Fuzzy min–max neural networks for categorical data: application to missing data imputation

Pilar Rey-del-Castillo · Jesús Cardeñosa

Abstract The fuzzy min–max neural network classifier is a supervised learning method. This classifier takes the hybrid neural networks and fuzzy systems approach. All input variables in the network are required to correspond to continuously valued variables, and this can be a significant constraint in many real-world situations where there are not only quantitative but also categorical data. The usual way of dealing with this type of variables is to replace the categorical by numerical values and treat them as if they were continuously valued. But this method, implicitly defines a possibly unsuitable metric for the categories. A number of different procedures have been proposed to tackle the problem. In this article, we present a new method. The procedure extends the fuzzy min–max neural network input to categorical variables by introducing new fuzzy sets, a new operation, and a new architecture. This provides for greater flexibility and wider application. The proposed method is then applied to missing data imputation in voting intention polls. The micro data—the set of the respondents' individual answers to the questions—of this type of poll are especially suited for evaluating the method since they include a large number of numerical and categorical attributes.

Keywords Classification · Fuzzy systems · Fuzzy min–max neural networks · Imputation · Missing data

1 Introduction

Missing information in datasets is a by no means uncommon scenario [1–4]. A frequently used procedure to deal with this problem in opinion polls is to replace each missing variable value with an estimated value or imputation obtained from the values of other variables in the same item [5, 6].

On the other hand, classification is one of the tasks involved in a data mining process. Classification can be defined as a procedure in which individual items are placed into groups or categories based on quantitative information on one or more characteristics inherent to the items (referred to as variables, characters, features, etc.,) and based on a training set of previously labeled items. Because of the appeal of simple rules that are easy to construct, fuzzy control systems have been used for the purpose of classification from the earliest days of fuzzy logic [7, 8]. These systems usually generate a rule for each classification category, specifying the rule's antecedent from fuzzy sets defined over the input variables set. The rules are easy to specify when there are not many categories, but this gets harder as the number grows. To overcome this problem, some hybrid approaches have been proposed to ease the learning of the fuzzy rules [9, 10]. These hybrid procedures are mainly based on the combination of fuzzy set theory with other methodologies, like evolutionary algorithms and neural networks. Neuro-fuzzy computation is one of the most popular hybridizations in the artificial intelligence literature [11–14], because it combines the merits of the

neural and fuzzy approaches. It has the generic benefits of neural networks—like massive parallelism and robustness—and, at the same time, uses fuzzy logic to model vague or qualitative knowledge and convey uncertainty [15].

The fuzzy min–max neural network classifier is a supervised learning method that takes the hybrid neural networks and fuzzy systems approach. The original fuzzy min–max neural networks model was developed by Simpson [16, 17], and was modified and improved in a later version [18, 19]. This version offers a new approach to dealing with missing input variables data. A number of modifications have also been put forward aimed at improving the fuzzy membership definition [20], and the effectiveness of some of the learning process steps [21–23].

A characteristic of the fuzzy min–max neural network classifier is that all the input variables for learning and classification are required to correspond to numerical, continuously valued variables. One typical way of dealing with this problem when there are categorical variables, is to replace the categorical by numerical values and treat them as if they were continuously valued. But this procedure implicitly defines a metric for the categories, which may not be suitable [24]. This suggests that a different procedure for dealing with categorical variables must be used.

In this article, we present a method that extends the fuzzy min–max neural network classifier input to categorical variables by introducing new fuzzy sets, a new operation, and a new architecture. This new procedure provides for greater flexibility and wider application, and also straightforwardly extends the treatment of the missing values in the input variables.

To test the proposed method, it will be used to tackle the problem of missing data. Specifically, it will be applied to non-response imputation in opinion polls. The micro data (the set of the respondents' individual answers to the questions) of this type of poll are especially suited for evaluating the method, since they include a large number of numerical and categorical attributes. To perform categorical variables imputation, every category or value of the variable to be imputed will be associated with a classifier class, and the estimation for a missing data input consists of the classification category [25].

The article is organized as follows. Section 2 gives a brief review of the architecture and operation of fuzzy min–max neural networks as a starting point for the new classifier. Section 3 describes the new fuzzy sets-based method used to define new networks and their architecture and operation. Section 4 shows the context of the imputation problem to be solved with the new method and presents the experimental results. Some conclusions are presented in Sect. 5. The results are also compared with the outcomes of applying traditional methods to the same data sets, resulting in some improvements as shown in the outlined experiment.

2 Fuzzy min–max neural network classifier

The original fuzzy min–max neural networks algorithm was introduced for the first time in two articles by Simpson [16, 17]. It is a classification method that separates the joint input variables space into classes of any size and shape with nonlinear boundaries. Here, we outline a later version that includes some improvements [18, 19].

2.1 Classification model

The n input variables must be numerical, and the output is a label or category of the discrete set of the categorical variable values. A hyperbox in R^n is a Cartesian product of closed intervals on the real line and is completely defined by its minimum and maximum points, as shown in the three-dimensional example in Fig. 1. Although it is possible to use hyperboxes with an arbitrary range of values in any dimension, min–max networks only use values that range from 0 to 1.

The operation is based on the hyperbox fuzzy sets defined in the n-dimensional pattern space. Thus, the input space is the n-dimensional unit cube $\mathbf{I^n} = [0, 1] \times [0, 1] \times \cdots \times [0, 1]$. The hyperbox fuzzy set B_j is defined by the ordered set

$$B_j = \left\{\mathbf{x}, \mathbf{v_j}, \mathbf{w_j}, b_j\left(\mathbf{x}, \mathbf{v_j}, \mathbf{w_j}\right)\right\}, \quad \forall \mathbf{x} \in \mathbf{I^n} \tag{1}$$

where $\mathbf{v_j} = (v_{j1}, \ldots, v_{jn})$ is the hyperbox minimum, $\mathbf{w_j} = (w_{j1}, \ldots, w_{jn})$ is the maximum, and $b_j(\mathbf{x}, \mathbf{v_j}, \mathbf{w_j})$ is the membership function, where all patterns within the hyperbox have full-class membership.

Figure 2 shows an example of how the hyperboxes are aggregated to form nonlinear boundaries in a two-class R^2 classification problem.

Pattern classification works in this type of networks by passing an input pattern through each characteristic function defining each class, and assigning the class with the

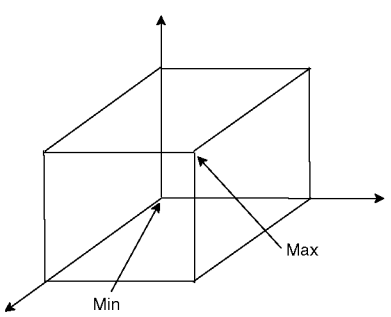

Fig. 1 Hyperbox in R^3 defined from its min and max points

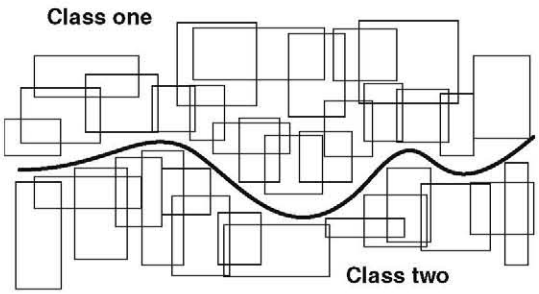

Fig. 2 Fuzzy min–max hyperboxes along the boundary of a two-class problem

largest value for these functions. Consequently, the first step for classifying an input pattern using the min–max neural networks classifier is to calculate its membership function of each class as the maximum of its membership functions of each of the hyperboxes defining this class (the maximum is the selected fuzzy union operator). The next step is to classify the point as the category corresponding to the class with the highest degree of membership.

One of Gabrys and Bargiela's improvements [18, 19], was to allow input patterns that are hyperboxes and not just numerical points. In this case, each input is specified by a vector $\mathbf{x_h}$, $h = 1, 2, \ldots, M$, where $\mathbf{x_h} = [\mathbf{x_h^l}, \mathbf{x_h^u}]$ is the hth input hyperbox defined by its minimum vector $\mathbf{x_h^l} = (x_{h1}^l, x_{h2}^l, \ldots, x_{hn}^l)$ and its maximum vector $\mathbf{x_h^u} = (x_{h1}^u, x_{h2}^u, \ldots, x_{hn}^u)$. When $\mathbf{x_h^l}$ and $\mathbf{x_h^u}$ are equal, the hyperbox shrinks to a point. The membership function of the hyperbox fuzzy set B_j for an input $\mathbf{x_h}$ is defined as

$$b_j(x_h) = \min_{i=1,\ldots,n} \{\min[(1 - g(x_{hi}^u - w_{ji}, \gamma)), (1 - g(v_{ji} - x_{hi}^l, \gamma))]\} \quad (2)$$

where γ is a parameter regulating how fast the membership function decreases and g is the ramp-threshold function of two parameters:

$$g(x, \gamma) = \begin{cases} 1 & \text{if } x.\gamma > 1 \\ x.\gamma & \text{if } 0 \le x.\gamma \le 1 \\ 0 & \text{if } x.\gamma < 0 \end{cases} \quad (3)$$

The membership function measures the degree to which the input pattern $\mathbf{x_h}$ falls inside of the B_j hyperbox fuzzy set. It takes the value 1—full membership—within the hyperbox and decays to zero as $\mathbf{x_h}$ moves away from the hyperbox. A two-dimensional example is shown in Fig. 3 for the hyperbox fuzzy set defined by the minimum $\mathbf{v_j} = (0.4, 0.2)$, the maximum $\mathbf{w_j} = (0.8, 0.4)$, and the parameter $\gamma = 3$.

The hyperboxes are incrementally trained by appropriately adjusting their number and volumes in a neural networks framework. This accounts for the name of fuzzy

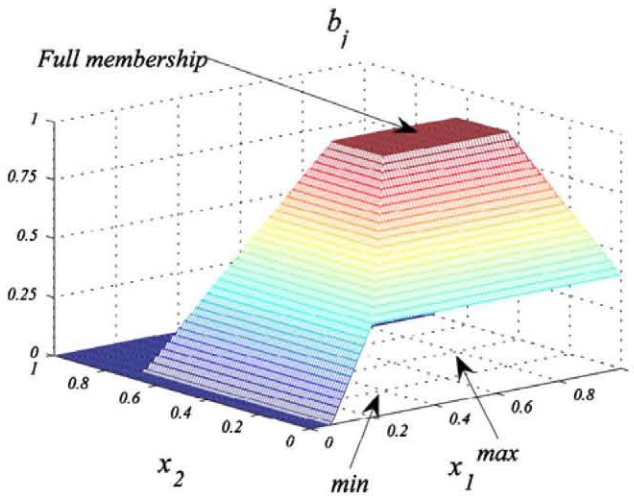

Fig. 3 Membership function of the hyperbox in I^2 defined by the minimum $\mathbf{v_j} = (0.4, 0.2)$, the maximum $\mathbf{w_j} = (0.8, 0.4)$, and the parameter $\gamma = 3$

min–max neural networks. The network architecture and learning are described next.

3 Network architecture

Figure 4 shows the three-layer feedforward neural network implementing Gabrys and Bargiela's fuzzy min–max neural classifier. Its topology grows adaptively to meet the problem requirements. The input layer has $2n$ nodes, two for each of the n input vector dimensions corresponding to the input hyperbox minimums (x_{hi}^l) and maximums (x_{hi}^u). Each intermediate layer node represents a hyperbox fuzzy set, where the connections with the input layer are the hyperbox fuzzy set minimum (v_{ji}) and maximum (w_{ji}) points, and the activation function is the hyperbox membership function (2).

Figure 5 shows the jth node of the intermediate layer in more detail. The connections between the second-layer and third-layer nodes are binary values, whose expression is

$$u_{jk} = \begin{cases} 1 & \text{if } B_j \text{ is a hyperbox for class } C_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where B_j is the jth intermediate layer node and C_k is the kth output layer node. The result of this last node represents the membership degree of input $\mathbf{x_h}$ to class k. The activation function for each output layer node is the fuzzy union of the hyperbox membership functions according to the expression $c_k = \max_{j=1,\ldots,m} b_j \cdot u_{jk}$. The classifier result for $\mathbf{x_h}$ is the class k with the greatest c_k value. The values for the connections are adjusted using the learning algorithm described next.

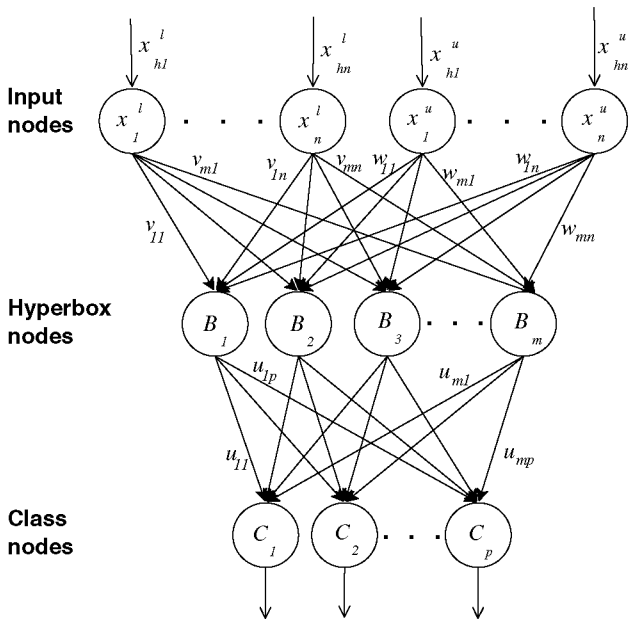

Fig. 4 Three-layer neural network implementing the fuzzy min–max neural network classifier

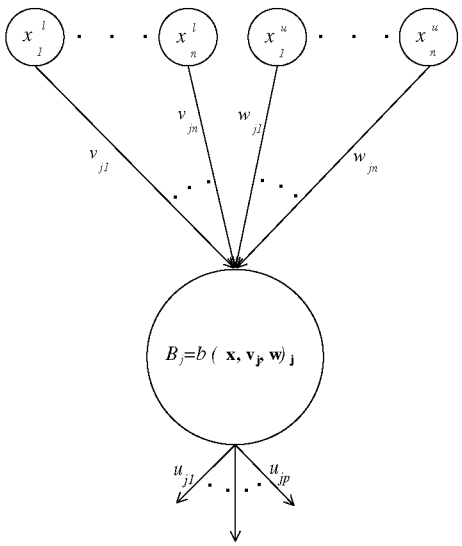

Fig. 5 Implementation of the jth node of the intermediate layer

3.1 Learning algorithm

Following Gabrys and Bargiela [18], the learning set consists of M ordered pairs

$$\{\mathbf{x_h}, d_h\}, \quad h = 1, \ldots, M \tag{5}$$

where $\mathbf{x_h} = [\mathbf{x_h^l}, \mathbf{x_h^u}]$ is the hth input defined by its minimum $\mathbf{x_h^l} = (x_{h1}^l, x_{h2}^l, \ldots, x_{hn}^l)$ and maximum $\mathbf{x_h^u} = (x_{h1}^u, x_{h2}^u, \ldots, x_{hn}^u)$ points, and $d_h \in \{1, 2, \ldots, p\}$ is the index of one of the p classes. The fuzzy min–max neural networks learning algorithm is a three-step expansion–contraction process:

1. Search for the closest expandable hyperbox (if necessary) and expand
2. Test for hyperbox overlap
3. Contract hyperbox

and it is repeated for each training input point. The process begins with the input of an ordered pair, searching the hyperbox with the highest membership degree that belongs to the same class and includes or allows expansion to include $\mathbf{x_h}$. If none of the hyperboxes satisfies the conditions, then a new hyperbox B_k for the input is created, adjusted, added to the neural network, and labeled by making class$(B_k) = d_h$.

The hyperbox is expanded by setting

$$v_{ji}^{\text{new}} = \min\left(v_{ji}^{\text{old}}, x_{hi}^l\right), \quad i = 1, \ldots, n \tag{6}$$

$$w_{ji}^{\text{new}} = \max\left(w_{ji}^{\text{old}}, x_{hi}^u\right), \quad i = 1, \ldots, n \tag{7}$$

and is constrained by a user-defined parameter θ, $(0 \leq \theta \leq 1)$, where $|w_{ji} - v_{ji}| \leq \theta, \quad \forall i = 1, \ldots, n$. The expansion can lead to an overlap between hyperboxes. This is not a problem when the overlap is between hyperboxes representing the same class. But when the overlap is between hyperboxes of different classes, it may mean that one input pattern belongs to two or more classes. So, when there is an overlap of this type, it is solved using a contraction process, following the principle of minimal adjustment where only the smallest overlap for one dimension is adjusted. The contraction process only eliminates the overlap between portions of the hyperbox fuzzy sets from separate classes that have full membership, allowing non-unit-valued portions of each of the hyperbox fuzzy sets to overlap. The boundaries between two classes are just the points with equal membership degree for both classes.

This learning process forms classes that are non-linearly separable. The existing classes can be refined over time and new classes can be added without retraining, thereby reducing total training time.

Concerning the issue of algorithm convergence, work by Zang et al. [26] is worth mentioning. They developed a rule for the min–max neural networks training and proved theoretically that converged using stochastic theory.

3.2 Numerical missing values treatment

A possible use of the min–max neural networks classifier is to perform imputation for categorical missing values as will be shown in Sect. 4. How the classifier deals with the missing values in the quantitative input variables is another question.

Thanks to the possibility of using hyperboxes as inputs [18, 19], missing values are easy to deal with: The missing features are represented as real-valued intervals spanning

the whole range of possible values. The procedure designed for learning and classification is to assign the minimum $x_{hi}^{l} = 1$ and the maximum $x_{hi}^{u} = 0$ to the ith numerical missing variable. Applying this strategy, the lower limit of the missing variable will never be less than v_{ji} and the upper limit will never be greater than w_{ji}, ensuring that the neural network structure will not have to be changed when processing inputs with missing values. It also has the advantage that when some limits for a missing feature are known, they can be used straightforwardly to contribute to the membership function.

According to Song and Shepperd's [27] missing data techniques taxonomy, this is a *toleration technique* because it does not impute missing data but works directly with data sets containing missing values. According to the same taxonomy, the proposed fuzzy min–max neural network algorithm that will be used in Sect. 4 is an *imputation technique* because it estimates each missing value.

4 New model with input of categorical variables

In contrast to the original fuzzy min–max neural networks classifier, the procedure proposed in this paper considers categorical as well as numerical variables as input. The problem with the categorical variable input is that there is no measure of distance between the different values or categories of the variables. This prevents the definition of hyperbox fuzzy sets membership functions.

The new method starts by defining such a distance to solve this problem. The following sections describe the proposed procedure according to the same framework as used in Gabrys and Bargiela's model. The basic process is divided into several stages:

1. Define distances between categories
2. Define hyperbox fuzzy sets in categorical variables
3. Extend network architecture and operation
4. Extend missing data treatment.

4.1 Defining distances between categories

To define a distance between the categories of a categorical variable, we will consider the relation of this variable to the classification variable, which must also be categorical. To illustrate this idea, Table 1 shows an example of a two-dimensional frequency table for the categorical variables *region* and *employment situation*.

Table 2 is calculated from Table 1 by just dividing the value of each cell by its row total. The vector $(q_1, \ldots, q_p)$ in each row of Table 2 contains the response rates for the *employment situation* categories in this *region*, referred to as the region's *employment situation* profile.

Table 1 Frequency table for region and employment situation variables

Region	Employment situation			
	Employed	Unemployed	Retired	Others
North	360	52	87	152
West	548	321	428	249
Center	132	16	27	48
East	811	723	543	703
South	264	178	227	136
Total	2,115	1,290	1,312	1,288

Table 2 Region's employment situation profiles

Region	Employment situation			
	Employed	Unemployed	Retired	Others
North	0.55	0.08	0.13	0.23
West	0.35	0.21	0.28	0.16
Center	0.59	0.07	0.12	0.22
East	0.29	0.26	0.2	0.25
South	0.33	0.22	0.28	0.17
Total	0.35	0.21	0.22	0.21

To define distances between *regions*, we examine their profiles, i.e., the *North* and *Center* regions have similar profiles (0.55, 0.08, 0.13, 0.23) and (0.59, 0.07, 0.12, 0.22), respectively. This means that the *employment situation* is similarly distributed across the categories in these regions. The profiles for the *West* and *South* regions are also similar, albeit different from the *North* and *Center* regions, whereas the *East* region is very different to the others. It could be said that, regarding the *employment situation*, the *North* and *Center* regions are closer to each other than to all the others; the *West* and *South* are also close, and so on.

The category profiles are points of the p-dimensional space R^p belonging to the hyperplane defined by $q_1 + \cdots + q_p = 1$. The distances between the profiles in this space can be used to define the distances between the categories. In this paper, we consider two distances:

$$\text{Euclidean distance:} \quad d_1(a_i, a_j) = \sqrt{\sum_{k=1}^{p} (p_{ik} - p_{jk})^2} \tag{8}$$

$$\text{Logarithmic distance:} \quad d_2(a_i, a_j) = \sum_{k=1}^{p} \left|\log p_{ik} - \log p_{jk}\right| \tag{9}$$

where a_i, a_j are the categories and (p_{ik}), (p_{jk}), $k = 1, \ldots, p$, are the corresponding profiles. As the proportions forming the profiles take values between 0 and 1, we consider the logarithmic distance in an attempt to prevent

proportionally short distances between high values from overdominating the calculations. To standardize and use the distances in the context of fuzzy set membership functions, they are also divided by their maximum:

$$c_k(a_i, a_j) = \frac{d_k(a_i, a_j)}{\max_{ij} d_k(a_i, a_j)}, \quad k = 1, 2 \tag{10}$$

This idea of distance between profiles appears well suited for classification purposes, because it takes into account the relation between each categorical variable to be measured and the classification variable. Correspondence analysis [28], for example, also exploits the same distance. Its use in a fuzzy min–max neural networks classifier is discussed next.

4.2 Defining hyperbox fuzzy sets in categorical variables

The next step after defining the distances between categories is to define the hyperbox fuzzy sets in the categorical dimensions.

This is not a straightforward step because, unlike numerical values, the categories or values of the categorical variable form a discrete rather than a dense set. This makes hyperboxes harder to create, update and modify. To do this, each hyperbox fuzzy set in the ith categorical dimension is defined by two categories e_{ji} and f_{ji} with a full membership function (equal to 1) similar to the two points—minimum and maximum—determining the hyperbox in the numerical dimensions. In any other category a_{ki}, this ith dimension membership function takes the value

$$b_{ji}(a_{hi}) = \min\left(1 - c\left(a_{hi}, e_{ji}\right),\ 1 - c\left(a_{hi}, f_{ji}\right)\right) \tag{11}$$

where function c refers to any of the normalized distances previously defined in (10), and the size of the hyperbox in each dimension is limited by a user-defined parameter η, $(0 \le \eta \le 1)$, where $c(e_{ji}, f_{ji}) \le \eta$.

Figure 6 is an example of the symmetric distance function $c(a_k, a_l)$ between the five categories of a variable and the membership function $b_j(a_k)$ obtained from the distance for the jth hyperbox that is determined by the two full-membership categories $e_j = a_3$ and $f_j = a_5$.

When there are numerical and categorical variables, the B_j hyperbox membership function—of all the dimensions—is defined by

$$\begin{aligned} &b_j(x_h, a_h) \\ &= \min\left\{ \min_{i=1,\ldots,n} \left[\min\left(1 - g\left(x_{hi}^u - w_{ji}, \gamma\right), 1 - g\left(v_{ji} - x_{hi}^l, \gamma\right)\right)\right], \right. \\ &\qquad \left. \min_{i=n+1,\ldots,n+r} \left[\min\left(1 - c_i\left(a_{hi}, e_{ji}\right), 1 - c_i\left(a_{hi}, f_{ji}\right)\right)\right] \right\} \end{aligned} \tag{12}$$

where n is the number of numerical variables and r is the number of categorical variables; g is the ramp-threshold function defined in (3); $c_i, i = n+1, \ldots, n+r$, are the normalized distances defined in (10) for the categorical dimensions; $\mathbf{x_h} = \left[\mathbf{x_h^l}\ \mathbf{x_h^u}\right]$ is the numerical input defined by its vectors of minimum $\left(x_{hi}^l\right)$ and maximum $\left(x_{hi}^u\right)$ points; $\mathbf{a_h} = (a_{hn+1}, \ldots, a_{hn+r})$ is the categorical input vector; v_{ji} is the minimum and w_{ji} is the maximum of the jth hyperbox in the ith numerical dimension, $i = 1, \ldots, n$; and e_{ji}, f_{ji} are the two categories defining hyperbox B_j in the ith categorical dimension, $i = n+1, \ldots, n+r$.

Note that the defined distance is suitable for categorical inputs with a lot of categories. When the categorical inputs are binary, the resulting distance will be the trivial:

$$c(a_{hi}, a_{hj}) = \begin{cases} 1 & \text{if } i \ne j, \quad i, j = 1, 2 \\ 0 & \text{if } i = j, \quad i, j = 1, 2 \end{cases} \tag{13}$$

When defining the hyperbox fuzzy sets for categorical variables, we also studied the use of other numbers of categories, especially just one, to determine the hyperbox fuzzy sets. But, we chose the number of two categories because it is similar to the numerical case with the maximum and minimum points, and also makes the hyperboxes in the categorical dimensions easier to update and refine during the learning step.

4.3 Extended network architecture and operation

The above membership function treats the categorical variables in a similar manner to how it processes numerical variables, where the inputs are categories in the first case and numerical hyperboxes in the second: the distances c_i play the role of functions g and they are combined by the same fuzzy operators. This straightforwardly extends neural network operation. Figure 7 shows the new network architecture including both types of variables, and Fig. 8 is the detail of an intermediate layer node.

The most important difference from Gabrys and Bargiela's network is the input layer, where, apart from the $2n$ numerical variable nodes, there are r additional nodes for the input categories, each having two connections with the second-layer nodes—one for each category e_{ji}, f_{ji} defining the B_j hyperbox.

As in the original network, the second layer maintains a node for each hyperbox. But, these are different hyperboxes because they now have categorical as well as numerical dimensions. The activation function of this second-layer is the membership function defined in (12). Its connections with the first layer are the $2(n + r)$ defined above. Apart from the $2n$ connections for the numerical features (the same B_j hyperbox minimums v_{ji} and maximums w_{ji}, $i = 1, \ldots, n$), there are the new $2r$ connections for the categorical dimensions, that is, the two categories e_{ji} and f_{ji} defining the B_j hyperbox in dimension i, $i = n+1, \ldots, n+r$.

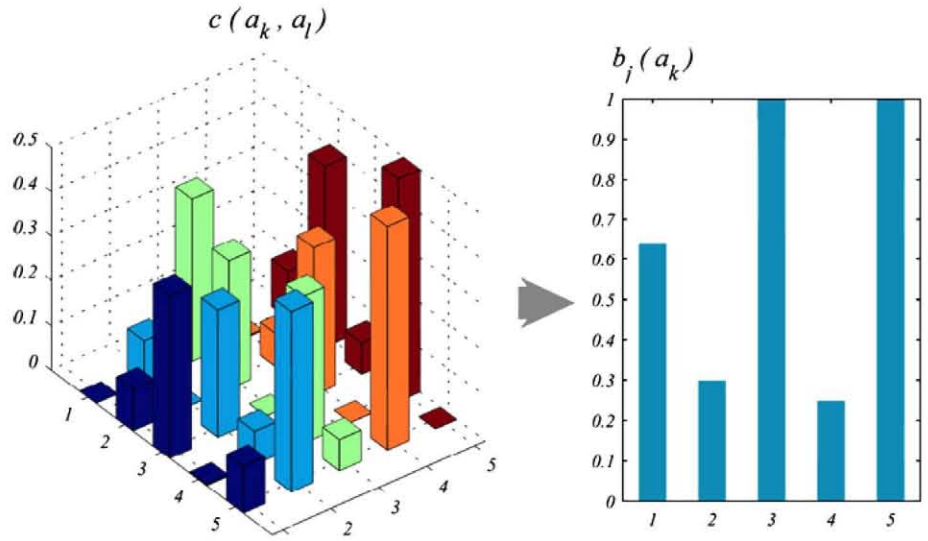

Fig. 6 The symmetric distance function between categories $c(a_k, a_l)$ and the derived membership function $b_j(a_k)$ of the hyperbox defined by categories $e_j = a_3$ and $f_j = a_5$

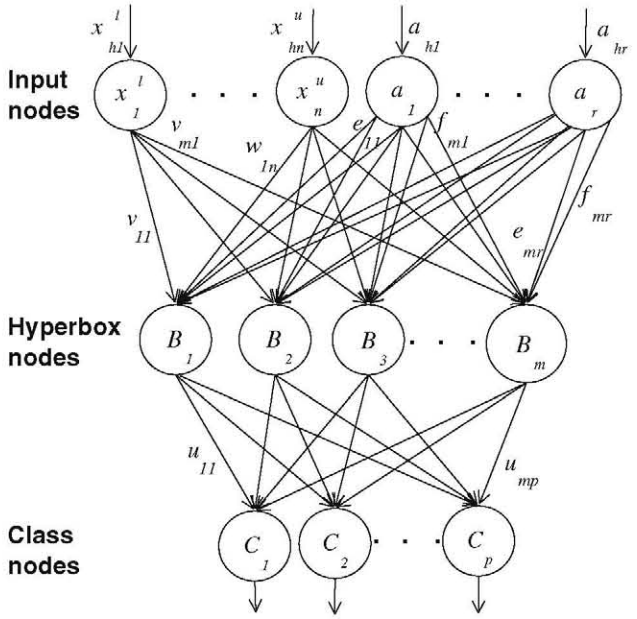

Fig. 7 Topology of the fuzzy min–max neural network implementing the new classifier

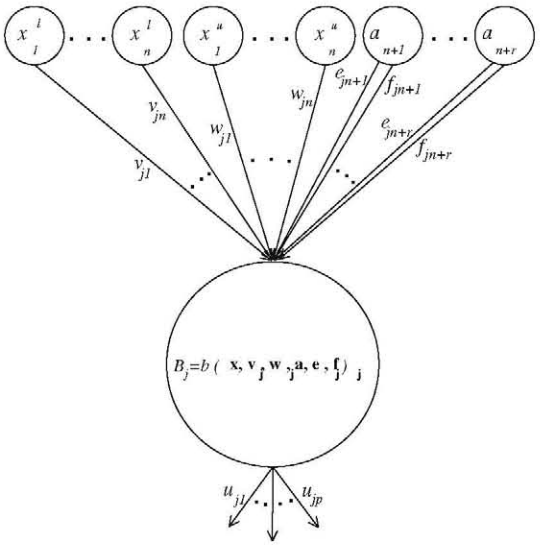

Fig. 8 Detail of the nodes connected with the jth node of the intermediate layer

Finally, like the original network, the third layer has a node for each one of the variable classification categories, and its connections with the intermediate layer are the same u_{jk} as defined in (4).

Learning in this three-layer feedforward neural network consists of creating and expanding or contracting hyperboxes. Its objective is to establish the connections v_{ji}, w_{ji}, e_{ji} and f_{ji}, that is, the hyperboxes defining each class. The first step—taken only once—is to calculate the distances between the categories of categorical variables and the resulting membership function, as described above.

This is followed by the iterative process to set and update the connection values. This process is repeated for each input and has the same steps as the original network. In fact, the procedures are exactly the same for the numerical dimensions and try to perform similar functions for the categorical dimensions. The new method proposed for the categorical dimensions results in a more complicated algorithm because of the difficulties in dealing with the finite number of categories and the more complex architectural design.

1. Initialization. When a new hyperbox B_j needs to be created for numerical dimensions, its minimum and maximum points are initially set—as in Gabrys and Bargiela's original network—to

$$v_{ji} = 1 \quad \text{and} \quad w_{ji} = 0, \quad \forall i = 1, \ldots, n \tag{14}$$

Applying this strategy, when the jth hyperbox is adjusted for the first time using the input

$\mathbf{x_h} = (x^l_{h1}, \ldots, x^l_{hn}, x^u_{h1}, \ldots, x^u_{hn})$, the minimum and maximum points of this hyperbox would be

$$v_{ji} = x^l_{hi} \quad \text{and} \quad w_{ji} = x^u_{hi} \tag{15}$$

The categorical dimensions are also initialized so that the expansion step can automatically use the hyperbox adjustment process. To do this, the new category a_{i0} is introduced in each ith categorical variable, $i = n+1, \ldots, n+r$, and each distance function definition is extended as

$$c(a_{i0}, a_{ik}) = c(a_{ik}, a_{i0}) = 0, \quad \forall i = n+1, \ldots, n+r, \ \forall k \tag{16}$$

In this way, the two categories initializing hyperbox B_j are

$$e_{ij} = a_{io} \quad \text{and} \quad f_{ij} = a_{i0}, \quad \forall i = n+1, \ldots, n+r \tag{17}$$

These values are later adjusted when the hyperbox is expanded for the first time. The role of the new category a_{i0} is just to improve the network operation, and it does not modify the aim of the learning and classification steps in any way.

2. Search for the expandable hyperbox with the highest membership degree, and expand. A network input now takes the form

$$\{\mathbf{x_h}, \mathbf{a_h}, d_h\} \tag{18}$$

where $\mathbf{x_h} = (x^l_{h1}, \ldots, x^l_{hn}, x^u_{h1}, \ldots, x^u_{hn})$, x^l_{hi} are the minimums and x^u_{hi} are the maximums of the input hyperboxes in dimension i, $i = 1, \ldots, n$; $\mathbf{a_h} = (a_{hn+1}, \ldots, a_{hn+r})$ are the input categories in dimension i, $i = n+1, \ldots, n+r$, and $d_h \in \{1, 2, \ldots, p\}$ is the index of one of the p classes. When the hth input pattern is presented, it searches the hyperbox B_j with the highest membership degree defined by (12). The first test run is to check whether the detected hyperbox and the input are members of the same class ($d_j = d_h$?). If not, it will search the hyperbox with the next highest membership degree. Once a hyperbox B_j from the same class of the input has been found, it must satisfy a number of different numerical and categorical data criteria before it can expand to include the input. For the numerical dimensions, it must meet the same condition as the original network:

$$\left(\max(w_{ji}, x^u_{hi}) - \min(v_{ji}, x^l_{hi})\right) \le \theta, \quad \forall i = 1, \ldots, n \tag{19}$$

where θ, $(0 \le \theta \le 1)$ is the user-defined parameter for the maximum size of the hyperbox in the numerical dimensions.As for the expansion of the categorical dimensions, there are different cases depending on the values of the two categories defining the hyperbox in each dimension.

Case 1: if the input value in a categorical dimension a_{hi} matches one of the values of the categories e_{ji} or f_{ji}, there is no need for expansion in this dimension.

Case 2: when $e_{ji} = a_{i0}$ and $f_{ji} = a_{i0}$, that is, neither of the two categories are preset, the hyperbox can be expanded without further testing.

Case 3: when $e_{ji} \neq a_{i0}$ and $f_{ji} = a_{i0}$, that is, when only one of the two categories defining the hyperbox is preset in the ith categorical dimension, the following criterion must be satisfied

$$c(e_{ji}, a_{hi}) \le \eta \tag{20}$$

before the hyperbox can expand, η being the user-defined parameter for the maximum size of the hyperbox in the categorical dimensions $(0 \le \eta \le 1)$.

Case 4: when $e_{ji} \neq a_{i0}$, $f_{ji} \neq a_{i0}$ and the input category for the ith dimension a_{hi} is not equal to either e_{ji} or f_{ji}, first check whether replacing either of the two categories e_{ji} or f_{ji} defining the hyperbox with the input category a_{hi} would increase the hyperbox size in this ith dimension. If so, later test criterion (20) defining the maximum size of the resulting hyperbox.

After verifying the criteria for the numerical and categorical dimensions, the expandable hyperbox B_j is adjusted to include the input by setting the numerical dimensions $i = 1, \ldots, n$ as

$$v^{\text{new}}_{ji} = \min\left(v^{\text{old}}_{ji}, x^l_{hi}\right) \tag{21}$$

$$w^{\text{new}}_{ji} = \max\left(w^{\text{old}}_{ji}, x^u_{hi}\right) \tag{22}$$

and setting the categorical dimensions $i = n+1, \ldots, n+r$ as

Case 1: $e_{ji} = a_{i0}$ and $f_{ji} = a_{i0} \Rightarrow e_{ji} = a_{hi}$

Case 2: $e_{ji} \neq a_{i0}$ and $f_{ji} = a_{i0} \Rightarrow f_{ji} = a_{hi}$

Case 3: $e_{ji} \neq a_{i0}$ and $f_{ji} \neq a_{i0}$ and $c(e_{ji}, a_{hi}) > c(e_{ji}, f_{ji}) \Rightarrow f_{ji} = a_{hi}$

Case 4: $e_{ji} \neq a_{i0}$ and $f_{ji} \neq a_{i0}$ and $c(a_{hi}, f_{ji}) > c(e_{ji}, f_{ji}) \Rightarrow e_{ji} = a_{hi}$

If neither of the existing hyperboxes include or can expand to include the input, then a new hyperbox B_j is initialized, adjusted, and labeled by setting

$$\text{class}(B_j) = d_h \tag{23}$$

3. Overlapping hyperboxes test. All the numerical and categorical dimensions must be checked for a

non-empty overlap between full-membership portions of hyperboxes representing different classes, in order to prevent an input pattern from being classified in two or more different classes at the same time. Hyperboxes with only one non-overlapping dimension—numerical or categorical—would pass the test.

4. Hyperboxes contraction according to the test result. Only if the overlap test result is positive, that is, when there is a non-empty overlap in all the numerical and categorical dimensions, are the hyperboxes contracted, following the minimum change principle, in a single dimension starting with the categorical dimensions. We try to change the overlapping category of the existing hyperbox for another one reducing the hyperbox size, that is, another category closer to the remaining category defining the hyperbox, in one of these dimensions. If this is possible, it is replaced—eliminating the overlap—and, if not, we try to contract in another dimension (it might not always be feasible to contract hyperboxes in this way in a given categorical dimension). When there are no more categorical dimensions left, we move on to the numerical dimensions. Contraction is always possible in numerical dimensions, and it is performed as defined for the original network, distributing the overlapping space between the two hyperboxes [18].

This learning algorithm is guaranteed to convergence because the extension designed for the categorical inputs is based on the previously defined metric between a finite number of categories.

Finally, the new network operates similarly to its predecessor in terms of classification: it is assigned the category corresponding to the class with the highest membership degree.

Let us look at a simple example based on data from Table 2, to illustrate this procedure. The *region* is the categorical input variable, whereas $X = \text{age}/100$ is the numerical input variable and *employment situation* is the categorical variable to be imputed. First, we calculate the Euclidean distances between the R^4 row profile vectors in Table 2, and then we divide by the greatest of these distances to get the distances between *regions* listed in Table 3.

Now, suppose that result of the above learning steps are the three hyperboxes shown in Table 4 defining three different classes:

Then, we calculate the three hyperbox membership degrees of the input case $z = (0.50, \text{West})$ to be imputed:

$$\begin{aligned} b_{H1}(z) &= \min\{1, \min[1 - 0.794943, 1 - 0.893085]\} \\ &= \min\{1, 0.106915\} = 0.106915 \\ b_{H2}(z) &= \min\{1, \min[1 - 0.392967, 1 - 0.794943]\} \\ &= \min\{1, 0.205057\} = 0.205057 \\ b_{H3}(z) &= \min\{0, \min[1 - 0.893085, 1 - 0.000000]\} \\ &= \min\{0, 0.106915\} = 0.000000 \end{aligned}$$

As hyperbox $H2$ defines the class with the highest membership degree, the category *Unemployed* is assigned to the z input case.

4.4 Categorical missing values treatment

Numerical missing data inputs are treated in the same way as proposed by Gabrys [19]. We also define a *toleration technique* [26] for the inputs with categorical missing values. This technique works directly with data sets containing missing data without making imputations as follows.

Categorical values could be missing at two different stages of the designed operation. First, they could be missing when calculating frequencies and distances between categorical variable categories. In this case, the calculations would be made using exclusively non-missing data, as is usual practice in most statistical software packages. Secondly, categorical data required to set and update the connections could also be missing during the iterative process. The method for dealing with this is also designed to use the other variables with non-missing data as though there were no missing attributes for this input.

Table 3 Distances between regions

	North	West	Center	East	South
North	0.000000	0.794943	0.119344	0.888457	0.839879
West	0.794943	0.000000	0.893085	0.392967	0.067065
Center	0.119344	0.893085	0.000000	1.000000	0.941306
East	0.888457	0.392967	1.000000	0.000000	0.346324
South	0.839879	0.067065	0.941306	0.346324	0.000000

Table 4 Example of hyperboxes created after the learning

Hyperbox	X input	Region input	Class or category of imputation
H1	[0.49, 0.52]	North, Center	Employed
H2	[0.46, 0.51]	East, North	Unemployed
H3	[0.86, 0.93]	Center, West	Retired

This is done by making the hyperbox membership degree equal to one for the corresponding dimension and all hyperboxes.

The designed method always takes advantage of all the available information. This is useful when there are a lot of variables or attributes and they all have missing values.

5 Case study: application to voting intention imputation in a political poll

A frequent procedure used to collect information about a population is to take a survey. When the questions refer to individual opinions or attitudes, these surveys are known as opinion polls [29, 30]. These polls have proven to be an especially fast and easy-to-use tool, because they simplify the most technical phases of the survey process. As in most surveys, there is usually total or partial non-response—when a respondent fails to answer all or some of the questions, respectively. The procedure for total non-response is usually addressed at the sampling design stage. This paper focuses on partial non-response.

Partial non-response is generally solved by imputing values to the missing variables from the answers of other respondents and from the non-missing variables in responses by the same individual. However, the usual way of dealing with non-response in polls is to add the "don't know/not applicable" category and treat it like any other category. Little and Rubin [31] argue that this is not a highly recommendable method because it can cause problems at the results analysis stage, but it is widely applied in polls due to its straightforwardness.

In election polls, though, there is one variable—*which political party do you intend to vote for in the next general elections* (*voting intention*, from now on)—for which the above procedure is not good enough, and missing values were imputed using other methods. Elsewhere, we presented a paper where fuzzy control procedures were used to estimate voting intention in an electoral poll [32]. It stressed the potential of using methods to automatically obtain fuzzy set membership functions. This is what we do now using neural networks, by imputing missing voting intention from the responses to other questions in the same survey.

Different procedures based on neural networks have been used to impute numerical variables from other likewise numerical values [33–35]. We are not aware of their use for imputing categorical variables from other numerical and categorical variables, as proposed in this paper.

To evaluate the operation of the proposed neuro-fuzzy classifier, we selected polls number 2555 and 2750 from the Sociological Research Center's catalog (the Sociological Research Center is an institution responsible for making opinion polls for the Spanish Public Administration). These surveys refer to the general elections held in Spain in 2004 and 2008. They contain 16,345 and 13,280 interviews, respectively, with an answer to the *voting intention* question. The chosen polls contain questions with different types of variables:

- *Quantitative variables.* Questions answered by entering a numerical value. They include questions referring to *ideological self-location* (the result of asking respondents to place themselves ideologically on a scale of 1–10, 1 being the extreme left and 10 the extreme right). Other possibilities are, the *rating of three specific political figures*, *likelihood to vote,* and *likelihood to vote for three specific political parties,* all of which are rated on a scale of 0–10.
- *Ordered categorical variables.* Questions answered by entering categories that are so well ordered that they are easy and straightforward to transform into quantitative variables. They refer to *government* and *opposition party ratings*. The answer categories are "*very good*", "*good*", "*fair*", "*bad*" and "*very bad*", which we transform into the values 1, 0.75, 0.5, 0.25 and 0, respectively, assuming they are ordered equidistantly. They should take values within the unit interval like the membership functions of fuzzy sets.
- *Categorical variables with non-ordered categories.* Questions including *voting intention* and similar, such as *vote memory* (party the respondent voted for at the last general election); the *Autonomous Community; which of the likely candidates the respondent would prefer to see as president of the government; how sure/definite the respondents' voting intention is; the political party the respondent tips to win* and *the political party the respondent would prefer to win.*

Although missing values are found in all the above variables, this paper focuses on the imputation of the categorical *voting intention* variable which is, thus, the classification feature. Our method will deal with missing data in other variables as explained in Sect. 3, depending on the variable type. We will explain the procedure for dealing with missing data when we present other methods for comparison.

For the purposes of imputation, each class or classification category is matched with one of the different values the variable to be imputed takes. So, the imputed value is the category corresponding to the class with the greatest membership degree.

Eleven categories have been taken for the *voting intention* variable, including the most important political parties' names, "*blank vote*", "*abstention*" and a category of "*others*". This would appear to be quite a good granularity level for obtaining reliable proportions for

nationwide *voting intention*, whereas a larger granularity would make the problem tougher. The sixteen numerical and ordered and non-ordered categorical variables described above are used as classifier inputs for both of the surveys.

The performance of the proposed method is then compared with other classical approaches. For the comparisons, we used an evaluation criterion frequently used in the supervised classification procedures area: the correctly imputed rate, that is, the percentage of imputed values that exactly match the original data over the inputs with non-missing voting intention. A tenfold cross-validation, partitioning the test data into ten parts (folds), is performed. We retain a single fold as the validation data for testing the model, whereas the remaining nine are used as training data. The cross-validation process is then repeated 10 times with each of the tenfolds, and the results are averaged to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. This procedure provides non-biased estimations of the correctly imputed rate [36].

One of the procedures used nowadays for single imputation of the voting intention variable is to make predictions from logistic regressions on other variables, and this is taken as a baseline for comparison. The tenfold cross-validation of the data sets with logistic regression and the sixteen variables (generated using SAS/STATS software, Version 9.1.3 of the SAS System for Windows. Copyright 2002–2003 by SAS Institute Inc., Cary, NC, USA), returns the results shown in Table 5.

Note that, we are likely to come across some problems using logistic regression to impute missing values. First of all, the likelihood equation for a logistic regression model does not always have a finite solution, making it difficult to estimate model parameters. Sometimes, there is a non-unique maximum on the boundary of the parameter space at infinity. The existence, finiteness, and uniqueness of maximum-likelihood estimates for the logistic regression model depend on the patterns of data points in the observation space. When there is a complete or quasi-complete separation, there exist infinite estimations, and only if there is an overlap of sample points do unique maximum likelihood estimates exist [37]. In our case, there is the possibility of separation because of the great many variables and categories, and the output models are questionable.

Table 5 Correctly imputed rate for the logistic regression imputations

Dataset	% Correctly imputed
2555	64.20
2750	63.05

A second problem with the use of logistic regression is that units with missing values in one or more input variables are deleted, reducing the learning set size.

To make an additional comparison using the same fuzzy min–max neural network classifier, we looked at another distance frequently used with categorical variables: if a_h, a_j are two categories, then

$$c_3(a_h, a_j) = 1 - \delta_{hj} \tag{24}$$

where δ_{hj} is the Kronecker delta. The resulting hyperbox membership function is then defined by

$$\begin{aligned} &b_j(x_h, a_h) \\ &= \min\left\{ \min_{i=1,\ldots,n} \left[\min\left(1 - f(x^u_{hi} - w_{ji}, \gamma), \left(1 - f(v_{ji} - x^l_{hi}, \gamma)\right)\right)\right], \right. \\ &\qquad \left. \min_{i=r+1,\ldots,n+r} \left[1 - c_3(a_{hi}, e_{ji})\right] \right\} \end{aligned} \tag{25}$$

where e_{ji} is the only category defining the hyperbox B_j in the ith dimension. (Note that this distance has no need of the η parameter because η does not make sense if there is only one category.) In this case, the membership function portion corresponding to a categorical dimension

$$\min_{i=n+1,\ldots,n+r} \left[1 - c_3(a_{hi}, e_{ji})\right] = \min_{i=n+1,\ldots,n+r} \left[1 - \delta(a_{hi}, e_{ji})\right] \tag{26}$$

takes only values 1 (when all the categorical inputs are equal to each matching hyperbox category) and 0. As a result, this Kronecker distance works by learning separate numerical variables for each combination of categorical variables.

The experiment run implements a classifier for each one of the three membership functions resulting from the three distances. As the designed networks have some user-defined parameters for adjustment (the maximum numerical hyperbox size θ, the numerical membership function decreasing parameter γ, and the maximum categorical hyperbox size η), estimations have been made for the set of parameter combinations resulting from $\gamma = 0.5$, 1.5, 2.5, 3.5, 4.5, $\theta = 0.25$, 0.35, 0.45, 0.55, 065 and $\eta = 0.25$, 0.35, 0.45, 0.55, 0.65.

Tables 6, 7 and 8 show the correctly imputed rates with the tenfold cross-validation for the parameter combinations returning the best results for each membership function and each dataset in decreasing order of these rates.

The level of the scores reached with each distance is similar for both datasets, but the combinations of the user-defined parameters with the best results are different. This reflects the fact that the input variables are not exactly the same in each dataset.

An important feature or weakness of this kind of learning method is that the learning set order may have an

Table 6 Correctly imputed rate for the proposed method imputations using the Euclidean distance

Dataset 2555				Dataset 2750			
γ	θ	δ	% Correctly imputed	γ	θ	δ	% Correctly imputed
1.5	0.35	0.65	85.63	2.5	0.55	0.55	86.06
0.5	0.45	0.55	85.54	2.5	0.65	0.35	85.95
1.5	0.35	0.55	85.54	2.5	0.65	0.25	85.94
0.5	0.45	0.45	85.46	2.5	0.55	0.35	85.93
0.5	0.35	0.25	85.34	2.5	0.55	0.45	85.93
0.5	0.45	0.35	85.21	1.5	0.45	0.55	85.91
1.5	0.45	0.65	85.21	2.5	0.55	0.25	85.91
0.5	0.45	0.25	85.19	1.5	0.45	0.65	85.89
0.5	0.35	0.45	85.17	1.5	0.25	0.65	85.88
1.5	0.55	0.55	85.16	1.5	0.35	0.65	85.88
1.5	0.35	0.45	85.05	1.5	0.45	0.45	85.88
0.5	0.45	0.65	84.98	2.5	0.45	0.45	85.88
0.5	0.35	0.55	84.97	1.5	0.35	0.45	85.81
1.5	0.55	0.45	84.94	2.5	0.35	0.65	85.79
1.5	0.45	0.25	84.92	2.5	0.45	0.55	85.79
0.5	0.25	0.65	84.9	1.5	0.65	0.25	85.76
1.5	0.55	0.25	84.9	2.5	0.45	0.35	85.76
1.5	0.45	0.35	84.87	2.5	0.45	0.65	85.76
0.5	0.35	0.65	84.86	1.5	0.45	0.25	85.68
0.5	0.35	0.35	84.85	1.5	0.45	0.35	85.66

impact on the results. The validation process has been repeated several times with a number of different randomizations of the input datasets to deal with this problem. The resulting rates were similar, thereby confirming the method's robustness.

6 Conclusions

We have shown how the fuzzy min–max neural network classifier could be extended to admit categorical inputs and the results of using the method for missing data imputation in opinion polls. It is possible to extract some conclusions from Tables 6, 7 and 8:

- The correctly imputed rates for the Euclidean and the logarithmic distance are significantly greater than for the Kronecker distance and logistic regression. Results are up around 11 percentage points over the Kronecker distance and 21 percentage points over logistic regression in each input dataset. The results range—up to 86%, even with a great many classification categories—is much better than what is usually achieved in similar polls.
- No significant difference has been found between the behavior of the Euclidean and logarithmic distances in

Table 7 Correctly imputed rate for the proposed method imputations using the logarithmic distance

Dataset 2555				Dataset 2750			
γ	θ	δ	% Correctly imputed	γ	θ	δ	% Correctly imputed
0.5	0.35	0.25	85.57	0.5	0.35	0.65	85.21
0.5	0.45	0.25	85.55	1.5	0.35	0.65	85.18
0.5	0.35	0.35	84.85	0.5	0.35	0.55	85.06
0.5	0.25	0.25	84.55	0.5	0.25	0.55	84.9
0.5	0.65	0.25	84.53	0.5	0.35	0.45	84.86
0.5	0.25	0.35	84.47	0.5	0.25	0.65	84.82
0.5	0.35	0.65	84.38	0.5	0.25	0.45	84.77
0.5	0.45	0.35	83.98	0.5	0.45	0.25	84.73
0.5	0.25	0.65	83.92	0.5	0.25	0.35	84.69
0.5	0.25	0.55	83.9	1.5	0.45	0.25	84.67
0.5	0.35	0.55	83.57	0.5	0.35	0.25	84.64
0.5	0.35	0.45	83.48	1.5	0.35	0.35	84.64
0.5	0.55	0.25	83.48	1.5	0.35	0.45	84.63
0.5	0.25	0.45	83.37	1.5	0.45	0.35	84.62
0.5	0.45	0.45	82.96	0.5	0.25	0.25	84.57
0.5	0.45	0.65	82.75	0.5	0.45	0.55	84.57
1.5	0.55	0.25	82.31	0.5	0.35	0.35	84.49
0.5	0.45	0.55	82.01	0.5	0.45	0.35	84.48
1.5	0.65	0.25	81.72	0.5	0.45	0.45	84.47
1.5	0.45	0.25	81.44	1.5	0.25	0.45	84.45

Table 8 Correctly imputed rate for the proposed method imputations using the Kronecker distance

Data set 2555			Data set 2750		
γ	θ	% Correctly imputed	γ	θ	% Correctly imputed
0.5	0.35	76.12	0.5	0.45	72.65
0.5	0.45	75.99	0.5	0.35	72.46
0.5	0.55	75.69	0.5	0.25	72.42
0.5	0.65	75.69	1.5	0.55	72.02
0.5	0.25	75.19	0.5	0.15	71.95
1.5	0.35	75.08	1.5	0.45	71.93
1.5	0.45	74.96	1.5	0.25	71.73
1.5	0.55	74.88	1.5	0.35	71.73
1.5	0.65	74.75	1.5	0.15	71.15
1.5	0.25	74.33	2.5	0.55	67.41
2.5	0.65	71.21	2.5	0.45	66.98
2.5	0.55	71.19	2.5	0.35	66.74
2.5	0.35	71.01	2.5	0.25	66.58
2.5	0.45	70.98	2.5	0.15	66.2
2.5	0.25	70.19	3.5	0.55	63.25
3.5	0.65	63.68	3.5	0.45	63.01
3.5	0.55	63.67	3.5	0.35	62.55
3.5	0.45	63.36	3.5	0.25	62.17
3.5	0.35	63.23	3.5	0.15	61.19
3.5	0.25	63.01	4.5	0.55	53.91

any of the datasets. Thus, the logarithmic distance does not appear to solve potential problems stemming from proportionally short distances between high input values. The question requires more thorough investigation before either of these distances is selected.

- Gabrys and Bargiela propose the use of different parameters θ and γ for each numerical dimension. The same parameters were used here, and we were able to improve results by varying the γ, θ and η thresholds in each dimension.
- The procedure presented here, proves to be especially apt if there is a relatively high number of classification categories, as opposed to the more commonly dealt with case of binary variables with just two categories.
- Also, note that the proposed neuro-fuzzy classifier is well suited when there are a lot of numerical and categorical input variables. In the case of missing values in input datasets, logistic regression estimations take into account only the complete data patterns. As a result, the number of inputs decreases dangerously when there are a lot of variables all with non-response. The proposed procedure always uses all the available data in the most efficient way, and the more variables there are, the better the results will be. Using this method, the select variables step could be eliminated, leading to more automatic imputation.
- Another important point is that the neuro-fuzzy classifier proposed here, works efficiently when there are the two types of inputs—numerical and categorical—in the learning dataset. It does not appear to be suitable when inputs are exclusively categorical variables because of the subsidiary role the categorical variables play at the contraction step. Further work will focus on testing the procedure in this case.

References

1. Rubin DB (1976) Inference and missing data. Biometrika 63:581–592
2. Rubin DB (1977) Formalizing subjective notions about the effect of non-respondents in sample surveys. J Am Stat Assoc 72(359): 538–543
3. Dempster P, Rubin DB (1983) Incomplete data in sample surveys. In: Madow WG, Olkin I, Rubin DB (eds) Sample surveys. II. Theory and Annotated Bibliograph. Academic Press, New York
4. Schafer JL, Graham JW (2002) Missing data: our view of the state of the art. Psychol Methods 7(2):147–177
5. Durrant GB (2005) Imputation methods for handling item-non-response in the social sciences: a methodological review. Tech. Rep. NCRM/002, National Centre for Research Methods and Southampton Statistical Sciences Research Institute, University of Southampton
6. Myrtveit I, Stensrud E, Olsson U (2002) Analyzing data sets with missing data: an empirical evaluation of imputation methods and likelihood-based methods. IEEE Trans Softw Eng 27(11):999–1013
7. Klir G, Yuan B (1995) Fuzzy sets and fuzzy logic, theory and applications. Prentice-Hall, New Jersey
8. Tanaka K (1997) An introduction to fuzzy logic for practical applications. Springer, New York
9. Yager RR, Filev DP (1996) Relational partitioning of fuzzy rules. Fuzzy Sets Syst 80(1):57–69
10. Dubois D, Prade H (1996) What are fuzzy rules and how to use them. Fuzzy Sets Syst 84(2):169–185
11. Pedrycz W (1992) Fuzzy neural networks with reference neurons as pattern classifiers. IEEE Trans Neural Netw 3(5):770–775
12. Mitra S, Pal SK (1994) Self-organizing neural network as a fuzzy classifier. IEEE Trans Syst Man Cybern A Syst Hum 24(3):385–399
13. Meneganti M, Saviello FS, Tagliaferri R (1998) Fuzzy neural networks for classification and detection of anomalies. IEEE Trans Neural Netw 9(5):848–861
14. Gabrys B (2004) Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine? Fuzzy Sets Syst 147:39–56
15. Mitra S, Pal SK, Mitra P (2002) Data mining in soft computing framework: a survey. IEEE Trans Neural Netw 13(1):3–14
16. Simpson PK (1992) Fuzzy min–max neural networks—part 1: classification. IEEE Trans Neural Netw 3:776–786
17. Simpson PK (1993) Fuzzy min–max neural networks—part 2: clustering. IEEE Trans Fuzzy Syst 1:32–45
18. Gabrys B, Bargiela A (2000) General fuzzy min–max neural network for clustering and classification. IEEE Trans Neural Netw 11:769–783
19. Gabrys B (2002) Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems. Int J Approx Reason 30:149–179
20. Quteishat M, Lim CP (2006) A modified fuzzy min–max neural network and its application to fault classification. In: 11th Online world conference soft computing in industrial applications (WSC11)
21. Gabrys B (2002) Agglomerative learning algorithms for general fuzzy min–max neural network. J VLSI Signal Process 32:67–82
22. Bargiela A, Pedrycz W, Tanaka M (2004) An inclusion/exclusion fuzzy hyperbox classifier. Int J Knowl Based Intell Eng Syst 8(2):91–98
23. Nandedkar P, Biswas PK (2007) A fuzzy min–max neural network classifier with compensatory neuron architecture. IEEE Trans Neural Netw 18(1):42–54
24. Brouwer RK (2002) A feed-forward network for input which is both categorical and quantitative. Neural Netw 15(7):881–890
25. Farhangfar A, Kurgan LA, Pedrycz W (2007) A novel framework for imputation of missing values in databases. IEEE Trans Syst Man Cybern A Syst Hum 37(5):692–709
26. Zhang X, Hang CH, Tan S, Wang P (1996) The min–max function differentiation and training of fuzzy neural networks. IEEE Trans Neural Netw 7(5):1139–1150
27. Song Q, Shepperd M (2007) Missing data imputation techniques. Int J Bus Intell Data Min 2(3):262–291
28. Greenacre MJ (1984) Theory and applications of correspondence analysis. Academic Press, London
29. Cox R (2006) Principles of statistical inference. Cambridge University Press, Cambridge
30. Allison P (2002) Missing data. Sage, California
31. Little RJ, Rubin DB (2002) Statistical analysis with missing data, 2nd edn. Wiley, New York
32. Cardeñosa J, Rey-del-Castillo P (2007) A fuzzy control approach for vote estimation. In: Proceedings of 5th international conference on information technologies and applications, Varna
33. Abdella M, Marwala T (2005) The use of genetic algorithms and neural networks to approximate missing data in databases. In: IEEE 3rd international conference on computational cybernetics, pp 207–212

34. Nelwamondo V, Mohamed S, Marwala T (2007) Missing data: a comparison of neural network and expectation maximization techniques. Curr Sci 93(11):1514–1521
35. Lingras P, Zhong M, Sharma S (2008) Evolutionary regression and neural imputations of missing values. In: soft computing applications in industry. Studies in Fuzziness and Soft Computing Series, vol 226. Springer, Berlin, pp 151–163
36. Witten H, Frank E (2005) Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, USA
37. Santner TJ, Duffy DE (1986) A note on A. Albert and J. A. Anderson's conditions for the existence of maximum likelihood estimates in logistic regression models. Biometrika 73:755–758