Abstract

Several classifiers are available for the identification of radar emitter types from their waveform parameters. In particular, these classifiers can be applied to data that is affected by some types of noise. This paper proposes a more efficient classifier, which uses on-line learning and is attractive for real time applications, such as electronic support measures. A self-organizing interval type-2 fuzzy neural network (ST2FNN) is proposed for radar emitter identification. The ST2FNN has both an on-line structure and parameter learning ability. The structural learning includes to add a new rule to an interval type-2 fuzzy neural network (IT2FNN) and to prune an inefficient rule from ITSFNN; and the parameter learning can improve the learning ability of IT2FNN. Then the developed ST2FNN is applied for the radar emitter identification. Simulation results indicate that the proposed ST2FNN can achieve satisfactory classification performance and has a consistent average error deviation level that is lower than that of other neural network classifiers.

Keywords: Type-2 fuzzy system, fuzzy neural network, self-organizing learning algorithm, emitter identification, electronic support measures.

1. Introduction

In certain situations, data cannot be expressed using precise scalar features, so interval data must be used. Intervals occur naturally in imprecise data, or when an estimation of a certain parameter must be performed using a confidence interval. When sensing in a noisy environment, uncertainty or vagueness always causes identification problems. For these problems, where the input data are scalar features, many techniques have been successfully applied, such as fuzzy classification [1], or neural network classification [2, 3], but these algorithms can accept only scalar input data to solve the classification problem. When the input data are interval-values, these algorithms cannot achieve efficient classification. Classification using interval data using different strategies appears in literature. In [4], three different techniques of linear discriminative analysis are used for this type of classification problem: assigning a uniform distribution to each interval, expanding the dataset into the corresponding set of vertices and describing each interval via its center and its range. In [5], a radial basis function kernel is built, using a Hausdorff distance between intervals, and is applied to the classification of interval data.

In view of the foregoing, it is apparent that there still exists a need for an efficient method of radar emitter identification. Most of the methods that use neural networks are designed to process numerical data. For example, neural networks are used for classification [6, 7]. Ishibuchi and Nojima extended a normal (scalar-type) back-propagation (BP) learning algorithm to train a feed-forward neural network with fuzzy inputs and fuzzy outputs [8]. Shieh and Lin adapted the error function and proposed a new, vector-type back-propagation (NVTBP) algorithm for classification [9], which uses a three-layer vector neural network (VNN) to accept interval-value input data, as well as scalar input data, to achieve better classification. To deal with the classification of interval data, Liu et. al. [10] proposed a new identification algorithm that uses a combination of vector neural networks (CVNN), which is deduced from the back-propagation vector neural network and which allows nonlinear mapping between the interval-values of the input data.

This study aims to improve emitter identification (EID), using a model with interval type-2 fuzzy sets (IT2FSs) [11-15]. Specifically, IT2FSs are used as interval-values, in an extension of type-1 fuzzy sets. IT2FSs appear to be a more promising prospect than their type-1 counterparts, in handling EID that is affected by noise, or frequency jitter. That is, IT2FSs allow re-
searchers to model and minimize the effects of uncertainties in radar emitter signals. A feed-forward multilayer network, which integrates the IT2FS into a network structure that has structure and parameter learning ability is designed, called an interval type-2 fuzzy neural network (IT2FNN) [16-20]. To provide a better structure, a structural learning algorithm has been used for the IT2FNN, called as self-evolving IT2FNN [21-22]. However, it takes long learning time of these ST2FNNs by using evolution algorithm so that they are not suitable for the real time application systems. In this paper a self-organizing IT2FNN (ST2FNN) is proposed, in which, all of the rules are generated on-line by the proposed self-evolving structure for learning, which not only performs the rule generation through clustering, but also prunes unavailable rules. Moreover, a back-propagation algorithm is derived for the parameter training to improve the learning ability of IT2FS. Simulations for radar emitter identification are conducted to verify the performance of the ST2FNN. For the purposes of comparison, other identification algorithms, such as VNN [9] and CVNN [10] algorithms, are also used.

This work is organized as follows: Section 2 outlines the background of this work. Section 3 describes the interval type-2 fuzzy sets and the fuzzy neural network. In Section 4, a self-organizing interval type-2 fuzzy neural network (ST2FNN) is developed for emitter identification. Section 5 shows the simulation results. Finally, conclusions are drawn in Section 6.

2. Background of Radar Emitter Identification

Radar reconnaissance equipment, known as electronic support measures (ESM), is one of the most important elements of electronic warfare (EW). It performs threat detection and area surveillance, to determine the bearing and identity of surrounding radar emitters. The ESM receiver, which is a passive radar receiver, picks up the pulses emitted by surrounding radars in the environment and measures their identifying parameters such as radio frequency (RF), pulse repletion interval (PRI), pulse width (PW), angle of arrival (AOA), and time of arrival (TOA). The receiver is designed to cover a parameter range that is wide enough to ensure the detection of all radars of interest. A critical function is the real-time identification of the radar emitter type that is associated with each pulse train. The typical measurement parameters are divided into two groups; interval-values and scalar-values. The set of measured points for RF, PRI, PW and so on is subordinated in the form of a measured interval-value, i.e., with a lower limit and an upper limit. Figure 1(a) shows the interval-value for RF. The intensity in one main lobe (or main beam) is considerably stronger than in the other side lobe. The main lobe has an interval-value around the direction of maximum radiation. Figure 1(b) shows the pattern with radiation concentrated in scalar-values.

In the dense electromagnetic environments encountered during war, a large number of surrounding emitters can cause an ESM to receive a seemingly random pulse stream consisting of interleaved pulse trains with high noise levels. A received PRI is derived from the measured pulse time-of-arrival (TOA) values (i.e., a sequence of activation times). The TOA measurements are corrupted by TOA jitter and by both noise and missing TOA data. A jittered PRI is a pulse train wherein the PRI value is switched randomly within the bounds of a maximum and a minimum PRI value and where there is a variation in the starting time for each successive pulse, relative to the time that it would start, if the pulse train occurred at regular intervals. Jittered PRI explicitly implies that there is a random deviation in the interval around a mean value and that this deviation is homogeneously distributed, as shown in Figure 2.

From the subfigures (a) to (d) in Figure 2, it can be concluded that uncertainty in the PRI estimation has a
A strong influence on ESM performance. Over the past few years, jitter has become a more important signal property for ESM designers. The phenomena of signal skew and data jitter in a waveform not only affect data integrity and set-up and hold times, but magnify the signaling rate vs. transmission distance tradeoff, ultimately resulting in an inferior ESM system.

Based on the uncertain data information of radar emitter, an interval type-2 fuzzy neural network will be developed to identify these uncertain data.

3. Interval Type-2 Fuzzy Sets and Fuzzy Neural Network

In this section, an interval type-2 fuzzy sets (IT2FSs) are described first. This leads to the construction of the proposed interval type-2 fuzzy neural network (IT2FNN).

A. Type-2 fuzzy sets and fuzzy logic system

A type-1 fuzzy set (T1FS) is defined, which is in terms of a single variable, \( x \in X \), and is characterized by a membership function that takes values in the interval \([0, 1]\). It is defined as:

\[
\mu(x) \in [0, 1]
\]

where \( 0 \leq \mu(x) \leq 1 \) is a membership function.

Type-2 fuzzy sets (T2FS) were originally proposed by Zadeh in 1975 and are essentially “fuzzy fuzzy” sets where the fuzzy degree of membership is a T1FS \([11, 12]\). T2FS is a high level representation of vague data, and can handle the uncertainties in T1FS.

A T2FS, denoted as \( \tilde{A} \), is characterized by a type-2 \((T2)\) membership function (see Figure 3) \( \mu_2(x, \mu) \), where \( 0 \leq \mu_2(x, \mu) \leq 1 \), where \( x \in X \) and \( u \in J_x \subseteq [0,1] \). Fuzzy set \( \tilde{A} \) can be defined as

\[
\tilde{A} = \{ (x, \mu, u) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \} \quad (2)
\]

The domain of a secondary membership function is called the primary membership function of \( x \), and \( J_x \) is a primary membership function at \( x = x' \) (see Figure 3). For simplicity, \( u_2(x, \mu) \) can be written as \( \mu_2(x) \).

That is,

\[
\mu_2(x') = \sum_{\mu \in J_x} \mu_2(x', \mu) / \mu
\]

for \( \mu \in J_x \subseteq [0,1] \) and \( x' \in X \) (3)

when \( \mu_2(x', \mu) = 1 \), \( \forall u \in J_x \subseteq [0,1] \), then the secondary membership functions are interval sets, and, if this is true for \( \forall x \in X \), then the case of an interval type-2 membership function is obtained. Interval secondary membership functions reflect a uniform uncertainty at the primary memberships of \( x \).

Uncertainty concerning the shape and position of \( \tilde{A} \) is conveyed by the union of all of the primary memberships. This is called the footprint of uncertainty (FOU).

\[
FOU(\tilde{A}) = \bigcup_{x \in X} J_x \quad (4)
\]

Figure 3 shows an IT2FS with an uncertain mean. The FOU can be described in terms of an upper membership function (UMF) and a lower membership function (LMF), denoted as \( \bar{\mu}_2(x) \) and \( \underline{\mu}_2(x) \), which are two type-1 (T1) membership functions that are bounds for the FOU of an IT2FS, so an IT2FS can also be defined as follows:
An interval type-2 (IT2) membership function is a measure of the fuzziness in a T1 membership function. The IT2 Gaussian membership function is constructed using a Gaussian primary MF and an IT2 secondary membership function [23]. Figure 3 shows a two-dimensional IT2 Gaussian membership function with an adjustable uncertain mean \([m_1, m_2]\) and standard deviation \(\sigma\). It is described as:

\[
\mu_\lambda(x) = \exp\left[-\frac{1}{2}\left(\frac{x - m}{\sigma}\right)^2\right], \quad m \in [m_1, m_2]
\]  

(6)

An interval type-2 fuzzy logic system (IT2FLS) has five subsystems; a fuzzifier, an inference system, a rule base, a type-reducer and a defuzzifier (see Figure 4). The structure of the rules in the IT2FLS and its inference engine are similar to those in T1FLS. The IT2FLS works as follows: the crisp inputs are first fuzzified into IT2FS, which then activates the inference engine and the rule base to produce an output IT2FS. These IT2FS outputs are then processed by a type-reducer, which combines the output sets and then performs a centroid calculation. This results in an interval T1FS, called the type-reduced set. A defuzzifier then defuzzifies the type-reduced set to produce crisp outputs [24, 25].

B. The structure of the IT2FNN

The structure of an interval T2 fuzzy neural network (IT2FNN) is shown in Figure 5 and it is built according to the mechanism shown in Figure 4. This is an implementation of IT2FLS, and some of the parameters and components are presented using fuzzy logic terms. The IF–THEN rule for the IT2FNN can be expressed as:

\[ R^j : \text{IF } x_1 \text{ is } \tilde{F}_{1i}^j \text{ and } \ldots \text{ and } x_n \text{ is } \tilde{F}_{ni}^j \]  

Then \( y_1 \) is \([w_{1L}^j, w_{1U}^j]\) and...and \( y_n \) is \([w_{nL}^j, w_{nU}^j]\) (7)

where \( R^j \) is the \( j \)th rule \((j = 1, 2, \cdots, m)\), \( x_i \in [x_{1i}, x_{ni}] \) is the input, \( y \in [y_{1i}, y_{ni}] \) is the output, \( F_{ni}^j \) \((i = 1, 2, \cdots, n)\) is the IT2 fuzzy membership function of the antecedent part, which is defined as a Gaussian membership function, and \([w_{ni}^j, w_{ni}^j]\) is a centroid set with the membership grade of the secondary membership function set to unity, which can be called a weighting interval set, derived from IT2Fs in the consequent part [26]. Here the superscripts (L) and (U) are used to denote the lower limit and upper limit curves of an input data. The parameters of lower and upper membership functions are denoted by an underline (_) and up line (¯), respectively. The structure of the IT2FNN is shown in Figure 5, assuming that the IT2FNN has \( n \) inputs, \( x_i \) \((i = 1, 2, \cdots, n)\) and one output, \( y \), the realization algorithm is described as follows:
Layer 1: Input layer

The inputs are crisp values. For input range unification, each node in this layer is a scalar input \( x_i \) (\( i = 1, 2, \ldots, n \)). The nodes in this layer only transmit input values to the next layer directly. The net input and output are represented as

\[
O_i^{(1)} = x_i^{(1)}
\]

where \( x_i^{(1)} \) represents the \( i \)th input to the \( j \)th node.

Layer 2: Membership layer

This layer performs the fuzzification operation. Each node in this layer defines an interval type-2 membership function. For the \( j \)th fuzzy set \( F_i^{(j)} \) in input variable \( x_i \), a Gaussian primary membership function with a fixed standard deviation \( \sigma \) and an uncertain mean that takes values in \([m_i, m_i] \) is used (see Figure 3), i.e., the type-2 fuzzy membership function adopts the Gaussian interval type-2 membership function of (6). The footprint of uncertainty (FOU) of this membership function can be represented as a bounded interval, in terms of an upper certainty \( \text{FOU} \) of this membership function can be defined as follows [18]

\[
\text{FOU} = \left[ \mu_{F_i^{(j)}}, \bar{\mu}_{F_i^{(j)}} \right]
\]

where \( \mu_{F_i^{(j)}}, \bar{\mu}_{F_i^{(j)}} \) are used to denote the left and right curves of a Gaussian MF. For any value \( x_i \in [m_i, m_i] \), the output of each node can be represented as an interval \([\mu_{F_i^{(j)}}, \bar{\mu}_{F_i^{(j)}}] \). Therefore, for the network input \( x_i \), the layer output is

\[
O_i^{(2)} = \begin{bmatrix} (O_i^{(2)}_{l}) \quad (O_i^{(2)}_{r}) \end{bmatrix}^T = \begin{bmatrix} \mu_{F_i^{(j)}(O_i^{(1)})} \quad \bar{\mu}_{F_i^{(j)}(O_i^{(1)})} \end{bmatrix}^T
\]

where \( i = 1, 2, \ldots, n \), \( j = 1, 2, \ldots, m \) (9)

Layer 3: Rule layer

Each node in this layer is a rule node and performs a fuzzy meet operation using an algebraic product operation. The output of a rule node in the IT2FNN is a firing strength \( F_i^{(j)} \), which is computed as follows [17]

\[
F_i^{(j)} = \prod_{r=1}^{m} \left[ \prod_{l=1}^{n} \mu_{F_i^{(j)}(x_l)} \right]
\]

where \( \prod \) is the meet operation and \( \bigcup \) is the join operation [23]. The result can be an interval type-1 set [24] as

\[
F_i^{(j)} = \left[ f_i^{(j)} \quad \bar{f}_i^{(j)} \right]^T
\]

where

\[
f_i^{(j)} = \mu_{F_i^{(j)}(x_{l_1})} \mu_{F_i^{(j)}(x_{l_2})} \quad \bar{f}_i^{(j)} = \bar{\mu}_{F_i^{(j)}(x_{l_1})} \bar{\mu}_{F_i^{(j)}(x_{l_2})}
\]

Therefore, a simple PRODUCT operation is used to get the output

\[
O_i^{(3)} = \begin{bmatrix} O_i^{(3)}_{l} \quad O_i^{(3)}_{r} \end{bmatrix}^T = \begin{bmatrix} \mu_{F_i^{(j)}(O_i^{(1)})} \quad \bar{\mu}_{F_i^{(j)}(O_i^{(1)})} \end{bmatrix}^T
\]

Layer 4: Type reduction layer

Each node in this layer is called a consequent node. Each rule node in layer 3 has its own corresponding consequent node in layer 4. The center-of-sets type-reduction is used to get the output \( y_{i_{cos}} \), which is also an interval T1 set determined by the left and right end points \( (y_i, y_r) \), and it can be derived from the consequent centroid set \( w_i^j = [w_i^j, \bar{w}_i^j] \) and firing strengths \( f_i^j = f_i^j(x) = [f_i^j, \bar{f}_i^j] \) [18]. Here the superscripts \( (l) \) and \( (r) \) are used to denote the left and right curves of a Gaussian MF. For any value \( y_i \in [y_{li}, y_{ri}] \), \( y_i \) can be expressed as

\[
y_i = \frac{\sum_{j=1}^{n} f_i^j w_i^j}{\sum_{j=1}^{n} f_i^j}
\]

where \( y_i \) is a monotonic increasing function with respect to \( w_i^j \). Also, \( y_r \) is the minimum value associated only with \( w_i^j \), and \( y_l \) is the maximum value associated only with \( \bar{w}_i^j \). Note that \( y_i \) and \( y_r \) depend only on the mixture value of \( f_i^j \) or \( \bar{f}_i^j \) values. Hence, the left-most point and the right-most point can be expressed as [25]

\[
y_l = \frac{\sum_{j=1}^{n} f_i^j \bar{w}_i^j}{\sum_{j=1}^{n} f_i^j}
\]

and

\[
y_r = \frac{\sum_{j=1}^{n} \bar{f}_i^j \bar{w}_i^j}{\sum_{j=1}^{n} \bar{f}_i^j}
\]

The \( y_l \) and \( y_r \) can be computed efficiently using the Karnik-Mendel (KM) algorithms [26]. Each output node corresponds to one output variable. The node in this layer computes the output variable \( y_i \) by a defuzzification operation. Because the output of layer 4 is an interval set \([y_l, y_r] \), the node defuzzifies it by computing the average of \( y_l \) and \( y_r \). The defuzzified crisp output from an interval type-2 FLS is the average of \( y_l \) and \( y_r \), i.e.,

\[
y_{li} = \frac{y_{li} + y_{ri}}{2}
\]

and

\[
y_{ri} = \frac{y_{li} + y_{ri}}{2}
\]

The layer output is

\[
O_i^{(4)} = \begin{bmatrix} O_i^{(4)}_{l} \quad O_i^{(4)}_{r} \end{bmatrix}^T = y_{li}
\]

Layer 5: Output layer

The layer performs the weighted fusion (WF) of \( y_i \).
Finally, the crisp output of this layer is
\[ y = \sum_{i=1}^{n} \beta_i y_i = \sum_{i=1}^{n} \beta_i \]  
(20)
where \( \beta_i (i=1, 2, \ldots, n) \) is a weight.

The decision making rule for the IT2FNN is shown below:
\[ y = \begin{cases} 1 & y \geq \alpha \\ 0 & y < \alpha \end{cases} \quad \alpha \in [0, 1] \]  
(21)
where \( \alpha \) is a threshold. From (20), it is seen that the decision-making rule uses the weighted fusion (WF) output of the output interval from each neural network. Although each of the rules has its own characteristic, they have similar performance in radar emitter identification.

4. Self-Organizing Interval Type-2 Fuzzy Neural Network

The learning algorithm of self-organizing IT2FNN is shown in Figure 6, which includes two parts: self-organizing learning for the determination of fuzzy rules and parameter learning for the adjustment of the parameters in the fuzzy rules.

A. Self-organizing learning of IT2FNN

In many previously published papers, the structure of NN has been determined by trial-and-error, because it is difficult to strike a balance between the number of rules and the desired performance. The structural learning includes to add a new rule to IT2FNN and to prune an inefficient rule from IT2FNN. When generating rules, the mathematical description of the existing rules can be expressed as a cluster. Since one cluster in the input space corresponds to one potential fuzzy logic rule, the firing strength of a rule for each incoming data \( x_i \) can be represented as the degree to which the incoming data belong to the cluster. If a new input data \( x_i \) falls within the boundary of the clusters, the ST2FNN does not generate a new rule but update the parameters of the existing rules.

The clustering uses the distance from the mean \( MD_k, (k=1, \ldots, n) \) as a measure \[ MD_k (x) = \| x - m_k \| \]  
(22)
where \( x = [x_1, x_2, \ldots, x_n]^T, m_k = [m_{k1}, m_{k2}, \ldots, m_{kn}]^T \). A MAX-MIN method is proposed for rule growth. Find \[ \hat{k} = \arg \min_{1 \leq k \leq n} MD_k (x) \]  
(23)
If \[ \min_{1 \leq k \leq n} MD_k (x) > K_g \]  
(24)
where \( K_g \) is the predefined generating threshold, then a new rule is generated. Therefore, for a new input data, if the distance between input data and the mean is too large for the existing clusters; that is, the excitation value of the existing membership function is too small, a new cluster is generated. For the new rule, the weight is randomly generated and the initial mean and variance of the Gaussian membership function is defined as \[ m_{iok} = x_i \]  
(25)
\[ \sigma_{iok} = 0 \]  
(26)
\[ w_{iok} = 0 \]  
(27)
where \( x_i \) is the new incoming data and \( \sigma_i \) is a pre-specified constant. In this case, the number of rule is increased as follows:
\[ n_k (t+1) = n_k (t) + 1 \]  
(28)
where \( n_k \) is the number of the existing rule at time \( t \).

Another self-organizing structure learning is to determine whether to delete an inappropriate existing rule. A MAX–MIN method is proposed for rule pruning. For the output of the self-organizing IT2FNN in (20), the ratio of the \( k \)th component is defined as \[ MM_k (x) = \frac{v_j^i}{y_j} \]  
(29)
where \( v_j^i = f_i^j w_i^j \).

The corresponding minimum component is defined as follows:
\[ \bar{k} = \arg \min_{1 \leq k < |n_c|} \max_{1 \leq j \leq c} MM_k (x) \]  
(30)
If \[ MM_k (x) < K_c \]  
(31)
where \( K_c \) is a predefined deleting threshold, then the \( k \)th rule is deleted. This means that for an output data, if the minimum contribution of a rule is less than the deleting threshold, then this rule is deleted. With these
automatically rule generating and pruning, the proposed self-organizing IT2FNN can achieve the best suitable number of rules.

B. Parameter learning of IT2FNN

In parameter adjustment, the backpropagation (BP) method is used. The error function is defined as

\[ e_i = \frac{1}{2} (y_i - d_i)^2 \]  

(32)

where \( d_i \) is the target value.

The weighting factor \( w_i^j \), the mean \( m_i^j \) and the standard deviation \( \sigma_i^j \) of the Gaussian MF in the \( j \)th rule are adjusted as

\[
\begin{align*}
    w_i^j (p+1) &= w_i^j (p) + \Delta w_i^j \\
    m_i^j (p+1) &= m_i^j (p) + \Delta m_i^j \\
    \sigma_i^j (p+1) &= \sigma_i^j (p) + \Delta \sigma_i^j
\end{align*}
\]

(33)\-

(35)

The training algorithms in (33)–(35) perform error back-propagation by using a chain rule, that is

\[
\Delta \sigma_i^j = -\eta \frac{\partial e_i}{\partial \sigma_i^j} = -\frac{1}{2} \eta (y(x_i) - d_i) \left[ \frac{N(m_i^j, \sigma_i^j; x_i)}{\prod_{i=1}^{n} \hat{w}_{F_i}^j} \right]
\]

(36)

\[
\Delta \sigma_i^j = -\eta \frac{\partial e_i}{\partial \sigma_i^j} = -\frac{1}{2} \eta (y(x_i) - d_i) \times
\]

\[
\left[ \frac{(w_i^j - y)(x_i - m_i^j)}{\prod_{i=1}^{n} \hat{w}_{F_i}^j} \right] \frac{N(m_i^j, \sigma_i^j; x_i)}{\sigma_i^j}
\]

(37)

\[
\Delta \sigma_i^j = -\eta \frac{\partial e_i}{\partial \sigma_i^j} = -\frac{1}{2} \eta (y(x_i) - d_i) \times
\]

\[
\left[ \frac{(w_i^j - y)(x_i - m_i^j)}{\prod_{i=1}^{n} \hat{w}_{F_i}^j} \right] \frac{N(m_i^j, \sigma_i^j; x_i)}{\sigma_i^j}
\]

(38)

where \( N(m_i^j, \sigma_i^j; x_i) = \frac{1}{\sqrt{2\pi\sigma_i^j}} e^{-\frac{(x_i - m_i^j)^2}{2\sigma_i^j}} \) and \( \eta \) is a learning-rate, and \( \mu_{F_i}^j \) can be \( \mu_{F_i}^j * \) or \( \mu_{F_i}^j / \).

5. Simulation Results and Discussion

In this section, two experiments are performed for scalar-value and interval-value input data to demonstrate the identification capability of an IT2FNN trained using the ST2FNN algorithm. The performance is compared to that of a vector neural networks (VNN) trained using a CVNN algorithm [10] and a NVTBP algorithm [9], using the same training and testing data. All reference data for the ESM radar emitters are given in references [10, 28]. The ESM repository used for this study contains three parameters: The pulse repletion interval (PRI) is in the range \( 1.0 \mu s - 10.0 \mu s \), the pulse width (PW) is in the range \( 0.1 \mu s - 10.0 \mu s \) and the radio frequency of the carrier wave (RF) is in the range 2.0GHz \~ 18.0GHz.

Prior to each simulation, the data set is partitioned into training and testing subsets. One third of the data from each radar type is selected at random, to form the training subset. The testing data are described further in Subsections 5.1 and 5.2. The third experiment, in Subsection 5.3, analyzes the effect of a jittered ESM received pulse train. Three experiments are performed to demonstrate the learning and identification capability of the proposed ST2FNN algorithm. The first one verifies whether the ST2FNN algorithm could process scalar input data, with and without noise. The second one examines whether the ST2FNN algorithm could process interval-valued input data, with and without noise. The last one examines whether the ST2FNN algorithm could process scalar/interval-valued input data with jitter.

A. Performance evaluation on scalar-valued samples

In this experiment, all training interval-valued samples and testing scalar-valued samples (see Table 1) are randomly detected, where there are 20 emitter samples belong to 5 types. The objective is to show the performance of the ST2FNN algorithm, the CVNN algorithm and the NVTBP algorithm with scalar-valued input data with different noises.

<table>
<thead>
<tr>
<th>Emitter sample</th>
<th>RF, GHz</th>
<th>PRI, µs</th>
<th>PW, µs</th>
<th>Emitter type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower limit</td>
<td>Upper limit</td>
<td>Lower limit</td>
<td>Upper limit</td>
</tr>
<tr>
<td>1</td>
<td>10.01</td>
<td>10.01</td>
<td>8.16</td>
<td>8.16</td>
</tr>
<tr>
<td>2</td>
<td>4.32</td>
<td>4.32</td>
<td>3.13</td>
<td>3.13</td>
</tr>
<tr>
<td>3</td>
<td>16.61</td>
<td>16.61</td>
<td>5.53</td>
<td>5.53</td>
</tr>
<tr>
<td>4</td>
<td>16.89</td>
<td>16.89</td>
<td>5.51</td>
<td>5.51</td>
</tr>
<tr>
<td>5</td>
<td>8.82</td>
<td>8.82</td>
<td>6.30</td>
<td>6.30</td>
</tr>
<tr>
<td>6</td>
<td>15.83</td>
<td>15.83</td>
<td>8.42</td>
<td>8.42</td>
</tr>
<tr>
<td>7</td>
<td>15.80</td>
<td>15.80</td>
<td>8.48</td>
<td>8.48</td>
</tr>
<tr>
<td>8</td>
<td>5.82</td>
<td>5.82</td>
<td>8.66</td>
<td>8.66</td>
</tr>
<tr>
<td>9</td>
<td>6.02</td>
<td>6.02</td>
<td>8.79</td>
<td>8.79</td>
</tr>
<tr>
<td>10</td>
<td>8.68</td>
<td>8.68</td>
<td>6.68</td>
<td>6.68</td>
</tr>
<tr>
<td>11</td>
<td>16.61</td>
<td>16.61</td>
<td>5.53</td>
<td>5.53</td>
</tr>
<tr>
<td>12</td>
<td>4.20</td>
<td>4.20</td>
<td>3.21</td>
<td>3.21</td>
</tr>
<tr>
<td>13</td>
<td>5.97</td>
<td>5.97</td>
<td>8.36</td>
<td>8.36</td>
</tr>
<tr>
<td>14</td>
<td>5.62</td>
<td>5.62</td>
<td>8.27</td>
<td>8.27</td>
</tr>
<tr>
<td>15</td>
<td>4.49</td>
<td>4.49</td>
<td>2.60</td>
<td>2.60</td>
</tr>
<tr>
<td>16</td>
<td>9.05</td>
<td>9.05</td>
<td>6.78</td>
<td>6.78</td>
</tr>
<tr>
<td>17</td>
<td>16.04</td>
<td>16.04</td>
<td>8.45</td>
<td>8.45</td>
</tr>
<tr>
<td>18</td>
<td>4.38</td>
<td>4.38</td>
<td>3.27</td>
<td>3.27</td>
</tr>
<tr>
<td>19</td>
<td>16.89</td>
<td>16.89</td>
<td>5.94</td>
<td>5.94</td>
</tr>
<tr>
<td>20</td>
<td>8.97</td>
<td>8.97</td>
<td>6.46</td>
<td>6.46</td>
</tr>
</tbody>
</table>

Three scenarios, including three-EID (Emitter Identification), four-EID and five-EID are examined to demonstrate the identification capability of the proposed IT2FNN, in order to examine whether the ST2FNN al-
algorithm could add a new type emitter into ESM repository. The self-organizing algorithm can dynamically add or prune the fuzzy rules, according to the influence of the firing strength on the third layer. The parameters $\eta = 0.5$, $K_p = 0.3$, $K_c = 0.1$, and $\alpha = 0.5$ were used for the ST2FNN algorithm. The results for the fuzzy rules updating process, using the ST2FNN algorithm, are shown in Figure 7. It can be seen that the ST2FNN produces the least number of rules and gives a clear illustration of the tendency for rule evolution between 0 and 10 observations and that ST2FNN can dynamically add or prune a rule, during structural learning.

$$EDL_i(\%) = \frac{\xi_{pi}}{x_{pi}} \times 100$$

(39)

where $x_{pi}^L$ is the data without noise and $\xi_{pi}$ is a random noise. The scalar-valued samples are $x_{pi} = (x_{p1}, x_{p2}, x_{p3})$ corresponding to RF, PRI and PW, respectively. $\xi_{pi} = (\xi_{p1}, \xi_{p2}, \xi_{p3})$ is the random noise corresponding to $x_p$, so the noisy scalar-valued sample is $(x_{p1} \pm \xi_{p1}, x_{p2} \pm \xi_{p2}, x_{p3} \pm \xi_{p3})$. The interval-valued samples is $x_p = (x_{p1}, x_{p2}, x_{p3})$, briefly, $x_{pi} = [x_{pi}^L, x_{pi}^U]$ where the superscript $L$ denotes a lower limit and the superscript $U$ denotes an upper limit. $\xi_{pi} = [\xi_{pi}^L, \xi_{pi}^U]$ is the random noise corresponding to $x_{pi}^L$, so the noisy test interval-valued sample is $x_{pi} = [x_{pi}^L \pm \xi_{pi}^L, x_{pi}^U \pm \xi_{pi}^U]$.

Table 2. Performances comparison for scalar-valued test samples.

<table>
<thead>
<tr>
<th>Test case</th>
<th>NVTBP (Total average EID rate, %)</th>
<th>CVNN (Total average EID rate, %)</th>
<th>ST2FNN (Total average EID rate, %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Three EID</td>
<td>Four EID</td>
<td>Five EID</td>
</tr>
<tr>
<td>EDL %</td>
<td>EID prob-lem len</td>
<td>EID prob-lem len</td>
<td>EID prob-lem len</td>
</tr>
<tr>
<td>20%</td>
<td>79.3</td>
<td>83.53</td>
<td>65.43</td>
</tr>
<tr>
<td></td>
<td>80.73</td>
<td>83.80</td>
<td>67.69</td>
</tr>
<tr>
<td>18%</td>
<td>85.30</td>
<td>84.99</td>
<td>71.91</td>
</tr>
<tr>
<td>16%</td>
<td>87.54</td>
<td>87.23</td>
<td>75.67</td>
</tr>
<tr>
<td>14%</td>
<td>90.52</td>
<td>90.12</td>
<td>81.64</td>
</tr>
<tr>
<td>12%</td>
<td>98.14</td>
<td>93.69</td>
<td>93.83</td>
</tr>
<tr>
<td>10%</td>
<td>99.07</td>
<td>94.55</td>
<td>93.77</td>
</tr>
<tr>
<td>8%</td>
<td>99.53</td>
<td>97.89</td>
<td>93.45</td>
</tr>
<tr>
<td>6%</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4%</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2%</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

The IT2FNN and VNN networks were trained using interval-valued samples and also tested using scalar-valued samples. In the training phase, 30 interval-valued training samples were selected for the 3-EID problem, 40 interval-valued training samples for the 4-EID problem and 50 interval-valued training samples for the 5-EID problem. The testing results for 100 Monte Carlo simulations are listed in Table 2. From Table 3, it can be seen that the proposed ST2FNN algorithm is more adaptable to noise. For the 3-EID problem with an EDL larger than 10%, the identification rate using the ST2FNN algorithm is approximate 10% better than with the NVTBP algorithm; for the 4-EID problem and the 5-EID problem, the ST2FNN algorithm performs better in both high noise environments and low noise environments than the NVTBP and CVNN algorithms.

Figure 7. Rule updating process using the self-organizing learning of the ST2FNN algorithm.

In this experiment, noisy testing samples with different error deviation levels (EDL) (from 0 to 20%) were used. The EDL is defined as:
B. Performance evaluation using interval-valued samples

In this experiment, all of the training interval-valued samples and testing interval-valued samples (see Table 3) were randomly selected from the ESM repository.

Table 3. Interval-valued test emitter samples from the ESM repository.

<table>
<thead>
<tr>
<th>Emitter sample</th>
<th>RF, GHz</th>
<th>PRI, µs</th>
<th>PW, µs</th>
<th>Emitter type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower limit</td>
<td>Upper limit</td>
<td>Lower limit</td>
<td>Upper limit</td>
</tr>
<tr>
<td>1</td>
<td>15.77</td>
<td>16.57</td>
<td>8.21</td>
<td>9.33</td>
</tr>
<tr>
<td>2</td>
<td>15.60</td>
<td>16.13</td>
<td>8.16</td>
<td>8.92</td>
</tr>
<tr>
<td>3</td>
<td>15.85</td>
<td>16.54</td>
<td>8.62</td>
<td>9.46</td>
</tr>
<tr>
<td>4</td>
<td>15.69</td>
<td>16.45</td>
<td>8.35</td>
<td>9.52</td>
</tr>
<tr>
<td>5</td>
<td>4.42</td>
<td>4.81</td>
<td>2.93</td>
<td>3.49</td>
</tr>
<tr>
<td>6</td>
<td>4.50</td>
<td>4.83</td>
<td>2.84</td>
<td>3.96</td>
</tr>
<tr>
<td>7</td>
<td>4.59</td>
<td>4.99</td>
<td>3.14</td>
<td>4.06</td>
</tr>
<tr>
<td>8</td>
<td>4.45</td>
<td>4.71</td>
<td>3.14</td>
<td>3.55</td>
</tr>
<tr>
<td>9</td>
<td>16.37</td>
<td>17.37</td>
<td>5.88</td>
<td>6.80</td>
</tr>
<tr>
<td>10</td>
<td>16.54</td>
<td>17.40</td>
<td>5.78</td>
<td>6.14</td>
</tr>
<tr>
<td>11</td>
<td>16.85</td>
<td>16.93</td>
<td>5.47</td>
<td>6.42</td>
</tr>
<tr>
<td>12</td>
<td>16.72</td>
<td>17.40</td>
<td>5.16</td>
<td>6.03</td>
</tr>
<tr>
<td>13</td>
<td>3.84</td>
<td>4.26</td>
<td>8.55</td>
<td>9.04</td>
</tr>
<tr>
<td>14</td>
<td>6.07</td>
<td>6.16</td>
<td>8.42</td>
<td>9.42</td>
</tr>
<tr>
<td>15</td>
<td>5.76</td>
<td>6.34</td>
<td>8.49</td>
<td>8.88</td>
</tr>
<tr>
<td>16</td>
<td>6.03</td>
<td>6.54</td>
<td>8.84</td>
<td>9.24</td>
</tr>
<tr>
<td>17</td>
<td>8.75</td>
<td>9.10</td>
<td>6.58</td>
<td>7.04</td>
</tr>
<tr>
<td>18</td>
<td>9.04</td>
<td>9.21</td>
<td>6.33</td>
<td>7.40</td>
</tr>
<tr>
<td>19</td>
<td>8.61</td>
<td>9.38</td>
<td>6.81</td>
<td>7.27</td>
</tr>
<tr>
<td>20</td>
<td>8.98</td>
<td>9.35</td>
<td>6.64</td>
<td>6.90</td>
</tr>
</tbody>
</table>

In the testing phase, 90 interval-valued samples were randomly selected for the 3-EID problem, 120 interval-value samples for the 4-EID problem and 150 interval-value samples for the 5-EID problem. In this experiment, noisy test samples with different EDLs (from 0 to 20%) are used for the performance testing. The testing results for 100 Monte Carlo simulations are listed in Table 4. From Table 4 it can be seen that the ST2FNN algorithm is more adaptable to noise. The ST2FNN algorithm performs better than the NVTBP and CVNN algorithms. When the EDL is equal to 20%, the identification rate of the ST2FNN algorithm is 93.02% and 84.09%, respectively, which is 5% better than the NVTBP algorithm and 22% better than the CVNN algorithm. For the 5-EID problem, with an EDL higher than 12%, the identification rate for the ST2FNN algorithm is approximate 5% higher than that of the CVNN algorithm and 25% better than the NVTBP. When the EDL reaches 20%, the identification rate for the ST2FNN algorithm is 81.37%, whereas that of the CVNN algorithm is approximate 77.67% and that for the NVTBP is 54.95%.

From Tables 2 and 4, it can be also seen that the performance of the ST2FNN decreases slowly as the EDL increases in noisy environments, so it is concluded that the proposed ST2FNN algorithm not only has better identification capability, but also is relatively less sensitive to noise than the CVNN and NVTBP algorithms.

C. Performance evaluation using scalar/interval-valued samples with jitter variation

This experiment demonstrates the identification capability of the ST2FNN algorithm using scalar/interval-valued data and interval-valued data with jitter variation. Because of its random nature, jitter is normally described in terms of its probability density function (PDF) on the lower limit (L) edge and the upper limit (U) edge. The measurement jitter due to noise is a white Gaussian noise [29], so the corresponding interval \( [x_{pi}^L, x_{pi}^U] \) is

\[
x_{pi}^L = x_{pi} - \text{rand} \left( [x_{pi} - \text{MinX} \cdot r] \right)
\]

(40)
\[ x^U_{\mu_i} = x_{\mu_i} + \text{rand} \left( [x_{\mu_i} - \text{MinX}] \cdot r \right) \]  

where \( \text{rand}(x) \in [0, x] \) is the random function, \( \text{MinX} \) is the minimum value of all \( x_{\mu_i} \) in the entire sequence, and \( r \in [0, x] \) is a real number, which is a relative random ratio \( (r = 0.2 \text{ in this experiment}) \).

The interval-valued samples are used for training, whereas both interval-valued and scalar-valued samples are used for testing. To perform the testing at different levels of additive noise, a jitter variation is also introduced to the testing samples. The test samples with different EDLs (from 0 to 20%) are presented to the trained IT2FNN for performance testing. The testing results for 100 Monte Carlo simulations are listed in Table 5. From Table 5, it is seen that the ST2FNN algorithm has a high identification rate for the testing samples. In conclusion, Table 5, it is seen that the ST2FNN algorithm has a high identification rate for the testing samples. In conclusion, the integration of the IT2FSs and the functions of fuzzy inference into a self-organizing IT2FNN effectively reduces the effect of uncertainties in radar emitter signals with significant noise and jitter variations.

6. Conclusions

An interval type-2 fuzzy neural network (IT2FNN) and a self-organizing IT2FNN algorithm are proposed for radar emitter identification. The main contribution of this paper is the method for the integration of interval-valued and scalar-valued data into a single processing system and the derivation of a ST2FNN learning algorithm solves the practical EID problems in real time. The simulations show that the proposed ST2FNN algorithm not only has better identification capability, but is also relatively less sensitive to noise than the other neural networks such as CVNN and NVTBP. These results show that the proposed IT2FNN is widely applicable to ESM or radar warning receiver applications, in order to achieve better identification performance. This paper shows that the proposed IT2FNN can be used to identify unambiguous emitters with jitter variations. In the future work, extra parameters for emitters such as the angle of arrival and amplitude will be used to form new enlarged input features, to address the problems associated with multiple ambiguous emitters.

Acknowledgment

This work was supported in part by the National Science Council of Taiwan under Grant NSC98-2221-E-15 5-058-MY3.

References


