ReLu-FCM trained by quasi-oppositional bare bone imperialist competition algorithm for predicting employment rate

Aihua Guo

LiRen College, Yanshan University, Qinhuangdao, Hebei, China

ahguo@ysu.edu.cn

Abstract

Fuzzy cognitive maps (FCMs) are a powerful tool for system modeling, which can be used for static and dynamic analysis. However, traditional FCMs are usually learned by gradient-based methods, and the adopted sigmoid nonlinear activation function frequently causes gradient saturation. These two shortcomings set a limit on the modeling accuracy. To overcome those problems, we propose in this paper a new FCM with two improvements. First, the rectified linear unit (ReLu) activation function is adopted to replace the sigmoid function. Second, a newly proposed quasi-oppositional bare bone imperialist competition algorithm (QBBICA) is used to learn the FCM. The improved FCM is used to predict the employment rate of graduates from Liren College, Yanshan University. Experimental results show that the improved FCM is effective in employment rate prediction.

1 Introduction

Improving the employment rate and employment quality of university graduates has been an important task for China’s education authority and universities. Based on the employment status of graduates, the education authority can adjust relevant policies and the university’s faculty can adjust their education and training plans to enhance the graduates’ employability. To guarantee sufficient employment of graduates, the responsible decision-makers need to make adjustments ahead of time. Adequate supporting data is necessary to ensure that the decisions are made in a scientific manner. Hence, accurately predicting the employment rate of graduates becomes an important topic of research.

Since the employment rate of graduates is affected by many factors, accurate prediction of graduates’ employment rate remains a difficult task. In the past years, many researchers have paid much attention to the prediction of employment rate of graduates. In [1], Wang adopted a neural network with an adaptive learning back-propagation framework to construct an intelligent employment rate prediction model. In [2], Guo et al. developed a framework called Multi-major Employment Status to predict graduates’ employment status when bias exists. In the framework, a long short term memory (LSTM) with dropout was adopted as the predictor. In [3], an employment rate prediction model was constructed using the back propagation (BP)
neural network based on employment rate data from 2002 to 2012. Usually, the BP neural network is trained using the gradient descent algorithm, which can be easily trapped in a local optimum. To overcome this drawback, Jiang et al. [4] proposed to use an artificial fish swarm algorithm (AFSA) to train a BP neural network for predicting university graduates’ employment rate. Zhang proposed to predict university graduates’ employment rate using regression analysis, and they constructed two regression models, i.e., the polynomial regression model and the multiple linear regression model [5]. To improve the prediction accuracy, a model that combines the B-mode relational degree and general induced order weighted averaging aggregation (GIOWA) operator was proposed in [6]. A decision tree model for employment prediction was presented in [7].

Despite much research progress in the prediction of graduates’ employment rate, there are still improvements to be made in the prediction accuracy. The reason for this problem is twofold. First, the employment rate data of a single university is not large enough. That is to say, one can not obtain enough samples to construct an artificial neural networks (ANN) based prediction model. As a consequence, the learned ANN model is under or over fit and the prediction accuracy decrease. Second, factors affecting the employment rate of graduates and the causality between these factors and the employment rate of graduates are still unknown. In other words, the employment rate data contains much uncertainties. The above mentioned neural networks and gray model based prediction methods are still weaker in handling uncertainty. Therefore, it is necessary to develop an alternative model that can handle different uncertainties and perform prediction with limited data.

Fuzzy cognitive maps (FCMs), which was first introduced by Kosko [8], are a powerful tool for modeling and simulating complex dynamic systems. In nature, an FCM is a directed graph consisting of a set of nodes and edges. The nodes stand for concepts, variables or factors in a system, and they are connected with each other through edges. The weights of the edges represent the degree of the interaction between these nodes. FCMs have several appealing characteristics in modeling the complex relationship between variables. One is that FCM is flexible and easy to be constructed. Second, FCM has powerful ability to handle uncertainties. Finally, FCM can be sufficiently trained with small samples. With these advantages, FCMs have been widely used to solve different scientific and engineering problems, for example, in [9], an enhanced FCM called E-FCMs is proposed to model the influence strength between brain activity associated with different regions of the brain. In [10], the FCM is applied to gene regulatory network reconstruction, and a preference-based evolutionary bi-objective algorithm was developed to learn the FCM. In [11], the FCM was used to predict the DNA-Binding residues in local segments of protein sequences. In [12], FCM was used to model the discontinuous behavior in a complex system. In [13], a higher-order FCM combined with sparse autoencoder was proposed for time series predicting and a batch gradient descent method is adopted to learn the FCM. In [14], a fast and effective method based on constrained convex optimization was developed for FCM learning. In [15], FCM was used to control wheeled mobile robot.

Although FCMs have found successful applications in many fields, they still suffer from several drawbacks. First, FCMs adopt the sigmoid function as the nonlinear transfer function. Recent deep learning researches have revealed that the sigmoid function can lead to poor performance due to its saturation near 0 and 1, which limits its ability to learn nonlinear mappings. Second, gradient-based methods are traditionally used as the learning algorithms for FCMs. Gradient-based methods are sensitive to noise and can easily be trapped in a local optimum. Recently, several population-based methods have been proposed to learn FCMs due to their global searching characteristics. In [16], the real-coded genetic algorithm (RCGA) was used to learn FCMs, where the weights of FCMs to be learned were encoded as an individual
of GA. Compared to GA, particle swarm optimization (PSO) has simpler evolutionary operations. Therefore, in [17], PSO was adopted to learn FCMs. The differential evolution (DE) [18], the asexual reproduction optimization (ARO) algorithm [19], and the imperialist competition algorithm (ICA) [20] have also been adopted to learn FCMs.

ICA is an effective evolutionary algorithm first proposed by Atashpaz-Gargari and his colleagues [21]. It shows great potential in solving different complex optimization problems, such as flexible job shop scheduling [22], neural network training [23], parameter optimal design of heat exchangers [24], controller parameter tuning [25], and node placement in wireless sensor networks [26]. However, like many other evolutionary algorithms, ICA is also plagued by the premature problem. That is to say, if ICA finds a local optimal solution and fails to jump out of it, ICA will converge to the local optimum, which is undesired. To enhance the search ability of ICA, several ICA variants have been proposed. These variants can be roughly classified into three classes. The first class is to change the assimilation strategy. For example, in [27], an ICA variant called AR-ICA was proposed, where the assimilation coefficients are dynamically adjusted based on the concept of attraction and repulsion to better balance the diversity and convergence of the algorithm. In [28, 29], a modified ICA was proposed, in which the colonies move to its imperialist using a velocity-position model like PSO algorithm. In [30], a bare-bone ICA (BB-ICA) was given. In BB-ICA, the new position of each colony is determined according to a Gaussian sampling. In [31], an ICA variant called Gbest-ICA was proposed via taking the difference between the colony’s current position and the global best position into account when colony moves towards its imperialist. An orthogonal ICA (O-ICA) was proposed in [32], where a new move direction orthogonal to the original move direction is added to the colonies’ movement formula. The other class of ICA variants is to modify the original ICA’s competition and revolution step. In [33], the chaotic map was used to generate the revolted colonies, while in [34], the GA mutation operator is used to generate new revolted colonies to replace some original colonies. In [35], a clone evolution operator is introduced into the competition phase. Apart from the above two classes of variants, some researchers hybrid ICA with other evolutionary algorithms to improve the performance of ICA. For instance, in [36], the hybrids of ICA and simplex method was proposed. In the hybrid algorithm, the simplex method is responsible for a local searching. In [37], the ICA is hybridized with simulating annealing (SA), in which the SA is used to improve the position of imperialist after assimilation step. In [38], ICA is mixed with pattern search algorithm, the results of ICA severing as initial position are further improved using pattern search.

With the above facts in mind, we develop in this paper an improved FCM in which two improvements are made. First, a ReLU function is adopted to replace the original sigmoid function to enhance the nonlinear mapping ability. Second, an improved ICA, which integrates the quasi-opposition learning and bare bone ICA, is proposed to learn FCMs’ weights. The proposed method is used to predict the employment rate of university graduates.

The rest of this paper is arranged as follows. Some basics about FCMs are provided in Section 2. The bare-bone imperialist competitive algorithm is detailed in Section 3. The proposed method for predicting the employment rate of graduates is presented in Section 4. The prediction results are shown in Section 4.4. Finally, the concluding remarks are given in Section 5.2.

2 ReLU-FCM

FCMs, as shown in Fig 1, can be viewed as a fuzzy version of Axelrod’s cognitive maps. The causality between the nodes in an FCM can be a fuzzy degree. Like cognitive maps, an FCM consists of a set of nodes and weighted edges. The nodes represent the concepts or variables of a system while the weighted edges denote the causality between these concepts or variables.
Suppose there are $n$ nodes in an FCM, denoted as $\{C_1, C_2, \cdots, C_n\}$. The weight of the edge from node $C_j$ to $C_i$ is denoted by $w_{ji}$ where $i, j = 1, 2, \cdots, n$. The value of $w_{ji}$ lies in $[-1, 1]$. The value of weight $w_{ji}$ explains the degree of the causality between the nodes, and the sign of weight $w_{ji}$ denotes the influential direction of the connected nodes. Specifically, if $w_{ji} > 0$, it means that an increase of node $C_j$ results in an increase of node $C_i$ with strength $w_{ji}$ and vice versa. If $w_{ji} < 0$, it means an increase of node $C_j$ results in a decrease of node $C_i$ with strength $|w_{ji}|$ and vice versa. If $w_{ji} = 0$ means there is no relation between node $C_j$ and node $C_i$, the weights of all edges in an FCM can be rewritten into a matrix form as follows:

$$
W = \begin{bmatrix}
w_{11} & w_{12} & \cdots & w_{1n} \\
w_{21} & w_{22} & \cdots & w_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
w_{n1} & w_{n2} & \cdots & w_{nn}
\end{bmatrix}
= \begin{bmatrix}w_1 & w_2 & \cdots & w_n\end{bmatrix}
$$

(1)

In the FCM, each node at time $t$ has a value, called activation or state value, denoted by $a_i(t)$. The state values of nodes vary with time. At time $t + 1$, the state value is calculated as:

$$
a_i(t + 1) = f \left( \sum_{j=1}^{n} a_j(t) w_{ji} \right), \, i = 1, 2, \cdots, n,
$$

(2)

where $f(\cdot)$ is the transfer function or activation function. Usually, the sigmoid function can be expressed as:

$$
f(x) = \frac{1}{1 + e^{-\lambda x}}
$$

(3)
is selected as the transfer function. As stated before, the sigmoid function has several drawbacks that limit its nonlinear mapping performance. In this paper, the ReLU function is adopted to replace the sigmoid function. The ReLU function is defined as:

$$f(x) = \max(0, x).$$  \hspace{1cm} (4)

Denote \(a(t) = [a_1(t), a_2(t), \cdots, a_n(t)]^T\). Then (2) can be written into the following form:

$$a_i(t+1) = f(a(t)w_i).$$  \hspace{1cm} (5)

In (5), the state value of each node at time \(t + 1\) depends on the state values of all nodes at time \(t\) only. If the state value of each node at time \(t + 1\) depends on the state values of all nodes at both time \(t\) and time \(t-1, \cdots, t-k+1\), i.e.,

$$a_i(t + 1) = f(a(t)w_i^1 + a(t-1)w_i^2 + \cdots + a(t-k+1)w_i^k),$$  \hspace{1cm} (6)

then, the FCM is called a high-order FCM. Eq (6) can be written in a matrix form as follows:

$$a_i(t + 1) = f\left(\sum_{j=1}^{k} a(t-j+1)w_i^j\right),$$ \hspace{1cm} (7)

To enable the FCM to describe the complex dynamics of a system, we must determine its weights first. The weights can be determined in two ways, by using expert knowledge and by learning from actual data. In this paper, the weights of the FCM are obtained by learning. An improved ICA called BICA is developed to perform the learning.

### 3 Quasi-oppositional bare-bone imperialist competitive algorithm

#### 3.1 Quasi-opposition learning

Opposition-based learning (OBL) was first introduced by Tizhoosh in 2005 [39] inspired by the Ying-Yang concept. OBL aims to find better solutions for machine learning tasks in an opposition direction. Since its introduction, OBL has been applied in several different fields such as neural network training and evolutionary algorithm construction [39]. OBL can be used to improve the performance of original methods. However, the opposition points frequently exceed the searching range and are therefore unable to produce higher performance. To mitigate this problem, in this paper, we focus on a variant of OBL, i.e., quasi-opposition based learning (QOBL). The related concepts are stated as follows.

**Definition 1 (Opposite number)** Let \(x \in [a, b]\) be a real number. Its opposite \(\bar{x}\) is defined as:

$$\bar{x} = a + b - x.$$  \hspace{1cm} (8)

**Definition 2 (Opposite point in the D-dimensional space)** [39] Let \(x = [x_1, x_2, \cdots, x_D]\) be a point in \(D\)-dimensional space and \(x_i \in [a_i, b_i],\) \(i = 1, 2, \cdots, D\). The opposition of \(x\) is denoted as \(\bar{x} = [\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_D]\), where each element of \(\bar{x}\) is given as:

$$\bar{x}_i = a_i + b_i - x_i.$$  \hspace{1cm} (9)
Definition 3 (Quasi-opposite number) Let \( x \in [a, b] \) be a real number. Its quasi-opposite number \( \tilde{x} \) is defined as:

\[
\tilde{x} = \begin{cases} 
\frac{a + b}{2} + r \left( \tilde{x} - \frac{a + b}{2} \right), & \tilde{x} < \frac{a + b}{2}; \\
\tilde{x} + \left( \frac{a + b}{2} - \tilde{x} \right), & \tilde{x} \geq \frac{a + b}{2}.
\end{cases}
\]  

(10)

Definition 4 (Quasi-opposite point in the D-dimensional space) Let \( \mathbf{x} = [x_1, x_2, \ldots, x_D] \) be a point in \( D \)-dimensional space and \( x_i \in [a, b], i = 1, 2, \ldots, D \). The quasi-opposition of \( \mathbf{x} \) is denoted as \( \tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_D] \), where each element of \( \tilde{\mathbf{x}} \) is given as:

\[
\tilde{x}_i = \begin{cases} 
\frac{a + b}{2} + r \left( \tilde{x}_i - \frac{a + b}{2} \right), & \tilde{x}_i < \frac{a + b}{2}; \\
\tilde{x}_i + \left( \frac{a + b}{2} - \tilde{x}_i \right), & \tilde{x}_i \geq \frac{a + b}{2}.
\end{cases}
\]  

(11)

Definition 5 (Opposition or quasi-opposition based optimization) Let \( \mathbf{x} = (x_1, x_2, \ldots, x_D) \) be a point in \( D \)-dimensional space and \( f(\mathbf{x}) \) the fitness function. The opposition and quasi-opposition point of \( \mathbf{x} \) is denoted as \( \mathbf{\tilde{x}} \) and \( \tilde{\mathbf{x}} \), respectively. If \( f(\mathbf{\tilde{x}}) \) or \( f(\tilde{\mathbf{x}}) \) is better than \( f(\mathbf{x}) \), then point \( \mathbf{x} \) is replaced with \( \mathbf{\tilde{x}} \) or \( \tilde{\mathbf{x}} \); otherwise, point \( \mathbf{x} \) is reserved.

3.2 Bare bone ICA (BBICA)

BICA involves the same main steps as basic ICA, including initialization, assimilation, imperialistic competition, and convergence. These steps are described in detail in the following sections.

3.2.1 Initialization. In this step, the main task is to generate initial countries, and then classify them into imperialists and colonies according to their power. Furthermore, the colonies are probably assigned to each imperialist to form empires. Suppose a country is represented by a vector \( \mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}] \) and each element \( x_{ij} \) of vector \( \mathbf{x}_i \) lies in \( [l_j, u_j] \), which is randomly generated as:

\[
x_{ij} = l_j + \text{rand} \times (u_j - l_j), \quad i = 1, 2, \ldots, N, j = 1, 2, \ldots, D,
\]  

(12)

where \( \text{rand} \) is a random number distributed in \( [0, 1] \). To evaluate each country, the cost of each country is calculated as \( c_i = \mathbb{P}(\mathbf{x}_i) \), where \( \mathbb{P}(\cdot) \) is called the cost function. Once \( N \) countries are generated, they will be classified into imperialists and colonies according to their power, i.e., the cost of each country. Usually, the first \( N_{imp} \) countries are selected as the imperialists and the remaining \( N_{col} = N - N_{imp} \) countries are classified as colonies. Then, the colonies are assigned to each imperialist. The \( n \)-th imperialist occupies \( NC_n \) colonies, which are calculated as:

\[
NC_n = \text{round}\{p_n \times N_{col}\},
\]  

(13)

where

\[
p_n = \left| \frac{c_n - \max_{i=1,2,\ldots,N} c_i}{\sum_{i=1}^{N_{imp}} (c_n - \max_{i=1,2,\ldots,N} c_i)} \right|
\]  

(14)

is the normalized power of an imperialist and \( c_n \) is the cost of the \( n \)-th imperialist. The \( NC_n \) colonies together with the imperialist make up the \( n \)-th empire.
3.2.2 Assimilation. After initialization, the colonies in each empire move towards the corresponding imperialists to increase the total power of the empire. This process is called the assimilation step. In the original ICA, the colonies move towards their corresponding imperialists as shown in Fig 2. However, the assimilation is an important step, which is responsible for the exploitation and exploration of the ICA. A good strategy should strike a good balance between exploration and exploitation. To this end, a Gaussian sampling technique is adopted as the assimilation strategy in the BICA. Specifically, the new position of a colony is determined as:

\[ x'_{ij} = N(\mu_j, \sigma_j), \]  

where \( x'_{ij} \) represents the \( j \)-th variable of \( x'_i \), i.e., the new position of the \( i \)-th colonies in an empire, and \( N(\mu_j, \sigma_j) \) is a Gaussian distribution random number with mean \( \mu_j \) and variance \( \sigma_j \). The mean and variance are calculated as:

\[ \mu_j = \frac{x_{ij} + x'_{ij}}{2} \]  

and

\[ \sigma_j = |x'_{ij} - x_{ij}|, \]  

where \( x'_{ij} \) is the \( j \)-th variable of the imperialist position.

3.2.3 Imperialistic competition. After assimilation, the power of each empire is enhanced. Then, the competition among empires begins. The competition leads more powerful empires to possess more colonies and the weaker ones to lose their colonies. The lost colonies from the weaker empires are reassigned to other more powerful empires via competition. Let the \( \ell \)-th empire be the weakest. The probability of the \( n \)-th empire to possess the colonies is given as:

\[ P_n = \frac{NTC_n}{\sum_{i=1}^{N} NTC_i}, \]
where $NTC_n = TC_n - \max_i \{TC_i\}, i = 1, 2, \cdots, N_{\text{imp}}$ and $TC_i$ is the total cost of the $i$-th empire. Define a probability vector $P$ as

$$P = [p_1, p_2, \cdots, p_{N_{\text{imp}}}],$$  \hspace{1cm} (19)$$
and a probability vector $R$ as

$$R = [r_1, r_2, \cdots, r_{N_{\text{imp}}}], r_i \sim U(0,1)$$ \hspace{1cm} (20)

The difference vector between $P$ and $R$ is

$$D = P - R.$$ \hspace{1cm} (21)

The empire whose relevant index is maximum in $D$ will occupy the weakest colony of the weakest empire.

3.2.4 Convergence. After empire competition, all the empires will collapse, except the most powerful one, and all the colonies will belong to this empire. As a consequence, all the colonies along with the unique imperialist will be located in the same position and have the same cost. In such a condition, the competition is stopped and the algorithm is terminated.

3.3 Quasi-oppositional bare-bone ICA (QBBICA)

To further improve the global searching performance of BBICA, in this paper, quasi-oppositional learning is combined with BBICA. The resulting algorithm is called QBBICA. The difference between BBICA and QBBICA lies in the initialization and assimilation steps.

3.3.1 Quasi-oppositional initialization. In QBBICA, after randomly generating $N$ countries according to Eq (12), another $N$ corresponding quasi-oppositional countries are generated according to the concept of quasi-oppositional point (11). Then, we calculate the fitness values of the $2N$ countries and sort them in an ascending order. The first $N$ countries are selected as the initial population.

3.3.2 Quasi-oppositional assimilation. In QBBICA, after all the colonies in an empire have moved towards their corresponding imperialists according to Eq (15), a set of quasi-oppositional colonies are generated according to Eq (11). Similarly, we calculate the fitness values of new colonies and their corresponding quasi-oppositional colonies, and sort the values in an ascending order. The first $N$ colonies are selected the final colonies in the empire. For complex optimization problems with many local optimum, exploring the information of oppositional point of current location may provide more help to find global optimum. The reasons are as follows. First, the oppositional point maybe result in a better function value, and can speed up the convergence of ICA algorithm. Second, the oppositional direction of current location maybe provide a better descent direction and thus has a large probability to escape the local optimal. Therefore, the oppositional operator can greatly improve the performance of ICA algorithm.

3.3.3 Computation complexity of QBBICA. QBBICA mainly consists of four steps, i.e., initialization, assimilation, revolution and imperialistic competition. For a $D$ dimension problem and QBBICA with $N$ countries and the maximum iteration number $\text{iter}_{\text{max}}$, the computation complexity for generating $N$ initial countries and evaluate its fitness is $O(N^* D + N^* nF)$, where $nF$ is a number of function evaluation. The computation complexity of generating $N$ quasi-oppositional countries and fitness evaluation is $O(N^* D + N^* nF)$. Therefore, in the initialization step, the computation complexity of QBBICA is $2O(N^* D + N^* nF)$. In the iterative process, the computation complexity of assimilation step is, similarly, $2O(\text{iter}_{\text{max}}^* (N^* D + N^* nF))$. The computation complexity of revolution step is $O(\text{iter}_{\text{max}}^* (rr^* N + nF^* rr^* N)$ and that of imperialistic competitive is $O(\text{iter}_{\text{max}}^* (N_{\text{imp}} + N_{\text{imp}}^* N_{\text{col}}))$, where $rr$ is the revolution
rate and \( N_{\text{imp}} \) the number of imperialist, \( N_{\text{col}} \) the number of colonies. Finally, the computation complexity of QBBICA is \( O(\text{iter}_{\text{max}} \times N \times D \times nF) \).

4 Predicting employment rate of graduates using proposed method

In this section, we describe in detail how our proposed method is used to predict the employment rate of graduates. Inspired by the work of Wojciech [40], we divide the prediction process into five steps. First, the historical employment rate series is transformed into a fuzzy time series (FTS). Second, the FTS is used to construct an FCM with the ReLU activation function. Third, the FCM is trained using QBBICA. Then, the trained FCM evolves into one or more steps to obtain a predicted fuzzy state. Finally, the predicted fuzzy state is defuzzified to obtain the numerical predicted values of employment rate.

4.1 Fuzzy time series

Let \( Y = \{ y(t), t = 1, 2, \cdots, n \} \) be the historical time series of employment rate, and \( \mu_{A_i} : \mathbb{R} \rightarrow [0, 1], i = 1, 2, \cdots, m \) a set of membership functions. The membership value of each \( y(t) \) belonging to fuzzy set \( A_i \subseteq \tilde{A} \) is evaluated using \( \mu_{A_i} \). Thus, the original time series \( y(t) \) is transformed into FTS \( F(t) = \{ f_1(t), f_2(t), \cdots, f_m(t) \} \). In this paper, the triangle membership function is adopted.

4.2 Construction of FCM

After the original time series \( y(t) \) is transformed into FTS \( F(t) \), an FCM is constructed. At any time \( t \), there are \( m \) membership values \( f_1(t), f_2(t), \cdots, f_m(t) \) corresponding to \( y(t) \). The \( m \) membership values represent the state of the FCM at time \( t \). Therefore, an FCM with \( m \) nodes is constructed, and the activation value of the \( i \)-th node equals to the membership value \( f_i(t) \) at time \( t \).

4.3 Training of FCM using QBBICA

After the FCM is constructed, the weights of FCM should be determined through training. In the literature, many learning algorithms have been developed for FCM learning, such as Hebbian-based, population-based and hybrid algorithms. See [41] for more details. In this paper, the proposed QBBICA is used to learn the FCM. The main steps are illustrated as follows.

**Step 1** Preparing employment rate data \( \{ y(t), t = 1, 2, \cdots, n \} \). The data is normalized into \([0, 1]\). The first 60\% is used as training samples and the remaining as test samples. The employment rate data is normalized and then transformed into \( m \) fuzzy sets with the triangle membership function. Consequently, the \( m \) FTS \( f_i(t) \) is obtained.

**Step 2** Constructing the FCM. According to the order \( k \) of the FCM and the node number \( m \), an FCM with \( m \) nodes is constructed. The status value of FCM at time \( t \) is \( f_1(t), f_2(t), \cdots, f_m(t) \).

**Step 3** Determining the algorithm parameters of QBBICA such as the number of countries, the number of empires, and the maximum number of iterations.

**Step 4** Solution representation. If the FCM has \( m \) nodes, then each weight matrix is a \( m \times m \) matrix. Furthermore, if the order is \( k \), then there are \( k \) weight matrices. Therefore, a vector with length \( k \times m \times m \) is used to represent each country in QBBICA.

**Step 5** Initialization. In the FCM, each weight lies in \([-1, 1]\). In this step, \( N \) vectors with length \( k \times m \times m \) in \([-1, 1]\) are randomly generated. The \( r \)-th country is denoted as \( \mathbf{x}_r \).
Step 6 Calculating the fitness of each country. To evaluate the power of each country, their fitness is calculated. First, each vector $\mathbf{x}_r$ is reshaped into $k$ matrices with size $m \times m$, which is denoted as $w'_{ji}$. Then, the fitness of each country $x_r$ is defined as:

$$F(x_r) = \frac{1}{ml} \sum_{l=1}^{l} \sum_{j=1}^{k} \sum_{i=1}^{m} (a_i(t) - \hat{a}_i(t))^2,$$

where

$$\hat{a}_i(t) = f \left( \sum_{j=1}^{l} a(t - j + 1) w'_{ji} \right)$$

is the estimated state and $a_i(t)$ is the target state.

Step 7 According the fitness values of the countries, the whole countries are classified into colonies and imperialists and assigned to their corresponding imperialists to form empires.

Step 8 Perform assimilation on each empire as described in subsection 3.2.2.

Step 9 Perform empire competition as described in subsection 3.2.3.

Step 10 Repeat Step 7 and Step 8 until the convergence condition is met.

Step 11 Output the best vector with the minimum fitness value. Reshaping the vector into weight matrices.

4.4 Predicting employment rate using FCM

After the FCM is trained, it is used to predict the employment rate. Suppose that the state value of the FCM at time $t$ is \{a_1(t), a_2(t), \ldots, a_m(t)\}, where each $a_i(t)$ represents the degree of $y(t)$ belonging to the $i$-th fuzzy set, it is a linguistic variable. After the weights of the FCM are determined, the FCM evolves into a step according to Eq (7) to obtain the predicted state values at $t + 1$, i.e., $\hat{a}_i(t + 1)$. The linguistic value $\hat{a}_i(t + 1)$ is transformed into a numerical value using the defuzzification technique. The predicted numerical value $\hat{y}(t + 1)$ is obtained as:

$$\hat{y}(t + 1) = \frac{\sum_{i=1}^{m} \hat{a}_i(t + 1) m p_i}{\sum_{i=1}^{m} a_i(t + 1)},$$

where $m p_i$ is the center of the $i$-th fuzzy set.

5 Results

5.1 Performance of QBBICA

Here, the optimization performance of the proposed QBBICA is validated using several benchmark functions. The selected benchmark functions are listed in Table 1. In those six functions, the Sphere function is unimodal and the others are multimodal. To show the advantage of the proposed QBBICA, we compare it with four other algorithms, i.e., GA, PSO, ICA, and BBICA. The parameter settings of these algorithms are as follows. For GA, the “tournament” selection, the “polynomial mutation” and the “Simulated Binary Crossover (SBX)” are adopted. For PSO, $c_1 = c_2 = 2$, and the inertia weight decreases linearly from $w_{\text{max}} = 0.9$ to $w_{\text{min}} = 0.4$. For DE, the crossover rate $CR = 0.8$, mutation factor $F = 0.8$, and “DE / rand-to-best / 1 / bin” strategy is adopted. The parameters for ICA, BBICA, and the proposed QBBICA are set as in [21],
which are set as follows, the assimilation coefficient $\gamma = 2$, revolution rate equals to 0.3, the number of initial imperialist is 8. For each algorithm, the population size is 50 and the maximum number of iterations is 1000. All the algorithms independently run 30 times.

Table 2 shows the optimization results of benchmark functions with 10 dimension obtained by each algorithm, including the best, the worst and the mean fitness. It can be seen that our proposed QBBICA achieves better optimization results than the other four algorithms. To compare the convergence speeds of these algorithms, Fig 3 shows the fitness convergence curves of the algorithms. It is clear that our proposed QBBICA has a faster convergence speed than the other algorithms. To sum up, our proposed QBBICA not only has a stronger global

Table 1. Benchmark functions.

| No. | Function name | Expression | Bounds | Global minimum |
|-----|---------------|------------|--------|----------------|
| $f_1$ | Sphere | $f(x) = \sum_{i=1}^{D} x_i^2$ | [-5.12, 5.12] | $f(x^*) = 0$ |
| $f_2$ | Rastrigin | $f(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10 \cos(2\pi x_i))$ | [-5.12, 5.12] | $f(x^*) = 0$ |
| $f_3$ | Griewank | $f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} + \prod_{i=1}^{D} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1$ | [-600, 600] | $f(x^*) = 0$ |
| $f_4$ | Ackley | $f(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^{D} \cos \left( 2\pi x_i \right) \right) + 20 + e$ | [-32,32] | $f(x^*) = 0$ |
| $f_5$ | Alpine | $f(x) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i|$ | [-10,10] | $f(x^*) = 0$ |
| $f_6$ | Pathologic | $f(x) = \sum_{i=1}^{D-1} \left( 0.5 + \frac{\sin^{2}(\sqrt{|x_i^2 + x_{i+1}^2|})}{1 + |x_i^2 + x_{i+1}^2|} \right)^{2}$ | [-100,100] | $f(x^*) = 0$ |

Table 2. Results of benchmark function optimization.

|          | GA      | PSO     | DE      | ICA     | BBICA   | QBBICA  |
|----------|---------|---------|---------|---------|---------|---------|
| Sphere   | Best    | 0.1910  | 2.0074E-45 | 1.0428E-14 | 7.4827E-65 | 0       |
|          | Worst   | 0.1910  | 1.1405E-43 | 7.9616E-11 | 1.6881E-56 | 0       |
|          | Mean    | 0.1910  | 2.3425E-44 | 4.6654E-12 | 5.9441E-58 | 0       |
| Rastrigin| Best    | 50.5737 | 8.0129  | 3.4928  | 2.3120E-10 | 1.9899  |
|          | Worst   | 50.5737 | 11.4694 | 9.3382  | 0.9950  | 14.9244 |
|          | Mean    | 50.5737 | 10.0255 | 6.4821  | 0.0664  | 5.9366  |
| Griewank | Best    | 1.7835  | 0.0495  | 0.0140  | 0.0074  | 0.0296  |
|          | Worst   | 1.7835  | 0.7751  | 0.1607  | 0.1205  | 0.3545  |
|          | Mean    | 1.7835  | 0.4382  | 0.0825  | 0.0658  | 0.1233  |
| Ackley   | Best    | 5.8070  | 8.8818E-16 | 4.4409E-15 | 1.15E-06 | 7.9936E-15 | 8.8818E-16 |
|          | Worst   | 5.8070  | 7.9936E-15 | 4.4409E-15 | 0.0001  | 5.0626E-14 | 8.8818E-16 |
|          | Mean    | 5.8070  | 4.0856E-15 | 4.4409E-15 | 1.8754E-05 | 1.6875E-14 | 8.8818E-16 |
| Alpine   | Best    | 1.9266  | 3.7499E-22 | 2.57E-14  | 2.8744E-09 | 1.7347E-18 | 0       |
|          | Worst   | 1.9266  | 0.0082  | 0.0006  | 1.1307E-05 | 4.8017E-14 | 0       |
|          | Mean    | 1.9266  | 0.0021  | 4.8514E-05 | 1.1562E-06 | 8.2664E-15 | 0       |
| Pathologic| Best    | 0.8563  | 0.1374  | 3.8916E-05 | 3.8916E-05 | 3.8916E-05 | 0       |
|          | Worst   | 0.8563  | 0.9631  | 0.7140  | 0.4760  | 0.6371  | 0       |
|          | Mean    | 0.8563  | 0.3071  | 0.1984  | 0.1190  | 0.1706  | 0       |

https://doi.org/10.1371/journal.pone.0272624.t001

https://doi.org/10.1371/journal.pone.0272624.t002
search ability but also a faster convergence speed than the other algorithms. To compare the superiority of the proposed QBBICA, a Wilcoxon sign-rank-sum test had been conducted and the $h$ values of the test are listed in Table 3. The $h$-value equaling to 1 indicates that QBBICA is better than other algorithm. From Table 3, one can see that the proposed QBBICA performs better than other algorithms except for PSO on sphere function. Furthermore, to demonstrate the computational efficiency, the average computation time of each algorithm during the 30 runs for each function are listed in Table 4.

5.2 Predicting employment rate

In this section, experiments are conducted to predict the employment rate of graduates from Liren College, Yanshan University. The employment rate is the ratio of the number of the employed graduates to the total number of graduates. The employment rate data of Liren College from 2005 to 2020 is used in the experiments, which is listed in Table 5.

Table 3. $h$-value of Wilcoxon test results.

| Comparison      | 1 | 1 | 1 | 1 | 1 | 1 |
|-----------------|---|---|---|---|---|---|
| QBBICA vs GA    |   |   |   |   |   | 1 |
| QBBICA vs PSO   | 0 | 1 | 1 | 1 | 1 | 1 |
| QBBICA vs DE    | 1 | 1 | 1 | 1 | 1 | 1 |
| QBBICA vs ICA   | 1 | 1 | 1 | 1 | 1 | 1 |
| QBBICA vs BBICA | 1 | 1 | 1 | 1 | 1 | 1 |

https://doi.org/10.1371/journal.pone.0272624.t003
The first 60% of the employment rate data is used as the training sample and the remaining 40% as the test sample.

In the experiments, five triangle membership functions are adopted to transform the employment rate into an FTS. Thus, there are five nodes in the FCM. The order of the FCM is set to 2 in this paper. Thus, there are two weight matrices and the size of each matrix is $5 \times 5$.

For the purpose of comparison, the RCGA, PSO, ICA, BBICA and the proposed QBBICA are implemented and used to train the ReLU-FCM and the original FCM, denoted as Sig-FCM. For the sake of convenience, the FCMs trained by these algorithms are denoted as RCGA-FCM, PSO-FCM, ICA-FCM, BBICA-FCM, and QBBICA-FCM, respectively. In these algorithms, each individual is represented by a vector with length 50. Each algorithm has 200 individuals, and the maximum number of iterations of each algorithm is 1000. In addition, to reveal its effectiveness in employment rate prediction, the FCM based prediction methods are also compared back propagation neural network, linear regression model and grey GM(1,1) model.

To objectively reflect the performance of the FCMs trained by the different algorithms, root mean square error (RMSE), coefficient of determination $R^2$, coefficient of efficiency $E_{sn}$, and mean absolute error (MAE) are used as the indices for performance evaluation. They are defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}},$$

where $y_i$ and $\hat{y}_i$ are the real and predicted values, respectively, and $N$ is the total number of data points used in the experiment.

Coefficient of determination:

$$R^2 = \frac{\sum_{i=1}^{N} (y_i - \bar{y})(\hat{y}_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^{N} (\hat{y}_i - \bar{y})^2}},$$

where $\bar{y}$ and $\bar{\hat{y}}$ are the mean values of the real and predicted sequences, respectively. The larger the $R^2$ is, the more variability is explained by the model.
Coefficient of efficiency:

\[ E_{sn} = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \]  

(26)

The larger the coefficient of efficiency is, the closer the predicted value is to the real value.

Table 6. Prediction results of ReLU-FCM trained by QBBICA, BBICA, ICA, RCGA, PSO and DE.

| Year | Actual | QBBICA | BBICA | ICA   | PSO   | GA    | DE    |
|------|--------|--------|--------|-------|-------|-------|-------|
| Train |        |        |        |       |       |       |       |
| 2005 | 84.68  | 84.68  | 84.68  | 84.68 | 84.68 | 84.68 | 84.68 |
| 2006 | 86.54  | 84.68  | 84.68  | 84.68 | 84.68 | 84.68 | 84.68 |
| 2007 | 88.32  | 88.3180| 88.3002| 88.4615| 88.1898| 88.3193| 88.2419|
| 2008 | 90.44  | 90.4419| 90.4015| 90.6640| 90.4548| 90.2367| 90.4090|
| 2009 | 88.11  | 88.1115| 88.1306| 88.1458| 88.3276| 88.1520| 88.1529|
| 2010 | 90.21  | 90.2005| 90.2243| 90.1962| 89.9994| 90.2616| 90.2727|
| 2011 | 88.93  | 88.9303| 88.9203| 88.8462| 88.7924| 88.9740| 88.8517|
| 2012 | 90.59  | 90.5905| 90.5807| 90.5989| 90.5021| 90.6107| 90.5244|
| 2013 | 88.59  | 88.5881| 88.5609| 88.5287| 88.4979| 88.5471| 88.5306|
| 2014 | 89.93  | 89.9346| 89.9045| 89.9950| 90.0028| 90.2458| 89.9977|
| 2015 | 90.19  | 90.1926| 90.1865| 90.1888| 90.0216| 90.1386| 90.0717|
| 2016 | 90.19  | 90.1926| 90.1894| 90.2852| 90.3621| 90.5059| 90.2670|
| 2017 | 91.07  | 91.0720| 91.1503| 90.9857| 91.0014| 90.7530| 91.1956|
| Test |        |        |        |       |       |       |       |
| 2018 | 90.83  | 89.9950| 89.4580| 89.3053| 88.4427| 88.9284| 87.4515|
| 2019 | 90.37  | 89.8360| 89.3722| 87.0088| 88.1153| 89.0952| 89.1950|
| 2020 | 95.31  | 92.6969| 89.3722| 88.0088| 88.1153| 89.2952| 91.0132|

Table 7. Prediction results of Sig-FCM trained by QBBICA, BBICA, ICA, RCGA, PSO and DE.

| Year | Actual | QBBICA | BBICA | ICA   | PSO   | GA    | DE    |
|------|--------|--------|--------|-------|-------|-------|-------|
| Train |        |        |        |       |       |       |       |
| 2005 | 84.68  | 84.68  | 84.68  | 84.68 | 84.68 | 84.68 | 84.68 |
| 2006 | 86.54  | 84.68  | 84.68  | 84.68 | 84.68 | 84.68 | 84.68 |
| 2007 | 88.32  | 89.2392| 88.7506| 88.7566| 88.6554| 88.8924| 88.7566|
| 2008 | 90.44  | 89.2882| 88.7260| 88.7380| 88.7466| 88.9968| 88.7380|
| 2009 | 88.11  | 88.2991| 88.4276| 88.4076| 88.5643| 89.0877| 88.4076|
| 2010 | 90.21  | 90.3039| 90.6749| 90.5749| 90.5912| 90.4488| 90.6749|
| 2011 | 88.93  | 88.5155| 88.3467| 88.2967| 88.4324| 89.4263| 88.1967|
| 2012 | 90.59  | 90.5271| 90.6203| 90.6905| 90.6524| 90.0731| 90.8805|
| 2013 | 88.59  | 88.4262| 88.9403| 88.9632| 89.1113| 89.3173| 89.9632|
| 2014 | 89.93  | 90.2199| 90.4105| 90.6213| 90.5154| 90.3009| 90.6213|
| 2015 | 90.19  | 89.4158| 88.8382| 88.8588| 89.2465| 88.8588|
| 2016 | 90.19  | 90.0177| 90.4904| 90.4995| 90.3952| 89.7990| 90.4995|
| 2017 | 91.07  | 91.2610| 90.5363| 90.5827| 90.3869| 89.8654| 90.5827|
| Test |        |        |        |       |       |       |       |
| 2018 | 90.83  | 89.7500| 89.1580| 89.2451| 87.8427| 88.6203| 90.0078|
| 2019 | 90.37  | 89.4668| 89.0722| 87.1088| 88.0153| 89.0450| 90.3982|
| 2020 | 95.31  | 91.8098| 89.1325| 87.2088| 87.1153| 89.1862| 90.4539|
Mean absolute error:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|,$$

(27)
In the training phase, all the algorithms are run independently for 30 times. The employment rate data from 2005 to 2017 is used as the training set and that from 2018 to 2020 as the test set. The individual corresponding to the best fitness value in the 30 runs is taken as the final weight of the FCM. With the obtained weight, numerical prediction values are calculated using Eq (23). The prediction results from the training and test phases for ReLU-FCM and Sig-FCM are listed in Tables 6 and 7, and also, they are shown in Fig 4 for the sake of intuition. The performance indices are listed in Tables 8 and 9, respectively. It can be seen that the ReLU-FCM trained by QBBICA delivers better results than other FCMs in both prediction values and performance indices. The main reason for this is that QBBICA has a stronger search ability, which allows it to better balance exploration and exploitation. Therefore, QBBICA can find more appropriate weights of FCMs. Furthermore, the ReLU activation function has no saturation phenomenon, which enables ReLu-FCM to deliver better performance than Sig-FCM. On the other hand, to illustrate the computational efficiency of QBBICA and other referenced algorithms in FCM training process, the average cpu time in 30 runs is shown in Fig 5. Obviously, one can see that the average cpu time of expanded expended by QBBICA is less than ICA, BBICA, PSO and GA, but is competitive to DE. For a more straightforward comparison, the evolution curves of those algorithms are plotted in Fig 6. From Fig 6, it can be seen that QBBICA not only has a faster convergence speed but also a stronger search ability.

In order to demonstrate the advantage of FMC in prediction, the prediction results obtained by our proposed QBBICA-FCM is compared to those by BP neural network, linear

### Table 8. Performance indices of ReLU-FCM.

|        | QBBICA | BBICA | ICA  | PSO  | GA   | DE   |
|--------|--------|-------|------|------|------|------|
| **Training** |  |  |  |  |  |  |
| RMSE   | 0.0035 | 0.0311| 0.0967| 0.1388| 0.1790| 0.049 |
| \( E_{sn} \) | 1.0000 | 0.9990| 0.9990| 0.9793| 0.9657| 0.9121 |
| \( R^2 \) | 1.0000| 0.9991| 0.9910| 0.9809| 0.9659| 0.9505 |
| MAE    | 0.0025| 0.0228| 0.0741| 0.1248| 0.1278| 0.0193 |
| **Test** |  |  |  |  |  |  |
| RMSE   | 1.6136| 3.5654| 4.7233| 4.5661| 3.7157| 3.2279 |
| \( E_{sn} \) | 0.4756| -1.5603| -3.4934| -3.1991| -1.7807| -1.0985 |
| \( R^2 \) | 0.9988| 0.1808| 0.0001| 0.1808| 0.7216| 0.685  |
| MAE    | 1.3274| 2.7692| 4.0624| 3.9456| 3.0637| 2.9501 |

### Table 9. Performance indices of Sig-FCM.

|        | QBBICA | BBICA | ICA  | PSO  | GA   | DE   |
|--------|--------|-------|------|------|------|------|
| **Training** |  |  |  |  |  |  |
| RMSE   | 0.5367| 0.7590| 0.7653| 0.7638| 0.8033| 0.8856 |
| \( E_{sn} \) | 0.6914| 0.3827| 0.3724| 0.3748| 0.3086| 0.8029 |
| \( R^2 \) | 0.7117| 0.4909| 0.4900| 0.4628| 0.3208| 0.8192 |
| MAE    | 0.4021| 0.5961| 0.6116| 0.6116| 0.7166| 0.6906 |
| **Test** |  |  |  |  |  |  |
| RMSE   | 2.1782| 3.7701| 5.1243| 5.2161| 3.8358| 2.8436 |
| \( E_{sn} \) | 0.0444| -1.8628| -4.2887| -4.4797| -1.9633| -0.6286 |
| \( R^2 \) | 0.9993| 0.0956| 0.1500| 0.9906| 0.3966| 0.2768 |
| MAE    | 1.8278| 3.0491| 4.3158| 4.5122| 3.2195| 1.9022 |
Fig 5. Average CPU time of QBBICA, BBICA, ICA, PSO, GA and DE in FCM training process.
https://doi.org/10.1371/journal.pone.0272624.g005

Fig 6. Fitness evolution curve of QBBICA, BBICA, ICA, PSO, GA and DE.
https://doi.org/10.1371/journal.pone.0272624.g006
regression (LR) model and grey GM(1,1) model, which are listed in Tables 10 and 11. From Table 10, one can see that the prediction value obtained by our QBBICA-FCM is more accurate than those by BP neural network, LR model and GM(1,1) model. Furthermore, the four performance indices of QBBICA-FCM are also better than those of BP, LR and GM(1,1). This proves that our proposed QBBICA-FCM can achieve more accurate prediction results. From the above experimental results and comparisons, one can draw a conclusion that our proposed method can give more accurate prediction results than the other methods.

6 Conclusion

In this paper, an improved FCM is proposed to predict the employment rate of university graduates. Two improvements are made in order to get more accurate prediction results. One is that the sigmoid activation function is replaced with the ReLu function. The other is that an improved ICA algorithm, called quasi-oppositional learning bare-bone ICA (QBBICA), is proposed to train the FCM. The QBBICA algorithm has a stronger search ability and can obtain more appropriate weights of the FCM. Experiments are conducted to predict the employment rate and the results are compared with those obtained by BP, LR and GM(1,1). From the experimental results and comparisons, it can be concluded that the QBBICA-FCM proposed in this paper can give more accurate prediction results than the other methods.

Table 10. Prediction results comparison of QBBICA-FCM between BP, LR and GM(1,1).

| Year | Actual | QBBICA-FCM | BP | LR | GM(1,1) |
|------|--------|------------|----|----|---------|
| Train |
| 2005 | 84.68  | 84.68      | 84.68 | 84.68 | 84.68   |
| 2006 | 86.54  | 84.68      | 84.68 | 84.68 | 88.07329|
| 2007 | 88.32  | 88.3180    | 88.5626 | 88.8343 | 88.3169 |
| 2008 | 90.44  | 90.4419    | 88.6727 | 89.0873 | 88.5613 |
| 2009 | 88.11  | 88.1115    | 88.3511 | 89.2250 | 88.8063 |
| 2010 | 90.21  | 90.2005    | 88.4289 | 90.6067 | 89.0519 |
| 2011 | 88.93  | 88.9303    | 88.7415 | 89.2037 | 89.2983 |
| 2012 | 90.59  | 90.5905    | 90.6362 | 90.3146 | 89.5453 |
| 2013 | 88.59  | 88.5881    | 88.5222 | 89.4168 | 89.7931 |
| 2014 | 89.93  | 89.9346    | 89.9140 | 90.5428 | 90.0415 |
| 2015 | 90.19  | 90.1902    | 90.1587 | 89.4545 | 90.2906 |
| 2016 | 90.19  | 90.1926    | 90.2730 | 89.8932 | 90.5403 |
| 2017 | 91.07  | 91.0720    | 90.3632 | 89.9910 | 90.7908 |
| Test |
| 2018 | 90.83  | 89.9950    | 89.5669 | 89.7703 | 91.0420 |
| 2019 | 90.37  | 89.8360    | 90.3269 | 90.1614 | 91.2938 |
| 2020 | 95.31  | 92.6969    | 90.3726 | 90.1866 | 91.5464 |

Table 11. Performance comparison of QBBICA-FCM and BP, LR and GM(1,1).

|         | QBBICA-FCM | BP      | LR       | GM(1,1) |
|---------|------------|---------|----------|---------|
| Train   |
| RMSE    | 0.5367     | 0.7956  | 0.7686   | 0.7032  |
| $E_{sn}$| 0.6914     | 0.1790  | -0.7247  | 0.6408  |
| $R^2$   | 0.7117     | 0.5003  | 0.3670   | 0.7358  |
| MAE     | 0.4021     | 0.4701  | 0.6799   | 0.6713  |
| Test    |
| RMSE    | 2.1782     | 2.9425  | 3.0230   | 2.2408  |
| $E_{sn}$| 0.0444     | -2.4051 | -2.5973  | -1.4016 |
| $R^2$   | 0.9993     | 0.2212  | 0.2240   | 0.6745  |
| MAE     | 1.8278     | 2.0812  | 2.1306   | 1.6331  |
rate of graduates from Liren College, Yanshan University. Results show that our proposed method can give accurate prediction results.

Author Contributions

Conceptualization: Aihua Guo.

Software: Aihua Guo.

Writing – review & editing: Aihua Guo.

References

1. Wang T. Intelligent employment rate prediction model based on a neural computing framework and human-computer interaction platform. Neural Computing and Applications. 2020; 32(21):16413–16426. https://doi.org/10.1007/s00521-019-04019-w

2. Guo T, Xia F, Zhen S, Bai X, Zhang D, Liu Z, et al. Graduate Employment Prediction with Bias. In: Proceedings of the 34-th AAAI Conference on Artificial Intelligence (AAAI), vol. 34; 2020. p. 670–677.

3. Ma D, Yu Z, Liu Y. Research on application of BP neural network in predicting employment rate of college graduates. Journal of Jiamusi University (Natural Science Edition). 2014; 32(5):751–753.

4. Jiang Z, Yuan Z, Yan R. College students employment forecasting model based on IAFSA-BP parallel integrated learning algorithm. Value Engineering. 2019; 38(19):232–234.

5. Wang Y. Research on University Students’ employment prediction model and Application based on Decision tree algorithm. Central China Normal University; 2018.

6. Pan W, Huang L. Combination forecast model based on B-mode relational degree and GIOWA operator. Statistics & Decision. 2018;(11):73–76.

7. Zhang W, Qianhan H. Prediction of the employment rate of graduates and research of the quality assessment. Computer Engineering & Science. 2009; 31(5):141–143.

8. Kosko B. Fuzzy cognitive maps. International Journal of Man-Machine Studies. 1986; 24(1):65–75. https://doi.org/10.1016/S0020-7373(86)80040-2

9. Kiarni M, Andreu-Perez J, Hagras H, Papageorgiou EI, Prasad M, Lin CT. Effective Brain Connectivity for fNIRS With Fuzzy Cognitive Maps in Neuroergonomics. IEEE Transactions on Cognitive and Developmental Systems. 2022; 14(1):50–63. https://doi.org/10.1109/TCDS.2019.2958423

10. Shen F, Liu J, Wu K. A Preference-Based Evolutionary Biojective Approach for Learning Large-Scale Fuzzy Cognitive Maps: An Application to Gene Regulatory Network Reconstruction. IEEE Transactions on Fuzzy Systems. 2020; 28(6):1035–1049. https://doi.org/10.1109/TFUZZ.2020.2975482

11. Amirkhani A, Kolahdozzi M, Wang C, Kurgan LA. Prediction of DNA-Binding Residues in Local Segments of Protein Sequences with Fuzzy Cognitive Maps. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2020; 17(4):1372–1382. PMID: 30602422

12. Rickard JT, Aisbett J, Morgensthaler DG, Yager RR. Modeling of Complex System Phenomena via Computing With Words in Fuzzy Cognitive Maps. IEEE Transactions on Fuzzy Systems. 2020; 28(12):3122–3132. https://doi.org/10.1109/TFUZZ.2019.2953615

13. Wu K, Liu J, Liu P, Yang S. Time Series Prediction Using Sparse Autoencoder and High-Order Fuzzy Cognitive Maps. IEEE Transactions on Fuzzy Systems. 2020; 28(12):3110–3121. https://doi.org/10.1109/TFUZZ.2019.2956904

14. Lu W, Feng G, Liu X, Pedrycz W, Zhang L, Yang J. Fast and Effective Learning for Fuzzy Cognitive Maps: A Method Based on Solving Constrained Convex Optimization Problems. IEEE Transactions on Fuzzy Systems. 2020; 28(11):2958–2971. https://doi.org/10.1109/TFUZZ.2019.2946119

15. Amirkhani A, Shirzadeh M, Shojaeefard MH, Abraham A. Controlling wheeled mobile robot considering the effects of uncertainty with neuro-fuzzy cognitive map. ISA Transactions. 2020; 100:454–468. https://doi.org/10.1016/j.isatra.2019.12.011 PMID: 31916988

16. Stach W, Pedrycz W, Kurgan LA. Learning of Fuzzy Cognitive Maps Using Density Estimate. IEEE Transactions on Systems Man & Cybernetics, Part B, Cybernetics. 2012; 42:900–912. https://doi.org/10.1109/TSMCB.2011.2182646 PMID: 22345544

17. Parsopoulos KE, Papageorgiou EI, Groumpos PP, Vrahatis MN. A first study of fuzzy cognitive maps learning using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, 2003. CEC’03. vol. 2; 2003. p. 1440–1447 Vol.2.
18. Ighravwe DE, Mashao D. Development of a Differential Evolution-based Fuzzy Cognitive Maps for Data Breach in Health-care Sector Fuzzy Cognitive Maps for Data Breach. In: 2019 IEEE AFRICON; 2019. p. 1–5.

19. Salmeron JL, Mansouri T, Moghadam MRS, Mardani A. Learning Fuzzy Cognitive Maps with modified asexual reproduction optimisation algorithm. Knowledge-Based Systems. 2019; 163:723–735. https://doi.org/10.1016/j.knosys.2019.09.034

20. Ahmadi S, Forouzideh N, Alizadeh S, Papageorgiou EI. Learning Fuzzy Cognitive Maps using Imperialist Competitive Algorithm. Neural Comput Appl. 2015; 26(6):1333–1354. https://doi.org/10.1007/s00521-014-1797-4

21. Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. 2007 IEEE Congress on Evolutionary Computation, CEC 2007. 2007; p. 4661–4667.

22. Li M, Lei D. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. Engineering Applications of Artificial Intelligence. 2021; 103:104307. https://doi.org/10.1016/j.engappai.2021.104307

23. Ahmadi N, Akbarizadeh G. Iris tissue recognition based on GLDM feature extraction and hybrid MLPNN-ICA classifier. Neural Computing and Applications. 2020; 32:2267–2281. https://doi.org/10.1007/s00521-018-3754-0

24. Yousefi M, Darus AN, Mohammad H. An imperialist competitive algorithm for optimal design of plate-fin heat exchangers. International Journal of Heat and Mass Transfer. 2012; 55(11-12):3178–3185. https://doi.org/10.1016/j.ijheatmasstransfer.2012.02.041

25. Elsisi M. Design of neural network predictive controller based on imperialist competitive algorithm for automatic voltage regulator. Neural Computing and Applications. 2019; 31:5017–5027. https://doi.org/10.1007/s00521-018-03995-9

26. Bankhoda W, Sheikh H. Immigrant imperialist competitive algorithm to solve the multi-constraint node placement problem in target-based wireless sensor networks. Ad Hoc Networks. 2020; 106:102183. https://doi.org/10.1016/j.adhoc.2020.102183

27. Afonso LD, Mariani VC, dos Santos Coelho L. Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization. Expert Systems with Applications. 2013; 40(9):3794–3802. https://doi.org/10.1016/j.eswa.2012.12.043

28. Rabiee A, Sadeghi M, Aghaei J. Modified imperialist competitive algorithm for environmental constrained energy management of microgrids. Journal of Cleaner Production. 2018; 202:273–292. https://doi.org/10.1016/j.jclepro.2018.08.129

29. Idoumghar L, Chérin N, Siarry P, Roche R, Mirouafi A. Hybrid ICA–PSO algorithm for continuous optimization. Applied Mathematics and Computation. 2013; 219(24):11149–11170. https://doi.org/10.1016/j.amc.2013.05.027

30. Chen CH, Chen WH. Bare-bones imperialist competitive algorithm for a compensatory neural fuzzy controller. Neurocomputing. 2016; 173:1519–1528. https://doi.org/10.1016/j.neucom.2015.09.025

31. Yang SD, Yi YL, Shan ZY. Fbest-guided Imperialist competitive algorithm for global numerical optimization. Proceedings—2012 International Conference on Computer Distributed Control and Intelligent Environmental Monitoring, CDICIEM 2012. 2012; p. 352–355.

32. Kaveh A, Talatahari S. Optimum design of skeletal structures using imperialist competitive algorithm. Computers & Structures. 2010; 88(21):1220–1229. https://doi.org/10.1016/j.compstruc.2010.06.011

33. Mortazavi A, Khanseh AA, Naderi B. A novel chaotic imperialist competitive algorithm for production and air transportation scheduling problems. Neural Computing and Applications. 2015; 26(7):1709–1723. https://doi.org/10.1007/s00521-015-1828-9

34. Goldansaz SM, Jolai F, Anaraki AHZ. A hybrid imperialist competitive algorithm for minimizing make-span in a multi-processor open shop. Applied Mathematical Modelling. 2013; 37(23):9603–9616. https://doi.org/10.1016/j.apm.2013.05.002

35. Yin Z, Gao Q. A Novel Imperialist Competitive Algorithm for Scheme Configuration Rules Extraction of Product Service System. Procedia CIRP. 2019; 80:762–767. https://doi.org/10.1016/j.procir.2019.01.103

36. Peri D. Hybridization of the imperialist competitive algorithm and local search with application to ship design optimization. Computers & Industrial Engineering. 2019; 137:106069. https://doi.org/10.1016/j.cie.2019.106069

37. Zandieh M, Khatami AR, Rahmati SHA. Flexible job shop scheduling under condition-based maintenance: Improved version of imperialist competitive algorithm. Applied Soft Computing. 2017; 58:449–464. https://doi.org/10.1016/j.asoc.2017.04.060
38. Moazzami M, Morshed MJ, Fekih A. A new optimal unified power flow controller placement and load shedding coordination approach using the Hybrid Imperialist Competitive Algorithm-Pattern Search method for voltage collapse prevention in power system. International Journal of Electrical Power & Energy Systems. 2016; 79:263–274. https://doi.org/10.1016/j.ijepes.2016.01.022

39. Tizhoosh HR. Opposition-Based Learning: A New Scheme for Machine Intelligence. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06). vol. 1; 2005. p. 695–701.

40. Stach W, Kurgan LA, Pedrycz W. Numerical and Linguistic Prediction of Time Series With the Use of Fuzzy Cognitive Maps. IEEE Transactions on Fuzzy Systems. 2008; 16(1):61–72. https://doi.org/10.1109/TFUZZ.2007.902020

41. Papageorgiou EI. Learning Algorithms for Fuzzy Cognitive Maps—A Review Study. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2012; 42(2):150–163. https://doi.org/10.1109/TSMCC.2011.2138694