77	62.54 ± 5.62	75.68 ± 3.37	88.81 ± 2.83		
EC	62.67 ± 4.69	**72.26 ± 3.39**	**77.13 ± 3.53**	**97.26 ± 3.00**		
KNNI	67.34 ± 2.73	67.52 ± 4.31	76.35 ± 2.85	90.26 ± 3.83		
WKNNI	67.00 ± 3.01	68.26 ± 3.75	76.10 ± 2.55	90.14 ± 3.62		
KMI	68.29 ± 2.95	67.96 ± 3.55	75.86 ± 3.37	89.77 ± 2.03		
FKMI	**69.68 ± 2.16**	68.24 ± 3.42	75.84 ± 3.02	89.30 ± 4.45		
SVMI	69.38 ± 2.74	67.44 ± 4.03	76.37 ± 2.74	89.45 ± 4.31		
EM	17.28 ± 8.76	68.52 ± 3.45	74.11 ± 3.31	82.54 ± 5.13		
SVDI	19.02 ± 8.15	68.12 ± 3.53	73.97 ± 3.01	82.06 ± 3.42		
BPCA	16.53 ± 0.79	64.12 ± 5.19	61.96 ± 1.54	62.90 ± 3.61		
MC	69.41 ± 2.83	67.92 ± 3.89	75.76 ± 2.58	87.81 ± 3.70		
CMC	69.46 ± 2.30	66.60 ± 3.54	76.39 ± 2.97	88.95 ± 4.33		
DNI	45.54 ± 5.22	67.36 ± 4.03	76.51 ± 2.97	86.50 ± 4.47		

3.2. Experiments and analysis

In this section we present the experimental results, and analyze them. We have summarized the percentage of well-classified instances in test. We have 50 runs of each problem (5 times per partition), the mean of these 50 experiments is shown as a representative value, and the standard deviations have been computed. We present the obtained results in Tables 3–5, a table per each RBFN model. The highest accuracy value is represented in bold, emphasizing the best method of a given data set.

From our study, we can point out the following:

- *DNI* (Do Not Impute) method is almost never the best method. This method informs us about the relevance of MV in the attributes.
- *IM* (Ignore instances with MVs) method has a very poor performance as well. Sometimes, the deletion of instances leads the RBFN method not to adjust itself to certain classes, since the information that describes them has been erased from the data set. In these situations, the results of the test accuracy are even poorer than the *DNI* method. However, data sets with a low percentage of MVs show a low disadvantage of this method from the rest.
- Simple methods as *MC* and *CMC* are competitive and can surpass more sophisticated methods. However, the use of these methods introduces bias in the data, thus the RBFN model can be penalized.
- The clustering methods based on K-Means and K-NN have an average performance, not always better than *DNI* or

Table 4
RBFN decremental test accuracy.

	AUT	BRE	CLE	CRX	WIS	BAN
IM	29.09 ± 12.13	62.01 ± 10.92	34.99 ± 13.70	59.32 ± 8.15	88.15 ± 11.11	56.03 ± 10.32
EC	**46.82 ± 13.71**	62.16 ± 10.42	**43.78 ± 15.53**	**76.39 ± 7.57**	**92.73 ± 6.03**	58.74 ± 8.94
KNNI	25.41 ± 11.22	62.89 ± 11.00	34.00 ± 11.47	59.72 ± 6.40	86.41 ± 12.50	60.41 ± 9.42
WKNNI	25.02 ± 9.51	64.50 ± 8.40	31.52 ± 11.34	59.16 ± 7.59	85.00 ± 16.21	60.70 ± 9.03
KMI	25.08 ± 10.49	62.53 ± 7.85	34.87 ± 12.51	59.29 ± 8.02	87.58 ± 11.75	62.81 ± 5.92
FKMI	25.18 ± 9.91	62.92 ± 9.49	33.61 ± 11.61	59.56 ± 7.86	88.27 ± 10.18	60.52 ± 8.45
SVMI	25.07 ± 9.77	64.54 ± 9.06	35.02 ± 12.41	60.79 ± 7.06	91.71 ± 5.46	62.67 ± 6.38
EM	33.87 ± 12.08	64.76 ± 8.90	34.90 ± 12.63	58.18 ± 7.29	91.13 ± 5.45	60.67 ± 7.06
SVDI	33.73 ± 12.35	62.95 ± 9.64	33.98 ± 11.91	59.81 ± 7.68	86.84 ± 10.99	63.00 ± 8.66
BPCA	29.87 ± 11.00	**70.99 ± 8.52**	33.08 ± 9.61	49.61 ± 6.23	46.33 ± 5.69	50.67 ± 6.97
MC	26.91 ± 11.38	62.84 ± 10.30	35.05 ± 11.31	57.11 ± 8.86	87.23 ± 10.15	60.44 ± 9.01
CMC	29.19 ± 9.42	65.19 ± 8.01	35.83 ± 11.44	59.46 ± 8.65	88.42 ± 11.98	**63.00 ± 8.41**
DNI	23.44 ± 11.43	62.86 ± 10.61	38.40 ± 12.30	60.42 ± 7.80	90.79 ± 8.53	59.30 ± 8.51
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	87.98 ± 10.74	78.15 ± 5.49	59.89 ± 16.53	**81.20 ± 10.05**	84.26 ± 15.62
EC	**66.80 ± 11.15**	88.07 ± 11.54	78.94 ± 5.29	58.00 ± 15.76	74.45 ± 11.95	86.59 ± 15.00
KNNI	60.98 ± 5.66	88.92 ± 9.33	76.49 ± 4.75	52.00 ± 17.84	62.01 ± 16.16	86.71 ± 13.28
WKNNI	59.46 ± 5.30	87.40 ± 13.08	76.34 ± 6.25	59.33 ± 16.72	65.09 ± 15.24	85.93 ± 13.89
KMI	58.58 ± 6.24	86.69 ± 10.86	76.49 ± 5.84	**60.89 ± 13.74**	63.74 ± 15.60	86.31 ± 10.61
FKMI	60.87 ± 6.01	**90.21 ± 9.65**	77.44 ± 5.22	56.67 ± 16.37	66.19 ± 12.51	84.14 ± 14.29
SVMI	58.55 ± 6.69	88.14 ± 12.66	77.55 ± 5.34	53.78 ± 17.69	71.37 ± 15.76	**88.40 ± 11.39**
EM	61.20 ± 4.73	83.37 ± 12.07	77.38 ± 5.46	58.44 ± 14.37	60.93 ± 18.23	85.56 ± 11.71
SVDI	60.75 ± 7.12	84.43 ± 12.08	77.61 ± 5.94	56.00 ± 14.22	62.68 ± 15.89	86.63 ± 12.34
BPCA	55.11 ± 7.50	50.99 ± 6.38	51.03 ± 4.71	46.67 ± 17.36	70.33 ± 12.54	50.81 ± 1.17
MC	60.72 ± 6.22	83.09 ± 16.46	76.55 ± 5.00	54.44 ± 18.36	64.97 ± 13.58	86.64 ± 12.25
CMC	59.41 ± 6.23	85.75 ± 14.19	**79.25 ± 4.74**	57.11 ± 16.48	69.64 ± 17.33	86.85 ± 13.80
DNI	58.86 ± 6.04	86.33 ± 11.81	76.82 ± 4.89	60.44 ± 15.74	66.67 ± 13.93	86.39 ± 13.15
	IRI	PIM	WIN	AUS	NTH	ECO
IM	92.93 ± 6.02	63.44 ± 9.37	66.84 ± 11.65	57.48 ± 6.02	77.30 ± 12.25	38.94 ± 11.65
EC	**94.40 ± 5.05**	**68.74 ± 5.31**	**89.75 ± 9.91**	**78.35 ± 10.44**	82.77 ± 16.19	43.38 ± 13.38
KNNI	93.73 ± 5.40	67.92 ± 6.36	66.89 ± 12.66	57.83 ± 7.57	80.01 ± 12.33	41.53 ± 7.54
WKNNI	92.27 ± 6.02	65.79 ± 6.77	66.01 ± 11.90	58.29 ± 7.28	79.52 ± 12.01	38.37 ± 11.02
KMI	91.47 ± 7.55	67.51 ± 7.36	68.90 ± 11.56	57.80 ± 6.95	79.60 ± 11.16	37.85 ± 12.02
FKMI	92.93 ± 6.72	66.04 ± 6.93	64.65 ± 12.31	56.06 ± 6.76	78.95 ± 10.81	39.04 ± 11.13
SVMI	94.27 ± 5.50	63.98 ± 8.21	69.20 ± 10.80	58.78 ± 6.09	81.76 ± 10.67	41.06 ± 10.42
EM	90.27 ± 9.35	66.19 ± 9.48	52.78 ± 13.09	54.46 ± 8.13	78.12 ± 15.60	40.02 ± 9.17
SVDI	91.07 ± 7.61	67.11 ± 5.93	57.88 ± 12.85	55.97 ± 6.86	77.61 ± 13.88	**44.46 ± 9.24**
BPCA	33.33 ± 0.00	66.59 ± 8.31	28.78 ± 3.51	49.65 ± 5.55	64.86 ± 8.15	38.44 ± 7.69
MC	94.27 ± 6.11	65.24 ± 5.55	64.04 ± 12.96	58.64 ± 7.72	**83.31 ± 13.40**	39.07 ± 11.14
CMC	93.73 ± 5.56	66.12 ± 7.74	67.85 ± 10.30	59.25 ± 7.22	81.03 ± 9.78	42.29 ± 8.22
DNI	92.40 ± 8.11	63.93 ± 9.11	63.63 ± 10.17	56.49 ± 7.41	78.46 ± 12.87	36.42 ± 14.05
	SAT	GER	MAG	SHU		
IM	33.25 ± 8.05	60.26 ± 5.12	64.56 ± 9.94	75.42 ± 17.14		
EC	35.55 ± 12.40	63.32 ± 7.26	**69.09 ± 10.17**	**87.45 ± 11.38**		
KNNI	38.47 ± 9.37	**68.68 ± 2.49**	63.74 ± 10.47	76.42 ± 16.53		
WKNNI	36.84 ± 9.26	68.04 ± 3.19	62.75 ± 10.38	74.02 ± 16.85		
KMI	37.93 ± 10.11	67.54 ± 4.15	63.95 ± 9.63	69.13 ± 18.36		
FKMI	33.50 ± 9.02	66.92 ± 5.44	64.90 ± 8.77	73.45 ± 15.11		
SVMI	**38.82 ± 10.82**	67.10 ± 3.67	64.46 ± 8.64	73.73 ± 12.91		
EM	23.63 ± 10.36	67.10 ± 3.40	64.13 ± 8.11	66.84 ± 16.96		
SVDI	24.36 ± 10.27	66.50 ± 6.01	64.59 ± 9.26	61.22 ± 21.72		
BPCA	15.63 ± 3.04	64.40 ± 8.63	57.78 ± 7.18	50.98 ± 12.03		
MC	36.86 ± 10.08	66.80 ± 5.39	62.58 ± 9.75	70.51 ± 14.42		
CMC	37.35 ± 8.98	67.86 ± 3.14	64.74 ± 8.36	78.16 ± 12.78		
DNI	30.43 ± 9.83	67.98 ± 6.73	62.66 ± 11.28	65.05 ± 19.37		

IM, and sometimes surpassed by simple methods like mean substitution.

- *SVMI* method does not perform very well, despite of the RBF Kernel we have chosen. This method appears to be very dependent on the data set.
- *EM* method does not perform as well as the best method, but it also never leads to the worse accuracy. Compared to *SVDI*, it obtains better results, although *SVDI* is based on *EM* imputation method.
- *BPCA* has a very poor performance sometimes. Except for RBFN Incremental in certain cases, their results are below the average.
- Finally, *EC* is the best method in many data sets. It is the best 13 times for RBFN, 11 times for RBFND and 13 times for RBFNI. It offers good stability in each data set, and can surpass the accuracy in more than 10% in some data sets respect to other imputation methods.

As a summary from the previous analysis we can observe that *DNI* is not the best option, since it can be improved by many methods, which are capable of capture the relationships between values of the instance. The use of case deletion (i.e. *IM*) is discouraged too, since it rarely outperforms the results of the imputation methods, and can seriously harm the test accuracy.

Table 5
RBFN incremental test accuracy.

	AUT	BRE	CLE	CRX	WIS	BAN
IM	36.69 ± 10.01	66.45 ± 8.48	35.33 ± 10.74	62.94 ± 5.38	96.02 ± 3.31	74.02 ± 7.61
EC	67.65 ± 9.77	61.75 ± 7.87	**54.00 ± 8.05**	**81.49 ± 4.08**	**96.23 ± 2.71**	**75.96 ± 5.97**
KNNI	34.52 ± 8.59	63.76 ± 7.13	35.89 ± 8.07	63.38 ± 5.18	95.66 ± 2.75	74.93 ± 6.43
WKNNI	34.56 ± 12.16	65.83 ± 8.87	35.46 ± 9.40	62.28 ± 5.95	95.77 ± 2.80	75.59 ± 5.96
KMI	34.23 ± 12.90	65.05 ± 6.19	34.61 ± 9.00	63.90 ± 4.93	96.20 ± 2.52	75.59 ± 6.30
FKMI	35.27 ± 11.66	65.63 ± 7.37	34.59 ± 9.96	63.69 ± 5.17	95.54 ± 4.02	74.70 ± 5.18
SVMI	36.60 ± 11.01	64.72 ± 7.88	35.74 ± 7.78	63.56 ± 6.29	96.20 ± 2.61	75.22 ± 6.05
EM	46.58 ± 10.17	64.33 ± 7.10	35.57 ± 9.61	63.15 ± 5.72	96.03 ± 2.46	75.41 ± 5.34
SVDI	46.56 ± 12.12	64.16 ± 8.27	35.75 ± 9.78	61.23 ± 5.42	95.88 ± 3.13	75.74 ± 6.57
BPCA	**76.96 ± 9.51**	**94.39 ± 3.85**	37.48 ± 8.95	51.97 ± 3.42	45.21 ± 3.25	57.41 ± 5.98
MC	34.45 ± 10.05	63.42 ± 8.13	36.10 ± 8.89	63.75 ± 5.44	95.83 ± 2.92	75.15 ± 5.12
CMC	36.72 ± 10.80	63.62 ± 7.37	36.60 ± 8.22	64.42 ± 5.74	95.85 ± 2.46	75.67 ± 6.73
DNI	32.52 ± 10.81	65.86 ± 7.36	35.36 ± 9.46	63.41 ± 5.64	96.00 ± 3.04	75.78 ± 4.88
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	95.42 ± 3.90	76.19 ± 4.49	59.44 ± 17.51	**77.70 ± 13.04**	**100.00 ± 0.00**
EC	**75.71 ± 6.13**	94.78 ± 4.04	**79.46 ± 6.15**	60.00 ± 15.56	75.33 ± 11.08	**100.00 ± 0.02**
KNNI	57.41 ± 5.70	94.23 ± 4.06	75.40 ± 4.20	59.11 ± 16.40	58.65 ± 13.77	**100.00 ± 0.00**
WKNNI	56.40 ± 8.25	94.50 ± 4.37	75.65 ± 4.41	**62.67 ± 16.14**	61.15 ± 14.23	**100.00 ± 0.02**
KMI	60.84 ± 8.31	94.64 ± 3.73	75.19 ± 4.85	60.89 ± 13.74	61.41 ± 13.15	99.99 ± 0.03
FKMI	58.02 ± 8.21	94.46 ± 3.65	75.73 ± 4.83	60.00 ± 16.33	58.11 ± 13.04	**100.00 ± 0.02**
SVMI	55.94 ± 7.59	94.73 ± 4.07	76.41 ± 4.57	60.89 ± 16.52	74.87 ± 13.53	**100.00 ± 0.00**
EM	60.12 ± 7.37	90.76 ± 6.33	75.40 ± 5.09	60.00 ± 17.07	63.25 ± 13.89	**100.00 ± 0.00**
SVDI	57.90 ± 6.99	90.66 ± 5.89	76.20 ± 4.53	58.89 ± 15.60	67.30 ± 12.19	99.99 ± 0.04
BPCA	54.59 ± 5.09	50.20 ± 7.36	50.11 ± 4.39	43.78 ± 17.41	73.67 ± 3.25	50.84 ± 0.68
MC	57.90 ± 6.82	94.64 ± 4.16	74.97 ± 4.09	60.22 ± 17.08	62.88 ± 13.56	**100.00 ± 0.02**
CMC	60.10 ± 8.51	**95.66 ± 3.76**	76.92 ± 5.13	58.44 ± 16.60	73.21 ± 10.39	99.99 ± 0.03
DNI	57.96 ± 8.16	94.18 ± 3.67	76.05 ± 4.36	62.00 ± 16.04	62.19 ± 13.19	**100.00 ± 0.00**
	IRI	PIM	WIN	AUS	NTH	ECO
IM	94.67 ± 5.66	65.98 ± 6.38	64.87 ± 11.53	59.68 ± 5.55	87.10 ± 7.87	50.34 ± 12.45
EC	94.13 ± 4.92	**70.26 ± 4.73**	**87.06 ± 17.67**	**82.43 ± 4.79**	90.91 ± 5.09	**67.70 ± 7.18**
KNNI	94.67 ± 4.42	67.44 ± 6.70	71.40 ± 8.20	59.83 ± 6.72	87.80 ± 8.28	54.26 ± 10.69
WKNNI	94.53 ± 4.36	65.08 ± 7.59	70.65 ± 8.89	60.72 ± 5.89	87.67 ± 6.69	54.40 ± 10.10
KMI	95.20 ± 5.17	65.94 ± 6.37	68.90 ± 9.18	58.81 ± 6.77	87.36 ± 7.65	54.16 ± 12.20
FKMI	95.20 ± 5.34	65.83 ± 5.57	69.92 ± 10.18	59.57 ± 6.14	88.02 ± 7.03	49.53 ± 11.89
SVMI	94.67 ± 6.11	64.72 ± 5.87	72.65 ± 8.85	59.25 ± 5.91	88.84 ± 6.05	57.06 ± 10.95
EM	94.67 ± 5.96	67.31 ± 5.13	56.69 ± 13.34	59.97 ± 5.38	86.32 ± 7.70	52.04 ± 10.02
SVDI	94.00 ± 7.45	64.56 ± 6.03	58.42 ± 14.04	60.00 ± 5.55	86.08 ± 8.08	51.65 ± 10.10
BPCA	33.33 ± 0.00	99.19 ± 0.82	33.06 ± 2.10	47.80 ± 6.56	69.62 ± 1.86	35.48 ± 5.83
MC	95.60 ± 4.54	63.93 ± 6.10	72.66 ± 9.29	59.74 ± 5.73	86.37 ± 7.72	50.87 ± 12.55
CMC	94.67 ± 5.33	67.30 ± 5.23	72.83 ± 9.31	61.10 ± 5.78	**90.93 ± 5.89**	56.11 ± 9.75
DNI	**95.73 ± 4.57**	66.73 ± 6.69	68.51 ± 10.59	60.17 ± 5.24	88.19 ± 6.47	53.55 ± 13.18
	SAT	GER	MAG	SHU		
IM	60.94 ± 3.16	53.98 ± 5.00	71.42 ± 3.63	93.23 ± 3.47		
EC	76.82 ± 1.51	**66.74 ± 5.18**	**76.20 ± 3.97**	**97.57 ± 5.32**		
KNNI	76.46 ± 2.74	55.64 ± 5.42	71.03 ± 3.25	95.42 ± 2.50		
WKNNI	76.86 ± 2.50	57.30 ± 4.79	72.84 ± 3.20	95.58 ± 1.54		
KMI	77.03 ± 2.82	56.06 ± 6.03	71.22 ± 3.39	95.17 ± 2.05		
FKMI	77.44 ± 3.49	56.98 ± 4.14	71.58 ± 3.54	95.71 ± 1.78		
SVMI	79.35 ± 2.17	57.02 ± 5.75	72.87 ± 3.32	95.68 ± 1.54		
EM	57.45 ± 6.68	55.64 ± 7.36	69.48 ± 3.60	92.35 ± 3.72		
SVDI	57.54 ± 7.01	56.94 ± 5.11	70.82 ± 3.84	93.82 ± 2.93		
BPCA	18.38 ± 0.88	59.74 ± 4.30	64.85 ± 0.32	65.37 ± 1.96		
MC	77.19 ± 2.19	56.74 ± 5.79	71.62 ± 3.15	95.31 ± 1.72		
CMC	**79.62 ± 1.74**	57.12 ± 5.41	71.82 ± 4.10	95.73 ± 1.53		
DNI	64.04 ± 3.00	56.90 ± 5.47	72.60 ± 3.92	95.11 ± 1.87		

EC has been capable of offering an outstanding performance in comparison with the other imputation models.

3.3. Statistical and graphical analysis of the EventCovering method

We have proceeded to a statistical analysis in order to establish the significance degree of differences in performance between the *EC* method and the other imputation methods. We have applied a non-parametric statistical test (Dĕmsar, 2006; García & Herrera, 2008), the Wilcoxon Signed Rank Test. We distinguish between the results obtained by the data sets with natural MVs and the data sets with induced MVs. The obtained results for *EC* versus the other methods can be found in Table 6 for the data sets with natural MVs, and Table 7 for the data sets with induced MVs.

As we can see from Tables 6 and 7, *EC* is the best method in almost all comparisons with statistical significance of $\alpha = 0.05$. There exist some exceptions:

- *IM* has no significant difference with *EC* for the RBFN and RBFNI models in the data sets with natural MVs.
- *SVMI* has no significant difference with *EC* for RBFN in the data sets with natural MVs. It also has a statistical significance with $\alpha = 0.1$ in RBFND method in both natural and artificial MV data sets.
- *MC* presents no significant difference with *EC* in the RBFN method for the data sets with natural MVs. For the artificial data sets, *MC* has statistical significance with $\alpha = 0.1$ in the RBFND method.

Table 6
Wilcoxon signed rank test results for EC (natural MVs).

	RBFN			RBFN decremental			RBFN incremental		
	R−	R+	*p*-value	R−	R+	*p*-value	R−	R+	*p*-value
IM	57	21	0.158	66	12	0.034	59.5	18.5	0.110
KNNI	68	10	0.023	68	10	0.023	71.5	6.5	0.013
WKNNI	70	8	0.015	66	12	0.034	65.5	12.5	0.041
KMI	70	8	0.015	65	13	0.041	67	11	0.028
FKMI	70	8	0.015	68	10	0.023	69.5	8.5	0.022
SVMI	53	25	0.272	61	17	0.084	64.5	13.5	0.050
EM	66	12	0.034	66	12	0.034	67.5	10.5	0.013
SVDI	64	14	0.050	69	9	0.019	71.5	6.5	0.008
BPCA	76	2	0.004	75	3	0.005	67	11	0.028
MC	68	10	0.230	72	6	0.010	68.5	9.5	0.021
CMC	63.5	14.5	0.050	64	14	0.050	68	10.	0.023
DNI	69	9	0.019	66	12	0.034	63.5	12.5	0.041

Table 7
Wilcoxon signed rank test results for EC (artificial MVs).

	RBFN			RBFN decremental			RBFN incremental		
	R−	R+	*p*-value	R−	R+	*p*-value	R−	R+	*p*-value
IM	55	0	0.005	55	0	0.005	54	1	0.007
KNNI	48	7	0.037	43	12	0.114	53	2	0.009
WKNNI	48	7	0.037	49	6	0.028	52	3	0.013
KMI	49	6	0.028	48	7	0.037	52	3	0.013
FKMI	50	5	0.022	51	4	0.017	52	3	0.013
SVMI	48.5	6.5	0.038	46	9	0.059	50	5	0.022
EM	55	0	0.005	52	3	0.013	54	1	0.007
SVDI	55	0	0.005	51	4	0.017	55	0	0.005
BPCA	55	0	0.005	54	1	0.007	51	4	0.017
MC	48	7	0.037	46	9	0.059	52	3	0.013
CMC	48.5	6.5	0.038	44	11	0.093	48	7	0.037
DNI	55	0	0.005	52	3	0.013	54	1	0.007

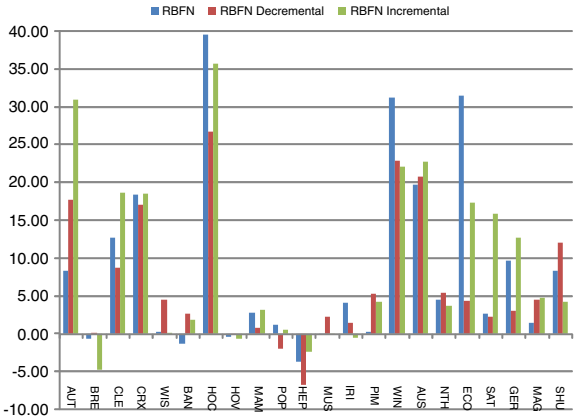

Fig. 1. EC vs. IM.

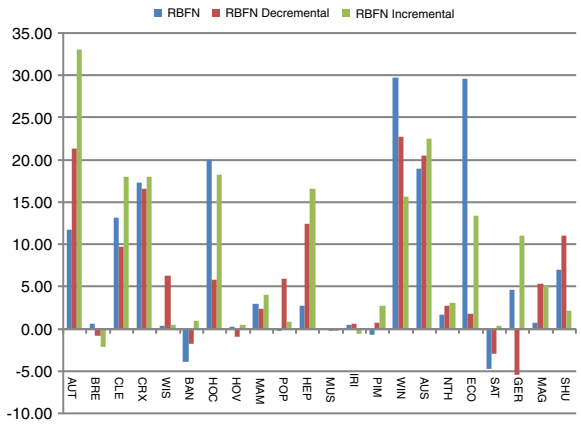

Fig. 2. EC vs. KNNI.

- Finally, *CMC* has a statistical significance with $\alpha = 0.1$ for the artificial data sets in the RBFND method.

These results confirm the previous analysis obtained from the result tables and figures, *EC* method is the best among all presented imputation methods in most cases for the RBFN models considered.

Finally we show graphically that *EC* method has got an outstanding performance with RBFNs. In Figs. 1–12 we use the test set accuracy as a performance measure, and the vertical bars represent the difference in accuracy between *EC* and the other methods for each data set.

- The positive values (bars above the baseline) represent the advantage of *EC* method. That is, the larger the bar, the higher the difference between *EC* method and the indicated one.
- A negative bar under the baseline indicates that the RBFN model for *EC* method has lower accuracy than the other imputation method.

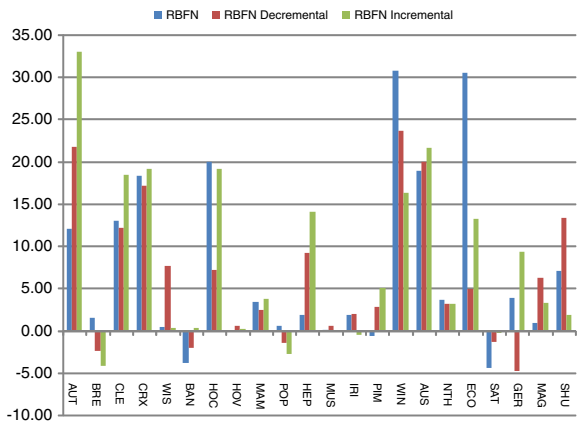

Fig. 3. EC vs. WKKNI.

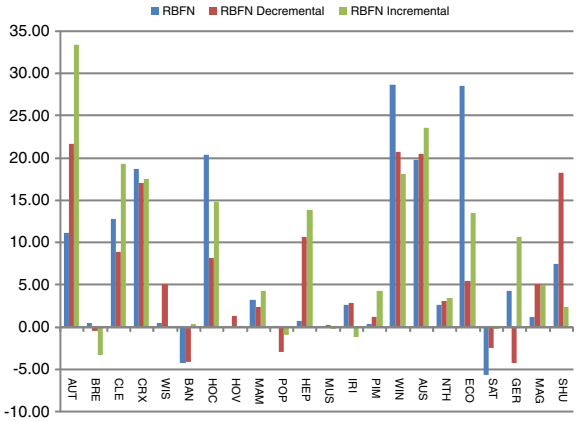

Fig. 4. EC vs. KMI.

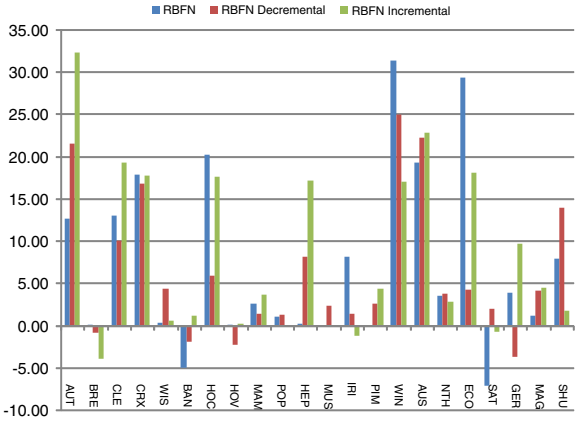

Fig. 5. EC vs. FKMI.

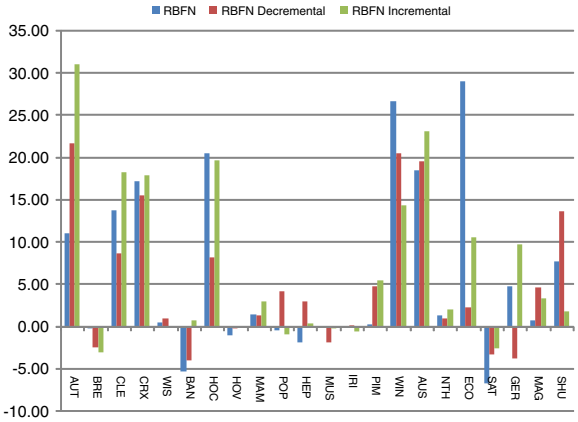

Fig. 6. EC vs. SVMI.

The best behavior of *EC* over all methods is present for almost every data set, since there are more "positive" bars, and with higher positive value. We can distinguish three different situations:

- The data sets AUT, CLE, CRX, HOC, HEP, WIN, AUS, ECO, AUS, GER and SHU present positive differences over the 5% for at least two models of the RBFNs.
- The data sets WIS, HAM, POP, MUS, IRI, PIM, NTH and MAG present little positive difference for *EC*, under the 5%, for at least two RBFN methods.
- Only the data sets BRE, BAN and SAT show a lower accuracy for the RBFN models in the case of *EC* method respect to the rest with a negative bar.

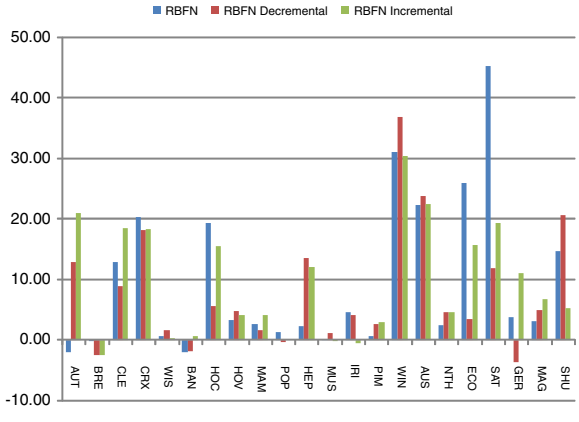

Fig. 7. EC vs. EM.

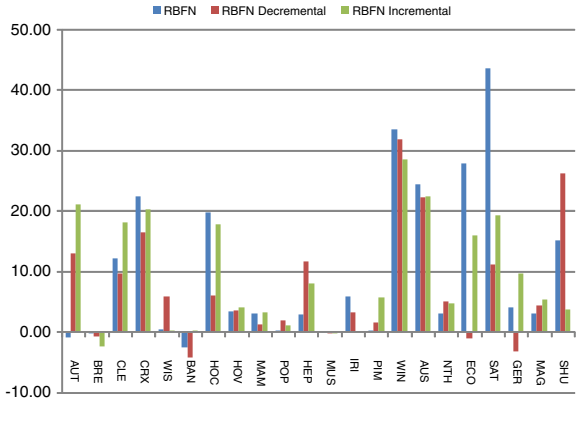

Fig. 8. EC vs. SVDI.

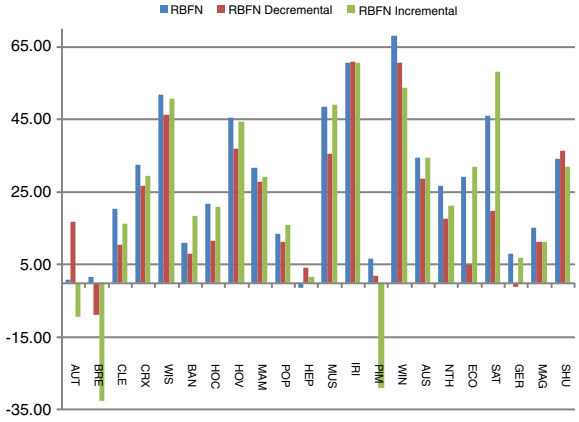

Fig. 9. EC vs. BPCA.

The behavior of the RBFN models in the different data sets is homogeneous, and we can find that the *EC* method outperforms the rest in the same data sets for all the three RBFN models. Therefore, from the figures we can observe that only for the 14% of the data sets, the *EC* imputation method is performing worse than the rest. From the remaining 86%, on the 36% of the data sets the *EC* method behaves better than the rest with little difference, and in the remaining 50% the improvements of accuracy are notorious.

4. Concluding remarks

We have studied the use of imputation techniques for the analysis of RBFN in classification problems, presenting a comparison

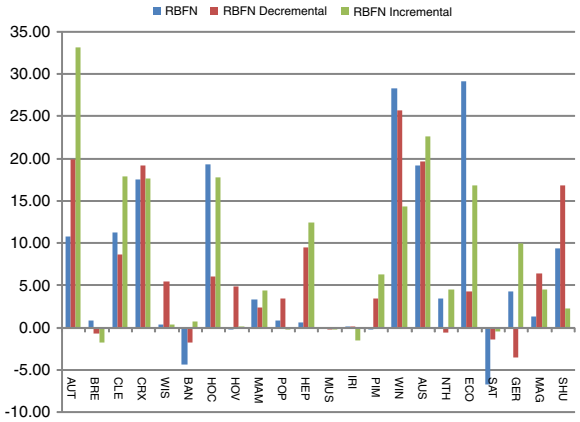

Fig. 10. EC vs. MC.

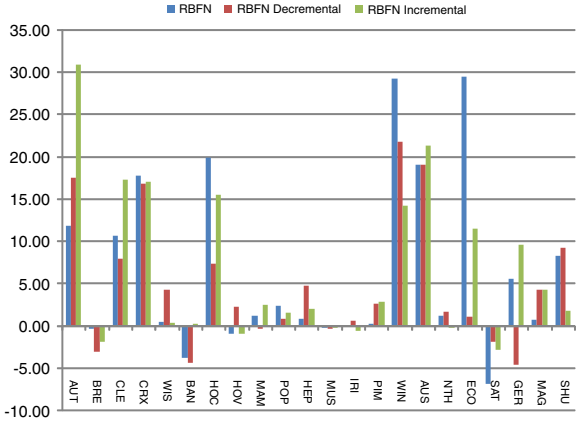

Fig. 11. EC vs. CMC.

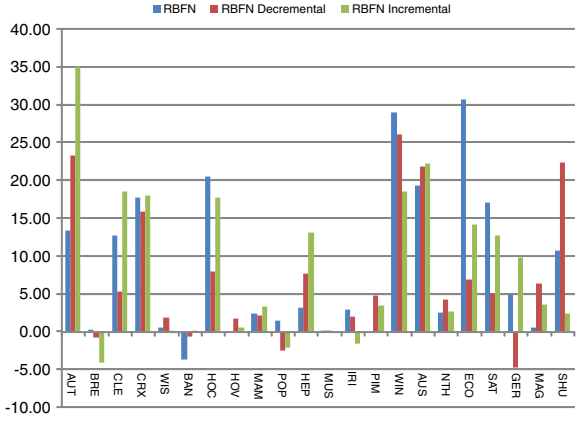

Fig. 12. EC vs. DNI.

between: (1)imputation, (2)do not impute, and (3)ignore cases with MVs.

When comparing the results of these three options, the need for using imputation methods is clear when analyzing RBFN for classification, since improvements in performance of the results are often achieved.

We must point out that the *EC* method is the best choice for carrying out an imputation of MVs when working with RBFNs in three considered variants of this model. Therefore, we can confirm the good synergy between RBFN models and the EventCovering method.

References

Acuna, E., & Rodriguez, C. (2004). The treatment of missing values and its effect in the classifier accuracy. In D. Banks, L. House, F. R. McMorris, P. Arabie, & W. Gaul (Eds.), *Classification, clustering and data mining applications* (pp. 639–648). Berlin, Germany: Springer-Verlag Berlin-Heidelberg.

Arenas-Garcia, J., Gomez-Verdejo, V., & Figueiras-Vidal, A. R. (2007). Fast evaluation of neural networks via confidence rating. *Neurocomputing*, *70*, 2775–2782.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository [WWW Page]. Irvine, CA: University of California, School of Information and Computer Science. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Batista, G. E. A. P. A., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*, 519–533.

Billings, S. A., Wei, H.-L., & Balikhin, M. A. (2007). Generalized multiscale radial basis function networks. *Neural Networks*, *20*(10), 1081–1094.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Buhmann, M. D. (2003). *Cambridge monographs on applied and computational mathematics. Radial basis functions: Theory and implementations*.

Děmsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Eickhoff, R., & Ruckert, U. (2007). Robustness of radial basis functions. *Neurocomputing*, *70*, 2758–2767.

Ennett, C. M., Frize, M., & Walker, C. R. (2001). Influence of missing values on artificial neural network performance. *Medinfo*, *10*, 449–453.

Er, M. J., Wu, S., Lu, J., & Hock-Lye, T. (2002). Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks*, *13*, 697–710.

Farhangfar, A., Kurgan, L., & Pedrycz, W. (2004). Experimental analysis of methods for imputation of missing values in databases. In K. L. Priddy (Ed.). *SPIE. Vol. 5421. Intelligent computing: Theory and applications II*. Michigan (pp. 172–182).

Feng, H. A. B., Chen, G. C., Yin, C. D., Yang, B. B., & Chen, Y. E. (2005). A SVM regression based approach to filling in missing values. In R. Khosla, R. J. Howlett, & L. C. Jain (Eds.), *Lecture notes in artificial intelligence*: *Vol. 3683. Knowledge-based intelligent information and engineering systems (KES 2005)* (pp. 581–587). Springer.

García, S., & Herrera, F. (2008). An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Ghodsi, A., & Schuurmans, D. (2003). Automatic basis selection techniques for RBF networks. *Neural Networks*, *16*(5–6), 809–816.

Grzymala-Busse, J. W., & Hu, M. (2000). A comparison of several approaches to missing attribute values in data mining. In W. Ziarko, & Y. Y. Yao (Eds.), *Lecture notes in computer science*: *Vol. 2005. Rough sets and current trends in computing : Second international conference (RSCTC 2000)* (pp. 378–385). Canada: Springer.

Grzymala-Busse, J. W., & Goodwin, L. K. (2005). Handling missing attribute values in preterm birth data sets. In D. Slezak, J. Yao, J. F. Peters, W. Ziarko, & X. Hu (Eds.), *Lecture notes in computer science*: *Vol. 3642. Rough sets, fuzzy sets, data mining, and granular computing (RSFDGrC 2005)* (pp. 342–351). Canada: Springer.

Harpham, C., & Dawson, C. W. (2006). The effect of different basis functions on a radial basis function network for time series prediction: A comparative study. *Neurocomputing*, *69*, 2161–2170.

Kros, J. F., Lin, M., & Brown, M. L. (2006). Effects of the neural network s-Sigmoid function on KDD in the presence of imprecise data. *Computers and Operations Research*, *33*, 3136–3149.

Lázaro, M., Santamaría, I., & Pantaleón, C. (2003). A new EM-based training algorithm for RBF networks. *Neural Networks*, *16*(1), 69–77.

Lendasse, A., Francois, D., Wertz, V., & Verleysen, M. (2005). Vector quantization: A weighted version for time-series forecasting. *Future Generation Computer Systems*, *21*, 1056–1067.

Li, D., Deogun, J., Spaulding, W., & Shuart, B. (2004). Towards missing data imputation: A study of fuzzy K-means clustering method. In S Tsumoto, R. Slowinski, J. Komorowski, & J. W. Grzymala-Busse (Eds.), *Lecture notes in computer science*: *Vol. 3066. Rough sets and current trends in computing (RSCTC 2004)* (pp. 573–579). Sweden: Springer-Verlag.

Liao, Y., Fang, S.-C., & Nuttle, H. L. W. (2003). Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks*, *16*(7), 1019–1028.

Lim, C. P., Leong, J. H., & Kuan, M. M. (2005). A hybrid neural network system for pattern classification tasks with missing features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 648–653.

Little, R. J., & Rubin, D. B. (1987). *Statistical analysis with missing data*. New York: John Wiley and Sons.

Markey, M. K., Tourassi, G. D., Margolis, M., & DeLong, D. M. (2006). Impact of missing data in evaluating artificial neural networks trained on complete data. *Computers in Biology and Medicine*, *36*, 516–525.

Mileva-Boshkoska, B., & Stankovski, M. (2007). Prediction of missing data for ozone concentrations using support vector machines and radial basis neural networks. *Informatica (Ljubljana)*, *31*, 425–430.

Morris, C. W., Boddy, L., & Wilkins, M. F. (2001). Effects of missing data on RBF neural network identification of biological taxa: Discrimination of microalgae from flow cytometry data. *International Journal of Smart Engineering System Design*, *3*, 195–202.

Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B., & Hummels, D. M. (1992). On the training of radial basis function classifiers. *Neural Networks*, *5*, 595–603.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007). Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, *93*, 1514–1521.
Oba, S., Sato, M., Takemasa, I., Monden, M., Matsubara, K., & Ishii, S. (2003). A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, *19*, 2088–2096.
Pearson, R. K. (2005). *Mining imperfect data*. SIAM.
Pelckmans, K., De Brabanterb, J., Suykensa, J. A. K., & De Moor, B. (2005). Handling missing values in support vector machine classifiers. *Neural Networks*, *18*, 684–692.
Pisoni, E., Pastor, F., & Volta, M. (2008). Artificial neural networks to reconstruct incomplete satellite data: Application to the mediterranean sea surface temperature. *Nonlinear Processes in Geophysics*, *15*, 61–70.
Plat, J. (1991). A resource allocating network for function interpolation. *Neural Computation*, *3*, 213–225.
Powell, M. J. D. (1987). Radial basis function for multivariate interpolation: A review. In J. C. Mason, & M. G. Cox (Eds.), *Algorithm for approximation* (pp. 143–168). Oxford, England: Clarendon Press.
Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers.
Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychol Methods*, *7*, 147–177.
Schneider, T. (2001). Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, *14*, 853–871.
Schwenker, F., Kestler, H. A., & Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural Networks*, *14*(4–5), 439–458.
Sun, Y. V., & Kardia, S. L. R. (2008). Imputing missing genotypic data of single-nucleotide polymorphisms using neural networks. *European Journal of Human Genetics*, *16*, 487–495.
Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, *17*, 520–525.
Uysal, M. (2007). Reconstruction of time series data with missing values. *Journal of Applied Sciences*, *7*, 922–925.
Wallace, M., Tsapatsoulis, N., & Kollias, S. (2005). Intelligent initialization of resource allocating RBF networks. *Neural Networks*, *18*(2), 117–122.
Wang, S. (2003). Application of self-organising maps for data mining with incomplete data sets. *Neural Computation & Applications*, *12*, 42–48.
Wei, H., & Amari, S.-i. (2008). Dynamics of learning near singularities in radial basis function networks. *Neural Networks*, *21*(7), 989–1005.
Wong, A. K. C., & Chiu, D. K. Y. (1987). Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*, 796–805.
Yeung, D. S., Ng, W. W. Y., Wang, D., Tsang, E. C. C., & Wang, X.-Z. (2007). Localized generalization error model and its application to architecture selection for radial basis function neural network. *IEEE Transactions on Neural Networks*, *18*, 1294–1305.
Yingwei, L., Sundararajan, N., & Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, *9*, 361–478.
Yoon, S. Y., & Lee, S. Y. (1999). Training algorithm with incomplete data for feed-forward neural networks. *Neural Processing Letters*, *10*, 171–179.