Intelligent Neuro-Fuzzy Computing with Asymmetric Neuro-Fuzzy System and Sphere Complex Fuzzy Sets

Student: Chia-Hao Tu Advisor: Dr. Chunshien Li

Department of Information Management, National Central University, Taiwan

Abstract. Artificial intelligence (AI) and machine learning have become essential items when companies invest in IT technologies. In which, neuro-fuzzy system (NFS) is designed to exploit the learning abilities of an artificial neural network (ANN) and the explicit knowledge of a fuzzy inference system (FIS). However, the two problems of symmetric structure and the multiple-input single-output architecture in traditional NFSs affect system efficiency.

An asymmetric NFS (ANFS) has been proposed in this dissertation to address the problems mentioned above with two mechanisms. Firstly, an asymmetric layer is added to the ANFS model, making the model has different neuron numbers in the premise and consequent layers. Secondly, the introduced sphere complex fuzzy sets (SCFSs) replace the traditional fuzzy sets, making the model output numbers adjustable for different applications. For resolving the cross-target feature selection problem existing in the multitarget prediction model, we proposed a feature selection algorithm based on influence information. Besides, a hybrid learning algorithm combining a state-based whale optimization algorithm with the recursive least-square estimator has been proposed to optimize the model.

Three experiments are designed to evaluate the proposed approach's performance, including the dual function approximation, two exchange rate, and four stock index predictions. The experimental results indicate that the proposed approach can predict multiple targets simultaneously, having a favorable performance better than conventional NFS and other methods in the literature.

Keywords: Neuro-fuzzy system (NFS), Asymmetric neuro-fuzzy system (ANFS), Aim-object-based ANFS (AANFS), Fast aim-object-based ANFS (FAANFS), Sphere complex fuzzy set (SCFS), Hybrid learning algorithm, State-based whale optimization algorithm (SWOA).

Chapter I. Introduction

1.1 Research background and motivation

Artificial intelligence (AI) indicates the analysis of data to create models of aspects of the world. These models are then applied to anticipate and predict future cases [1]. AI is not a single technology; it involves the subfields of perception, automated reasoning, cognitive systems, robotics, natural language processing, machine learning, and related fields. Among these fields, machine learning is responsible for recognizing all helpful information hidden in big data and implements improvements automatically based on experience [2], which is considered the most crucial part of AI. In recent years, AI and machine learning have become essential items when companies invest in IT technologies [3]. They can reshape the economic scape by generating new possibilities of value creation and cost reduction for businesses [4]. Therefore, today's businesses urgently require to create a machine learning model with powerful productivity and performance. Artificial neural network (ANN) is broadly discussed for it is the foundation of deep learning. However, ANN designs make complex internal logic difficult to explain clearly, they are often called black boxes. Researchers have criticized this disadvantage for a long time [5, 6]. Compared with ANNs, fuzzy inference systems (FISs), which are based on fuzzy logic and use rules related to linguistic variables and fuzzy membership functions, are easier to explain and more intuitive. If an FIS with two inputs, x_1 and x_2; x_1 has two linguistic values, A_1 and A_2; x_2 has two linguistic values, B_1 and B_2. Each linguistic value can be represented by a membership function. Assume that the FIS has two if-then rules, and each consequent part (or called then part) of the rules is represented as a function of inputs, as shown in Equation (1).

$$\begin{matrix} \text{IF} \quad x_1 = A_1 \quad \text{and} \quad x_2 = B_1 \quad \text{Then} \quad y_1 = p_1 x_1 + q_1 x_2 + r_1 \\ \text{IF} \quad x_1 = A_2 \quad \text{and} \quad x_2 = B_2 \quad \text{Then} \quad y_2 = p_2 x_1 + q_2 x_2 + r_2 \end{matrix}, \tag{1}$$

where y_1 is the output of the first rule; y_2 is the output of the second rule; $\{p_1,p_2,q_1,q_2,r_1,r_2\}$ are the

parameters of Equation (1) and will be optimized using learning algorithms. The premise part (or called if part) outputs of the rules can be obtained as Fig. 1.

IF $x_1 = A_1$ and $x_2 = B_1$

IF $x_1 = A_2$ and $x_2 = B_2$

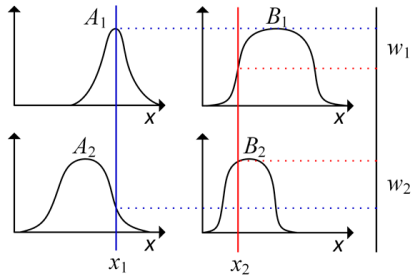

Fig. 1. The calculation of premise part outputs.

Where w_1 is the output of the first premise part; w_2 is the output of the second premise part. Then the system output, y, of the FIS can be obtained in Equation (2).

$$y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2}. \tag{2}$$

Nevertheless, the conventional FIS structure is less flexible and output just one scalar value, restricting its applications and productivity. Jang introduced an adaptive network-based FIS (ANFIS) that combines the properties of an ANN and FIS [7]. An ANFIS equivalent to the FIS mentioned in the previous paragraph can be illustrated as Fig. 2.

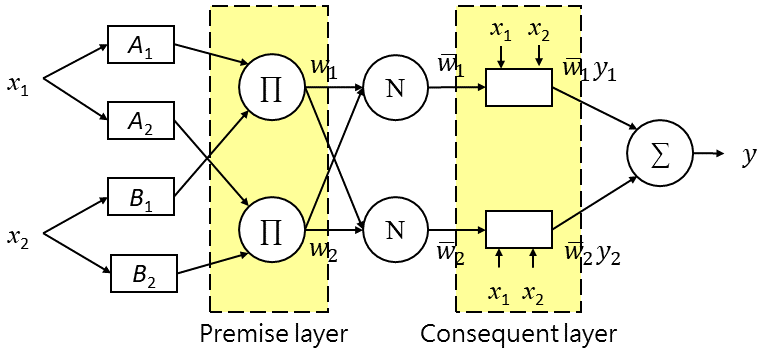

Fig. 2. The ANFIS model [7].

Where a premise layer's neuron is equivalent to a premise part of the FIS; a consequent layer's neuron is equivalent to a consequent part of the FIS; $\bar{w}_1$ and $\bar{w}_2$ are the normalized firing strengths of w_1 and w_2, respectively.

An ANFIS can be used to develop an ANN for performing fuzzy reasoning. Such an ANN is called a neuro-fuzzy system (NFS) [8]. NFSs are designed to exploit the learning abilities of an ANN and the explicit knowledge of an FIS [9,10]. However, traditional NFSs have low system efficiency due to two reasons. First, in NFSs, fuzzy if-then rules are divided into the consequent and premise layers. Therefore, the aforementioned layers must contain the same number of neurons. Such a symmetric structure is a universal characteristic of NFSs. The designs of ANNs and FISs are based on the operations of the human brain, which does not possess a symmetric structure. Unnecessary symmetric structures produce redundant parameters, which influence the accuracy and efficiency of a machine learning model because mapping the relationship between an error and these parameters is difficult [11]. Moreover, traditional NFSs possess a multiple-input, single-output architecture, which limits their application and model productivity.

1.2 Research objectives

This dissertation proposes a novel NFS architecture called an asymmetric NFS (ANFS). In ANFS, we insert an asymmetric layer into the premise and consequent layers to transform the premise layer's output and feed it to the consequent layer as input. With the asymmetric layer, an ANFS can have unequal neuron numbers in the premise and consequent layers, further reducing the model's parameter numbers and the effect of

redundant parameters on mapping proficiency between inputs and outputs. We also present two methods, the aim-object and fast aim-object, to construct the asymmetric layer. The models created through these two methods are called the aim-object-based ANFS (AANFS) and fast aim-object-based ANFS (FAANFS), respectively. In the two models, the clustering results of the input feature variables are applied to decide the neuron numbers in the premise layers, and the clustering results of the target variables are applied to decide the neuron numbers in the consequent layers. Conventional fuzzy sets are substituted by a new fuzzy set concept called sphere complex fuzzy sets (SCFSs) [12] in the fuzzy-set layers of the two models. The difference between the SCFS and conventional fuzzy set is that the SCFS can flexibly modify the output amount of membership values. By harnessing the ability of the SCFS, the two models can change the system output numbers to predict multiple targets simultaneously. The operations of AANFS, FAANFS, and SCFS are based on complex fuzzy sets (CFSs), which are fuzzy sets with complex-valued membership functions proposed by Ramot et al. [13]. Researchers have proven that CFSs have more freedom degrees in relation to adaption flexibility for optimizing the machine learning models than standard fuzzy sets [14-16].

In multitarget prediction, crucial features must be chosen amongst the input features from different targets to build a training data set. An influence information based feature selection algorithm is presented to resolve this matter. The proposed feature selection algorithm belongs to the filter method, which uses influence information as the metric to evaluate the significance of input feature variables and then selects crucial features to develop the training data set. Influence information is derived from entropy and mutual information [17, 18], which have been the accepted metrics in studies that have obtained favorable results [19-21]. The most notable benefit of the proposed algorithm is that its operation is separate from modeling; hence, the extracted features can be utilized in machine learning models developed through various techniques.

In the ANFS, the premise layer parameters are nonlinear, and the consequent layer parameters are linear. Based on the divide-and-conquer strategy, we combine the state-based whale optimization algorithm (SWOA) and recursive least-square estimator (RLSE) to present an innovative hybrid learning algorithm named the SWOA-RLSE to optimize the proposed AANFS and FAANFS models. In 2016, Mirjalili and Lewis [22] introduced the whale optimization algorithm (WOA) to resolve global optimization problems by imitating the humpback whales hunting behavior. The WOA can be implemented quickly and has outstanding performance in various studies, showing its excellent parameter learning ability [23-25]. However, two problems exist in the original version of WOA and haven't been explained explicitly. First, it decides the behavior of the whales on the basis of external variables, which cannot reflect their actual status. Second, the whales tend to use the spiral method rather than the shrinking encircling method or random method without a reasonable explanation. Hence, the SWOA is proposed to solve the aforementioned issues in this dissertation. The RLSE is the recursive version of the least-square estimator algorithm [26]. It can efficiently solve the linear-model optimization problems. In subsequent experiments, the nonlinear parameters in the premise layer are updated by the SWOA, and the linear parameters in the consequent layer are calculated using mathematical formulas by the RLSE.

Three experiments from two-function approximation to two- and four-target predictions of real-world problems will be devised to verify the proposed approaches.

1.3 *Organization of the dissertation* (Ignored)

Chapter II. Related works (Ignored)

Chapter III. Methodology

In this Chapter, the proposed methods are explained in detail (Fig. 3), including: a cross-target feature selection algorithm, the multiple-output neurons in the fuzzy-sets layer, the asymmetric layer, the ANFS framework, and the SWOA–RLSE hybrid learning algorithm.

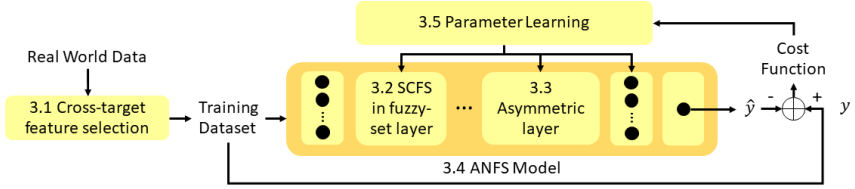

Fig. 3. Schematic of the methodology in this dissertation.

3.1 Cross-target feature selection (Ignored)

3.2 Multiple-output neurons in the fuzzy-sets layer

The SCFS, a new type of fuzzy set, is introduced in this Section [12]. The SCFS was inspired by the mechanism of converting a polar coordinate system into a Cartesian coordinate system. With SCFS, the model can adjust output numbers flexibly.

(Ignored)

In the experiments of this dissertation, the proposed model predicts up to four targets at the same time; hence we need to employ a four-dimensional SCFS here. The included angle of a four-dimensional SCFS between *r* and the *x*–*y* plane is θ_3; the polar coordinate value that *r* maps to the *x*–*y* plane is denoted as r'; the included angle between r' and the *w*–*x* plane is represented as θ_2; the polar coordinate value that r' maps to the *w*–*x* plane is denoted as r''; the included angle between r'' and the *w*-axis is θ_1; the value that *r* maps to the *z*, *y*, *x*, and *w* axes can be calculated using Equations (25) to (28) (a more detailed calculation is found in Appendix A.2):

$$z = r\sin\theta_3, \tag{25}$$
$$y = r'\sin\theta_2 = r\cos\theta_3\sin\theta_2, \tag{26}$$
$$x = r''\sin\theta_1 = r\cos\theta_3\cos\theta_2\sin\theta_1, \tag{27}$$
$$w = r''\cos\theta_1 = r\cos\theta_3\cos\theta_2\cos\theta_1. \tag{28}$$

The SCFS algorithm is presented as followed based on the introduction above:

Step 1. Decide the position in the polar coordinate, r: The model input is substituted into a real-valued Gaussian membership function (GMF):

$$\mathrm{GMF} = e^{\frac{-(f-c)^2}{2s^2}}, \tag{29}$$

where $\{c,s\}$ represent the parameters optimized by learning algorithm; *f* indicates the model input.

Step 2. Obtain θ_1, θ_2 *and* θ_3: In this dissertation, θ_1, θ_2, *and* θ_3 are defined as

$$\theta_1 = \mathrm{GMF}'(f)\xi_1, \tag{30}$$
$$\theta_2 = \mathrm{GMF}''(f)\xi_2, \tag{31}$$
$$\theta_3 = \mathrm{GMF}'''(f)\xi_3, \tag{32}$$

where $\mathrm{GMF}'(.)$ indicates the first derivatives of Equation (29); $\mathrm{GMF}''(.)$ indicates the second derivatives; $\mathrm{GMF}'''(.)$ indicates the third derivatives; $\{\xi_1,\xi_2,\xi_3\}$ are parameters optimized by learning algorithms and hired to increase the variation of these equations.

Step3. Get the output of SCFS: We can calculate the mapping value on each axis through Equation (25) to (32) and get the output of SCFS as follow:

$$\mathrm{SCFS} = \begin{bmatrix} w + jx \\ y + jz \end{bmatrix}, \tag{33}$$

where $j = \sqrt{-1}$.

The most important advantage of the SCFS is that it can flexibly increase output numbers by expanding the mapping dimension. With the help of SCFSs, a model can easily have the multiple-output capability and modify the output numbers in different applications.

3.3 Asymmetric layer

In the origin NFS, the neuron numbers in the premise and consequent layers are equal. We propose a new structure, named the asymmetric layer, inserted into the premise and consequent layers to break the neuron connections between them. Hence the consequent layer can have a chance with different neuron numbers with the premise layer and further reduce the parameter number of the model.

The asymmetric layer has Q neurons. Each neuron uses K normalized firing strength vectors as its inputs, and these vectors have L elements, the same as the system output numbers, and are generated by the premise layer's K neurons. The output of each neuron is a vector with L elements. Based on the description above, we can consider each asymmetric layer neuron as an L-layers transformation function, converting K vector inputs with L complex numbers into one vector output with L complex numbers. This section introduces two methods of establishing the asymmetric layer.

3.3.1 Aim-object method

The aim-object method can be detailed as stepwise herein.

Step 1. Determine the neuron numbers of the asymmetric layer. We collect all target variables into one set, named the target combined set. Because we have no idea about how many clusters should be, we choose subtractive clustering (SC) [73] to cluster the target combined set to obtain the number of clusters, Q. Q also represents the number of neurons in the asymmetric layer. SC is a fast algorithm used to estimate the number of clusters for a given data set if the user is unsure how many clusters of the data set.

Step 2. Create complex-valued target variables. Suppose that a training data set expressed as $TD_{Original}$ has M input feature variables and N target variables, which is represented as follows:

$$TD_{Original} = \{(\mathbf{h} = [f_1\ f_2\ \dots\ f_M], \mathbf{t} = [t_1\ t_2\ \dots\ t_N])\}. \tag{34}$$

The SCFSs, complex-valued fuzzy sets, are employed by the proposed model; hence, we need to transform the real-valued target variables of $TD_{Original}$ to complex-valued variables to fit the model operation, represented as follows:

$$tc_l = t_i + t_{i+1}\sqrt{-1}, \tag{35}$$

where tc_l indicates the l^{th} complex-valued target variable, $l = 1,2,\dots,L$; t_i indicates the i^{th} target variable in $TD_{Original}$. If N is even, then $L = \frac{N}{2}$, $i = 1,3,\dots,N-1$. If N is odd, then $L = \frac{N+1}{2}$, $i = 1,3,\dots,N$, and $t_{N+1} = t_N$.

The combined training data set, expressed as TD, is shown as follows:

$$TD = \{(\mathbf{h} = [f_1\ f_2\ \dots\ f_M], \mathbf{t} = [tc_1\ tc_2\ \dots\ tc_L])\}. \tag{36}$$

Step 3. Determine the original spread values of the transformation functions of Q neurons, denoted as $\mathbf{S}$, as follows:

$$\mathbf{S} = [\vec{S}_l, l = 1,2,\dots,L], \tag{37}$$

$$\vec{S}_l = [S_l^{(q)}, q = 1,2,\dots,Q]^{\mathrm{T}}, \tag{38}$$

$$S_l^{(q)} = S_{t_i} + S_{t_{i+1}} j, \tag{39}$$

$$S_{t_i} = \sqrt{\frac{1}{|TD|}\sum_{\varpi=1}^{|TD|}\left(t_i^{\{\varpi\}} - \mu_{t_i}\right)}, \tag{40}$$

where $S_l^{(q)}$ indicates the original spread value of the l^{th} layer in the q^{th} neuron transformation function. In this dissertation, $S_l^{(1)} = S_l^{(2)} = \cdots = S_l^{(Q)}$; S_{t_i} indicates the spread of the i^{th} real-valued target variable; $j = \sqrt{-1}$; $t_i^{\{\varpi\}}$ indicates the ϖ^{th} target value of the i^{th} real-value target variable; μ_{t_i} indicates the mean of the i^{th} real-valued target variable; and $|TD|$ denotes the number of training data pairs.

Step 4. Calculate the original center values of the transformation functions of Q neurons. Fuzzy c-means clustering is used, and the number of clusters is set to Q. We can obtain the cluster center $\mathbf{C}$ after clustering the complex-valued target variable tc_l as follows:

$$\mathbf{C} = \left[\vec{C}_l, l = 1,2,\dots,L\right], \tag{41}$$

$$\vec{C}_l = [C_l^{(q)}, q = 1,2,\dots,Q]^{\mathrm{T}}, \tag{42}$$

where $C_l^{(q)}$ indicates the original center value of the l^{th} layer in the q^{th} neuron transformation function.

Step 5. Build neuron transformation functions in the asymmetric layer. we can consider each asymmetric layer neuron as an L-layers transformation function, converting K vector inputs with L complex numbers into one vector output with L complex numbers. The transformation function can be expressed as follows:

$$\vec{\lambda}^{(q)} = [\lambda_1^{(q)}, \lambda_2^{(q)}, \dots, \lambda_L^{(q)}]^T, \tag{43}$$

$$\lambda_l^{(q)} = \sum_{k=1}^{K} \gamma_1(\eta_l^{(k)}) \exp\left(\omega_1\left(\eta_l^{(k)}\right) j\right), \tag{44}$$

$$\gamma_1\left(\eta_l^{(k)}\right) = exp\left(-\frac{(\eta_l^{(k)} - CT_l^{(q)})^2}{2(ST_l^{(q)})^2}\right), \tag{45}$$

$$\omega_1\left(\eta_l^{(k)}\right) = \gamma_1{}'\left(\eta_l^{(k)}\right), \tag{46}$$

where $\vec{\lambda}^{(q)}$ indicates the q^{th} neuron output in the asymmetric layer; $\lambda_l^{(q)}$ indicates the l^{th} element in $\vec{\lambda}^{(q)}$, $l = 1,2,\dots,L$; $\eta_l^{(k)}$ indicates the l^{th} output of the k^{th} normalization layer neuron, $k = 1,2,\dots,K$, K is the neuron numbers in the normalization layer; $j = \sqrt{-1}$; $\gamma_1{}'(.)$ indicates the first derivative of Equation (45); $CT_l^{(q)}$ is obtained by substituting Equation (42) into Equations (47) to (49); and $ST_l^{(q)}$ is obtained by substituting Equation (39) into Equations (50) to (52).

$$CT_l^{(q)} = \gamma_2(C_l^{(q)}) \exp\left(\omega_2\left(C_l^{(q)}\right) j\right), \tag{47}$$

$$\gamma_2\left(C_l^{(q)}\right) = \text{real}\left(\exp\left(-\frac{\left(C_l^{(q)} - \mu_{tc_l}\right)^2}{2\left(\sigma_{tc_l}\right)^2}\right)\right), \tag{48}$$

$$\omega_2\left(C_l^{(q)}\right) = \text{real}\left(\gamma_2{}'\left(C_l^{(q)}\right)\right). \tag{49}$$

In Equations (47) to (49), μ_{tc_l} indicates the mean of tc_l; σ_{tc_l} indicates the standard deviation of tc_l; $\text{real}(.)$ indicates the real part of the result; $\gamma_2{}'(.)$ indicates the first derivative of Equation (48).

$$ST_l^{(q)} = \gamma_3(S_l^{(q)}) \exp\left(\omega_3\left(S_l^{(q)}\right) j\right), \tag{50}$$

$$\gamma_3\left(S_l^{(q)}\right) = \text{real}\left(\exp\left(-\frac{(S_l^{(q)})^2}{2(\sigma_{tc_l})^2}\right)\right), \tag{51}$$

$$\omega_3\left(S_l^{(q)}\right) = \text{real}\left(\gamma_3{}'\left(S_l^{(q)}\right)\right). \tag{52}$$

In Equations (50) to (52), $\gamma_3{}'(.)$ indicates the first derivative of Equation (51).

3.3.2 Fast aim-object method

In 2020, we proposed a novel ANFS model called the AANFS [70], which employed the aim-object method to establish an asymmetric layer. However, the operation of the aim-object method based on the derivative of functions is excessively complicated and increases the training time required for the AANFS model. In addition, the prediction performance of the AANFS is not significantly different from that of a conventional NFS. Therefore, the fast aim-object method is proposed to address this problem.

Steps 1 to 4 of the fast aim-object method are identical to those of the aim-object method. The transformation function in step 5 differs. Accordingly, we describe step 5 and explain the difference between the two methods as follows:

Step 5. Build neuron transformation functions in the asymmetric layer. Each asymmetric layer neuron can be considered as an L-layer transformation function. The transformation function can input K vectors containing L complex numbers and generate one vector output. Equations (53) to (56) are utilized to express the transformation function:

$$\vec{\lambda}^{(q)} = [\lambda_1^{(q)}, \lambda_2^{(q)}, \dots, \lambda_L^{(q)}]^T, \tag{53}$$

$$\lambda_l^{(q)} = \sum_{k=1}^{K} \gamma_1(\eta_l^{(k)}) \exp\left(\omega_1\left(\eta_l^{(k)}\right) j\right), \tag{54}$$

$$\gamma_1\left(\eta_l^{(k)}\right) = \exp\left(-\frac{(\eta_l^{(k)} - CT_l^{(q)})^2}{2(ST_l^{(q)})^2}\right) \cdot \rho_1, \tag{55}$$

$$\omega_1\left(\eta_l^{(k)}\right) = \exp\left(-\frac{(\eta_l^{(k)} - CT_l^{(q)})^2}{2(ST_l^{(q)})^2}\right) \cdot \rho_2, \tag{56}$$

where $\vec{\lambda}^{(q)}$ indicates the q^{th} neuron output in the asymmetric layer; $\lambda_l^{(q)}$ indicates the l^{th} element in $\vec{\lambda}^{(q)}$, where $l = 1,2,\dots,L$; $\eta_l^{(k)}$ indicates the l^{th} output of the k^{th} normalization layer neuron, where $k = 1,2,\dots,K$, K is the neuron numbers in the normalization layer; $j = \sqrt{-1}$; ρ_1 and ρ_2 are the parameters of Equations (55) and (56) learned by learning algorithms; $CT_l^{(q)}$ is obtained by substituting Equation (42) into Equations (47) to (49); and $ST_l^{(q)}$ is obtained by substituting Equation (39) into Equations (50) to (52).

In the fast aim-object method, $\omega_1(.)$ use the same primary function as $\gamma_1(.)$ to replace the complicated first derivatives of $\gamma_1(.)$, and ρ_1 and ρ_2 are added to the equations to increase the output difference of the two functions and the mapping ability to the target.

3.4 Asymmetric neuro-fuzzy system

A novel NFS architecture named the asymmetric neuro-fuzzy system (ANFS) is presented based on the approaches described in this Section. The ANFS is a seven-layer structure model, shown in Fig. 8. We describe each layer as follows.

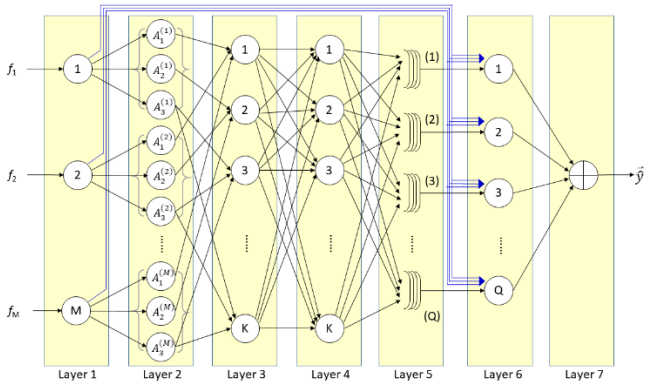

Fig. 8. ANFS architecture.

Layer 1: Layer 1 is the input layer. Each input feature variable maps to an input layer neuron. The neuron sends the input to the next layer and does not execute any processing. The input vector is given as follows:

$$\mathbf{h} = [f_1\ f_2 \dots f_m \dots f_M], \tag{57}$$

where f_m indicates the m^{th} input feature variable, $m = 1,2,\dots,M$, M is the input feature variable numbers.

Layer 2: Layer 2 is the fuzzy-set layer. Conventionally, the neuron numbers of this layer are determined by the user building the model or by experts in that domain. However, general users lack domain knowledge, and the subjectively determined structure is often not the optimal structure. Furthermore, employing domain experts to assist in building the model structure increases costs. In this dissertation, we employed SC to assist users who lack domain knowledge in determining the number of neurons in the premise layer. The steps of the method are explained as follows.

Step 1. SC is used to cluster each input variable to determine the cluster numbers of each dimension.

Step 2. Each cluster obtained in Step 1 is a fuzzy-set layer neuron and represents a linguistic value that is characterized by an SCFS introduced in Section 3.2, expressed as $A_i^{(m)}$, meaning the i^{th} fuzzy set of f_m, $i = 1,2,\dots,\Gamma_m$, Γ_m is the fuzzy set numbers of f_m. The parameters of $A_i^{(m)}$, $\{c, s, \xi_1, \xi_2, \xi_3\}$, are optimized by the learning algorithm proposed in Section 3.5. The output of $A_i^{(m)}$ is a complex-valued membership degree vector, shown as follows.

$$\vec{\mu}A_i^{(m)} = [\mu A_i^{(m)(1)}, \mu A_i^{(m)(2)}, \dots, \mu A_i^{(m)(L)}], \tag{58}$$

where $\mu A_i^{(m)(l)}$ indicates the l^{th} output element of $A_i^{(m)}$, $l = 1,2,\dots,L$, L is the output numbers of ANFS.

Layer 3: Layer 3 is the premise layer. Each neuron represents one premise part of an FIS, composed by the fuzzy sets of each input feature variable. A construct matrix, **CM**, can be used to express the relationship.

$$\mathbf{CM} = \begin{matrix} \begin{matrix} 1 & 2 & \cdots & m & \cdots & M \end{matrix} \\ \begin{bmatrix} c^{(1,1)} & c^{(1,2)} & \cdots & c^{(1,m)} & \cdots & c^{(1,M)} \\ c^{(2,1)} & c^{(2,2)} & \cdots & c^{(2,m)} & \cdots & c^{(2,M)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ c^{(k,1)} & c^{(k,2)} & \cdots & c^{(k,m)} & \cdots & c^{(k,M)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ c^{(K,1)} & c^{(K,2)} & \cdots & c^{(K,m)} & \cdots & c^{(K,M)} \end{bmatrix} \end{matrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ k \\ \vdots \\ K \end{matrix}, \tag{59}$$

where the element in the k^{th} row and the m^{th} column in **CM**, expressed as $c^{(k,m)}$, is a positive integer value, indicating that the k^{th} premise layer neuron employs the $c^{(k,m)^{th}}$ fuzzy set of f_m as one of the inputs, $k = 1,2,\dots,K$, $m = 1,2,\dots,M$.

In this layer, the "fuzzy and" operator is employed by the neurons to compute the rule firing strengths, expressed as follows:

$$\vec{\beta}^{(k)} = \left[\beta_1^{(k)}, \beta_2^{(k)}, \dots, \beta_L^{(k)}\right], \tag{60}$$

$$\beta_l^{(k)} = \prod_{m=1}^{M} \mu A_{c(k,m)}^{(m)}{}^{(l)}, \tag{61}$$

where $\vec{\beta}^{(k)}$ indicates the firing strength vector of the k^{th} neuron; $\beta_l^{(k)}$ indicates the l^{th} element of $\vec{\beta}^{(k)}$. In Equation (61), we employ multiplication as the "fuzzy and" operator to compute the rule firing strengths because it can fully utilize the membership degree of each input variable.

Layer 4: Layer 4 is the normalization layer. Each neuron is in charge of calculating the normalized firing strength vector of a single rule, expressed as follows.

$$\vec{\eta}^{(k)} = \left[\eta_1^{(k)}, \eta_2^{(k)}, \dots, \eta_L^{(k)}\right], \tag{62}$$

$$\eta_l^{(k)} = \frac{\beta_i^{(k)}}{\sum_{i=1}^{K} \beta_l^{(i)}}, \tag{63}$$

where $\vec{\eta}^{(k)}$ indicates the output, a normalized firing strength vector, of the k^{th} neuron; $\eta_l^{(k)}$ indicates the l^{th} element of $\vec{\eta}^{(k)}$.

Layer 5: Layer 5 is the asymmetric layer. Each neuron uses the normalized firing strength vectors generated from the normalization layer as its inputs. We proposed two algorithms to construct the asymmetric layer in Section 3.3, and the neuron structure is illustrated in Fig. 9, where $\vec{\lambda}^{(q)}$ indicates the q^{th} neuron output. Note that there are two parameters, $\{\rho_1, \rho_2\}$, requiring to be optimized by the learning algorithm proposed in Section 3.5 if we hire the fast aim-object method to build this layer.

Fig. 9. Neuron structure in the asymmetric layer. (Ignored)

Layer 6: Layer 6 is the consequent layer. Each neuron is a T-S function, equal to a consequent part in a traditional FIS. The q^{th} neuron output of the consequent layer is expressed as $\vec{\hat{y}}^{(q)}$, which is a $L \times 1$ vector, obtained as follows.

$$\vec{\hat{y}}^{(q)} = \vec{\Psi}^{(q)}.\ \vec{a}^{(q)}, \tag{64}$$

where $\vec{a}^{(q)}$ indicates a $(M+1) \times 1$ matrix containing parameters of the q^{th} T-S function, which can be optimized by the learning algorithm proposed in Section 3.5, $\vec{a}^{(q)} = \left[a_0^{(q)}, a_1^{(q)}, \dots, a_M^{(q)}\right]$; $\vec{\Psi}^{(q)}$ is defined as follows.

$$\vec{\Psi}^{(q)} = \vec{\lambda}^{(q)}.H, \tag{65}$$

where $\vec{\lambda}^{(q)}$ is the output of q^{th} neuron of layer 5, which is a $L \times 1$ vector shown in Equation (43); H is a $1 \times (M+1)$ vector, $H = [1, f_1, f_2, \dots, f_M]$; $\vec{\Psi}^{(q)}$ is a $L \times (M+1)$ matrix.

Layer 7: Layer 7 is the output layer. Only one neuron exists in this layer and is in charge of summarizing the outputs of the consequent layer to generate system output $\vec{\hat{y}}$, w. The output of the q^{th} neuron of the consequent layer is denoted as $\vec{\hat{y}}^{(q)}$, which is represented as follows.

$$\vec{\hat{y}} = \sum_{q=0}^{Q} \vec{\hat{y}}^{(q)}, \tag{66}$$

where $\vec{\hat{y}}$ indicates a vector with L complex-valued elements. Each element can be divided into a real part and imaginary part corresponding to two targets we want to forecast, respectively.

3.5 Parameter learning

In this Section, a novel hybrid learning algorithm, SWOA-RLSE, is proposed. We will explain SWOA, RLSE, the hybrid learning algorithm of these two methods, and the setting of SWOA-RLSE in the following, respectively.

Mirjalili and Lewis [22] presented the WOA, which emulates the hunting behavior of humpback whales, to find the global optimal solution. Whales swim around their prey $\vec{X}^*$ in a shrinking circle or along a spiral-shaped path, where $\vec{X}^*$ is the optimal solution of the problem. In addition, the WOA retains random search behavior. The mathematical functions of these behaviors in original version of WOA are expressed as follows:

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D}_i, if\ p < 0.5\ and\ \left|\vec{A} < 1\right| \\ \vec{X}^*(t) + \vec{D}_i' \cdot e^{bl} \cdot \cos(2\pi l), if\ p \geq 0.5 \\ \vec{X}_{rand}(t) - \vec{A} \cdot \vec{D}_{i,rand}, if\ p < 0.5\ and\ \left|\vec{A} \geq 1\right| \end{cases}, \tag{67}$$

where the first part of Equation (67) mimics the shrinking encircling method, the second part mimics the spiral method, and the third part is the random method; $\vec{X}_i(t+1)$ indicates the i^{th} whale's position in the $(t+1)^{th}$ iteration; $\vec{X}^*(t)$ indicates the optimal solution in the t^{th} iteration; $\vec{A}$ indicates a coefficient vector, $\vec{A} = 2\vec{\imath} \cdot \vec{r} - \vec{\imath}$, $\vec{r}$ is a random vector, and each element of $\vec{r}$ is located in [0,1]. The value of $\vec{\imath}$ decreases linearly from 2 as the iteration proceeds; $\vec{D}_i = \left|\vec{B} \cdot \vec{X}^*(t) - \vec{X}_i(t)\right|$, $\vec{B} = 2\vec{r}$; $\vec{D}_i' = \left|\vec{X}^*(t) - \vec{X}_i(t)\right|$; b is a constant defining the shape of the logarithmic spiral; l is a random number within [−1,1]; $\vec{X}_{rand}(t)$ is the position of a randomly selected whale in the t^{th} iteration; $\vec{D}_{i,rand} = \left|\vec{B} \cdot \vec{X}_{rand}(t) - \vec{X}_i(t)\right|$; and the variable p is a random number within [0,1].

The original version of the WOA has the following problems: First, the whales' behavior is determined on the basis of the external variables $\vec{A}$ and p, which cannot reflect the whales' real status. Second, the whales use the spiral method more than they use the shrinking encircling or random methods without a reasonable explanation. Finally, the random method conditions of $p < 0.5$ and $\left|\vec{A}\right| \geq 1$ appear only in the early stages of learning, causing the WOA to avoid local optimization difficultly. On the basis of the aforementioned problems, we propose a SWOA inspired by that of Zhan [74] to improve the original WOA. It decides the whales' behavior according to their actual dispersed state and adjusts the random method's conditions to appear in all stages of learning. The SWOA algorithm is described as follows:

Step 1. Calculate the average distance between whale i and the other whales, denoted as d_i:

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2}, \tag{68}$$

where $i = 1,2,\dots,N$; N is the number of whales; x_i^k is the location of whale i in dimension k, $k = 1,2,\dots,D$, and D is the dimensions of a problem.

Step 2. Calculate the degree of distribution, denoted as δ, as follows:

$$\delta = \frac{d_g - d_{min}}{d_{max} - d_{min}} \in (-\infty, \infty), \tag{69}$$

where d_g is the average distance between the optimal position and the other whales; d_{max} and d_{min} are the maximum and minimum average distances among the whales, respectively. When the whales are in the convergence state, δ is small because they surround the optimal position. By contrast, δ is large when the whales are in the exploration state.

Step 3. Determine the state of the whales on the basis of δ obtained in the previous step, denoted as S, as follows:

$$S = \begin{cases} 1, if\ \delta \geq \tau \\ 2, if\ \delta < \tau \end{cases}, \tag{70}$$

where τ is the threshold for separating the status of the whales and can be adjusted according to different applications. In this dissertation, τ is set to 0.1.

Step 4. Determine the behavior of the whales on the basis of S. When $S = 1$, the whales are in the exploration state and will use the random, shrinking encircling, or spiral methods depending on random variable $p1$ and $p2$. When $S = 2$, the whales are in the convergence state. If the algorithm identifies a new optimal solution in this iteration in the convergence state, the whales will use the shrinking encircling or spiral methods depending on random variable $p2$ to continually narrow the search area. Otherwise, the whales will use one of the three behaviors depending on random variables $p1$ and $p2$. The mathematical functions are expressed as follows:

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}^*(t) - \left(\vec{A} \cdot \vec{D}_i\right), \text{condition 1} \\ \vec{X}^*(t) + \left(\vec{D}_i' \cdot e^{bl} \cdot \cos(2\pi l)\right), \text{condition 2} \\ \vec{X}_{rand}(t) - \left(\vec{A} \cdot \vec{D}_{i,rand}\right), \text{condition 3} \end{cases}, \tag{71}$$

where condition 1 is $if\ (S = 1\ \&\&\ p_1 < 0.5\ \&\&\ p_2 \geq 0.5)\ ||\ (S = 2\ \&\&\ ((\varsigma = \text{true}\ \&\&\ p_2 \geq 0.5)\ ||\ (\varsigma = \text{false}\ \&\&\ p_1 < 0.5\ \&\&\ p_2 \geq 0.5)))$, where ς is a flag indicating whether the SWOA obtains a new optimal solution in this iteration; condition 2 is $if\ (S = 1\ \&\&\ p_1 < 0.5\ \&\&\ p_2 < 0.5)\ ||\ (S = 2\ \&\&\ ((\varsigma = \text{true}\ \&\&\ p_2 < 0.5)\ ||\ (\varsigma = \text{false}\ \&\&\ p_1 < 0.5\ \&\&\ p_2 < 0.5)))$; and condition 3 is $if\ (S = 1\ \&\&\ p_1 \geq 0.5)\ ||\ (S = 2\ \&\&\ \varsigma = \text{false}\ \&\&\ p_1 \geq 0.5)$. We show the conditions mentioned above in Fig. 10 for a more explicit expression.

Fig. 10. Determine the behavior of the whale based on the whales' state. (Ignored)

The parameters of an NFS model can be separated into two parts, nonlinear and linear. The SWOA is in charge of optimizing the premise layer's nonlinear parameters. Then, these parameters are used to calculate the normalized firing strengths by the model. Further, the normalized firing strengths are sent to the asymmetric layer to calculate the layer's outputs. Finally, the input variables and asymmetric layer's outputs are sent to the RLSE to compute the optimal solution of the consequent layer's linear parameters.

The consequent layer's optimal solution, $\boldsymbol{\theta}$, can be obtained by the following RLSE equations [26]:

$$\mathbf{P}(l+1) = \mathbf{P}(l) - \frac{\mathbf{P}(l)\mathbf{b}(l+1)\mathbf{b}(l+1)^T\mathbf{P}(l)}{\mathbf{I} + \mathbf{b}(l+1)^T\mathbf{P}(l)\mathbf{b}(l+1)}, \tag{72}$$

$$\boldsymbol{\theta}(l+1) = \boldsymbol{\theta}(l) + \mathbf{P}(l+1)\mathbf{b}(l+1)\left(\vec{t}(l+1) - \mathbf{b}(l+1)^T\boldsymbol{\theta}(l)\right), \tag{73}$$

where $\mathbf{I}$ indicates the identity matrix; $\mathbf{b}(l+1)$ is defined in Equation (74), where $l = 0,1,\ldots,(|TD| - 1)$; $\vec{t}(l+1)$ is the $(l+1)^{th}$ row of $\mathbf{T}$, where $\mathbf{T} = [\mathbf{t}^{\{1\}}\ \ \mathbf{t}^{\{2\}}\ \ \ldots\ \ \mathbf{t}^{\{|TD|\}}]^{\mathrm{T}}$ and $\mathbf{t}^{\{i\}}$ indicates the target vector from the i^{th} data pair of *TD*.

To initialize RLSE, $\boldsymbol{\theta}(0)$ is set as a zero matrix, and $\mathbf{P}(0) = \alpha\mathbf{I}$, where α must be a large positive value. In Equations (72) and (73), vector $\mathbf{b}$ and vector $\boldsymbol{\theta}$ are arranged as follows:

$$\mathbf{b}(l+1) = \left[\mathbf{bb}^{(1)}(l+1)\ \mathbf{bb}^{(2)}(l+1)\ \ldots\ \mathbf{bb}^{(Q)}(l+1)\right], \tag{74}$$

$$\mathbf{bb}^{(q)}(l+1) = \left[\vec{\lambda}^{(q)}(l+1)\ \vec{\lambda}^{(q)}(l+1)f_1^{\{l+1\}}\ \ldots\ \vec{\lambda}^{(q)}(l+1)f_M^{\{l+1\}}\right], \tag{75}$$

$$\boldsymbol{\theta}(l) = \left[\vec{a}^{(1)}(l)\ \vec{a}^{(2)}(l)\ \ldots\ \vec{a}^{(Q)}(l)\right], \tag{76}$$

$$\vec{a}^{(q)}(l) = \left[a_0{}^{(q)}(l)\ a_1{}^{(q)}(l)\ \ldots\ a_M{}^{(q)}(l)\right], \tag{77}$$

where q and l denote the q^{th} neuron in the consequent layer and the l^{th} repetition of machine learning, respectively, where $q = 1,2,\ldots,Q$ and $l = 0,1,\ldots,(|TD| - 1)$; $\vec{\lambda}^{(q)}$ is the output of the q^{th} neuron in the asymmetric layer and was defined in Equation (43); $f_m^{\{l+1\}}$ is the m^{th} input feature variable from the $(l+1)^{th}$ data pair of *TD*.

The SWOA-RLSE hybrid learning algorithm is illustrated as Fig. 11 and detailed herein:

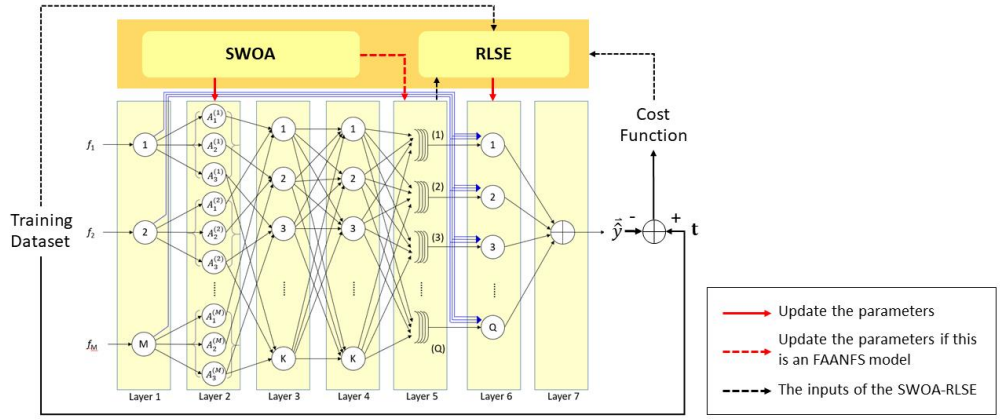

Fig. 11. Schematic of the SWOA-RLSE.

Step 1. Initialize SWOA to obtain the premise layer's initial parameters.

Step 2. Substitute the parameters generated by SWOA to the fuzzy-set layer of the model, then determine the normalized firing strength vectors using the steps described in Section 3.4 Layer 1 to Layer 4.

Step 3. If necessary, substitute the parameters generated by SWOA to the asymmetric layer, then hire the normalized firing strength vectors as inputs to calculate the asymmetric layer's output vectors using the method in Section 3.3 and Section 3.4 Layer 5.

Step 4. Use the RLSE, defined in Equations (72) to (77), to obtain the consequent layer's parameters.

Step 5. Substitute the training data into the model and compute the system output of each target.

Step 6. Repeat steps 2 to 6 to compute the costs for all whales. In this dissertation, we hire the root mean square error (RMSE) as the cost function, expressed as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{|TD|}(\boldsymbol{\varepsilon}^{\{i\}})^{*}\boldsymbol{\varepsilon}^{\{i\}}}{|TD|}}, \tag{78}$$

$$\boldsymbol{\varepsilon}^{\{i\}} = \mathbf{t}^{\{i\}} - \vec{\hat{y}}^{\{i\}}, \tag{79}$$

where $\{\boldsymbol{\varepsilon}^{\{i\}}, \mathbf{t}^{\{i\}}, \vec{\hat{y}}^{\{i\}}\}$ are the error vectors, target, and model output, respectively, for the i^{th} data pair; $(\boldsymbol{\varepsilon}^{\{i\}})^{*}$ is the complex conjugate and transposition of $\boldsymbol{\varepsilon}^{\{i\}}$.

Step 7. If necessary, update the optimal position, $\vec{X}^{*}$, for SWOA and the premise layer's parameters.

Step 8. Repeat steps 2 to 8 until one of the stopping criteria is met. The frequently used stopping criteria include a total number of learning iterations and a model performance accuracy threshold. These criteria should be predefined by the user before machine learning.

Some parameters need to be set before SWOA-RLSE running, which are listed in Table 1. In this table, the values are given based on the architecture of the proposed model.

Table 1. The SWOA-RLSE settings (Ignored)