Neural Computing and Applications (2023) 35:7267–7281
https://doi.org/10.1007/s00521-021-06267-1

S.I. : NEURO, FUZZY AND THEIR HYBRIDIZATION

Classification of incomplete data integrating neural networks and evidential reasoning

Suvra Jyoti Choudhury[1] · Nikhil R. Pal[1]

Received: 13 October 2020 / Accepted: 26 June 2021 / Published online: 25 October 2021

Abstract

When missing data are imputed by any method, there is some uncertainty associated with the imputed value. Consequently, when such imputed data are classified, some uncertainty will be propagated to the classifier output. This leads to two issues to address. First, reducing the uncertainty in the imputed value. Second, modeling and processing of the uncertainty associated with the classifier output to arrive at a better decision. To deal with the first issue, we use a latent space representation, while for the second issue we use Dempster-Shafer evidence theory. First, we train a neural network using the data without any missing value to generate a latent space representation of the input. The complete data set is now extended by deleting every feature once. These missing values are estimated using a nearest neighbor-based scheme. The network is then refined using this extended dataset to obtain a better latent space. This mechanism is expected to reduce the effect of the missing data on the latent space representation. Using the latent space representation of the complete data, we train two classifiers, support vector machines and evidential *t*-nearest neighbors. To classify an input with a missing value, we make a rough estimate of the missing value using the nearest neighbor rule and generate its latent space representation for classification by the classifiers. Using each classifier output, we generate a basic probability assignment (BPA) and all BPAs are combined to get an overall BPA. Final classification is done using Pignistic probabilities computed on the overall BPA. We use three different ways to defining BPAs. To avoid some problems of Dempster's rule of aggregation, we also use several alternative aggregations including some T-norm-based methods. Note that, T-norm has been used for combination of belief function in Pichon and Denœux (in: NAFIPS 2008: 2008 annual meeting of the North American fuzzy information processing society, pp 1–6, 2008). To demonstrate the superiority of the proposed method, we compare its performance with four state-of-the-art techniques using both artificial and real datasets.

Keywords Missing data · Neural networks · Autoencoder · Basic probability assignment · Belief function · Evidential K-NN · Evidential reasoning · Latent space representation

1 Introduction

Missing data are often encountered in many real life problems [11]. Let $\mathbf{x}_k$ be a data point, $\mathbf{x}_k \in \mathbf{X} \subseteq R^p$, $\mathbf{X}$ is the data set. If $\mathbf{x}_i$, for at least one i, has l, $0<l<p$ missing values then $\mathbf{X}$ is an incomplete dataset. Missing data are generally grouped into three types [19]: (1) MCAR (Missing completely at random), (2) MAR (Missing at random) and (3) NMAR (Not missing at random). Most missing value prediction methods are for MCAR and MAR types of missing data.

The MCAR and MAR type of missing data can be easily dealt with by restricting the analysis only to data points without any missing information [19, 33]. But, this is useful only when a small number of instances have missing values. One can also predict (impute) missing values and then the analysis can be done on that data. In [11], imputation techniques are divided into two groups: statistical imputation methods [1, 19, 33] and imputation based on machine learning [8, 25]. An example of the statistical imputation method is mean value imputation, where the

✉ Suvra Jyoti Choudhury
suvra_r@isical.ac.in

Nikhil R. Pal
nikhil@isical.ac.in

[1] Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700108, India

missing value of a feature is imputed by the mean value of that feature computed over the instances with all features present. Another example is cold and hot-deck imputation [14] which imputes missing values with the feature value of the closest complete data point where the closeness is computed using the feature values that are present. The closest data point can be chosen from the same data set or different data sets. In multiple imputations techniques, a missing value is imputed by a set of probable values [15, 36] and thereby generating multiple datasets.

Now, we discuss a few imputation methods using machine learning procedures. In the t-nearest neighbors (t-NN) [8] based approach, the missing value is replaced by the nearest neighbor ($t = 1$) and the distance is computed in the observed subspace. In [25], a distance-weighted modified t-nearest neighbor-based rule is proposed to classify instances with missing data. On the other hand, in [8], missing data are imputed using the t-most significant eigenvectors. Many other machine learning techniques have been used for handling missing values. For example, self-organizing maps (SOMs), multilayer perceptron (MLP) [10, 12, 13, 22, 28, 32, 36, 37, 43], and autoencoders (AEs) [5, 16, 23, 24, 26, 27, 31, 41] have been used.

Evidential reasoning has been used in many fields like data clustering, classification, and decision-making [20]. The evidence theory has also been used in handling missing data[21] in the context of classification. In [21], first, a prototype is generated for each class. These prototypes are then used to impute incomplete data. Thus, for an r-class problem, for an incomplete instance, this method will generate r complete data points. After that, each new data point is classified using any well-known classifier and the results of the classifier are merged based on a new prototype-based credal classification (PCC) method. Here, first the probabilistic output of a test point is determined by each of the r classifiers. The probabilistic output of each classifier is multiplied by the weight of the classifier which is determined by the distance of the test point to the prototype of that classifier. Using this weighted probabilistic output of a classifier, BPA for a particular test point for that classifier is determined. Then, the authors proposed a new type of aggregation rule to join different BPAs found from different results. Note that the aggregation rule is almost the same as Dempster's rule for joining BPAs. The PCC method is used to find the right class label of the incomplete data point. Here, a test data point may belong to more than one class. To account for this in [21], the authors defined two types of errors: normal classification error and the belongingness of a test point to more than one class which includes the correct class. However, only the normal classification error is used to compare their method with other methods [21].

Here, we use evidential reasoning to develop classification algorithms that can deal with missing data. First, we train an autoencoder using the complete data and an extended version of the complete data. If there are r classes, $(r + 1)$ classifiers are trained using the latent space representation of the complete data. As classifiers we use SVMs and EV-NN. Now to classify a test data point with missing attributes, we impute it using our method and generate its latent space representation, which is then classified using the r SVMs and the EV-NN. The output of each classifier is given a probabilistic representation. These probabilistic outputs are used to assign $r + 1$ BPAs, which are aggregated using Dempster's rule as well as several other rules. Here, we propose two methods to define BPA from the classifier outputs, which are comparatively simpler than the method in [21]. We also generate two types of errors as in [21]. As in [21], we have done four experiments to check the performance of our algorithms with others and compared the results of the proposed algorithm with four state-of-the-art techniques in terms of the usual classification errors. Our results revealed that the performance of the proposed method is better compared with the other methods. We have also done various experiments to check the influence of each component of the proposed method. In [2], some preliminary results of these investigations are reported, where we use only one way to define BPAs and use only the Dempster's rule of aggregation.

Many authors used auto-encoders with a bottleneck layer for missing data handling [5, 16, 22–24, 26, 27, 31, 41]. The use of auto-encoders with a bottleneck layer is useful for many applications, like data compression and feature extraction. But the primary objective of the proposed method is a better representation of the inputs. For this, we do not use a bottleneck layer for the encoding of inputs but a latent space of much higher dimension than that of the input. As in [2, 4], in our two-stage training scheme, we innovatively use the complete data so that the autoencoder is better equipped to deal with missing values. Like [2, 4] we use here a modified form of the 1-nearest-neighbor rule for imputing missing values. Moreover, we have proposed a very simple scheme for assigning the mass functions using the probabilistic outputs from SVMs and the probabilistic output of the trained EV-NN. Here, we combine latent space representation of auto-encoders with evidence theory via classifiers' outputs. We use the latent space representation of missing values to represent the imputed dataset in a better way. An imputed missing data point may be part of different classes. To handle this, we use evidence theory.

Note that in place of SVM or EV-NN, we can use any other classifiers whose outputs are probabilistic in nature or which can be given a probabilistic interpretation with a suitable transformation. For example, in case of a multilayer perceptron, we can use the soft-max operator to convert its output to a probabilistic one and then use the proposed method. The conversion into probabilistic outputs makes it convenient to define BPAs, but it is not necessary. Any mechanism that can define sensible BPAs from the classifier output can be used in our framework.

The remaining part of the article is organized as follows. The basics of evidential reasoning are described in Sect. 2 and the proposed method in Sect. 3. In Sect. 4, we present experimental results and their analysis, while our conclusions are drawn in Sect. 5.

2 Basics of evidential reasoning

Evidential reasoning is a popular technique for decision making under uncertainty [35, 38–40]. Let, an input $\mathbf{x} \in \mathbb{R}^p$ belong to one of the r classes in an r-class classification problem. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_r\}$ be the set of classes. The available evidence may suggest that a data point belongs to more than one class due to imprecision or some other uncertainty asociated with the input. Thus a test point may belong to any one of the $2^{|\Omega|}$ sets of classes. As an example, let there be three classes and the classes are $\Omega = \{\omega_1, \omega_2, \omega_3\}$. So, a given test point may belong to any one of the $2^{|\Omega|}$ sets of the classes: $2^\Omega = \{\emptyset, \{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_1, \omega_2\}, \{\omega_1, \omega_3\}, \{\omega_2, \omega_3\}, \{\omega_1, \omega_2, \omega_3\}\}$. Thus, in evidential reasoning, a test point may belong to meta-classes or in no class.

In evidential reasoning, BPA is defined by a function $m(\cdot) : 2^\Omega \rightarrow [0, 1]$ satisfying two properties: $\sum_{A \in 2^\Omega} m(A) = 1$ and $m(\emptyset) = 0$.

If there are multiple sources of evidence and each is represented by a BPA then the multiple evidences are combined by the Dempster–Shafer (DS) rule [35] and the combination operator is denoted by the $\oplus$. If $m_1(\cdot)$ and $m_2(\cdot)$ are two BBAs over 2^Ω then the combined BPA $m = m_1 \oplus m_2$ is defined as:

$$m(A) = [m_1 \oplus m_2](A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{B \cap C = \emptyset} m_1(B) m_2(C)}; \qquad \text{and} \quad m(\emptyset) = 0. \tag{1}$$

The denominator in Eq. (1) is used for normalization, and $\sum_{B \cap C = \emptyset} m_1(B) m_2(C)$ is the total conflicting belief mass.

In our problem $\Omega = \{1, 2, \ldots, r\}$, the set of classes. Given a test point $\mathbf{x}$, we shall generate several BPAs and combine them. Finally, we shall compute the Pignistic Probability using the combined BPA for class k as in (2) to decide the classes.

$$P_x\{k\} = \sum_{\mathbf{A \subseteq \Omega, k \in A m_x (A) / |A|}} . \tag{2}$$

2.1 Issues with Dempster's rule and possible solutions

So far, we restricted our discussion to Dempster's rule of aggregation. However, this rule does not produce desirable results when the pieces of evidence have high conflicts [35]. To demonstrate this, consider an example suggested by Zadeh [45]. Suppose we have two BPAs, m_1 and m_2, defined on the same universe of three diseases, $\Omega = \{a, b, c\}$ as follows:

$\text{BPA}_1 : m_1(\{a\}) = 0.99; m_1(\{b\}) = 0.01.$
$\text{BPA}_2 : m_1(\{b\}) = 0.01; m_2((\{c\}) = 0.99.$

Now, if we combine the two BPAs using Dempster's rule, we get $m_{12}\{(b)\} = 1.0$, and the confidence on a and c or any other subsets are zero. Given that the disease b was practically rejected by both doctors, this is counter-intuitive. To overcome such issues, many aggregation rules have been proposed [18, 34]. For example, in [44], Yager proposed a new aggregation rule to combine the pieces of evidence. Let there be k BPAs, $m_1, m_2, \ldots, m_k$ defined on the same universe Ω. Yager rule of aggregation to combine the k pieces of evidence or BPAs is define as:

$$\begin{aligned} m(A) &= m_1 \oplus m_2 \oplus \cdots \oplus \; m_k(A) \\ &= \sum_{\substack{A_1 \cap A_2 \cdots \cap A_k = A; \\ A_i \subseteq \Omega; A \subset \Omega}} \Pi_{i=1}^k m_i(A_i); \\ m(\Omega) &= \Pi_{i=1}^k m_i(\Omega) + \sum_{\substack{B_1 \cap B_2 \cdots \cap B_k = \emptyset \\ B_i \subseteq \Omega}} \Pi_{i=1}^k m_i(B_i) \\ &\text{and} \quad m(\emptyset) = 0. \end{aligned} \tag{3}$$

Using Yager's rule, in the Zadeh-example mentioned above $m_{12}\{(b)\} = 0.001$, which is more plausible than 1.0.

Dubois and Prade, on the other hand, dealt with the conflict between evidences in a different manner [9, 45]. They combined multiple pieces of evidence as:

$$\begin{aligned} m(A) = [m_1 \oplus m_2](A) &= \sum_{B \cap C = A} m_1(B) m_2(C) \\ &+ \sum_{B \cup C = A, B \cap C = \emptyset} m_1(B) m_2(C); \\ &\text{and} \quad m(\emptyset) = 0. \end{aligned} \tag{4}$$

All the previous rules, use the product of BPAs to combine BPAs. The underlying assumption is that the sources of evidence are independent. The quantity $m_1(B) m_2(C)$ contributes to mass on A by the combined BPA $m_1 \oplus m_2(A), B \cap C = A$. This can be viewed as the extent to which m_1 and m_2 focus on A. The product is a T-Norm. T-Norm is used to model the AND operation. This raises an interesting question: If we change the multiplication operator with a T-Norm along with appropriate changes in the rule of combination, how will it affect the performance? The reason is $m_1(B).m_2(C)$ with $B \cap C = A$ gives an indication of the extent A is supported by both B and C and the same is represented by $\mathrm{T}(m_1(B), m_2(C))$ when $B \cap C = A$. For T-norm, we use the following rule of combination:

$$\begin{aligned} m(A) &= m_1 \oplus m_2 \oplus \cdots \oplus m_k(A) \\ &= \sum_{\substack{A_1 \cap A_2 \cdots \cap A_k = A; \\ A_i \subseteq \Omega; A \subseteq \Omega}} T_{i=1}^{k} m_i(A_i); \\ k &= \sum_{A \subseteq \Omega} m(A); \\ &\text{if } k \neq 0, m(A) = m(A)/k \quad \forall A \subseteq \Omega; \text{else } m(\Omega) = 1; \\ &\text{and} \quad m(\emptyset) = 0. \end{aligned} \tag{5}$$

In (5), the division by k is needed to ensure that we get a valid BPA. We use four different **T-norm**s for combining the evidences. These are: $T_M(x, y) = min(x, y)$, $T_{H_{r_t}}(x, y) = \frac{xy}{r_t + (1 - r_t)(x + y - xy)}$, $T_{F_{r_t}}(x, y) = log_{r_t}(1 + \frac{(r_t{}^x - 1)(r_t{}^y - 1)}{r_t - 1})$ and $T_L(x, y) = max(x + y - 1, 0)$. We discuss the experiments using **T-norm**s in Sect. 4. For an easier presentation, we denote $Rule_1 = T_M(x, y)$, $Rule_2 = T_{H_{r_t}}(x, y)$, $Rule_3 = T_{F_{r_t}}(x, y)$ and $Rule_4 = T_L(x, y)$. It is worth noting here that in [29] T-norm and uninorm based combination of belief functions has been studied.

In Method$_3$, we use Dubois and Prade's rule [9, 45].

3 Proposed method

Since in our method we use an auto-encoder to generate a useful representation of the input data, we first discuss that.

3.1 The architecture of an autoencoder

An autoencoder is an MLP [17] that learns the identity function, i.e., the input and output of the MLP are the same. We follow the notations as in [17]. We consider linear nodes in the input layer:

$$\mathcal{S}(x_{ki}) = x_{ki}; \quad i = 1, 2, \ldots, p; \mathcal{S}(x_{k0}) = 1; \quad \forall k. \tag{6}$$

Here, $\mathcal{S}(\cdot)$ denotes the activation function of a node, and $\mathbf{x}_k$ and p are as defined earlier. For convenience, let us consider a single hidden layer network with sigmoidal activation function as follows:

$$z_{kh} = \sum_{i=0}^{p} w_{ih}^{I} \mathcal{S}(x_{ki}); \quad h = 1, 2, \ldots, q; \tag{7}$$

$$\begin{aligned} \widetilde{z}_{kh} &= \mathcal{S}(z_{kh}) = \frac{1}{1 + e^{-z_{kh}}}; \quad h = 1, 2, \ldots, q; \\ \widetilde{z}_{k0} &= \mathcal{S}(z_{k0}) = 1, \quad \forall k. \end{aligned} \tag{8}$$

Here, for $\mathbf{x}_k$, z_{kh} is the net input to the hth hidden node and $\mathcal{S}(z_{kh})$ is its output. Moreover, w_{ih}^{I} is the connection weight between the ith input node and the hth hidden node, and $w_{0h}^{I}, \forall h$, is a bias. The neurons in the output layer also use sigmoidal functions as follows:

$$y_{kj} = \sum_{h=0}^{q} w_{hj}^{H} \widetilde{z}_{kh}; \quad j = 1, 2, \ldots, p; \tag{9}$$

$$\mathcal{S}(y_{kj}) = \frac{1}{1 + e^{-y_{kj}}}; \quad j = 1, 2, \ldots, p. \tag{10}$$

For a given $\mathbf{x}_k$, y_{kj} is the net input to the jth output node and $\mathcal{S}(y_{kj})$ is its output. Furthermore, w_{hj}^{H} is the connection weight between the hth hidden node and the jth output node and $w_{0j}^{H}, \forall j$, is a bias. The network error for the kth input pattern is computed as:

$$E^k = \frac{1}{2} \sum_{j=1}^{p} \left(x_{kj} - \mathcal{S}(y_{kj})\right)^2. \tag{11}$$

3.2 Training of the network

For every training data point $\mathbf{x}_i \in R^p$, there is a class label c_i, where $c_i \in \{1, \ldots, r\}$. Let $X \subseteq R^p$ be the training data. We randomly divide X into two equal (to the extent possible) parts: $X = X_{\mathrm{TR}} \cup X_{\mathrm{TE}}, X_{\mathrm{TR}} \cap X_{\mathrm{TE}} = \phi$. We use X_{TR} for the training and X_{TE} for testing. The training procedure described here is the same as the training procedure in [2].

Now, X_{TR} has data from all r classes. $X_{\mathrm{TR}} = \cup_{i=1}^{r} \hat{X}_i$, where $\hat{X}_i = \{\mathbf{x}_k | \mathbf{x}_k \in \text{the } i\text{thclass}\}$. Let, the cardinality of $\hat{X}_i$ be n_i, i.e., $n_i = |\hat{X}_i|$. Now, we find out ith *class* center $\widetilde{\mathbf{v}}_i$ as,

$$\widetilde{\mathbf{v}}_i = \frac{1}{n_i} \sum_{\mathbf{x}_j \in \hat{X}_i} \mathbf{x}_j. \tag{12}$$

We get r class centers $\widetilde{V} = \{\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, \ldots, \widetilde{\mathbf{v}}_r\}; \widetilde{\mathbf{v}}_i \in R^p$. Further we cluster $\hat{X}_i$ into n_c clusters using the k-means algorithm. This produces $(n_c \times r)$ cluster centers $\hat{V} = \left\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_{(n_c \times r)}\right\}; \hat{\mathbf{v}}_i \in R^p$. Let $V = \hat{V} \cup \widetilde{V}$. This produces $(r + n_c \times r)$ *cluster* centers $V = \left\{\mathbf{v}_1, \ldots, \mathbf{v}_{(r + n_c \times r)}\right\}; \mathbf{v}_i \in R^p$. Now from X_{TR} we generate p modified data sets $X^s, s = 1, 2, \ldots, p$ as follows. To generate X^s for $\mathbf{x}_j \in X_{\mathrm{TR}}$, we replace x_{js} by v_{ls} if $l = \mathrm{argmin}_{k'} \left\{ ||\mathbf{x}_j - \mathbf{v}_{k'}||_*^2 \right\}$.

Here we assume as if the the sth feature is missing in $\mathbf{x}_j$ and the sth feature value is imputed by the sth feature value of the centroid which is closest to $\mathbf{x}_j$ in terms of distance measure $||.||_*$ computed using all but the sth feature.

Now we train an auto-encoder using X_{TR} for N_1 epoches. Next we retrain the same network for N_2 epoches with the data set $X^{\mathrm{Total}} = X_{\mathrm{TR}} \cup_{s=1}^{p} X^s$. For any $\mathbf{x}_j \in X_{\mathrm{TR}}$ as well as any $\mathbf{x}_j^s \in X^s$, the target vector is taken as $\mathbf{x}_j \in X_{\mathrm{TR}}$.

This training method is summarized as Algorithm 1 [2–4]. Such a procedure for training AEs to deal with missing values has been used in our previous studies and we have found it to be very effective [2–4].

Algorithm 1 : Training of the Auto-Encoder to generate the Latent Space

```
INPUT: $X_{TR}$: Training data set; $p$: Number of features; $r$: Number of classes; $n_c, c = 1, 2, \cdots, r$: Number of clusters in class $c$; $N_1$: Number of epochs for the first phase of training; $N_2$: Number of epochs for the second phase of training.
BEGIN
Set $\hat{V} = \emptyset$, $\widetilde{V} = \emptyset$.
for i=1 to $r$ do
    Set $\hat{X}_i = \{\mathbf{x}|\mathbf{x} \in X_{TR}$ and $\mathbf{x}$ belongs to the $i^{th}$ class$\}$.
    Set $\widetilde{\mathbf{v}}_i = \frac{1}{|\hat{X}_i|} \sum_{\mathbf{x} \in \hat{X}_i} \mathbf{x}$.
    $\widetilde{V} = \widetilde{V} \cup \widetilde{\mathbf{v}}_i$.
    Cluster $\hat{X}_i$ into $n_c$ clusters by the $k$-means algorithm to generate $n_c$ cluster centers $\hat{V}_i = \{\hat{\mathbf{v}}_{i1}, \hat{\mathbf{v}}_{i2}, \cdots, \hat{\mathbf{v}}_{in_c}\}; \hat{\mathbf{v}}_i \in \mathbb{R}^p$.
    $\hat{V} = \hat{V} \cup \hat{V}_i$.
end for
Set $V = \hat{V} \cup \widetilde{V}$.
for s=1 to $p$ do
    To generate $X^s$, for every $\mathbf{x} \in X_{TR}$, we obtain $\mathbf{x}^s$ replacing $x_s$ (the $s^{th}$ component of $\mathbf{x}$) by $v_{ls}$ if $l = argmin_{k'} \{||\mathbf{x} - \mathbf{v}_{k'}||_*^2\}$, where $||.||_*$ is computed using all but the $s^{th}$ feature.
end for
With $X_{TR}$ an autoencoder is trained for $N_1$ epoches.
The same autoencoder is further trained for $N_2$ epoches using $X^{Total} = X_{TR} \cup \{X^1 \cup X^2 \cup \cdots \cup X^p\}$. For any $\mathbf{x} \in X_{TR}$ and its corresponding $\mathbf{x}^s \in X^s$, the target vector is $\mathbf{x} \in X_{TR}$.
END
```

3.3 Imputation and training of classifiers

Let X_{TR} be the training data set and X_{TE} be the test data set. We assume that X_{TR} is complete and missing values are present only in X_{TE}. So, each training data point $\mathbf{x}_i \in X_{\mathrm{TR}}$ has all p features present and $|X_{\mathrm{TR}}| = n_1$. Let $Y_{\mathrm{TR}} = \{\mathbf{y}_i \in R^r\}$. y_i is the label vector of $\mathbf{x}_i \in X_{\mathrm{TR}}; i = 1, 2, \ldots, n_1$. We pass each point $\mathbf{x}_i$ through the trained network and get the output of the hidden layer, $\widetilde{z'}_{ih}$ as the latent space representation:

$$z'_{ih} = \sum_{k=0}^{p} w^I_{kh} \mathcal{S}(x_{ik}), \quad h = 1, \ldots, q$$
$$\widetilde{z'}_{ih} = \mathcal{S}\left(z'_{ih}\right) = \frac{1}{1 + e^{-z'_{ih}}}, \quad h = 1, \ldots, q. \tag{13}$$

Let, Z_{TR} be the latent space representation of X_{TR}, $Z_{\mathrm{TR}} = \cup_{i=1}^{n_1} \{\widetilde{\mathbf{z}}'_i\}, \widetilde{\mathbf{z}}'_i$ is the representation for $\mathbf{x}_i \in X_{\mathrm{TR}}; i = 1, 2, \ldots, n_1$. The set of label vectors associated with Z_{TR} is the same as Y_{TR}, i.e., if $\mathbf{x}_i \in X_{\mathrm{TR}}$ has a label vector $\mathbf{y}_i \in Y_{\mathrm{TR}}$ then $\mathbf{z}_i \in Z_{\mathrm{TR}}$ has the same label vector $\mathbf{y}_i$. Now using $(Z_{\mathrm{TR}}, Y_{\mathrm{TR}})$, we train r-SVMs using the one-vs-all strategy.

On the other hand, each test data point $\mathbf{x}_i \in X_{\mathrm{TE}}$ may have up to $(p-1)$ features missing. We impute the missing values of $\mathbf{x}_i$ using $\mathbf{v}_i$, $(i = 1, \ldots, (r + n_c \times r))$ as follows. The sth missing value of $\mathbf{x}_i$, x_{is} is imputed by v_{ls} if $l = \mathrm{argmin}_j \left\{ ||\mathbf{x}_i - \mathbf{v}_j||_*^2 \right\}$ where $||.||_*$ computed using all but the missing feature value of $\mathbf{x}_i$. After imputation, we pass $\mathbf{x}_i$ through the trained autoencoder and generate its latent space representation, $\widetilde{\mathbf{z}}_i$, as the output of the hidden layer. We use $\widetilde{Z}_{\mathrm{TE}}$, (where $\widetilde{Z}_{\mathrm{TE}} = \cup_i \ \{\widetilde{\mathbf{z}}_\mathbf{i}\}$) for testing. For testing, we take each point of $\widetilde{Z}_{\mathrm{TE}}$ and pass it through each trained SVMs. Let, for the jth test point, the probabilistic output from the cth SVM for class -1 and $+1$ be $P_{jc}^{(-1)}$ and $P_{jc}^{(+1)}$ as describe in the Sect. 3.5.

3.4 BPA Assignments using t-nearest neighbor rules

In [7], authors used the information about the t-nearest neighbors of a test point to define r BPAs. We are given a point $\mathbf{x}_j$ whose class is to be determined. Let Φ^j be the set of t nearest neighbours of $\mathbf{x}_j$ and $\Phi_c^j \subseteq \Phi^j$ be the set of $c \leq t$ nearest neighbours of $\mathbf{x}_j$ that are from class c. Following [7] the mass assigned by the points in Φ_c^j to the class c (we call here the positive class) is defined as

$$P_c^j = \frac{m_c^j(\{c\}) \prod_{l \neq c} m_l^j(\Omega)}{K}; \forall c = 1, \ldots, r; \tag{14}$$

where K is a normalizing factor. Here, $m_c^j(\{c\}) = 1 - \prod_{x_i \in \Phi_c^j} (1 - \alpha_0 \phi_c(d^{j,i}))$, $m_c^j(\Omega) = \prod_{x_i \in \Phi_c^j} (1 - \alpha_0 \phi_c(d^{j,i}))$, and $d^{j,i}$ are the distance between $\mathbf{x}_j$ and the ith element of Φ_c^j. Later using $P_c^j; c = 1, \ldots, r$, we shall define $r-$BPAs.

3.5 Probabilistic output of SVM

Typically, an SVM classifier does not produce probability as its output. There are various methods to convert SVM output to a probability [30]. In the proposed method, we calculate $P_{jc}^{(-1)}$ and $P_{jc}^{(+1)}$ from cth SVM output as follows. According to [30] if $f(\mathbf{x})$ is the output of SVM for a test data point $\mathbf{x}$, then $f(\mathbf{x}) = h(\mathbf{x}) + b$, where $h(\mathbf{x}) = \sum_i y_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$. Here, b is the bias of SVM, y_i is the class label of ith data point, α_i is the coefficient of ith training data point, $k(.,.)$ is the kernel function of SVM. We choose the Gaussian kernel for our problem. The probability of a test point to a particular class should be higher if it is more close to the plane containing the support vector of that class. Keeping this in mind we use the probabilistic output from SVM as in [30] and calculate the probabilistic output for class $+1$, $P_{jc}^{(+1)}$ as: $P_{jc}^{(+1)} = \frac{1}{1+exp(Af+B)}$ where A and B are two parameters to be estimated. Consequently, the probabilistic output for class -1 is $P_{jc}^{(-1)} = 1 - P_{jc}^{(+1)}$.

3.6 Final classification using evidence theory

The probabilistic output for class $+1$ using cth SVM $P_{jc}^{(+1)}$ indicates probability of the point to belong to the cth class, while probability of class -1, $P_{jc}^{(-1)}$ is probability of not belonging to class $c; c = 1, 2, \ldots, r$. Now, if we try to combine outputs of cth SVM and lth$(l \neq c)$ SVM it is reasonable to expect that in $P_{jc}^{(+1)}$ some influence of $P_{jl}^{(-1)}$ is present and in $P_{jl}^{(+1)}$ some influence of $P_{jc}^{(-1)}$ is present. Keeping this in mind, from the probabilistic output of the jth test point we define the ith BPA $m_i^1; i = 1, \ldots, r$ from the SVM output as follows.

$$m_i^1 : \begin{cases} m_i^1(\{i\}) = \dfrac{P_{ji}^{(+1)} + \sum_{\substack{c=1 \\ c \neq i}}^{r} \dfrac{P_{jc}^{(-1)}}{r-1}}{r}; \\ m_i^1(\{\Omega - i\}) = 1 - m_i^1(\{i\}). \end{cases} \tag{15}$$

Similarly, for the jth test point, using equation (14) for the EV-NN we can assign ith; $i = 1, \ldots, r$ BPA, m_i^2 as follows.

$$m_i^2 : \begin{cases} m_i^2(\{i\}) = P_i^j; \\ m_i^2(\{\Omega - i\}) = 1 - m_i^2(\{i\}). \end{cases} \tag{16}$$

We shall propose three methods, of which Method$_2$ will use both m_i^1 and m_i^2, while the other two methods will not use the BPAs derived from the EV-NN outputs.

Method$_1$: Here using Dempster's rule, we compute the composite BPA as $m = m_1^1 \oplus m_2^1 \oplus \ldots \oplus m_r^1$. This composite BPA m is then used to compute the pignistic probability for the final decision making.

Method$_2$: Here also, we use Dempster's rule to join m_i^1 and m_i^2 to get $m_i = m_i^1 \oplus m_i^2; \forall i = 1, 2, \ldots, r$. Then, the composite BPA is computed by further aggregating the r BPAs as $m = m_1 \oplus m_2 \oplus \cdots \oplus m_r$ using again Dempster's rule of aggregation.

Method$_3$: Here, we use Dubois and Prade's rule [9, 45] of aggregation to obtain the composite BPA as $m = m_1^1 \oplus m_2^1 \oplus \cdots \oplus m_r^1$.

Now to find the overall belongingness of a test point $\mathbf{x}$ to the class c, we compute the Pignistic probability [6] using equation (2). Thus, we have a set of Pignistic probabilities $\mathcal{P} = \{P_x\{1\}, P_x\{2\}, \ldots, P_x\{r\}\}$. Let, $l = \text{argmax}_i\{P_x\{i\}; \forall i = 1, 2, \ldots, r\}$ and $d = \text{argmax}_i\{P_x\{i\}; \quad \forall i = 1, 2, \ldots, r; i \neq l\}$. Now, if $(P_x\{l\} - P_x\{d\}) > \epsilon$, we decide that $\mathbf{x}$ belongs to the class l. Otherwise, it is reasonable to assume that $\mathbf{x}$ belongs to both the classes l and d. Here, ϵ is a user defined threshold. This is quite intuitive because if the Pignistic probability of the best class is significantly different from the rest we are more confident of assigning the associated point to the best class.

4 Experiments

To demonstrate the effectiveness of the proposed methods following [21], our experiments are divided into four parts and we compare our methods with four state-of-the-art methods. For comparison, we use two error terms as in [21] and use the results of [21]. In the results Tables, an entry P in $\{P, Q\}$ indicates the miss-classification error (Re), and Q indicates the error rate of test points belonging to more than one class (Ri_2; here we use only two classes) including the correct class.

4.1 Experimental set up

We made a few experiments and based on that we chose the following architecture of the network (AE) for all datasets: $(p) - (10 \times p) - (p)$, where p is the number of input nodes or features. Also in each class for every data set, we find $n_c = (r \times 5)$ clusters. The learning rate is chosen as $\eta = 0.9$. Based on a few trial and error experiments for all datasets, we choose $N_1 = N_2 = 10{,}000$. Each experiment is repeated 10 times each with a different weight initialization and we report the average results. While calculating probability from EV-NN, for the first three experiments we take $t = 10$. But, in the last experiment (i.e., in Experiment 4) only for the Yeast dataset, we take $t = 20$ and for the other datasets, we take $t = 10$. For comparison of our method, we consider four missing value handling algorithms in conjunction with two classifiers. The different combinations used are: k-NN imputation (KNNI) method with EK-NN, FCM imputation (FCMI) with EK-NN, prototype-based credal classification (PCC) with EK-NN as well as with evidential neural network (ENN), KNNI method with ENN, both FCMI and MI have been considered with ENN and the mean imputation (MI) with EK-NN. The detailed information about these algorithms is available in [21].

4.2 Experiment 1

Similar to experiment 1 in [21], here we consider a three-class dataset with circular shape. Every class contains 305 training and 305 test instances. The radius of each circle is 3 unit, and the centers of the three circles are $c_1 = (3,3)^T$, $c_2 = (13,3)^T$, $c_3 = (8,8)^T$, where T denotes transpose. Similar to experiment 1 in [21] the values in the second dimension of all test samples are missing, and thus there is only one known value, the x-coordinate for each test sample. For this experiment, to define meta-classes we consider the following two thresholds: $\epsilon = 0.30$ and $\epsilon = 0.45$. This will help to understand the influence of the threshold on the performance. Figure 1 depicts the classification performance of Method$_1$ on the three class data set using the proposed method with $\epsilon = 0.45$. In Table 1 and in other tables, the bold face values indicate the best results for the corresponding experiments. From Table 1, we may conclude that the performance of Method$_1$ and Method$_3$ is better than other methods in terms of Re, whereas the performance of Method$_2$ is worse than PCC, Method$_1$, Method$_3$ and better than other methods in terms of Re. Note that in Table 1 as well as in other Tables, except for the PCC based methods, the reported values are the best class performance, i.e., the accuracy [21].

Table 1 Performance comparison of different methods in Experiment 1

Method	$\epsilon = 0.30$ $\{Re, Ri_2\}$	$\epsilon = 0.45$ $\{Re, Ri_2\}$
Method$_1$	{**0.87**, 0.87}	{**0.87**, 0.87}
Method$_2$	{1.75, 0.00}	{1.75, 0.00}
Method$_3$	{1.73, 0.00}	{**0.87**, 0.87}
PCC [21]	{1.75, 4.81}	{**0.87**, 8.31}
FCMI [21]	4.15	
KNNI [21]	4.15	
MI [21]	8.52	

4.3 Experiment 2

Like Experiment 2 in [21] we consider a four-class dataset for this experiment. Each class contains 100 training and 100 test instances. The uniform distribution of the instances is characterized by the interval bounds as shown in Table 2.

As in [21], the values in the x-coordinates of the test samples are all missing. Thus, there is only one known value, i.e., the y coordinate of each test sample. In this experiment, we use three different meta-class selection thresholds $\epsilon = 0.15$, $\epsilon = 0.30$ and $\epsilon = 0.45$ to show their influences on the results. In Fig. 2, we show the classification results of Method$_1$ on this four-class data set with $\epsilon = 0.45$.

Table 3 compares performance of different methods on this data set. Form Table 3, we find that in terms of Ri_2, all three proposed methods perform better than the two PCC-based methods. In terms Re, the $Method_1$ performs noticeably better than nine methods including $Method_2$ for $\epsilon = 0.30$. On the other hand, for $\epsilon = 0.45$, $Method_3$ performs better than nine Methods including Method$_2$. We note here that the dataset used in the comparing methods and the dataset used in our method are not exactly the same, but they are generated using the same guidelines.

4.4 Experiment 3

This experiment is similar to Experiment-3 in [21], here, we use a three-class problem, where a class is represented by random instances drawn from a four-dimensional

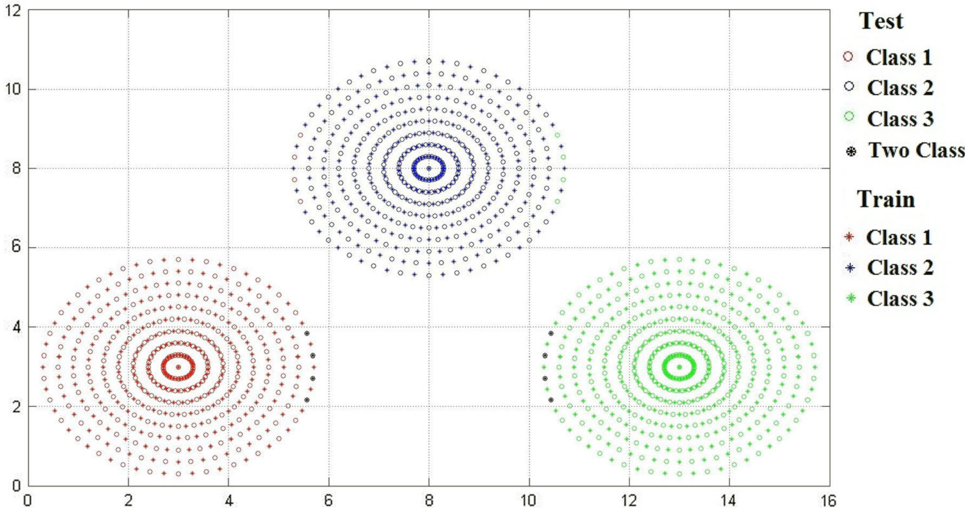

Fig. 1 Classification result of the proposed method of three class data set

Table 2 Details dataset description of Experiment 2

Class	x-label interval	y-label interval
Class 1	(10, 20)	(5, 65)
Class 2	(10, 20)	(110, 170)
Class 3	(35, 45)	(50, 120)
Class 4	(55, 65)	(150, 230)

Gaussian distribution with the following parameters: The means of the three classes are $(1, 5, 10, 10)^T$, $(10, 3, 2, 1)^T$, and $(15, 15, 1, 15)^T$, while the respective covariance matrices are $6 \cdot I$, $5 \cdot I$, and $7 \cdot I$; I is the 4×4 identity matrix. We generate two datasets, one with 100 points from each class and the other with 200 points from each class. We consider three different missing value scenarios. In the three cases, we randomly drop exactly 1, 2, and 3 features, respectively, from each data point. Thus, we have six cases: (100, 1), (100, 2), (100, 3), (200, 1), (200, 2), and (200, 3) where in (M, N), M indicates the number of data points in a dataset and N indicates the number of missing features from each data point of that dataset.

In Table 4, we compere results of the proposed methods with those of others using threshold $\epsilon = 0.45$. As in [21], we report the averages over ten trials performed with ten independent random generations of the data sets in Table 4. From Table 4, we can conclude that all three proposed methods perform better than other methods in terms of Re for all six cases of this experiment. Also of the three proposed methods, $Method_3$ performs the best in 4 cases and $Method_1$ performs the best in 3 cases.

4.5 Experiment 4

In Experiment-4 like [21], we use the same four real data sets (Breast cancer, Seeds, Yeast, and Wine) to compare the performance of the proposed methods with that of others. Breast cancer is a two-class data set having 9 attributes and 699 instances; Seeds, Wine, and Yeast are three class data sets with a number of attributes 7, 13, and 8, respectively.

Like [21] we perform twofold cross-validation experiment. Similar to [21] each test sample has some missing (unknown) values, and they are missing completely at random covering all dimensions. In Table 5, we summarize the results.

Table 5 shows that Method$_1$ performs the best in 8 out of 12 cases with respect to Re (i.e., normal classification error), Method$_2$ performs the best in 2 out of 12 cases with respect to Re and PCC(ENN) performs the best in 2 out of 12 cases with respect to Re. The next best performing method is Method$_3$.

As already mentioned, in [21], for all but the PCC-based methods the best class performance, i.e., accuracy is reported. So, for a fair comparison of the proposed methods with others (FCMI, KNNI, and MI) in Table 6 we report the results of the proposed methods with $\epsilon = 0.0$, i.e, the best class performance. From Table 6, we observe that the Method$_2$ performs the best in 3 out of 12 cases. From other methods, FCMI (ENN) performs the best in 4 cases followed by KNNI (ENN) which yields the best result in 2 cases; while $Method_1$, $Method_3$ and FCMI (EK-NN) perform better than others in 1 case each.

In Table 7, we compare the performance of different methods according to the average rank of the algorithms in

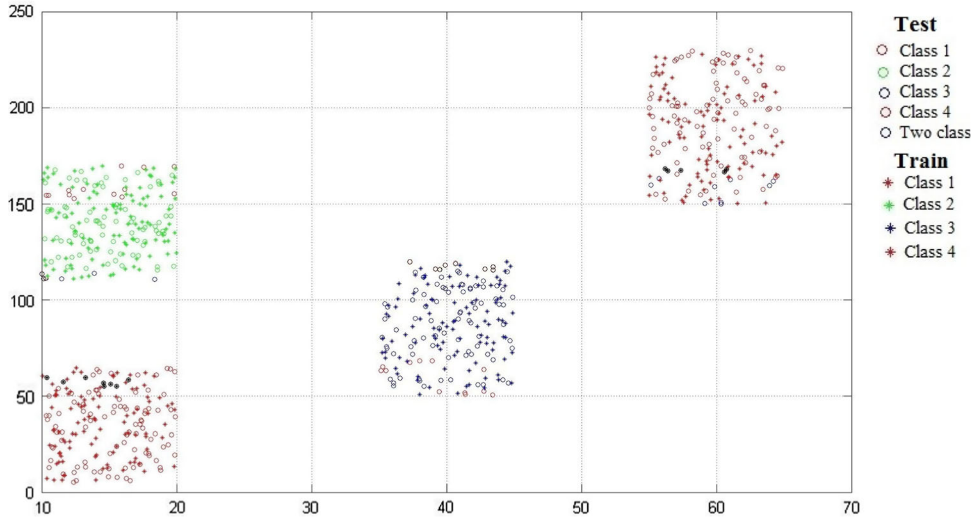

Fig. 2 Classification result of the proposed method of four class data set

Table 3 Performance comparison of different methods in Experiment 2

Method	$\epsilon = 0.15$ $\{Re, Ri_2\}$	$\epsilon = 0.30$ $\{Re, Ri_2\}$	$\epsilon = 0.45$ $\{Re, Ri_2\}$
Method$_1$	{16.50, 0.00}	{11.35, 5.15}	{11.33, 5.18}
Method$_2$	{16.50, 0.00}	{16.50, 0.00}	{16.50, 0.00}
Method$_3$	{16.50, 0.00}	{11.40, 5.10}	{11.30, 5.20}
PCC (EK-NN) [21]	{**8.62**,13.87}		
PCC (ENN) [21]	{8.75, 11.75}		
FCMI (EK-NN) [21]	16.53		
FCMI (ENN) [21]	16.01		
KNNI (EK-NN) [21]	13.04		
KNNI (ENN) [21]	13.63		
MI (EK-NN) [21]	28.37		
MI (ENN) [21]	18.90		

terms of misclassification error (i.e., *Re*) considering all experiments on all data sets. From Table 7, we can find that the performance of Method$_1$ is the best among all compering methods, the second-best performer is Method$_3$ and this is followed by Method$_2$. The next best performers are the PCC-based methods, while the worst performer in terms of this indicator is MI (EK-NN). The performance of other algorithms is in between as depicted in Table 7.

4.6 Effects of other aggregation rules

In this sub-section, we try to illustrate the effect of using different aggregation rules when we combine the pieces of evidence. From our experiments, we find that for Method$_2$ generally the difference between the probability of the best class and probability of the second best class is larger than that for Method$_1$ and Method$_3$. In general, it can be seen in Table 6 that in terms of *Re*, Method$_2$ performs better than Method$_1$ and Method$_3$. From Table 6, we find that Method$_2$ performs better than Method$_1$ in eight cases and better than Method$_3$ also in eight cases. Also comparing with all the methods, Method$_2$ performs the best in 3 cases. Considering this, in this sub-section we use Method$_2$ for all experiments.

In Table 8, we compare classification error (*Re*) between Yager's rule of aggregation and that of Dempster's method with $\epsilon = 0.00$. From Table 8, we find that Yager's method is better only in 3 out of 12 cases. Thus, we find that Dempster's rule performs better than Yager's rule for this particular problem; however, generally, the difference in performance between the two is marginal.

As said earlier, we use four different **t-norms** (instead of product) for combining the BPAs in Method$_2$. They are: $T_M(x,y)$, $T_{H_{r_t}}(x,y)$, $T_{F_{r_t}}(x,y)$ and $T_L(x,y)$ as defined in Sect. 2.1. For each data set, we choose r_t using two-fold cross validation over the set of $r_t : \{0.5, 1.5, 3.5, 4.5, 5.5$ and $6.5\}$.

In Table 9, we display the results obtained using various rules (**t-norms**) with $\epsilon = 0.00$. Table 9 reveals that Method$_2$ performs the best in 8 out of 12 cases for each of the four rules. This suggests that product, i.e., Dempster's rule is a better choice than the four **t-norms** that we have explored.

4.7 Importance of each component of the proposed method

In the proposed method, we use three components: AE, SVM, and Evidential Reasoning. After imputation, we use the AE to represent a data point to its latent space representation. Then, we apply SVM on the latent space representation to find the probability of the data point to a particular class and after this, we use evidential reasoning to find the final classification of a point. This step-by-step procedure raises a few questions. For example, if we drop any component from the above three, will it affect the performance? To check this we have done some experiments and check the effect on the performance in Experiment 4. As in the previous sub-section, here also, we use Method$_2$ in most of the experiments.

Table 10 compares the results between Method$_2$ and Method$_2$ without using the auto-encoder (with $\epsilon = 0.00$). From Table 10, we find that out of the 12 cases our original proposed method performs better in 10 cases. However, the differences between two methods is marginal.

In Table 11, we demonstrate the results of the proposed methods and the proposed methods without using Evidential Reasoning (with $\epsilon = 0.00$). From Table 11, we see that here of the 12 cases, Method$_1$ performs the best only in 2 cases, Method$_2$ performs the best in 6 cases and Method$_3$

Table 4 Performance comparison of different methods in Experiment 3

	(100,1) $\{Re, Ri_2\}$	(100,2) $\{Re, Ri_2\}$	(100,3) $\{Re, Ri_2\}$	(200,1) $\{Re, Ri_2\}$	(200,2) $\{Re, Ri_2\}$	(200,3) $\{Re, Ri_2\}$
Method$_1$	{**0.03**, 0.23}	{1.52, 2.15}	{**9.82**, 11.45}	{0.08, 0.34}	{1.40, 2.45}	{**9.99**, 11.26}
Method$_2$	{0.30, 0.00}	{3.90, 0.00}	{21.90, 0.00}	{0.51, 0.01}	{4.02, 0.00}	{21.37, 0.00}
Method$_3$	{**0.03**, 0.37}	{**1.49**, 2.54}	{11.36, 11.94}	{**0.07**, 0.38}	{**1.25**, 2.69}	{10.04, 10.91}
PCC (EK-NN) [21]	{11.67, 12.60}	{16.72, 12.01}	{29.00, 14.39}	{12.65, 11.34 }	{15.73, 14.86}	{29.82, 14.86}
PCC (ENN) [21]	{14.67, 5.33}	{16.85, 11.00}	{27.70, 15.07}	{15.17, 4.50}	{18.27, 8.83}	{29.67, 15.13}
FCMI (EK-NN) [21]	18.17	24.27	40.06	18.81	24.90	39.59
FCMI (ENN) [21]	17.50	24.13	38.43	17.50	23.22	37.85
KNNI (EK-NN) [21]	18.28	24.57	41.00	18.83	25.00	40.86
KNNI (ENN) [21]	17.99	24.32	39.91	17.60	23.52	38.99
MI (EK-NN) [21]	19.24	25.94	41.85	19.62	28.06	41.34
MI	19.33	25.20	40.57	17.83	23.85	39.90

performs the best in 2 out of the 12 cases. In the Yeast dataset when 1 and 3 features are missing, none of the proposed methods perform well in terms of the SVM output.

In Table 12, we compare Method$_2$ and Method$_2$ using a Bottleneck layer in AE ($\epsilon = 0.00$). The number of nodes in the hidden layer (to represent the latent space) for Breast, Seeds, Wine, and Yeast data sets are taken as 7, 5, 10, and 6, respectively. Note that, there is no special reason to choose these values - these values are less than the number of features of the corresponding data set. Table 12 reveals that of the 12 cases Method$_2$ performs better in 11 cases. In this case, some of the differences are significant. Based on our limited experiments, we can say that the use of a bottleneck layer in the AE is not an efficient option at least for this problem.

In Table 13, we compare between Method$_2$ and Method$_2$ without using auto-encoder where we use all training data for imputation. From Table 13, we observe that of the 12 cases Method$_2$ performs better in 10 cases. So, AE plays a very important role in the proposed method even if we use all training data for imputation.

In Table 14, we depict results of Method$_2$ and Method$_2$ without using Evidential Reasoning and without using AE. Here also, we use all training data for imputation. From Table 14, we observe that in 7 cases Method$_2$ performs better.

In conclusion, we say that every component plays a role but the AE, with a large latent presentation and the use of evidential reasoning has a stronger influence on the outcome.

4.8 Influence of the number of centroids on the performance of the proposed method

It is expected that if we increase the number of centroids, the performance will improve because we are likely to get better imputation of the missing values. As an extreme case, we can use the number of points in the training set as the number of centroids. In Table 15, we compare the performance of the proposed original Method 2 (which uses $(n_c + 1) * r$ centroids) with the case when every training point is used as a centroid. From Table 15, we observe that in 10 of the 12 cases when we use all training data for imputation, Method$_2$ performs much better. Thus, as expected, use of more centroids helps at the cost of more computation time and space.

5 Conclusion

Here, we presented three methods to classify incomplete data using an evidential reasoning framework. Each of the proposed methods has three major components : use of an

Table 5 Performance comparison of different methods in Experiment 4 (With $\epsilon = 0.45$)

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$	$\{Re, Ri_2\}$
Method$_1$	{2.82, 1.19}	{3.98, 1.20}	{7.28, 0.86}	{**4.97**, 32.98}	{**9.78**, 34.48}	{**12.85**, 37.97}	{**2.62**, 7.38}	{**7.95**, 6.86}	{18.48, 9.43}	{**0.79**, 5.34}	{**4.10**, 12.19}	{**17.92**, 11.12}
Method$_2$	{**2.79**, 1.33}	{3.82, 1.23}	{**6.71**, 1.36}	{30.24, 8.40}	{36.26, 8.40}	{41.50, 8.26}	{7.86, 1.24}	{14.24, 0.67}	{27.43, 0.33}	{4.72, 0.85}	{12.92, 3.20}	{28.99, 0.34}
Method$_3$	{2.83, 1.39}	{4.06, 1.20}	{7.11, 1.34}	{11.58, 26.16}	{18.17, 26.03}	{22.69, 28.00}	{6.57, 3.00}	{14.05, 1.43}	{26.29, 1.57}	{3.09, 3.65}	{10.34, 6.24}	{27.53, 1.85}
PCC (EK-NN) [21]	{4.10, 3.38}	{4.38, 4.69}	{7.91, 8.05}	{34.36, 6.95}	{34.71, 18.00}	{33.46, 31.01}	{7.14, 3.72}	{9.67, 6.70}	{16.79, 12.77}	{26.05, 1.05}	{26.62, 0.84}	{25.84, 3.86}
PCC (ENN) [21]	{3.81, 2.34}	{**3.81**, 6.00}	{6.88, 12.44}	{32.67, 6.19}	{34.19, 14.95}	{32.29, 27.62}	{9.05, 2.86}	{9.52, 9.05}	{**16.19**, 14.76}	{26.97, 1.69}	{27.53, 1.12}	{27.53, 3.93}
FCMI (EK-NN) [21]	3.95	5.07	13.00	38.54	45.95	51.11	12.46	20.08	21.75	30.15	32.12	32.30
FCMI (ENN) [21]	3.81	5.27	11.42	36.19	41.33	46.00	13.33	20.00	20.95	26.97	32.02	31.46
KNNI (EK-NN) [21]	6.10	8.15	14.35	38.13	44.29	50.95	9.68	12.54	25.87	26.59	25.84	30.90
KNNI (ENN) [21]	3.95	5.76	11.54	36.70	40.90	49.22	11.19	12.14	25.71	26.97	28.09	31.18
MI (EK-NN) [21]	4.71	8.20	38.33	37.59	45.08	51.16	21.03	33.49	40.71	30.71	34.93	39.23
MI (ENN)[21]	4.25	6.44	14.64	37.71	42.10	49.33	21.43	31.43	39.52	29.78	33.71	37.64

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 6 Comparing the performance of *Re* in Experiment 4 using $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_1$	4.22	5.15	8.17	37.61	44.12	50.53	9.48	15.10	27.62	**5.49**	15.28	29.49
Method$_2$	4.12	5.12	**8.08**	38.85	44.69	49.69	**8.90**	14.57	27.81	5.80	**15.14**	29.44
Method$_3$	4.11	5.39	8.31	37.95	44.33	50.89	9.90	15.76	27.95	6.63	16.29	**29.27**
FCMI (EK-NN) [21]	3.95	**5.07**	13.00	38.54	45.95	51.11	12.46	20.08	21.75	30.15	32.12	32.30
FCMI (ENN) [21]	**3.81**	5.27	11.42	**36.19**	41.33	**46.00**	13.33	20.00	**20.95**	26.97	32.02	31.46
KNNI (EK-NN) [21]	6.10	8.15	14.35	38.13	44.29	50.95	9.68	12.54	25.87	26.59	25.84	30.90
KNNI (ENN) [21]	3.95	5.76	11.54	36.70	**40.90**	49.22	11.19	**12.14**	25.71	26.97	28.09	31.18
MI (EK-NN) [21]	4.71	8.20	38.33	37.59	45.08	51.16	21.03	33.49	40.71	30.71	34.93	39.23
MI (ENN) [21]	4.25	6.44	14.64	37.71	42.10	49.33	21.43	31.43	39.52	29.78	33.71	37.64

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 7 Comparing the performance of different methods using $\epsilon = 0.45$ using average rank

Method	Average Rank
Method$_1$	1.80
Method$_2$	3.90
Method$_3$	2.60
PCC (EK-NN)	4.40
PCC (ENN)	4.63
FCMI (EK-NN)	8.20
FCMI (ENN)	6.37
KNNI (EK-NN)	7.45
KNNI (ENN)	6.42
MI (EK-NN)	10.30
MI (ENN)	9.16

autoencoder for the latent space representation of the data, outputs of several classifiers, and the evidence theory.

To show the effectiveness of our algorithms, we compare their performance with four state-of-the-art techniques with four experiments. In this context, in addition to Dempster-Shafer's rule, we have also used the aggregation rules proposed by Dubois and Prade as well as that by Yager. If we consider the typical misclassification error where the class label is decided by the maximum Pignistic Probability, then proposed *Method*$_2$ is found to be better. On the other hand, if we exploit further the power of evidence theory to allow a data point to lie in more than one (we considered up to two) class when the difference between the maximum and the next maximum Pignistic Probabilities is small, then proposed Method$_1$ works better.

We have also experimented with aggregation rules using four different **t-norms**. Based on our limited experiments, our observation is that the product T-norm, which is the original Dempster-Shafer's rule of combination, usually performs better than other T-norms. Finally, we have demonstrated the roles played by different components of the method. One interesting finding is that unlike, feature extraction applications, where an AE with a bottleneck layer performs better, for the present problem the use of a hidden layer of bigger size performs better.

Table 8 Comparing classification error (*Re*) between Yager and Dempster's method using $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Dempstar (Method$_2$)	**4.12**	5.12	**8.08**	38.85	**44.69**	**49.69**	**8.90**	**14.57**	27.81	**5.80**	**15.14**	**29.44**
Yagar (Method$_2$)	4.22	**4.96**	8.13	**38.56**	45.00	50.56	9.19	14.76	**27.62**	6.57	15.17	29.66

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 9 Comparing classification error (*Re*) various rules using $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	4.12	**5.12**	**8.08**	38.85	44.69	**49.69**	**8.90**	**14.57**	27.81	**5.80**	**15.14**	**29.44**
Rule$_1$	4.21	5.41	8.57	**37.77**	**43.82**	50.55	10.48	14.86	**27.62**	6.46	15.79	29.49
Rule$_2$	**4.08**	5.52	8.24	41.08	45.83	52.84	12.33	15.86	28.19	8.03	16.91	30.00
Rule$_3$	4.21	5.48	8.34	37.79	44.20	50.65	11.95	15.90	28.19	6.97	16.29	29.49
Rule$_4$	4.26	5.29	8.36	55.35	56.02	58.20	14.24	18.62	28.76	15.84	19.21	29.61

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 10 Comparing classification error (*Re*) between the proposed method and the proposed method without using auto-encoder; $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	**4.12**	**5.12**	**8.08**	38.85	**44.69**	**49.69**	**8.90**	**14.57**	27.81	**5.80**	**15.14**	**29.44**
Without AE+Method$_2$	4.25	5.37	8.19	**38.25**	45.34	50.41	10.17	14.67	**27.62**	6.43	16.11	30.07

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 11 Comparing classification error (*Re*) between the proposed methods and SVM error; $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_1$	4.22	5.15	8.17	37.61	44.12	50.53	9.48	15.10	**27.62**	**5.49**	15.28	29.49
Method$_2$	4.12	**5.12**	**8.08**	38.85	44.69	**49.69**	**8.90**	**14.57**	27.81	5.80	**15.14**	29.44
Method$_3$	**4.11**	5.39	8.31	37.95	44.33	50.89	9.90	15.76	27.95	6.63	16.29	**29.27**
With AE+SVM-Error	4.24	5.30	8.32	**37.57**	**43.86**	49.96	9.76	15.14	27.82	5.51	16.18	29.72

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 12 Comparing classification error (*Re*) between the proposed method and proposed method with Bottleneck layer; $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	4.12	**5.12**	**8.08**	**38.85**	**44.69**	**49.69**	**8.90**	**14.57**	**27.81**	**5.80**	**15.14**	**29.44**
Method$_2$ with Bottlenack	**4.05**	5.46	8.80	41.81	46.32	51.57	14.83	19.33	30.23	14.51	19.29	29.45

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 13 Comparing classification error (*Re*) between the proposed method and proposed method without using auto-encoder; $\epsilon = 0.00$ (here we use all training data for imputation)

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	**3.39**	**3.58**	**9.37**	**34.71**	**35.09**	43.74	**7.91**	**7.71**	11.52	**5.42**	**17.56**	**27.98**
Without AE+Method$_2$	3.53	3.81	9.55	35.32	35.33	**43.51**	8.02	7.81	**11.31**	6.10	17.61	28.13

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 14 Comparing classification error (*Re*) between the proposed method and SVM error on imputed data without using auto-encoder; $\epsilon = 0.00$ (here we use all training data for imputation)

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	**3.39**	**3.58**	**9.37**	34.71	35.09	**43.74**	7.91	**7.71**	**11.52**	**5.42**	17.56	27.98
Without AE+SVM-Error	3.78	3.85	9.74	**34.17**	**33.88**	43.86	**6.62**	7.91	11.62	5.62	**16.89**	**27.39**

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Table 15 Comparing classification error (*Re*) between the proposed method with all training data for imputation and the original method; $\epsilon = 0.00$

DataSet	B-3	B-5	B-7	Y-1	Y-3	Y-5	S-3	S-5	S-6	W-3	W-6	W-10
Method$_2$	**3.39**	**3.58**	9.37	**34.71**	**35.09**	**43.74**	**7.91**	**7.71**	**11.52**	**5.42**	17.56	**27.98**
Original Method$_2$	4.12	5.12	**8.08**	38.85	44.69	49.69	8.90	14.57	27.81	5.80	**15.14**	29.44

B-#: Breast-#; Y-#: Yeast-#; S-#: Seeds-#; W-#: Wine-#; #: No of missing features

Any imputation method is always associated with some uncertainty. In the proposed method that uncertainty is reduced to some extent because of the very special way we use the AE. Since during training, the AE uses an imputed input vector as input and produces the latent space representation using the actual input vector as the target, the latent space is able to capture the loss of information due to imputation. The use of limited training data also adds some uncertainty in the classifier outputs, which we try to reduce using the theory of evidence. However, the main problem associated with the use of evidence theory is its computational overhead. In general, a BPA may focus on 2^r possible subsets, where r is the number of classes. To combine two such BPAs, we have to consider $2^r * 2^r = 2^{2r}$ pairs. This goes on increase as we combine more BPAs. In order to reduce this we have defined BPA using a very simple concept which focuses only on two sets : a singleton set (a class) and a set consisting of the remaining classes. So two combine two such BPAs using Dempster's rule, we shall need to consider only 4 possibilities. Thus to combine r such BPAs, we need $O(r)$ computation, while the overhead of computing each BPA using the SVM output is also $O(r)$ and there are r such BPAs making the complexity $O(r^2)$. The other computational overhead is associated with training of r SVMs and training of an autoencoder. The SVM complexity depends on the implementation. However, typical implementation has a complexity of $O(n^3)$ where n is the number of training instances [42] and we need to train r such SVMs. The training complexity of an autoencoder is the same as that of an MLP, which again depends on the learning algorithm used. Assuming that all operations including exponentiation take the same time, the time complexity of backpropagation algorithm will be $O(tNq(p+R))$, where we have one hidden layer with q nodes, R is the number of output nodes, here $R = p$, the same as the number of input nodes, and N is the size of the augmented data set, so $N = n(p+1)$, n is the number of instances in the original training data. So the complexity becomes $O(tnqp^2)$ where t is the number of iterations.

In this investigation, we have used simple ways to define BPAs. More involved ways of exploiting detailed structural information from data can be used to define BPAs. The use of both information of data in the input space as well as in the latent space to define BPAs is likely to improve performance. In the future, we like to explore some of these ideas.

Declarations

Conflict of interest The authors declare that they have no conflict of interest regarding the publication of this paper.

References

1. Allison PD (2001) Missing data: Sage university papers series on quantitative applications in the social sciences (07–136), Thousand Oaks, CA
2. Choudhury SJ, Pal NR (2019) Classification of incomplete data using autoencoder and evidential reasoning. In: IFIP international conference on artificial intelligence applications and innovations. Springer, pp 167–177
3. Choudhury SJ, Pal NR (2021) Deep and structure-preserving autoencoders for clustering data with missing information. IEEE Trans Emerg Top Comput Intell 5(4):639–650. https://doi.org/10.1109/TETCI.2019.2949264
4. Choudhury SJ, Pal NR (2019) Imputation of missing data with neural networks for classification. Knowl Based Syst. https://doi.org/10.1016/j.knosys.2019.07.009
5. Chung D, Merat FL (1996) Neural network based sensor array signal processing. In: IEEE/SICE/RSJ international conference on multisensor fusion and integration for intelligent systems, 1996. IEEE, pp 757–764
6. Cobb BR, Shenoy PP (2003) A comparison of methods for transforming belief function models to probability models. In: European conference on symbolic and quantitative approaches to reasoning and uncertainty. Springer, pp 255–266
7. DENOEUX T (1995) A k-nearest neighbor classification rule based on Dempster–Shafer theory. IEEE Trans Syst Man Cybern 25(5):804–813
8. Dixon JK (1979) Pattern recognition with partly missing data. IEEE Trans Syst Man Cybern 9(10):617–621
9. Dubois D, Prade H (1988) Representation and combination of uncertainty with belief functions and possibility measures. Comput Intell 4(3):244–264
10. Fessant F, Midenet S (2002) Self-organising map for data imputation and correction in surveys. Neural Comput Appl 10(4):300–310
11. García-Laencina PJ, Sancho-Gómez JL, Figueiras-Vidal AR (2010) Pattern classification with missing data: a review. Neural Comput Appl 19(2):263–282
12. Gautam C, Ravi V (2015) Counter propagation auto-associative neural network based data imputation. Inf Sci 325:288–299
13. Gautam C, Ravi V (2015) Data imputation via evolutionary computation, clustering and a neural network. Neurocomputing 156:134–142
14. Kalton G (1983) Compensating for missing survey data. Inst for Social Research the Univ
15. Kofman P, Sharpe IG (2003) Using multiple imputation in the analysis of incomplete observations in finance. J Financ Econom 1(2):216–249
16. Krstulovic J, Miranda V, Costa AJS, Pereira J (2013) Towards an auto-associative topology state estimator. IEEE Trans Power Syst 28(3):3311–3318
17. Kumar S (2004) Neural networks: a classroom approach. Tata McGraw-Hill Education, New York
18. Lefevre E, Colot O, Vannoorenberghe P (2002) Belief function combination and conflict management. Inf fusion 3(2):149–162
19. Little RJ, Rubin DB (2014) Statistical analysis with missing data. Wiley, Hoboken
20. Liu Z.g, Dezert J, Pan Q, Mercier G (2011) Combination of sources of evidence with different discounting factors based on a new dissimilarity measure. Decis Support Syst 52(1):133–141
21. Liu ZG, Pan Q, Mercier G, Dezert J (2015) A new incomplete pattern classification method based on evidential reasoning. IEEE Trans Cybern 45(4):635–646

22. Marseguerra M, Zoia A (2005) The autoassociative neural network in signal analysis: II. Application to on-line monitoring of a simulated BWR component. Ann Nucl Energy 32(11):1207–1223
23. Marwala T, Chakraverty S (2006) Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. Curr Sci 90:542–548
24. Miranda V, Krstulovic J, Keko H, Moreira C, Pereira J (2012) Reconstructing missing data in state estimation with autoencoders. IEEE Trans Power Syst 27(2):604–611
25. Morin R, Raeside B (1981) A reappraisal of distance-weighted *k*-nearest neighbor classification for pattern recognition with missing data. IEEE Trans Syst Man Cybern 3:241–243
26. Narayanan S, Marks R, Vian JL, Choi J, El-Sharkawi M, Thompson BB (2002) Set constraint discovery: missing sensor data restoration using autoassociative regression machines. In: Proceedings of the 2002 international joint conference on neural networks, 2002. IJCNN'02, vol 3. IEEE, pp 2872–2877
27. Narayanan S, Vian JL, Choi J, Marks R, El-Sharkawi M, Thompson BB (2003) Missing sensor data restoration for vibration sensors on a jet aircraft engine. In: Proceedings of the international joint conference on neural networks, 2003, vol 4. IEEE, pp 3007–3010
28. Nowicki R (2009) Rough neuro-fuzzy structures for classification with missing data. IEEE Trans Syst Man Cybern Part B (Cybern) 39(6):1334–1347
29. Pichon F, Denœux T (2008) T-norm and uninorm-based combination of belief functions. In: NAFIPS 2008: 2008 annual meeting of the North American fuzzy information processing society, pp 1–6
30. Platt J et al (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv Large Margin Classif 10(3):61–74
31. Qiao W, Gao Z, Harley RG, Venayagamoorthy GK (2008) Robust neuro-identification of nonlinear plants in electric power systems with missing sensor measurements. Eng Appl Artif Intell 21(4):604–618
32. Samad T, Harp S.A (1992) Self-organization with partial data. Netw Comput Neural Syst 3(2):205–212
33. Schafer JL (1997) Analysis of incomplete multivariate data. CRC Press, Cambridge
34. Sentz K, Ferson S et al (2002) Combination of evidence in Dempster–Shafer theory, vol 4015. Citeseer, Princeton
35. Shafer G (1976) A mathematical theory of evidence, vol 42. Princeton University Press, Princeton
36. Silva-Ramírez EL, Pino-Mejías R, López-Coello M (2015) Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. Appl Soft Comput 29:65–74
37. Silva-Ramírez EL, Pino-Mejías R, López-Coello M, Cubiles-de-la Vega MD (2011) Missing value imputation on missing completely at random data using multilayer perceptrons. Neural Netw 24(1):121–129
38. Smarandache F, Dezert J (2009) Advances and Applications of DSmT for Information Fusion Collected works. American Research Press, vol 3, p 760
39. Smets P (1990) The combination of evidence in the transferable belief model. IEEE Trans Pattern Anal Mach Intell 12(5):447–458
40. Smets P (2007) Analyzing the combination of conflicting belief functions. Inf Fusion 8(4):387–412
41. Thompson BB, Marks R, El-Sharkawi MA (2003) On the contractive nature of autoencoders: application to missing sensor restoration. In: Proceedings of the international joint conference on neural networks, 2003, vol 4. IEEE, pp 3011–3016
42. Tsang I, Kwok J, Cheung P, Cristianini N (2005) Core vector machines: fast SVM training on very large data sets. J Mach Learn Res 6:363–392
43. Westin LK (2004) Missing data and the preprocessing perception, page 3, Umea University, ISSN-0348-0542
44. Yager RR (1987) Quasi-associative operations in the combination of evidence. Kybernetes 16(1):37–41
45. Yang JB, Xu DL (2013) Evidential reasoning rule for evidence combination. Artif Intell 205:1–29

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.