Received January 31, 2020, accepted February 21, 2020, date of publication February 27, 2020, date of current version March 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976731

The Soft Sets and Fuzzy Sets-Based Neural Networks and Application

ZHICAI LIU[1,2], JOSÉ CARLOS R. ALCANTUD[3], KEYUN QIN[2], AND LING XIONG[1]
[1]School of Computer and Software Engineering, Xihua University, Chengdu 610039, China
[2]School of Mathematics, Southwest Jiaotong University, Chengdu 610031, China
[3]BORDA Research Unit and Multidisciplinary Institute of Enterprise (IME), University of Salamanca, 37008 Salamanca, Spain

Corresponding author: Ling Xiong (lingdonghua99@163.com)

This work was supported in part by the National Science Foundation of China under Grant 61175044, Grant 61372187, and Grant 61473239, and in part by the Civil Aviation Administration of China under Grant PSDSA201802.

ABSTRACT This paper reviews and compares theories of fuzzy sets and soft sets from the perspective of transformation, and a machine learning model—SF-ANN (the soft sets and fuzzy sets based artificial neural network) is proposed. Liu et al. proved that every fuzzy set on a universe U can be considered as a soft set, and show that any soft set can be regarded as even a fuzzy set. Inspired by this idea, we construct a neuron-like structure based on soft sets and fuzzy sets, and we get a more practical fuzzy learning model—SF-ANN. In practical applications, it can be used as a general methodology for establishing the membership function of fuzzy sets, and it also can be applied to pattern recognition, decision-making, etc. In general, it provides a new perspective to observe the relationship between soft sets and fuzzy sets, and it is easy to relate soft set theory and fuzzy set theory to machine learning methods. To a certain extent, it reveals that the research of fuzzy sets and artificial neural networks do lead to the same destination.

INDEX TERMS Fuzzy sets, soft sets, neural networks.

I. INTRODUCTION

Artificial intelligence is rapidly coming of age, and we live in the midst of a data deluge. In this period, we may face all kinds of challenges. That is, to deal with this big data, methods in classical mathematics are not always successful because of various types of uncertainties presented in these problems, such as fuzziness, randomness, roughness, ambiguity, inaccuracy, incompleteness and so on. There are some mathematical tools for dealing with uncertainties, among which fuzzy set theory introduced by Zadeh [1], and soft set theory initiated by Molodtsov [2] have paramount importance. The theory of fuzzy sets initiated by Zadeh provides an appropriate framework for representing and processing vague concepts by allowing partial memberships. From a more descriptive perspective, Molodtsov proposed the theory of soft sets, which has proven useful in many fields such as decision-making [3]–[10], data mining [11], [12], business [13], social choice [14], forecasting [15], clinical diagnosis [16], etc.

The associate editor coordinating the review of this manuscript and approving it for publication was M. Venkateshkumar.

Many references have discussed the relationship between fuzzy sets and soft sets. In [2] Molodtsov points out that Zadeh's [1] fuzzy set is a special case of the soft set. Aktaş and Çağman [17] proved that fuzzy set and rough set are all special cases of the soft set. Yao [18] studies relationships and differences between theories of fuzzy sets and rough sets and points out that a fuzzy set can be interpreted by a family of crisp sets, and fuzzy set operators can be defined using standard set operators. Alcantud [19] discusses some transformation relations between fuzzy sets and soft sets. He proves that every fuzzy set can be considered as a special case of a soft set; every soft set on a finite field U can be considered as a fuzzy set. Bustince *et al.* [20] proved that fuzzy sets are intuitionistic fuzzy sets. Torra [21] shows that hesitant fuzzy sets can be expressed as fuzzy sets and 2-valued fuzzy sets, and Alcantud and Torra [22] prove novel decomposition theorems for hesitant fuzzy sets. Bustince *et al.* [20] show that fuzzy sets and interval-valued fuzzy sets are particular cases of interval type-2 fuzzy sets.

Molodtsov proposed that Zadeh's fuzzy set may be considered as a special case of soft sets. Here we extend this conclusion and compare the soft sets and the fuzzy sets from the perspective of conversion. The attribute sets of the soft sets and the membership function of the fuzzy sets are all being used to describe a characteristic or feature of an object. For soft sets, each attribute represents one view of the vague concept, and then a fuzzy set may be interpreted as a weighted combined view. From the perspective of establishing the membership function of the fuzzy sets, this is the 'other way' proposed by Molodtsov to build the membership function of a fuzzy set. From the viewpoint of the decomposition of the attributes, there is a relationship between 'integration' and 'decomposition.' In this paper, the structure of $\mu(x) = (F, A)W$ is discussed, and realize the transformation through aggregation operators. Based on the above research, a more practical fuzzy learning model is generated, which is composed of soft sets, fuzzy sets, and neural networks. Let us succinctly explain this later framework.

McCulloch and Pitts [23] introduced the first artificial neurons in 1943. Over the years the neural networks have been widely used for the solution of complex non-linear problems such as system game-playing and decision making [24], pattern recognition [25], medical diagnosis [26]–[30], diverse applications in control [31], and so on. The formula $\mu(x) = (F, A)W$ shows that the fuzzy sets can be generated from soft sets by 'learning.' Inspired by this, we construct a neural network based on soft set and fuzzy set. In this model, we take the soft set as input and the fuzzy set as the output. Through the existing research on neural networks, we know that neurons (single layer neural network) can only be used as linear classifications. One can also extend it to two or more layers network, such as $\mu(x) = f(f((F, A)W_1)W_2 + b)$, which can be used to solve more complex (non-linear) problems. By using this model, when solving practical problems, all we need to do is focus on the front-end 'S' (soft sets) and back-end 'F' (fuzzy sets), and the neural network used as a tool for solving $\{(W_1, b_1), (W_2, b_2), \cdots, (W_n, b_n)\}$. At the front-end and back-end, we can take full advantage of the soft sets and fuzzy sets theories, to realize the information fusion/reduction, etc. Since we integrate soft sets, fuzzy sets, and neural network theory by using SF-ANN, these theories can be viewed from a holistic perspective, which provides a more practical model for solving practical problems. Finally, the proposed algorithm is simulated and uploaded to [32], which can be easily downloaded to validate all the examples.

The rest of this paper is organized as follows. Section II reviews the basic concepts of fuzzy set, soft set, and induced soft set. Section III discusses the existing transformation methods between fuzzy sets and soft sets. In Section IV, we propose a fuzzy learning model–SF-ANN based on the neural network, including examples. Finally, Section VI presents the conclusion and future work.

II. PRELIMINARIES

In this section, we will briefly recall some related basic concepts. These concepts include fuzzy sets, soft sets, and some examples are given to illustrate the concepts.

A. FUZZY SETS

In 1965, Zadeh [1] proposed a mathematical method of describing the fuzzy phenomenon in mathematics: fuzzy set theory. A fundamental notion in this theory is as follows:

Definition 1 [1]: Let U be a set, called a universe. A fuzzy set μ on U is defined by a membership function $\mu : U \rightarrow [0, 1]$. For any $x \in U$, the value $\mu(x)$ represents the extent to which x belongs to the fuzzy set μ.

The fuzzy set μ is also denoted as follows:

$$\mu = \{(x, \mu(x)) : x \in U\} \tag{1}$$

A fuzzy set can be either discrete or continuous. For discrete fuzzy sets, $\mu(x)$ can also be expressed as follows:

$$\mu(x) = \sum_{i=1}^{n} (\mu(x_i)/x_i) \tag{2}$$

n being the number of elements in U.

There are several forms of operations on fuzzy sets. According to maximum - minimal operator proposed by Zadeh [1], the intersection, union, and complement of fuzzy sets are defined as follows:

$$(\mu \cap \upsilon)(x) = \mu(x) \wedge \upsilon(x),$$
$$(\mu \cup \upsilon)(x) = \mu(x) \vee \upsilon(x),$$
$$\mu^c(x) = 1 - \mu(x).$$

B. SOFT SETS AND INDUCED SOFT SETS

Let U be the universe set and E the set of all possible parameters under consideration with respect to U. Usually, parameters are attributes, characteristics, or properties of objects in U. (U, E) will be called a soft space. Let $\mathcal{P}(U)$ denote the set of all subsets of U. Molodtsov defined the notion of a soft set in the following way:

Definition 2 [2]: A pair (F, A) is called a soft set over U, when $A \subseteq E$ and F is a mapping given by $F : A \rightarrow \mathcal{P}(U)$.

That is, a soft set over U is a parameterized family of subsets of U. A is called the parameter set of the soft set (F, A). For $e \in A$, $F(e)$ may be considered as the set of $e-$approximate elements of (F, A).

Definition 3 [33]: For any soft set (F, E) over U, a pair (F^{-1}, U) is called an induced soft set over E of (F, E), where $F^{-1}(x) = \{a \in E : x \in F(a)\}$ for each $x \in U$.

Definition 4: The set X is indexed by a set I, if there is an onto function $f : I \rightarrow X$. Let $C = \{X_i : i \in I\}$ be an indexed collection of sets X_i. The Cartesian product of C is the set of functions that assign to each index $i \in I$ an element $x_i \in X_i$. That is,

$$\prod_{i \in I} X_i = \left\{ f : I \rightarrow \bigcup_{i \in I} X_i \text{ such that } f(i) \in X_i \text{ for every } i \in I \right\}$$

Let $\varsigma = \{0, 1\}$ be a set with two elements. Let $\Sigma = \varsigma^N = \{(s_1, s_2, s_3, \cdots, s_k, \cdots) : s_k \in \varsigma\}$ denote the set of all binary sequences, that is, sequences whose terms are either 0 or 1.

Proposition 1: Let I be a set. A subset $A \in I$ can be identified with its characteristic function $\chi_A : I \rightarrow \varsigma$ defined

TABLE 1. An induced soft set represented as a Boolean-valued information system.

U	e_1	e_2	e_3
x_1	0	0	0
x_2	0	1	0
x_3	0	1	1
x_4	1	0	0
x_5	1	0	1

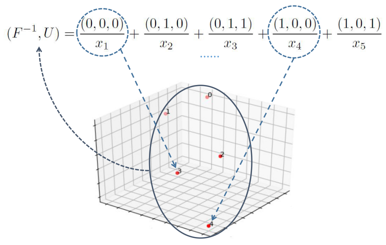

$(F^{-1},U) = \frac{(0,0,0)}{x_1} + \frac{(0,1,0)}{x_2} + \frac{(0,1,1)}{x_3} + \frac{(1,0,0)}{x_4} + \frac{(1,0,1)}{x_5}$
......

FIGURE 1. The induced soft sets in three-dimensional space.

by: $i \in A$ if and only if $\chi_A(i) = 1$. Thus, $A \mapsto \chi_A$ defines a one-to-one map from $\mathcal{P}(I)$ onto 2^I, the set of functions from I to ς.

When I is is finite with cardinality n, each χ_A can be identified with a sequence from ς^n. When I is denumerably infinite, each χ_A can be identified with a sequence from Σ.

In our example, one readily checks that we can represent the induced soft set in Table 1 by the following expression:

$$(F^{-1}, U) = \frac{(0,0,0)}{x_1} + \frac{(0,1,0)}{x_2} + \frac{(0,1,1)}{x_3} + \frac{(1,0,0)}{x_4} + \frac{(1,0,1)}{x_5}$$

The induced soft set can be regarded as a group of points in n-dimensional space composed of parameter sets, as shown in Figure 1. Therefore, the soft set and machine learning methods can be intuitively linked.

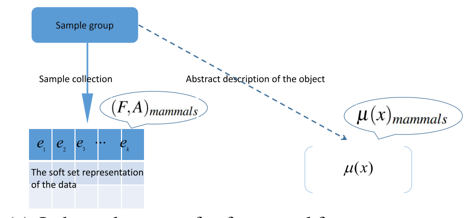
III. THE NEURON BASED ON SOFT SET AND FUZZY SET AND ITS LIMITATIONS

In practical applications, fuzzy sets and soft sets are often used as independent tools to solve practical problems. As shown in Figure 2(a), the current difficulties are:

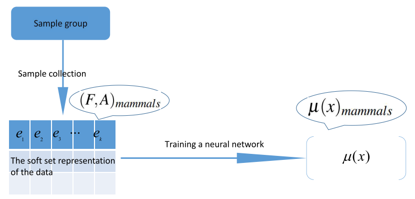
(1) When fuzzy sets are used to solve practical problems, the construction of membership functions is difficult and subjective;

(2) When using soft sets to solve practical problems, it usually has a large amount of data, and the essential characteristics behind the data are not revealed. That means, all operations are based on the original data, this is usually not a good idea under big data conditions.

Taking the experimental V as an example, it is challenging for us to build the membership function $\mu(x)_{mammals}$ directly, but it is easier to describe 'mammals' with $(F, A)_{mammals}$. According to the existing research on soft sets, we can use soft set $(F, A)_{mammals}$ to perform similarity measurement, classification, etc. on 'mammals,' but this usually has a large


Sample group
Sample collection
Abstract description of the object
$(F,A)_{mammals}$
$\mu(x)_{mammals}$
e_1 e_2 e_3 ... e_k
The soft set representation of the data
$\mu(x)$

(a) Independent use of soft sets and fuzzy sets


Sample group
Sample collection
$(F,A)_{mammals}$
$\mu(x)_{mammals}$
e_1 e_2 e_3 ... e_k
The soft set representation of the data
Training a neural network
$\mu(x)$

(b) Constructing fuzzy sets by using soft sets

FIGURE 2. The logical relationship of soft set and fuzzy set.

amount of computation, and the implied 'mammals' abstract characteristics of these samples are not revealed. Inspired by the idea of transformation between soft sets and fuzzy sets [34], we can find an appropriate method to convert soft sets into fuzzy sets, thus constructing the membership function $\mu(x)_{mammals}$ of the fuzzy set about 'mammals.' This idea can be expressed in Figure 2.

This method of constructing membership functions can also be considered as the original intention of Molodtsov's proposed soft set: 'In classical mathematics, we construct a mathematical model of an object and define the notion of the exact solution of this model. Usually, the mathematical model is too complicated, and we cannot find the exact solution. So, in the second step, we introduce the notion of an approximate solution and calculate that solution' [2].

The idea of using the SF-ANN model to solve the problem is similar to the neural network. As shown in Figure 2:

1) As previously described, an induced soft set can be represented as a set of points in an n-dimensional space, as shown in Figure 1;
2) Training the soft set by using neural networks, we can construct a fuzzy set $\mu(x)$. From the perspective of classification, $\mu(x)$ is actually a curve, surface or hypersurface in n-dimensional space;
3) For a new sample T, $\mu(T)$ indicates the extent to which T belongs to that category.

The SF-ANN model has the following advantages:

1) It can be used as a general method to construct the membership function of the fuzzy set;
2) It reveals the relationship between ambiguity and accuracy, and provides a new perspective for the relationship between fuzzy set and soft set;
3) It provides a further reference for the application of soft set and fuzzy set in the field of artificial intelligence.

Generally speaking, we give a general way to construct membership functions of fuzzy sets, and this paper reveals the implicit relationship between soft set, fuzzy set, and neural network.

A. THE CONSTRUCTION OF FUZZY SETS FROM SOFT SETS BASED ON MODAL LOGIC OPERATOR

Klir [35] proposed a formulation of fuzzy sets based on modal logic. In this model, a vague concept is characterized by some, possibly different, crisp sets. Let $W = \{w_1, \cdots, w_n\}$ denote a set of n possible states of the world or simply states. With respect to W, a vague concept is represented by n crisp sets: $\mathcal{A} = (A_{w_1}, \cdots, A_{w_n})$. In each world w_i, the vague concept is described precisely by a crisp set A_{w_i}.

Let U be a finite and non-empty set called universe. The set all families of n crisp sets is given by the n-fold Cartesian product of 2^U, $\Pi_n 2^U = 2^U \times \cdots \times 2^U$. Suppose $\Omega : W \rightarrow [0, 1]$ is a weighting function satisfying the condition: $\sum_{i=1}^{n} \omega_i = 1$. For an element of $\Pi_n 2^U$ representing a vague concept, Yao [18] proposed a fuzzy set model defined by:

$$\mu_A(x) = \sum_{i=1}^{n} \omega_i \mu_{A_{w_i}}(x), \tag{3}$$

where $\mu_{A_{w_i}}$ is the characteristic function of A_{w_i}. If each crisp set A_{w_i} represents one view of the vague concept, a fuzzy set may be interpreted as a weighted combined view [18].

An aggregation operator is mapping a function, which assigns a real number to any n-tuple $(e_1, e_2, \cdots, e_n)$ of real numbers [36]:

$$\mu = Aggreg(e_1, e_2, \cdots, e_n)$$

Thus, each soft set can be transformed into a fuzzy set by using an aggregation operator. The different aggregation operators have different physical meanings. By Eq. (3), which interprets a fuzzy set as a weighted combined view of soft set, as shown in Figure 3.

$$\mu(x) = (F, A)W = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \end{pmatrix}$$
$$= \begin{pmatrix} \sum_{i=1}^{k} \omega_i u_{1i} \\ \sum_{i=1}^{k} \omega_i u_{2i} \\ \vdots \\ \sum_{i=1}^{k} \omega_i u_{ni} \end{pmatrix} \tag{4}$$

where (F, A) and W are described as matrices.

B. THE SOFT SETS AND FUZZY SETS BASED PERCEPTRON

In 1943, McCulloch and Pitts [23] introduced the first artificial neurons. A neural network is a bio-inspired system with several single processing elements, called neurons. The neurons are connected to each other by a joint mechanism which is consisted of a set of assigned weights. The neuron calculates the weighted sum of the inputs and compares the result with a threshold value Θ. If the final sum is less than

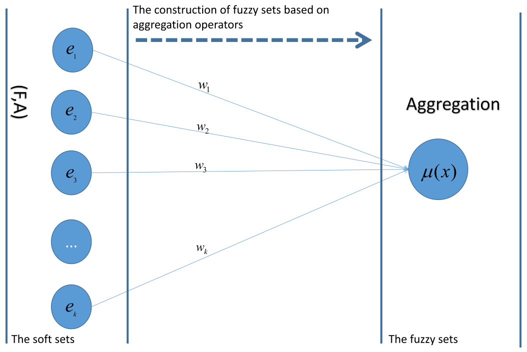


FIGURE 3. The construction of fuzzy sets from soft sets based on modal logic operator.

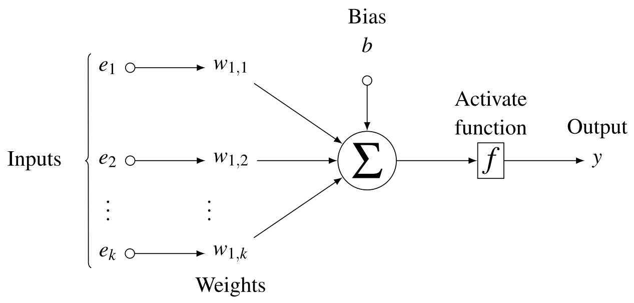


FIGURE 4. The perceptron.

the threshold, the output is set to -1, otherwise to $+1$. It can be represented as:

$$Y = sign[\sum_{i=1}^{n} x_i \omega_i - \Theta] \tag{5}$$

that is,

$$X = \sum_{i=1}^{n} x_i \omega_i, \quad Y = \begin{cases} +1 & if\ X \geq \Theta \\ -1 & if\ X < \Theta \end{cases}$$

A neuron with k inputs is shown in Fig.(4), it's a multiple-input neuron [37].

Let $U = \{u_1, u_2, \cdots, u_n\}$, (F, A) be a soft set with k attributes $E = \{e_1, e_2, \cdots, e_k\}$, and $\mu(x)$ be a fuzzy set. They can be expressed as a matrix. We consider using the soft set as input and fuzzy set as output and using the sigmoid function as the activation function. The inputs $e_1, e_2, \cdots, e_k$ each weighted by corresponding elements $w_1, w_2, \cdots, w_k$ of the weight matrix W. The b is the bias of the neuron, and it sums the weighted input to form a net input n. As shown in Figure 3.

$$n = w_1 e_1 + w_2 e_2 + \cdots + w_k e_k + b$$

This expression can be written in matrix form:

$$n = WE + b \tag{6}$$

The neuron output can be written as

$$y = f(WE + b) \tag{7}$$

where $f = \frac{1}{1+e^{-X}}$.

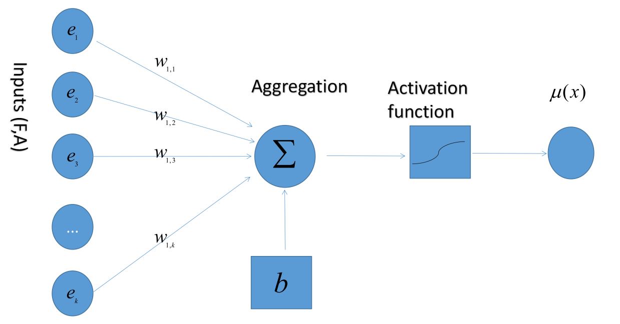


FIGURE 5. The perceptron based on soft set.

By Eq. (4), we have:

$$\mu(x) = f((F, A)W + b) \tag{8}$$

As shown in the Fig.(5).

Fig. (4) and (5) shows that the conversion of soft sets to fuzzy sets has the same structure as the multiple-input neuron. That means we can use the soft set (F, A) as input and get the membership function $\mu(x)$ of fuzzy set by training the neural networks.

C. THE LIMITATIONS OF THE SOFT SETS AND FUZZY SETS BASED PERCEPTRON

Although Eq. 8 can already be used to calculate membership function and apply it to classifications. But based on existing research of neural network, we already know that the structure of the perceptron is limited. That is:

1) Marvin and Seymour [38] in 1969 illustrated the shortcomings of Rosenblatt's [39] single-layer perceptron. The multiple-input neuron is not enough to solve complex classification problems, and a single-layer perceptron can only learning linearly separable patterns.
2) For $\mu(x) = (F, A)W$, it is actually a single layer perceptron, and this model also can perform pattern classification only on linearly separable patterns. This means, $\mu(x)$ is just a line in two-dimensional space or a hyperplane in a multi-dimensional space. Sometimes there are complex problems, and the data for these problems cannot be divided into separate classes by using a single straight line.

The multi-layer perceptron and the back-propagation algorithm overcomes many of the shortcomings of the single-layer perceptron. In practice, many complex problems need to have multiple decision lines for a well acceptable solution. Multiple-layer perceptrons realize this task by the introduction of one or more hidden layers. So, we introduce the soft sets and fuzzy sets based multi-layer perceptron in subsection (IV-A).

IV. THE SF-ANN MODEL

In this section, the soft sets and fuzzy sets based neural network (SF-ANN) is proposed. As shown in Fig. (6). As can be seen from the discussion above, the soft sets and fuzzy sets interconnected each other through W. And the key is W, a properly configured W can generate membership function of $\mu(x)$, which can be used to solve complex problems, such as decision-making, pattern recognition, and so on.

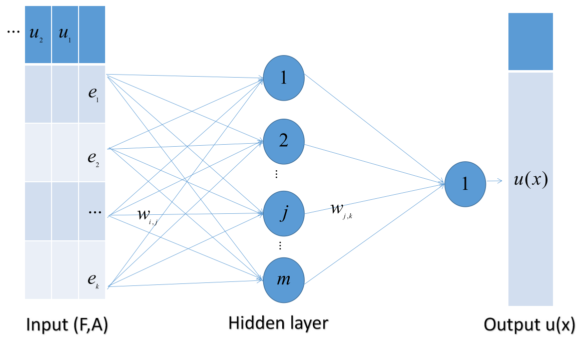


FIGURE 6. The soft sets and fuzzy sets based neural network.

The discussion mentioned in the previous chapters reveals that the data behind the soft set is the abstract features of samples. That is, the soft set (F, A) can be used as a training data set, this 'training set' can be 'trained' by a certain method and abstracted into an abstract description function of an object, that is, the fuzzy set $\mu(x)$. As shown in Fig.(6).

By using this model, we can achieve the following effects:

1) For any soft set (F, A), take it as a neural network input, we can generate the membership function of the fuzzy set $\mu(x)$.
2) As an abstract description of an object, $\mu(x)$ can be used directly in classification, pattern recognition, decision-making and so on.
3) The neural network is used as a black box for solving W. It's flexible and substitutable, we can easily replace it without affecting existing applications.

All we need to do is focus on the soft sets and solve practical problems conveniently. That is, in the practical application, we only need to prepare the front-end 'S' (soft sets), and using the back-end 'F' (fuzzy sets).

A. THE PRINCIPLE AND IMPLEMENTATION OF SF-ANN

A multilayer perceptron is a class of feedforward neural networks [40]. The network consists of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer of computational neurons. The input signals are propagated in a forward direction on a layer-by-layer basis.

Let $U = \{u_1, u_2, \cdots, u_n\}$ and (F, A) be a soft set with k attributes $E = \{e_1, e_2, \cdots, e_k\}$, $\mu(x)$ be a fuzzy set. Similar to the multilayer perceptron, the SF-ANN always takes soft sets as input and fuzzy sets as output. As shown in the Figure (6).

For convenience, let (F, A) be an $n \times k$ matrix,and $\mu(x)$ be a $n \times 1$ matrix. The W_1 and W_2 are also represented as matrices based on the number of neurons nodes. Here we only use the two-layer neural network, which can be easily extended to multi-layer neural networks.

The SF-ANN is formulated as follows:

$$\mu(x) = f(f((F,A)\cdot W_1)\cdot W_2) \tag{9}$$

where,

$$u(x) = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

$$(F,A) = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1k} \\ u_{21} & u_{22} & \dots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \dots & u_{nk} \end{pmatrix}$$

$$W_1 = \begin{pmatrix} w_{1,1} & w_{2,1} & \dots & w_{m,1} \\ w_{1,2} & w_{2,2} & \dots & w_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,k} & w_{2,k} & \dots & w_{m,k} \end{pmatrix}$$

$$W_2 = \begin{pmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,m} \end{pmatrix}$$

m is the number of neurons.

First, it computes the net weighted input as before:

$$X = (F,A)W - \Theta = \sum_{i=1}^{k} e_i\omega_i - \Theta \tag{10}$$

where k is the number of attributes, and Θ is the threshold applied to the neuron.

This input value is passed through the activation function, and the neurons use a sigmoid activation function:

$$\mu(x) = Y^{sigmoid} = \frac{1}{1+e^{-X}} \tag{11}$$

A significant advantage of the sigmoid function is the derivative is easy to compute, and it also guarantees that the $\mu(x)$ output is bounded between 0 and 1.

To derive the back-propagation learning law, we refer to the idea of the three-layer network shown in Figure (6). The indices i, j and k refer to neurons in the input, hidden and output layers. The symbol w_{ij} indicates the weight for the connection between neuron i and neuron j, and the symbol w_{jk} the weight between neuron j and neuron k. To propagate error signals, we start at the output layer and work backward to the hidden layer. The error signal at the output of neuron k at iteration p is defined by

$$err_k(p) = y_{d,k}(p) - y_k(p) \tag{12}$$

where $err_k(p)$ is the error signals, and $y_{d,k}(p)$ is the desired output of neuron k at iteration p, $y_k(p)$ is the output of neuron k at iteration p.

We use a straightforward procedure to update weight w_{jk}.

$$w_{jk}(p+1) = w_{jk}(p) + \Delta W_{jk}(p) \tag{13}$$

where $\Delta W_{jk}(p)$ is the weight correction.

The weight correction in the multilayer network is computed by (Fu, 1994) [41]:

$$\Delta W_{jk}(p) = \alpha \times y_i(p) \times \delta_k(p) \tag{14}$$

where $\delta_k(p)$ is the error gradient at neuron k in the output layer at iteration p.

The error gradient is determined as the derivative of the activation function multiplied by the error at the neuron output. For neuron k in the output layer, we have

$$\begin{aligned}\delta_k(p) &= \frac{\partial y_k(p)}{\partial X_k(p)} \times err_k(p) \\ &= \frac{\partial\{\frac{1}{1+exp[-X_k(p)]}\}}{\partial X_k(p)} \times err_k(p) \\ &= \frac{exp[-X_k(p)]}{1+exp[-X_k(p)]} \times err_k(p)\end{aligned} \tag{15}$$

where $y_k(p)$ is the output of neuron k at iteration p, and $X_k(p)$ is the net weighted input to neuron k at the same iteration. Thus,

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times err_k(p) \tag{16}$$

where

$$y_k(p) = \frac{1}{1+exp[-X_k(p)]}$$

To calculate the weight correction for the hidden layer, we can apply the same equation as for the output layer:

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \tag{17}$$

where $\delta_j(p)$ represents the error gradient at neuron j in the hidden layer:

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) w_{jk}(p) \tag{18}$$

where l is the number of neurons in the output layer;

$$y_j(p) = \frac{1}{1+e^{-X_j(p)}} \tag{19}$$

where

$$X_j(p) = \sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \Theta_j$$

and n is the number of neurons in the input layer.

Now we can derive the SF-ANN training algorithm:

1) Initialisation
Input the soft set (F,A) and the fuzzy set $\mu(x)$ for training. Set all the weights and threshold levels of the network to random numbers. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer [42].

2) Activation
   a) Calculate the outputs in the hidden layer: $y_j(p) = f(\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \Theta_j)$, where n is the number of inputs of neuron j in the hidden layer, and f is the sigmoid activation function.
   b) Calculate the outputs in the output layer: $y_k(p) = f(\sum_{i=1}^{n} x_{ij}(p) \times w_{jk}(p) - \Theta_k)$, where m is the number of inputs of neuron k in the output layer.
3) Weight training
   a) Calculate the error gradient in the output layer: $\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times err_k(p)$, where $err_k(p) = y_{d,k}(p) - y_k(p)$
   Update the weights at the output layer: $w_{jk}(p+1) = w_{jk}(p) + \Delta_{jk}(p)$ where, $\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$,
   b) Calculate the error gradient in the hidden layer: $\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) \times w_{wj}(p)$
   Update the weights at the hidden neurons: $w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$
   where, $\Delta_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$
4) Iteration
   Go back to Step 2 and repeat the process until the selected error criterion is satisfied.

Until all training instances have been trained, the SF-ANN is ready for classification. In the classification phase, the network only executes feed-forward to get the final classified result.

For n-layer artificial neural network, input (F, A) and $\mu(x)$, $u(x) = f(f(f((F, A) \cdot W_1) \cdot W_2) \cdots W_n + \Theta), f$ is the sigmod function, until all training instances have been trained, the $W_1, W_2, \cdots, W_n, \Theta$ is generated. Now $\mu(x)$ can be used for decision-making, classification, pattern recognition and so on. Let T be a sample to be tested, then $\mu(T) = f(f(f(T \cdot W_1) \cdot W_2) \cdots W_n + \Theta)$ used to determine if this sample T belongs to 'this class'. The algorithm can be expressed as Algorithm (1), and the source code can be found at [32].

Advantages of the SF-ANN:

1) This model $\mu(x) = (F, A)W$ has the same structure as the single layer neural network, which can be easily extended to two layers or multilayer neural networks.
2) With the help of the model, we can focus on the soft set and the fuzzy set when solving the practical problems, and reduce the difficulty and complexity of constructing the neural network. Correspondingly, our focus is on the data itself.
3) A single (F, A) can be considered a 'class', a combination of multiple classes can make up more complex things. Let (F, A),(G, B) be two soft sets, through SF-ANN, we can generate $\mu_{(F,A)}(x)$ and $\mu_{(G,B)}(x)$. According to the research of soft set, we can also generate $\mu_{(F,A)\cup(G,B)}(x)$ or $\mu_{(F,A)\cap(G,B)}(x)$. On the other hand, the attribute reduction [43] can be performed on (F, A), and then we can greatly reduce the neuron input during training.

Algorithm 1 The SF-ANN Training and Testing

```
1  function SF-ANN-Training < $(F, A)$, $\mu(x)$ >;
   Input  : The matrix expressed $(F, A)$ and the
            corresponding $\mu(x)$, as show in Eq. 10
   Output: $W_1$ and $W_2$, as show in Eq. 10
2  Initialize $W_1$ and $W_2$ based on number of neurons;
3  while Error > Tolerance do
       // dot:Multiplying a Matrix;f:The
          Sigmod Function
4      Calculate hidden layer values:
       $Z_1 = f(dot((F, A), W_1))$
5      Calculate output layer values: $Z_2 = f(dot(Z_1, W_2))$
6      Observe Error:
       $\begin{cases} Err_{Z_2} = \mu(x) - Z_2;\ \Delta_{Z_2} = Err_{Z_2} \times Z_2 \times (1 - Z_2) \\ Err_{Z_1} = dot(\Delta_{Z_2}, W_2);\ \Delta_{Z_1} = Err_{Z_1} \times Z_1 \times (1 - Z_1) \end{cases}$
7      Update hidden layer weights: $W_1 += dot(Z_1^T, \Delta_{Z_1})$
8      Update output layer weights: $W_2 += dot(Z_2^T, \Delta_{Z_2})$
9  end
10 function SF-ANN-Testing $(F, A)$;
   Input  : A test sample $T$
   Output: The value of $\mu(T)$
11 Initialize the threshold $\lambda$ ;
12 Calculate output value:
   $\mu(T) = f(dot(f(dot(T, W_1), W_2))$
13 if $1 - \mu(T) > \lambda$ then
14     return True;
15 else
16     return False;
17 end
```

V. NUMERICAL EXPERIMENTS

In this application, we will be testing SF-ANN with Zoo Dataset [44]. The Zoo Database from the UCI Machine Learning Database Repository, which contains 101 tuples on 15 Boolean attributes and two numerical attributes, such as hair (Boolean), feathers (Boolean) and the number of legs (numeric). All tuples are classified into seven categories (mammals, birds, reptiles, fish, amphibians, insects, and invertebrates). The task of the classifier is to produce the classification number given the animal name or id. For the 7 categories, we will create $\mu_{mammals}(x)$, $\mu_{birds}(x)$, $\mu_{reptiles}(x)$, $\mu_{fish}(x)$, $\mu_{amphibians}(x)$, $\mu_{insects}(x)$, $\mu_{invertebrates}(x)$ by using SF-ANN, respectively.

The experimental of 'mammals' described below, the other experiments are similar. We wrote programs for all experiments and uploaded it to [32] for easy validation and improvement.

In order to test the SF-ANN model, there are four steps to be made:

1) Normalizing the data
   First, the dataset must be normalized. In this case, the thirteenth column must be normalized. After we

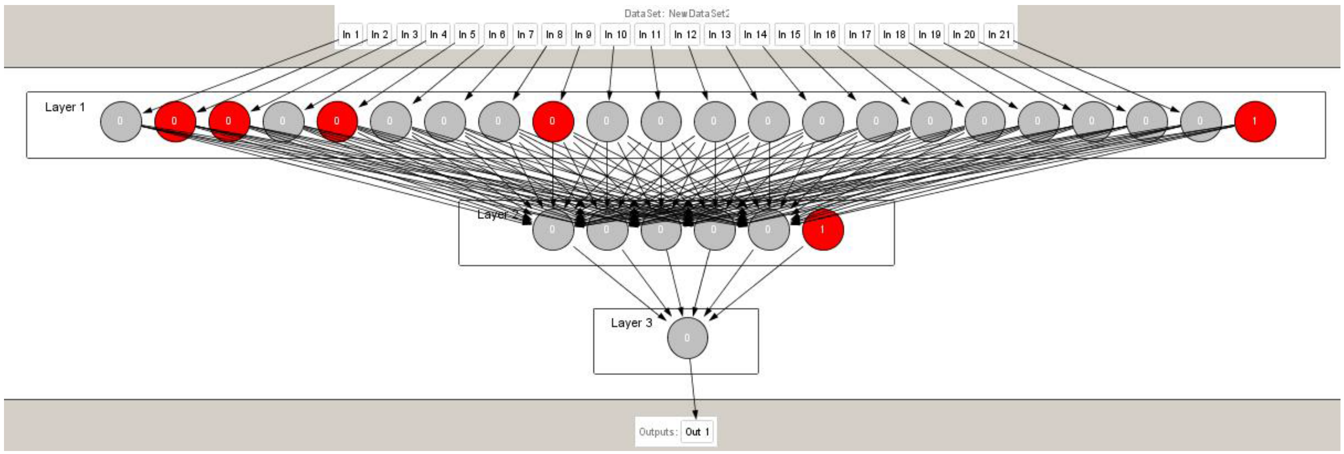


FIGURE 7. The SF-ANN for training mammals with NeurophStudio.

TABLE 2. Zoo dataset.

U	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_{17}	e_{18}	e_{19}	e_{20}	e_{21}	type
aardvark	1	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	1	1
frog	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	5
frog	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	5
tortoise	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1	3
pitviper	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	3
antelope	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	1	1
bass	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	0	0	1	1	0	0	4
...																						

TABLE 3. The training set $(F, A)_{mammals}$ and $\mu_{mammals}(x)$.

U	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_{17}	e_{18}	e_{19}	e_{20}	e_{21}	$\mu(x)$
aardvark	1	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	1	1
frog	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
frog	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0
tortoise	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0
pitviper	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0
antelope	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	1	1
bass	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0
...																						

finished that, we got 22 columns. Exactly we got 21 inputs ((F, A)) and 1 output ($\mu(x)$). As shown in Table (2).

2) Create the training dataset
Generate soft Set (F, A) and $\mu(x)$ as training samples based on Zoo Dataset. We sampled 30 samples from the dataset as training samples and established the first type of animal training sample $(F, A)_{mammals}$. $(F, A)_{mammals}$ contains 21 attributes ($e_i, 0 < i < 22$) and 30 objects ($u_i, 0 < i < 31$). We extract the last column from the dataset to establish $\mu_{mammals}(x)$, if u_i is the first type of animal mammals, then $\mu_{mammals}(u_i) = 1$, otherwise $\mu_{mammals}(u_i) = 0$. As shown in Table (3).

3) Training the SF-ANN
The SF-ANN automatically sets the number of nodes of the network according to the number of samples and attributes of the input soft set. The number of inputs is 21, and hidden neurons number is 6, the number of outputs is 1. As shown in Figure (7).

4) Testing the neural network
After the network is trained, we use the remaining 101 datasets to test the SF-ANN, and the test results are shown in Table (4) (See Appendix for details).

The SF-ANN is sensitive to the problem of class imbalance in training samples. Balanced categories tend to get the best performance, while unbalanced categories tend to reduce the effect of the model. As shown in Table (4), the experimental results of 'REPTILES' and 'INSECTS' appear to be relatively poor because they have fewer samples in the datasets (The number of 'REPTILES' and 'INSECTS' is 5 and 4, respectively).

Remark 1: The SF-ANN is essentially a two-class classifier (in the training samples,the value of $\mu(x)$ is either 0 or 1), which has its own advantages:

(1) The scale of training is small, and we have narrowed down the problem at the data level.

(2) The number of hidden layers decreases, and the computational complexity is reduced.

TABLE 4. The accurate of testing SF-ANN.

CLASS	PRECISION	RECALL	F1-SCORE
MAMMALS	1.00	1.00	1.00
BIRDS	1.00	0.95	0.974
REPTILES	0.75	0.60	0.667
FISH	0.929	1.00	0.963
AMPHIBIANS	1.00	0.75	0.857
INSECTS	1.00	0.625	0.769
INVERTEBRATES	1.00	0.70	0.824

(3) In practice, samples are more comfortable to collect and fewer samples are needed.

(4) According to the meaning of $\mu(x)$, we can set the threshold value to improve the recognition rate.

We can solve the complex classification problem with fuzzy sets and soft sets and avoid the construction of complex neural networks. That is, in the practical application, we only need to focus on the front-end 'S' (soft sets), and take advantage of the back-end 'F' (fuzzy sets).

VI. CONCLUSION

This paper focuses on the relationship between fuzzy sets and soft sets. Inspired by the idea of transformation between fuzzy sets and soft sets, we construct a neuron-like structure and extend it to multilayer neural network structures. Furthermore, a neural network learning model—SF-ANN is constructed. By using soft set theory and machine learning method, this paper gives the general construction method of membership function of fuzzy set. An illustrative example shows that the model can be used to establish the membership function of fuzzy sets. In general, it provides a new perspective to observe the relationship between soft sets and fuzzy sets and a practical model for solving practical problems.

Further, the SF-ANN model can be studied in three aspects. Firstly, at the front-end, the soft sets can be used for information fusion, attribute reduction, and so on. The information fusion based on the soft sets will play a significant role in promoting the model proposed in this paper. Secondly, at the back-end, we can study and replace the sigmoid functions from the perspective of membership function construction. This is also indirectly beneficial to the neural network and improves its learning ability. Finally, by using the SF-ANN model, the researchers of fuzzy sets or soft sets can focus on their field and avoid constructing complex neural networks. As a tool, the neural network is flexible and substitutable, and the support vector machine may be considered as an alternative.

APPENDIX
THE EXPERIMENT OF 'MAMMALS'

```
1. Train the SF—ANN
Softset ==
[1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1]
[0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
[1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0]
[0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
[0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
Fuzzyset ==
[1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 0, 0, 1, 1]
W1==
[[−0.46374472   1.1011822   −1.14662219   0.51217834
−1.30600343  −1.74795964  −1.17362203  −0.2059837  ]
[−0.16589491  −0.43320794  −0.03762993  −0.12038066
−0.39589952   1.07887511  −0.86764188   0.22990682]
[  0.07800391  −1.3161552   −0.43658647  −2.19229382
1.31737883   2.61284989   0.43359323   0.09325996]
[  0.34655106   1.93867029  −0.99199591   0.24742641
−1.3467796   −0.6121514   −1.73276001  −0.04332664]
[  0.87364505  −0.25208039   0.47404662  −0.58256035
0.36918331   0.69278321  −0.90817993   0.4189295  ]
[  1.0665062    0.17159622  −0.34485928  −0.15843924
−0.39634673   0.59698584   1.07856339  −0.5559415  ]
[−0.42061591  −0.83890556  −0.90120004  −0.14995517
−0.31857957   0.01149887   0.06829555  −1.01788612]
[−0.08005091  −0.2172667    0.08681253   0.75936532
−1.0623397   −0.53608754   0.04249259  −0.16256934]
[−1.08780906   0.05022755   0.35977179  −0.10119284
0.81737693   0.1315508    0.53807904  −0.82700666]
[−1.0427852    0.87793024  −0.15395053  −0.3047882
0.51822182  −0.98393879  −0.18586671   0.39838517]
[  0.78082391   0.15215159   0.51886844  −0.39030799
−0.41656958   0.84596421  −0.11054856   0.91268806]
[  0.36659524   0.29196333  −0.74621382   0.55261134
0.12674328   0.56566582  −0.05852341  −0.59336137]
[  0.84745287   0.16938979  −0.96280508  −0.16654548
−0.10774737   0.50206771   0.91212072  −0.37159248]
[  0.77493629  −0.07169073  −0.87806512   0.64528285
0.37797528   1.01817757  −0.60007547  −0.80708763]
[  0.68361681   0.5962679   −0.94416315   0.8773105
0.23214675   0.40752333  −0.00485294  −0.71644802]
[−0.96023973  −0.94757803  −0.94338702  −0.50757786
0.7200559    0.07766213   0.10564396   0.68406178]
[−0.73147201  −0.63196196   0.24649072   0.76805407
0.19197104  −0.68777272   0.71097897  −0.57849753]
[  0.61421039  −0.22427871   0.72708371   0.49424329
0.11248047  −0.72708955  −0.88016462  −0.75731309]
[−1.02274043  −0.75407113  −0.51144923   0.23332069
0.06181266  −0.85669132  −0.92989794   0.83452803]
[  0.07966276  −0.61981983  −0.48610253   0.4644802
−0.62872566   0.05466       0.77654787   0.66799597]
[−0.66288097   0.29976419   0.22114261   0.83142582
−0.79337536  −1.12556746  −1.20793854  −0.01633509]]
W2==
[[−0.64034914]
[  3.62908286]
[−0.72785483]
[  2.54136244]
[−2.04055295]
[−4.06408276]
[−2.03575245]
[  0.10484768]]

2.Test the SF—ANN on the test data
aardvark ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,1   mux=0.99667  DesireOutput:1
Output:1   OK
antelope ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99703  DesireOutput:1
Output:1   OK
bear ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,1   mux=0.99667  DesireOutput:1
Output:1   OK
boar ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99680  DesireOutput:1
Output:1   OK
buffalo ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99703  DesireOutput:1
Output:1   OK
calf ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1   mux=0.99713  DesireOutput:1
Output:1   OK
cavy ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,1   mux=0.99455  DesireOutput:1
Output:1   OK
cheetah ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99680  DesireOutput:1
Output:1   OK
deer ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99703  DesireOutput:1
Output:1   OK
dolphin ,0 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.98725  DesireOutput:1
Output:1   OK
elephant ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99703  DesireOutput:1
Output:1   OK
fruitbat ,1 ,0 ,0 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,1   mux=0.99149  DesireOutput:1
Output:1   OK
giraffe ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1   mux=0.99703  DesireOutput:1
Output:1   OK
girl ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,1 ,1   mux=0.99602  DesireOutput:1
Output:1   OK
goat ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1   mux=0.99713  DesireOutput:1
Output:1   OK
gorilla ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,1   mux=0.99630  DesireOutput:1
Output:1   OK
hamster ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,0 ,1   mux=0.99593  DesireOutput:1
Output:1   OK
hare ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,1   mux=0.99569  DesireOutput:1
Output:1   OK
```

```
leopard ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
lion ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
lynx ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
mink ,1 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99662  DesireOutput:1
Output:1   OK
mole ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,1     mux=0.99527  DesireOutput:1
Output:1   OK
mongoose ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
opossum ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,1     mux=0.99527  DesireOutput:1
Output:1   OK
oryx ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99703  DesireOutput:1
Output:1   OK
platypus ,1 ,0 ,1 ,1 ,0 ,1 ,1 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.95632  DesireOutput:1
Output:1   OK
polecat ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
pony ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1     mux=0.99713  DesireOutput:1
Output:1   OK
porpoise ,0 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.98725  DesireOutput:1
Output:1   OK
puma ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
pussycat ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1     mux=0.99662  DesireOutput:1
Output:1   OK
raccoon ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK
reindeer ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1     mux=0.99713  DesireOutput:1
Output:1   OK
seal ,1 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,1     mux=0.99410  DesireOutput:1
Output:1   OK
sealion ,1 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99624  DesireOutput:1
Output:1   OK
squirrel ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,1     mux=0.99472  DesireOutput:1
Output:1   OK
vampire ,1 ,0 ,0 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,1     mux=0.99149  DesireOutput:1
Output:1   OK
vole ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,1     mux=0.99569  DesireOutput:1
Output:1   OK
wallaby ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99664  DesireOutput:1
Output:1   OK
wolf ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,1 ,1     mux=0.99680  DesireOutput:1
Output:1   OK

3.The  results
Total  number:  41
Find   numper:  41
Error  numper:  0
Corre  number:  41
PRECISION:1.000   RECALL:1.000   F1−SCORE:1.000
```

REFERENCES

[1] C. V. Negoita, "Fuzzy sets," *Fuzzy Sets Syst.*, vol. 8, no. 3, pp. 338–353, 1965.

[2] D. Molodtsov, "Soft set theory-first results," *Comput. Math. With Appl.*, vol. 37, nos. 4–5, pp. 19–31, Feb. 1999.

[3] J. C. R. Alcantud and G. Santos-García, "A new criterion for soft set based decision making problems under incomplete information," *Int. J. Comput. Intell. Syst.*, vol. 10, no. 1, pp. 394–404, 2017.

[4] A. O. Atagün, H. Kamacı, and O. Oktay, "Reduced soft matrices and generalized products with applications in decision making," *Neural Comput. Appl.*, vol. 29, no. 9, pp. 445–456, Aug. 2016.

[5] R. B. F. Hakim, E. N. Sari, and T. Herawan, "On if-then multi soft sets-based decision making," in *Information and Communication Technology*, Linawati, M. S. Mahendra, E. J. Neuhold, A. M. Tjoa, and I. You, Eds. Berlin, Germany: Springer, 2014, pp. 306–315.

[6] R. B. Fajriya Hakim, E. N. Sari, and T. Herawan, "Soft solution of soft set theory for recommendation in decision making," in *Recent Advances on Soft Computing and Data Mining*. T. Herawan, R. Ghazali, and M. M. Deris, Eds. Cham, Switzerland: Springer, 2014, pp. 313–324.

[7] F. Feng, Y. B. Jun, X. Liu, and L. Li, "An adjustable approach to fuzzy soft set based decision making," *J. Comput. Appl. Math.*, vol. 234, no. 1, pp. 10–20, May 2010.

[8] X. Ma, "Applications of rough soft sets in BCI-algebras and decision making," *J. Intell. Fuzzy Syst.*, vol. 29, no. 3, pp. 1079–1085, Oct. 2015.

[9] X. Ma, Q. Liu, and J. Zhan, "A survey of decision making methods based on certain hybrid soft set models," *Artif. Intell. Rev.*, vol. 47, no. 4, pp. 507–530, Jun. 2016.

[10] X. Peng and Y. Yang, "Interval-valued hesitant fuzzy soft sets and their application in decision making," *Fundamenta Informaticae*, vol. 141, no. 1, pp. 71–93, Sep. 2015.

[11] F. Feng, J. Cho, W. Pedrycz, H. Fujita, and T. Herawan, "Soft set based association rule mining," *Knowl.-Based Syst.*, vol. 111, pp. 268–282, Nov. 2016.

[12] T. Herawan and M. M. Deris, "A soft set approach for association rules mining," *Knowl.-Based Syst.*, vol. 24, no. 1, pp. 186–195, Feb. 2011.

[13] W. Xu, Z. Xiao, X. Dang, D. Yang, and X. Yang, "Financial ratio selection for business failure prediction using soft set theory," *Knowl.-Based Syst.*, vol. 63, pp. 59–67, Jun. 2014.

[14] F. Fatimah, D. Rosadi, R. B. F. Hakim, and J. C. R. Alcantud, "A social choice approach to graded soft sets," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2017, pp. 1–6.

[15] Z. Xiao, K. Gong, and Y. Zou, "A combined forecasting approach based on fuzzy soft sets," *J. Comput. Appl. Math.*, vol. 228, no. 1, pp. 326–333, Jun. 2009.

[16] J. C. R. Alcantud, G. Santos-García, and E. Hernández-Galilea, "Glaucoma diagnosis: A soft set based decision making procedure," in *Advances in Artificial Intelligence*, J. Puerta, J. A. Gámez, B. Dorronsoro, E. Barrenechea, A. Troncoso, B. Baruque, and M. Galar, Eds. Cham, Switzerland: Springer, 2015, pp. 49–60.

[17] H. Aktaş and N. Çağman, "Soft sets and soft groups," *Inf. Sci.*, vol. 177, no. 13, pp. 2726–2735, Jul. 2007.

[18] Y. Yao, "A comparative study of fuzzy sets and rough sets," *Inf. Sci.*, vol. 109, nos. 1–4, pp. 227–242, Aug. 1998.

[19] J. C. R. Alcantud, "Some formal relationships among soft sets, fuzzy sets, and their extensions," *Int. J. Approx. Reasoning*, vol. 68, pp. 45–53, Jan. 2016.

[20] H. Bustince, E. Barrenechea, M. Pagola, J. Fernandez, Z. Xu, B. Bedregal, J. Montero, H. Hagras, F. Herrera, and B. De Baets, "A historical account of types of fuzzy sets and their relationships," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 1, pp. 179–194, Feb. 2016.

[21] V. Torra, "Hesitant fuzzy sets," *Int. J. Intell. Syst.*, vol. 25, no. 6, pp. 529–539, 2010.

[22] J. C. R. Alcantud and V. Torra, "Decomposition theorems and extension principles for hesitant fuzzy sets," *Inf. Fusion*, vol. 41, pp. 48–56, May 2018.

[23] W. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biol.*, vol. 52, nos. 1–2, pp. 99–115, 1990.

[24] N. Sengupta, M. Sahidullah, and G. Saha, "Lung sound classification using cepstral-based statistical features," *Comput. Biol. Med.*, vol. 75, pp. 118–129, Aug. 2016.

[25] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks–a review," *Pattern Recognit.*, vol. 35, no. 10, pp. 2279–2301, 2002.

[26] D. N. Ganesan, D. K. Venkatesh, D. M. A. Rama, and A. M. Palani, "Application of neural networks in diagnosing cancer disease using demographic data," *Int. J. Comput. Appl.*, vol. 1, no. 26, pp. 81–97, Feb. 2010.

[27] L. Bottaci, P. J. Drew, J. E. Hartley, M. B. Hadfield, R. Farouk, P. W. Lee, I. M. Macintyre, G. S. Duthie, and J. R. Monson, "Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions," *Lancet*, vol. 350, no. 9076, pp. 469–472, Aug. 1997.

[28] E. Alizadeh, S. M. Lyons, J. M. Castle, and A. Prasad, "Measuring systematic changes in invasive cancer cell shape using zernike moments," *Integrative Biol.*, vol. 8, no. 11, pp. 1183–1193, 2016.

[29] S. M. Lyons, E. Alizadeh, J. Mannheimer, K. Schuamberg, J. Castle, B. Schroder, P. Turk, D. Thamm, and A. Prasad, "Changes in cell shape are correlated with metastatic potential in murine and human osteosarcomas," *Biol. Open*, vol. 5, no. 3, pp. 289–299, Feb. 2016.

[30] G. Santos-García and E. H. Galilea, *Using Artificial Neural Networks to Identify Glaucoma Stages*. in *The Mystery of Glaucoma*. Rijeka: IntechOpen, 2011, pp. 331–352.

[31] D. Zissis, E. K. Xidias, and D. Lekkas, "A cloud based architecture capable of perceiving and predicting multiple vessel behaviour," *Appl. Soft Comput.*, vol. 35, pp. 652–661, Oct. 2015.

[32] Z. Liu, *Sf-ANN*. Accessed: 2018. [Online]. Available: https://github.com/idle010/the-implementation-of-SF-ANN

[33] Z. Liu, K. Qin, Z. Pei, and J. Liu, "On induced soft sets and topology for the parameter set of a soft set," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervas. Intell. Comput.*, Oct. 2015, pp. 1349–1353.

[34] Z. Liu, J. C. R. Alcantud, K. Qin, and Z. Pei, "The relationship between soft sets and fuzzy sets and its application," *J. Intell. Fuzzy Syst.*, vol. 36, no. 4, pp. 3751–3764, Apr. 2019.

[35] G. Klir, "Multivalued logics versus modal logics: Alternative frameworks for uncertainty modelling," in *Proc. Adv. Fuzzy Theory Technol.*, 1994, pp. 3–47.

[36] A. Kolesárová, "Limit properties of quasi-arithmetic means," *Fuzzy Sets Syst.*, vol. 124, no. 1, pp. 65–71, Nov. 2001.

[37] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, *Neural Network Design*. Martin Hagan, 2014. [Online]. Available: https://books.google.nl/books?id=4EW9oQEACAAJ

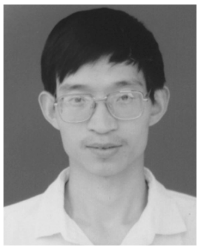
[38] M. Marvin and A. P. Seymour, *Perceptrons*. Cambridge, MA, USA: MIT Press 1969.

[39] F. Rosenblatt, *The perceptron, a perceiving recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

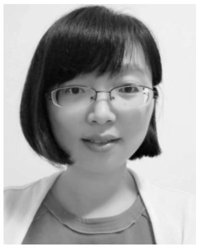
[40] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*. London, U.K.: Pearson, 2005.
[41] L. Fu, *Neural Networks in Computer Intelligence*. New York, NY, USA: McGraw-Hill, 1994.
[42] J. Heaton, *Introduction to Neural Networks With Java*. Heaton Research, 2008. [Online]. Available: https://books.google.com/books?id=Swlcw7M4uD8C
[43] J. Zhan and J. C. R. Alcantud, "A survey of parameter reduction of soft sets and corresponding algorithms," *Artif. Intell. Rev.*, vol. 52, no. 3, pp. 1839–1872, Nov. 2017.
[44] D. Dua, and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California Irvine, Irvine, CA, USA, Tech. Rep., 2017.

ZHICAI LIU received the Ph.D. degree from the School of Information Science and Technology, Southwest Jiaotong University (SWJTU), Chengdu, China. He is currently an Associate Professor Fellow with the School of Computer and Software Engineering, Xihua University. His current researches include decision-making, cryptographic protocol, and intrusion detection.

JOSÉ CARLOS R. ALCANTUD received the M.Sc. degree in mathematics from the University of Valencia, Spain, in 1991, and the Ph.D. degree in mathematics from the University of Santiago de Compostela, Spain, in 1996. He is currently a Professor with the Department of Economics and Economic History, University of Salamanca, Spain, where he also acts as the Head of the Department. His current research interests include fuzzy preference modeling, decision-making, and social choice.



KEYUN QIN received the M.Sc. and Ph.D. degrees in applied mathematics from Southwest Jiaotong University, China, in 1994 and 1997, respectively. He is currently a Full Professor with the College of Mathematics, Southwest Jiaotong University. He has published nearly 80 research articles in academic journals or conferences. His current research interests include rough set theory, soft set theory, fuzzy logic-based systems, and formal concept analysis.



LING XIONG received the M.S. and Ph.D. degrees from the School of Information Science and Technology, Southwest Jiaotong University (SWJTU), Chengdu, China. She is currently an Associate Professor Fellow with the School of Computer and Software Engineering, Xihua University. She is also a Postdoctoral Researcher with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu. Her research interest includes the security and privacy in Internet of Things and blockchain.