Knowledge Discovery by an Intelligent Approach Using Complex Fuzzy Sets

Chunshien Li and Feng-Tse Chan

Laboratory of Intelligent Systems and Applications
Department of Information Management, National Central University, Taiwan (R.O.C.)
jamesli@mgt.ncu.edu.tw

Abstract. In the age of rapidly increasing volumes of data, human experts have come to the urgent need to extract useful information from the huge amount of data. Knowldege discovery in databases has obtained much attention for researches and applications in business and in science. In this paper, we present a neuro-fuzzy approach using complex fuzzy sets (CFSs) for the problem of knowledge discovery. A CFS is an advanced fuzzy set, whose membership is complex-valued and characterized by an amplitude function and a phase function. The application of CFSs to the proposed complex neuro-fuzzy system (CNFS) can increase the functional mapping ability to find missing data for knowledge discovery. Moreover, we devise a hybrid learning algorithm to evolve the CNFS for modeling accuracy, combining the artificial bee colony algorithm and the recursive least squares estimator method. The proposed approach to knowledge discovery is tested through experimentation, whose results are compared with those by other approaches. The experimental results indicate that the proposed approach outperforms the compared approaches.

Keywords: complex fuzzy set (CFS), complex neuro-fuzzy system (CNFS), knowledge discovery, artificial bee colony (ABC), recursive least squares estimator (RLSE).

1 Introduction

In the era of the modern information world, data have been accumulating exponentially in various forms. We now have been facing the urgent need to develop new theories and tools to extract useful information from data automatically and effectively. If it is understandable and interpretable by human beings, information can be turned into knowledge. For knowledge discovery and system modeling, several artificial intelligence (AI) based approaches have been presented, where fuzzy logic, neural networks, and neuro-fuzzy systems (NFSs) have been playing significantly important roles for modeling and applications. In general, fuzzy-system-based approaches have the advantage of representing knowledge in the form of If-Then rules, which are transparent to human beings. In contrast, neural-network-based approaches are with the merit of adaptability. The hybrid of fuzzy and neural models has been obtaining the popularity in the community of research, although each of

J.-S. Pan, S.-M. Chen, N.T. Nguyen (Eds.): ACIIDS 2012, Part I, LNAI 7196, pp. 320–329, 2012.

AI-based approaches still has its own important feature in the perspective of research. Zhang et al. [1] used granular neural networks for data fusion and knowledge discovery. Fayyad et al. [2] presented a good overview for knowledge discovery, where the various methods of data mining provide just a single step for the whole process of information discovery. Castellano et al. [3] presented a neuro-fuzzy modeling framework for knowledge discovery. Qin et al. [4] proposed a kernel-based imputation to deal with missing data in knowledge discovery. In [5], Zhang et al. presented a fuzzy modeling approach for training data selection and multi-object optimization. In [6], Rezaee and Zarandi developed a data-driven TSK fuzzy approach for fuzzy modeling. Juang [7] presented the design of recurrent neural network using a hybrid method, which uses genetic algorithm and particle swarm optimization for modeling. Kurban and Beşdok [8] investigated several training methods for a RBF neural network in the application of terrain classification. Boskovitz and Guterman [9] used a neuro-fuzzy system for image segmentation and edge detection. Cpałka [10] designed a neuro-fuzzy classification system. Jang [11] presented the famous adaptive neural-network-based fuzzy inference system (ANFIS), which molds the hybrid of a neural network and a fuzzy inference system, for system modeling and forecasting. Scherer in [12] used a neuro-fuzzy system for nonlinear modeling. In general, neural fuzzy systems [13] are excellent tools for modeling and for knowledge discovery. Qin and Yang [14] studied a neuro-fuzzy method for image noise cancellation. In [15], Zounemat-Kermani and Teshnehlab presented a neuro-fuzzy approach to time series forecasting.

In this paper, we present a framework called the complex neuro-fuzzy system (CNFS) for knowledge discovery. A CNFS is a neuro-fuzzy based system whose kernel is composed of fuzzy If-Then rules which are characterized by complex fuzzy sets (CFSs). Ramot at al. [16] proposed the theory of CFS. A CFS is an advanced fuzzy set whose membership degree is complex-valued and characterized by an amplitude function and a phase function in the unit disc of the complex plane. The property of complex membership state of a CFS indeed makes difference from a traditional type-1 fuzzy set, whose membership degree is defined normally in the real-valued unit interval of [0, 1]. The application of CFSs [17]-[19] to the design of adaptive systems can increase their adaptability for learning, so that the functional mapping capability of the adaptive systems can be augmented. For this motivation, we proposed the CNFS approach using CFSs for the process of knowledge discovery. For knowledge discovery, the goal of the study is that with the proposed CNFS approach we try to represent numerical-linguistic records [1] by rules. In the perspective of If-Then rules, we apply the proposed CNFS models to create a framework of knowledge discovery, which can transform these numerical-linguistic records into fuzzy If-Then rules. And, in the principle of divide-and-conquer, the parameters of CNFS are imaginatively separated into two groups: the premise parameters and the consequent parameters. For the parameter learning of CNFS, we devise a hybrid ABC-RLSE learning method, which integrates the famous artificial bee colony (ABC) algorithm [20]-[22] and the well-known recursive least squares estimator (RLSE) method [13]. The ABC-RLSE algorithm is applied to evolve the parameters of CNFS, in the way that the ABC is used to update the premise parameters and the RLSE is used to update the consequent parameters. The ABC-RLSE can achieve fast training for the proposed approach to the application of knowledge discovery.

We arrange the rest of the paper in the following. In Section 2, we present the knowledge discovery framework. In Section 3, the proposed CNFS and the ABC-RLSE learning method for knowledge discovery are specified. In Section 4, Experimentation is conducted to test the proposed approach for knowledge discovery. The experimental results are compared to other approaches. Finally, the paper is concluded.

2 Framework for Knowledge Discovery

In this paper, for knowledge discovery, we adopt and improve the framework in [1]. The improved framework has three parts, including the feature extraction for numerical-linguistic records from a dataset, the CNFS modeling with the extracted records, and the linguistic records output. The framework is shown in Fig. 1.

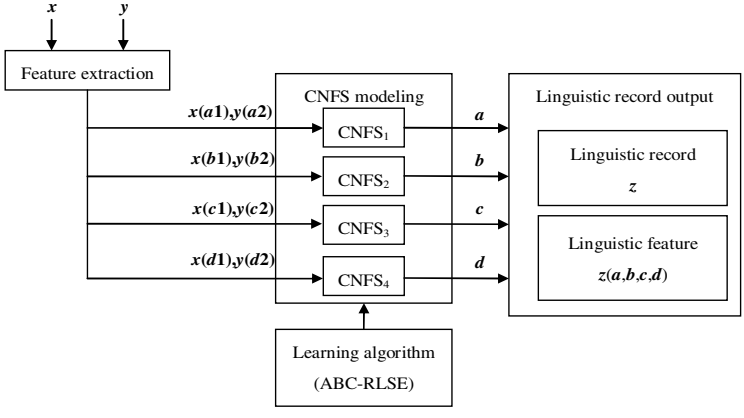

Fig. 1. CNFS-based framework for knowledge discovery

(A) *Feature Extraction*. The block of feature extraction for numerical-linguistic records, shown in Fig. 1, is used to extract feature for each datum. By the concept of trapezoidal feature vector in [1], we can have a feature vector (w, w/10, w/2, w/2) for a datum w. For example, in Fig. 1, there are two inputs, x and y, to the block of feature extraction. For the first input, x, the trapezoidal feature vector is denoted as ($x(a1)$, $x(b1)$, $x(c1)$, $x(d1)$)= (x, x/10, x/2, x/2). Similarly, the feature vector for the second input, y, is denoted as ($y(a2)$, $y(b2)$, $y(c2)$, $y(d2)$)= (y, y/10, y/2, y/2).

(B) *CNFS Modeling*. As shown in Fig. 1, the block of CNFS modeling consists of four CNFS models, each of which is composed of a set of fuzzy If-Then rules. For the first CNFS (denoted as $CNFS_1$), the first components of the feature vectors for x and y, which are the $x(a1)$ and the $y(a2)$, are used as two inputs to $CNFS_1$, whose output is then used as the first component of the feature vector for z. Similar operations can be applied to the others {$CNFS_i$, i=2,3,4}. The goal for the block of CNFS modeling is to produce a feature vector for z, which is the output of the system for knowledge

discovery. To optimize these CNFS models, the ABC-RLSE hybrid learning method is applied. The optimization is based on the measure of root mean square error (RMSE). Note that the details for theory of CNFS and the ABC-RLSE learning method are specified later in the following section.

(C) *Linguistic Record Output.* The block of linguistic record output has two subparts. One is to generate the linguistic record z, and the other is to produce the feature vector, denoted as (a,b,c,d), for z.

3 Methodology

Based on the concept of CFS, the membership function of a CFS is complex-valued and characterized within the unit disc of the complex plan. The general form for the complex-valued membership function of a CFS is given below.

$$\begin{aligned}\mu_A(h) &= r_A(h)\exp\left(j\omega_A(h)\right)\\ &= \mathrm{Re}\left(\mu_A(h)\right) + j\mathrm{Im}\left(\mu_A(h)\right)\\ &= r_A(h)\cos\left(\omega_A(h)\right) + jr_A(h)\sin\left(\omega_A(h)\right),\end{aligned} \tag{1}$$

where $\mu_A(h)$ is the membership function; $j=\sqrt{-1}$; A denotes the CFS; h is the input base variable; $r_A(h)$is the amplitude function; $\omega_A(h)$is the phase function. We devise a Gaussian complex fuzzy set (GCFS) for the design of the premises of CNFS. The general form of a GCFS is characterized below.

$$\mathrm{GCFS}(h,m,\sigma) = \exp\left[-\frac{(h-m)^2}{2\sigma^2}\right] + j\frac{-(h-m)}{\sigma^2}\exp\left[-\frac{(h-m)^2}{2\sigma^2}\right], \tag{2}$$

where $j=\sqrt{-1}$; m and σ are the mean and the spread of the GCFS, respectively.

Suppose that we have a CNFS that consists of K first-order Takagi-Sugeno (T-S) fuzzy rules, given as follows.

$$\begin{aligned}&\text{Rule } i:\ \text{IF } \left(x_1 \text{ is } A_1^{(i)}(h_1)\right) \text{ and } \left(x_2 \text{ is } A_2^{(i)}(h_2)\right) \cdots \text{and } \left(x_M \text{ is } A_M^{(i)}(h_M)\right)\\ &\qquad\text{Then } z^{(i)} = a_0^{(i)} + \textstyle\sum_{j=1}^{M} a_j^{(i)} h_j,\end{aligned} \tag{3}$$

for i =1,2,...,K, where i is the index for the ith fuzzy rule; M is the number of inputs; x_j is the jth input linguistic variable for j=1,2,…,M; h_j is the jth input base variable; $A_j^{(i)}(h_j)$ is the jth premise, which is defined by a GCFS in (2); $z^{(i)}$ is the nominal output; $\{a_j^{(i)}$, j=0,1,…,$M\}$ are consequent parameters. The complex fuzzy inference process can be cast into a neural-net structure to become a CNFS with six layers, specified below.

Layer 0: This layer receives the inputs and transmits them to Layer 1. The input vector at time t is given as follows.

$$H(t) = [h_1(\mathrm{t})\ h_2(\mathrm{t}) \cdots h_M(t)]^{\mathrm{T}}. \tag{4}$$

Layer 1: This layer is called the complex fuzzy-set layer, which is used to calculate complex membership degrees of GCFSs. **Layer 2:** This layer is called the firing-strength layer. The firing strength of the *i*th rule is calculated and defined as fellow.

$$\beta^{(i)}(t) = \prod_{j=1}^{M} r_j^{(i)}\big(h_j(t)\big) \exp\left(j\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}\right), \tag{5}$$

where $r_j^{(i)}$ and $\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}$ are the amplitude function and the phase function of the firing strength of the *i*th rule. **Layer 3:** This layer is used for the normalization of the firing strengths. The normalized firing strength for the *i*th rule is given as follows.

$$\lambda^{(i)}(t) = \frac{\beta^{(i)}(t)}{\sum_{i=1}^{K} \beta^{(i)}(t)} = \frac{\prod_{j=1}^{M} r_j^{(i)}(h_j(t)) \exp\left(j\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}\right)}{\sum_{i=1}^{K} \prod_{j=1}^{M} r_j^{(i)}(h_j(t)) \exp\left(j\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}\right)}. \tag{6}$$

Layer 4: The layer is called the consequent layer. The normalized consequent of the *i*th rule is represented as follows.

$$\begin{aligned} \xi^{(i)}(t) &= \lambda^{(i)}(t)\ z^{(i)}(t) = \lambda^{(i)}(t)\big(a_0^{(i)} + \textstyle\sum_{j=1}^{M} a_j^{(i)} h_j(t)\big), \\ &= \frac{\prod_{j=1}^{M} r_j^{(i)}(h_j(t)) \exp\left(j\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}\right)}{\sum_{i=1}^{K} \prod_{j=1}^{M} r_j^{(i)}(h_j(t)) \exp\left(j\omega_{A_1^{(i)} \cap \ldots \cap A_M^{(i)}}\right)} \big(a_0^{(i)} + \textstyle\sum_{j=1}^{M} a_j^{(i)} h_j(t)\big). \end{aligned} \tag{7}$$

Layer 5: This layer is called the output layer. The normalized consequents from Layer 4 are congregated into the layer to produce the CNFS output, given as follows.

$$\xi(t) = \sum_{i=1}^{K} \xi^{(i)}(t) = \sum_{i=1}^{K} \lambda^{(i)}(t)\ z^{(i)}(t). \tag{8}$$

The complex-valued output of the CNFS can be expressed as follows.

$$\xi(t) = \xi_{\mathrm{Re}}(t) + j\xi_{\mathrm{Im}}(t) = |\xi(t)|\cos\big(j\omega_{\xi}\big) + j|\xi(t)|\sin\big(j\omega_{\xi}\big), \tag{9}$$

where $\xi_{\mathrm{Re}}(t)$ is the real part of the output, and $\xi_{\mathrm{Im}}(t)$ is the imaginary part. Based on (8), the CNFS can be viewed as a complex-valued function, expressed as follows.

$$\xi(t) = \mathrm{F}(\mathbf{H}(t), \mathbf{W}) = \mathrm{F}_{\mathrm{Re}}(\mathbf{H}(t), \mathbf{W}) + j\mathrm{F}_{\mathrm{Im}}(\mathbf{H}(t), \mathbf{W}), \tag{10}$$

where $\mathrm{F}_{\mathrm{Re}}(.)$ is the real part of the CNFS output; $\mathrm{F}_{\mathrm{Im}}(.)$ is the imaginary part; $\mathbf{H}(t)$ is the input vector to the CNFS; $\mathbf{W}$ denotes the parameter set of the CNFS, including the premise parameters and the consequent parameters, denoted as $\mathbf{W}_{\mathrm{If}}$ and $\mathbf{W}_{\mathrm{Then}}$, respectively.

$$\mathbf{W} = \mathbf{W}_{\mathrm{If}} \cup \mathbf{W}_{\mathrm{Then}}. \tag{11}$$

For the training of the proposed CNFS, we apply the ABC-RLSE hybrid learning method, where $\mathbf{W}_{\text{If}}$ and $\mathbf{W}_{\text{Then}}$ are updated by the ABC and RLSE, respectively, in a hybrid way.

The artificial bee colony (ABC) algorithm is an optimization method which simulates the nectar-searching behavior by bees. Basically, the bees of a bee colony can be separated into three groups: the employed bees (EBs), the onlooker bees (OBs), and the scout bees (SBs). The EBs perform their job by flying to the food sources and bringing back food to the colony. Moreover, these EBs reveal the locations of food sources to the OBs by dancing. With the dancing information, each of the OBs selects a food source to go. The selection of a food source is dependent on the nectar amount of each food source. For the SBs, they fly randomly to look for new food sources. Note that the nectar amount of each food source corresponds to a fitness value for a specific optimization problem and the food source location represents a candidate solution to the optimization problem. For the ABC algorithm, assume there are S food sources. The location of the ith food source is expressed as $\mathbf{X}_i$=[x_{i1}, x_{i2}, …, x_{iQ}] for i=1,2,…,S. In the ABC algorithm, the location of the ith food source is updated by the following equation.

$$x_{ij}(t+1) = x_{ij}(t) + \varphi_{ij}\left(x_{ij}(t) - x_{kj}(t)\right), \tag{12}$$

for i=1,2,…,S and j=1,2,…,Q, where $x_{ij}(t)$ is the jth dimensional coordinate of the ith food source at iteration t, k is a random integer in the set of {1,2,…,S} with the constraint of $i \neq k$, and φ_{ij} is a value between [-1, 1]. An onlooker bee goes to the vicinity of X_i with the probability given below.

$$p_i = \frac{F_{fitness}(\mathbf{X}_i)}{\sum_{j=1}^{S} F_{fitness}(\mathbf{X}_j)}, \tag{13}$$

where $F_{fitness}(.)$ is the fitness function. In the ABC, if the fitness of a food source is not improved further through a predetermined number of cycles, called *limit*, then that food source is assumed to be abandoned. The operation of the ABC is specified in steps. **Step 1:** initialize the locations of bees. **Step 2:** send employed bees to food sources and compute their fitness values. **Step 3:** send onlooker bees to the food sources, according to (13), and then compute their fitness values. **Step 4:** send scout bees for other food sources to help find better food source as possible. **Step 5:** update the locations of food sources and save the best one so far. **Step 6:** if any termination condition is met, stop; otherwise increase the iteration index and go back to **Step 2** to continue the procedure.

The recursive least squares estimator (RLSE) is a recursive method for the problem of least squares estimation (LSE). The model of LSE is given below.

$$y = \theta_1 f_1(u) + \theta_2 f_2(u) + \cdots + \theta_m f_m(u) + \varepsilon, \tag{14}$$

where y is the target; u is the input to model; {$f_i(u)$, i=1,2,..,m} are known functions of u; {θ_i , i=1,2,…,m} are unknown parameters to be estimated; ε is the model error. Note that the parameters {θ_i, i=1,2,…,m} can be viewed as the consequent parameters

of the proposed CNFS. The observed samples are collected to be used as training data for the proposed CNFS. The training data (TD) is denoted as follows.

$$\text{TD} = \{(u_i, y_i), i = 1,2, \dots, N\}, \tag{15}$$

where (u_i, y_i) is the ith data pair in the form of (*input*, *target*). With (15), the LSE model can be expressed in matrix form, $\mathbf{y}=\mathbf{A}\boldsymbol{\theta}+\boldsymbol{\varepsilon}$, where $\mathbf{y} = [y_1\ y_2 \dots y_N]^{\mathrm{T}}$; $\boldsymbol{\theta} = [\theta_1\ \theta_2 \dots \theta_m]^{\mathrm{T}}$; $\boldsymbol{\varepsilon} = [\varepsilon_1\ \varepsilon_2 \dots \varepsilon_N]^{\mathrm{T}}$; $\mathbf{A}$ is the matrix which is formed by the known functions $\{f_i(u_j), i=1,2,..,m\}$ for j=1,2,…,N.

The optimal estimation for $\boldsymbol{\theta}$ can be calculated using the following RLSE equations recursively.

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{b}_{k+1} \mathbf{b}_{k+1}^T \mathbf{P}_k}{1+\mathbf{b}_{k+1}^T \mathbf{P}_k \mathbf{b}_{k+1}}, \tag{16}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{P}_{k+1}\mathbf{b}_{k+1}(y_{k+1} - \mathbf{b}_{k+1}^T \boldsymbol{\theta}_k), \tag{17}$$

for k = 0,1,2,…,(N-1), where $\boldsymbol{\theta}_k$ is the estimator at the kth iteration; $\left[\mathbf{b}_{k+1}^{\mathrm{T}}, y_{k+1}\right]$ is the (k+1)th row of [**A, y**]. To implement the RLSE, we initialize $\boldsymbol{\theta}_0$ with the zero vector and $\mathbf{P}_0=\alpha\mathbf{I}$, where α is a large positive number and $\mathbf{I}$ is the identity matrix.

4 Experimentation

The target of this experiment is to find the unknown records and represents the unknown records by *IF-Then* rules. We use the trapezoidal feature vector to represent a linguistic record [1]. Suppose we have a data set for (x, y, z) that includes numerical and linguistic data, as shown in Table 1, in which x and y are the two variables for inputs; z is the variable for output linguistic record; $\tilde{d}$ denotes a input linguistic record, meaning it is around the value d; "?" denotes that the output linguistic record is missing. For the linguistic record of x, there is a feature vector by ($x(a1)$, $x(b1)$, $x(c1)$, $x(d1)$). Similarly, for the linguistic record of y, there is a feature vector by ($y(a2)$, $y(b2)$, $y(c2)$, $y(d2)$). For the record of z, a multiplication z= xy is implied, whose feature vector for linguistic representation is represented by (a, b, c, d). There are nine fuzzy If-Then rules to represent these known records, shown in the Table 2.

Table 1. Known and unknown numerical-linguistic records

		y				
		$\tilde{1}$	$\widetilde{1.5}$	$\tilde{2}$	2.5	$\tilde{3}$
x	$\tilde{1}$	$\tilde{1}$	?	$\tilde{2}$	?	$\tilde{3}$
	1.5	?	?	?	?	?
	$\tilde{2}$	$\tilde{2}$	?	$\tilde{4}$	?	$\tilde{6}$
	$\widetilde{2.5}$	?	?	?	?	?
	$\tilde{3}$	$\tilde{3}$	?	$\tilde{6}$	?	$\tilde{9}$

$\tilde{d}$: linguistic record around a numerical value d.

? : unknown record.

Table 2. Nine fuzzy rules of known linguistic records

x, ($x(a1)$, $x(b1)$, $x(c1)$, $x(d1)$)	y, ($y(a2)$, $y(b2)$, $y(c2)$, $y(d2)$)	$z = xy$, (a, b, c, d)
around 1, (1, 0.1, 0.5, 0.5)	*around* 1, (1, 0.1, 0.5, 0.5)	*around* 1, (1, 0.1, 0.5, 0.5)
around 1, (1, 0.1, 0.5, 0.5)	*around* 2, (2, 0.2, 1.0, 1.0)	*around* 2, (2, 0.2, 1.0, 1.0)
around 1, (1, 0.1, 0.5, 0.5)	*around* 3, (3, 0.3, 1.5, 1.5)	*around* 3, (3, 0.3, 1.5, 1.5)
around 2, (2, 0.2, 1.0, 1.0)	*around* 1, (1, 0.1, 0.5, 0.5)	*around* 2, (2, 0.2, 1.0, 1.0)
around 2, (2, 0.2, 1.0, 1.0)	*around* 2, (2, 0.2, 1.0, 1.0)	*around* 4, (4, 0.4, 2.0, 2.0)
around 2, (2, 0.2, 1.0, 1.0)	*around* 3, (3, 0.3, 1.5, 1.5)	*around* 6, (6, 0.6, 3.0, 3.0)
around 3, (3, 0.3, 1.5, 1.5)	*around* 1, (1, 0.1, 0.5, 0.5)	*around* 3, (3, 0.3, 1.5, 1.5)
around 3, (3, 0.3, 1.5, 1.5)	*around* 2, (2, 0.2, 1.0, 1.0)	*around* 6, (6, 0.6, 3.0, 3.0)
around 3, (3, 0.3, 1.5, 1.5)	*around* 3, (3, 0.3, 1.5, 1.5)	*around* 9, (9, 0.9, 4.5, 4.5)

In order to discover the 16 unknown linguistic records for z in Table 1, we use four CNFS models to discover their feature vectors. The framework of knowledge discovery is shown in Fig. 1. The feature vectors of the known (x, y, z) linguistic record pairs are used as the training data for the four CNFS models, for which the ABC-RLSE method is applied for the training purpose. The settings for the ABC-RLSE method are shown in Table 3.

Table 3. Settings of the ABC-RLSE method

Settings of ABC		Settings of RLSE	
Bee colony dimensions	20	Number of consequent parameters	75
Bee colony size	4	θ	75×1
Scout bee	1	A	10^8
Maximal iterations	200	I (identity matrix)	75×1

After learning, the CNFS models can generate feature vectors for the sixteen unknown linguistic records, respectively, as shown in Table 4. The discovered linguistic records are represented by *If-Then* rules, as shown in Table 5. The results are compared to other approaches. Performance comparison in the measure of average absolute error (ABE) [1] is given in Table 6.

Table 4. New linguistic records discovered by the proposed approach

	$\tilde{1}$	$\widetilde{1.5}$	$\tilde{2}$	2.5	$\tilde{3}$
$\tilde{1}$	$\tilde{1}$	$\widetilde{1.4940}$	$\tilde{2}$	$\widetilde{2.5067}$	$\tilde{3}$
1.5	$\widetilde{1.4888}$	$\widetilde{2.2241}$	$\widetilde{2.9770}$	$\widetilde{3.7311}$	$\widetilde{4.4649}$
$\tilde{2}$	$\tilde{2}$	$\widetilde{2.9880}$	$\tilde{4}$	$\widetilde{5.0136}$	$\tilde{6}$
$\widetilde{2.5}$	$\widetilde{2.5114}$	$\widetilde{3.7523}$	$\widetilde{5.0235}$	$\widetilde{6.2969}$	$\widetilde{7.5359}$
$\tilde{3}$	$\tilde{3}$	$\widetilde{4.4818}$	$\tilde{6}$	$\widetilde{7.5207}$	$\tilde{9}$

Table 5. New 16 rules discovered by the proposed approach

x	*y*	*z*, (*a*, *b*, *c*, *d*)
around 1	*around* 1.5	*around* 1.4940, (1.4940, 0.1499, 0.7498, 0.7542)
around 1	2.5	*around* 2.5067, (2.5067, 0.2500, 1.2501, 1.2409)
1.5	*around* 1	*around* 1.4888, (1.4888, 0.1499, 0.7489, 0.7482)
1.5	*around* 1.5	*around* 2.2241, (2.2241, 0.2249, 1.1231, 1.1288)
1.5	*around* 2	*around* 2.9970, (2.9970, 0.3000, 1.4978, 1.4965)
1.5	2.5	*around* 3.7311, (3.7311, 0.3750, 1.8724, 1.8570)
1.5	*around* 3	*around* 4.4649, (4.4649, 0.4500, 2.2467, 2.2448)
around 2	*around* 1.5	*around* 2.9880, (2.9880, 0.2999, 1.4997, 1.5085)
around 2	2.5	*around* 5.0136, (5.0136, 0.5000, 2.5002, 2.4819)
around 2.5	*around* 1	*around* 2.5114, (2.5114, 0.2500, 1.2511, 1.2521)
around 2.5	*around* 1.5	*around* 3.7523, (3.7523, 0.3750, 1.8764, 1.8888)
around 2.5	*around* 2	*around* 5.0235, (5.0235, 0.4999, 2.5023, 2.5043)
around 2.5	2.5	*around* 6.2969, (6.2969, 0.6249, 3.1282, 3.1079)
around 2.5	*around* 3	*around* 7.5359, (7.5359, 0.7499, 3.7535, 3.7564)
around 3	*around* 1.5	*around* 4.4818, (4.4818, 0.4500, 2.2496, 2.2626)
around 3	2.5	*around* 7.5207, (7.5207, 0.7499, 3.7503, 3.7231)

Table 6. Performance comparison

	ABE	
Method	Training phase	Testing phase
CGNN [1]	0.0001	0.303
FGNN [1]	0.0001	0.224
Proposed	5.58×10^{-8}	0.0194

5 Conclusion

The complex neuro-fuzzy system (CNFS) based approach using complex fuzzy sets for knowledge discovery has been presented. The ABC-RLSE hybrid learning method has been devised for training the proposed CNFS models in the framework of knowledge discovery. Through experimentation, the proposed approach has shown excellent performance in finding missing data for knowledge discovery. The experimental results indicate that our proposed approach outperforms the compared methods in performance comparison.

References

1. Zhang, Y.Q., Fraser, M.D., Gagliano, R.A., Kandel, A.: Granular neural networks for numerical-linguistic data fusion and knowledge discovery. IEEE Transactions on Neural Networks 11, 658–667 (2000)
2. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The KDD process for extracting useful knowledge from volumes of data. Commun. ACM 39(11), 27–34 (1996)

3. Castellano, G., Castiello, C., Fanelli, A.M., Mencar, C.: Knowledge discovery by a neuro-fuzzy modeling framework. Fuzzy Sets and Systems 149, 187–207 (2005)
4. Qin, Y., Zhang, S., Zhu, X., Zhang, J., Zhang, C.: POP algorithm: Kernel-based imputation to treat missing values in knowledge discovery from databases. Expert Systems with Applications 36, 2794–2804 (2009)
5. Zhang, Q., Mahfouf, M.: A hierarchical Mamdani-type fuzzy modelling approach with new training data selection and multi-objective optimisation mechanisms: A special application for the prediction of mechanical properties of alloy steels. Applied Soft Computing 11, 2419–2443 (2011)
6. Rezaee, B., Zarandi, M.H.F.: Data-driven fuzzy modeling for Takagi–Sugeno–Kang fuzzy system. Information Sciences 180, 241–255 (2010)
7. Juang, C.F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Transactions on Systems, Man, and Cybernetics 34, 997–1006 (2004)
8. Kurban, T., Beşdok, E.: A comparison of RBF neural network training algorithms for inertial sensor based terrain classification. Sensors, 6312–6329 (2009)
9. Boskovitz, V., Guterman, H.: An adaptive neuro-fuzzy system for automatic image segmentation and edge detection. IEEE Transactions on Fuzzy Systems 10, 247–262 (2002)
10. Cpałka, K.: A new method for design and reduction of neuro-fuzzy classification systems. IEEE Transactions on Neural Networks 20, 701–714 (2009)
11. Jang, S.R.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics 23, 665–685 (1993)
12. Scherer, R.: Neuro-fuzzy relational systems for nonlinear approximation and prediction. Nonlinear Analysis: Theory, Methods & Applications 71, 1420–1425 (2009)
13. Jang, J.S.R., Sum, C.T., Mizutani, E.: Neuro-fuzzy and soft computing. Prentice-Hall, Englewood Cliffs (1997)
14. Qin, H., Yang, S.X.: Adaptive neuro-fuzzy inference systems based approach to nonlinear noise cancellation for images. Fuzzy Sets and Systems 158, 1036–1063 (2007)
15. Zounemat-Kermani, M., Teshnehlab, M.: Using adaptive neuro-fuzzy inference system for hydrological time series prediction. Applied Soft Computing 8, 928–936 (2008)
16. Ramot, D., Milo, R., Friedman, M., Kandel, A.: Complex fuzzy sets. IEEE Transactions on Fuzzy Systems 10, 171–186 (2002)
17. Chen, Z., Aghakhani, S., Man, J., Dick, S.: ANCFIS: A neurofuzzy architecture employing complex fuzzy sets. IEEE Transactions on Fuzzy Systems 19, 305–322 (2011)
18. Dick, S.: Toward complex fuzzy logic. IEEE Transactions on Fuzzy Systems 13, 405–414 (2005)
19. Aghakhani, S., Dick, S.: An on-line learning algorithm for complex fuzzy logic. In: IEEE International Conference on Fuzzy Systems (FUZZ), pp. 1–7 (2010)
20. Irani, R., Nasimi, R.: Application of artificial bee colony-based neural network in bottom hole pressure prediction in underbalanced drilling. J. Petroleum Science and Engineering 78, 6–12 (2011)
21. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optimization 39, 171–459 (2007)
22. Ozturk, C., Karaboga, D.: Hybrid artificial bee colony algorithm for neural network training. In: IEEE Congress on Evolutionary Computation (CEC), pp. 84–88 (2011)