Applications of Complex Fuzzy Sets in Time-Series Prediction

by

Omolbanin Yazdanbakhsh Poodeh

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Complex fuzzy sets are a recent extension of type-1 fuzzy sets, whose membership functions have the unit disc of the complex plane as their co-domain. In the same vein, complex fuzzy logic is a new multi-valued logic whose truth valuation set is the unit disc. Prior research has indicated that machine-learning algorithms built using complex fuzzy logic could be very accurate in time-series forecasting. This Ph.D. dissertation investigates different designs of machine learning algorithms based on complex fuzzy logic to develop reliable and fast algorithms for time-series prediction.

The machine learning algorithms designed in this dissertation are inferred from Adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS). ANCFIS was the first neuro-fuzzy system to combine complex fuzzy sets and rule interference for time-series forecasting. ANCFIS uses a hybrid learning rule where consequent parameters are updated on the forward pass, and antecedent parameters on the backward pass. Some recent findings, however, indicate that published results on ANCFIS are sub-optimal. First, we propose to improve the performance of the ANCFIS by changing how we define an input window, or even using sub-sampled windows. We compare the performance of ANCFIS using three different approaches to defining an input window, across six time-series data sets.

Then, we evaluate the performance of ANCFIS for univariate time-series prediction using a photovoltaic power data set. We compare the results of ANCFIS against well-known machine learning and statistical learning algorithms.

As ANCFIS has not been designed to work with multivariate time-series, we extend the ANCFIS learning architecture to the multivariate case. We investigate single-input-single-output,

multiple-input-single-output, and multiple-input-multiple-output variations of the architecture, exploring their performances on four multi-variate time-series. We also explore modifications to the forward- and backward-pass computations in the architecture. We find that our best designs are superior to the published results on these data sets, and at least as accurate as kernel-based prediction algorithms.

We also propose and evaluate a randomized-learning approach to training this neuro-fuzzy system. A number of recent results have shown that assigning fixed, random values to a subset of the adaptive parameters in a neural network model is an effective, simpler, and far faster alternative to optimizing those same parameters. We study mechanisms by which randomized learning may be combined with our system, and evaluate the system on both univariate and multivariate time-series. In general, we find that our proposed architecture is far faster than the original system, with no statistically significant difference in accuracy.

Finally, we propose a machine learning algorithm, which is designed for fast training of a compact, accurate forecasting model. We use the Fast Fourier Transform algorithm to identify the dominant frequencies in a time-series, and then create complex fuzzy sets to match them as the antecedents of a complex fuzzy rule. Consequent linear functions are then learned via recursive least-squares. We evaluate this algorithm on both univariate and multivariate time-series, finding that this incremental-learning algorithm is as accurate and compact as its slower predecessor, and can be trained much more quickly.

Preface

This is an original work by Omolbanin Yazdanbakhsh Poodeh. Chapter 2 of this thesis has been published as O. Yazdanbakhsh and S. Dick, "A Systematic Review of Complex Fuzzy Sets and Logic," *Fuzzy Sets and Systems,* 2017. Chapter 4 has been published as O. Yazdanbakhsh and S. Dick, "Time-Series Forecasting via Complex Fuzzy Logic," *Frontiers of Higher Order Fuzzy Sets*, A. Sadeghian and H. Tahayori, Eds., Heidelberg, Germany: Springer, 2015. Chapter 5 has been presented as O. Yazdanbakhsh, A. Krahn and S. Dick., "Predicting Solar Power Output using Complex Fuzzy Logic," *Fuzzy Information Processing Society (NAFIPS)*, Annual Meeting of the North American Edmonton, Alberta, 2013. A shorter version of Chapter 6 has been presented as O. Yazdanbakhsh and S. Dick, "Multi-variate time-series forecasting using complex fuzzy logic," *Fuzzy Information Processing Society (NAFIPS)* held jointly with 5th World Conference on Soft Computing (WConSC),Washington, Seattle, 2015, pp. 1-6, and the longer version has been published as O. Yazdanbahksh and S. Dick, "Forecasting of multivariate time-series via complex fuzzy logic," *IEEE Transactions on Systems, Man and Cybernetics: Systems*. Chapter 7 has been presented as O. Yazdanbakhsh and S. Dick, "ANCFIS-ELM: A Machine Learning Algorithm based on Complex Fuzzy Sets," *World Congress on Computational Intelligence*, Vancouver, Canada, 2016.

Dedication

This thesis is dedicated to my husband, Ali, who has been a constant source of support and encouragement during my Ph.D. This work is also dedicated to my mother, Maryam, and my sister, Fatemeh, who have always loved me unconditionally, and have always been there for me. Thank you for all of your support along the way.

Acknowledgments

I would like to thank my supervisor Dr. Scott Dick for his great help and encouragement through the course of my Ph.D. I am sincerely grateful to Dr. Scott Dick for his excellent guidance and supervision.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Complex fuzzy sets are a recent extension of type-1 fuzzy sets, whose membership functions have the unit disc of the complex plane as their co-domain. In the same vein, complex fuzzy logic is a new multi-valued logic whose truth valuation set is the unit disc [1, 2]. Prior research has indicated that machine-learning algorithms built using complex fuzzy logic could be very accurate in time-series forecasting. Adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS), previously developed in Dr. Dick's research group [3], is a time-series forecasting algorithm based on complex fuzzy logic, and shows excellent accuracy in univariate time-series forecasting using only a few rules to model even chaotic data sets.

This dissertation develops several machine learning algorithms inspired by ANCFIS using complex fuzzy logic. The evolution of these machine learning algorithms leads to a final system applicable for data-stream mining.

1.1. Contribution of the Thesis

Briefly, this dissertation mainly focuses on developing machine learning algorithms based on complex fuzzy logic for time-series forecasting. In the following, a summary of each chapter of the dissertation is given, contributions of the chapters are described, and the papers published based on each chapter are listed.

Chapter 2 presents a complete review on the truth valuation sets and complex fuzzy membership functions that have appeared in the literature, the existing results on complex fuzzy set operations and complex fuzzy relations, complex fuzzy logic and the existing applications of

complex fuzzy sets and logic. the main contribution of this chapter is a systematic review on complex fuzzy sets and logic to provide a framework to position new research in the field, consolidate the available theoretical results, catalogue the current applications of complex fuzzy sets and logic, identify the key open questions facing researchers in this area and suggest possible future directions for the research in this field. The Chapter 2 has been published in *Fuzzy Sets and Systems* [3].

Chapter 3 provides the background needed to follow this dissertation.

Chapter 4 investigates different approaches for presenting time-series to the ANCFIS machine-learning algorithm. The main contribution in this chapter is comparing three different approaches to define input-window for machine learning algorithms based on complex fuzzy set and logic. These three approaches are compared against each other by applying them to six univariate time-series data sets. Chapter 4 has been published as a book chapter in *Frontiers of Higher Order Fuzzy Sets* [4]. The input-window approaches presented in this chapter are used to create training and testing input vectors to train and evaluate the machine learning algorithms designed in this dissertation.

In the Chapter 5, ability of the ANCFIS machine-learning algorithm for one-step-ahead forecasting of univariate time-series is investigated. The main contribution in this chapter is evaluating the existing architecture of ANCFIS for univariate time-series prediction. A shorter version of Chapter 5 has been presented in *Fuzzy Information Processing Society (NAFIPS)*, Edmonton, Alberta, 2013 [5], and the longer version has been submitted to *Journal of Multiple-Valued Logic and Soft Computing* [6]. However, the existing architecture of ANCFIS has been

designed for univariate time-series prediction and it cannot be used for multivariate time-series prediction. Chapter 6 extends the ANCFIS machine-learning algorithm, designed for univariate time-series, for multivariate time-series forecasting.

Our first contribution in the Chapter 6 is to expand the known design space for machine learning algorithm based on complex fuzzy logic, by extending and evaluating the ANCFIS architecture for multivariate time-series forecasting. We evaluate three fundamental design options, as well as four more minor modifications. These three main options themselves form the second contribution: the design and evaluation of three new approaches for multivariate time-series forecasting. A shorter version of Chapter 6 has been presented in *Fuzzy Information Processing Society (NAFIPS)* held jointly with 5th World Conference on Soft Computing (WConSC), Washington, Seattle, 2015 [7], and the longer version has been published in *IEEE Transactions on Systems, Man and Cybernetics: Systems* [8]. However, the main disadvantages of ANCFIS is its slow learning algorithm. In Chapter 7, a new machine learning algorithm based on complex fuzzy sets and logic is developed that solves the slow learning problem.

The main contribution in the Chapter 7 is designing and developing a neuro-complex fuzzy system implementing randomized learning algorithms. The proposed system is applied for univariate and multivariate time-series forecasting. A shorted version of Chapter 7 has been presented in *World Congress on Computational Intelligence (WCCI)*, Vancouver, Canada, 2016 [9], and the longer version has been submitted in *IEEE Transactions on Fuzzy Systems* [10]. However, the weakness of randomized learning algorithms is that there is no way to assure that the randomly-chosen parameters are optimal for a given data set; this would manifest as a need for a larger number of complex fuzzy sets. This additional complexity is undesirable for large-scale

learning. Chapter 8 designs a compact, fast and accurate learning algorithm suitable for data stream prediction.

The main contribution of Chapter 8 is designing a system for fast training of a compact and accurate forecasting model to be used for data streams. We apply this architecture to time-series stream prediction problem and its performance evaluated on real-world univariate and multivariate time-series. A manuscript based on Chapter 8 is currently in preparation.

Chapter 2

Literature Review

A type-1 fuzzy set is descried by a membership function mapping the elements of a universe of discourse X to the unit interval [0,1]. Complex fuzzy sets and logic are an extension of type-1 fuzzy sets wherein memberships may be complex-valued. This has been an area of growing research focus in the fuzzy systems community for over a decade, with successful applications in time-series forecasting and other areas. In this section, we review available theoretical results and catalogue the current applications of complex fuzzy sets and logic.

2.1. Introduction

In 2002, Ramot et al. [1] defined a Complex Fuzzy Set (CFS) as an extension of type-1 fuzzy sets in which the co-domain of the membership function was the unit disc of the complex plane (the set of complex numbers with modulus ≤ 1). CFS are distinct from Buckley's fuzzy complex numbers, which are type-1 fuzzy subsets of the complex numbers [11]. While complex-valued memberships had been explored in a few earlier works [12-14], it was Ramot's paper and the companion piece on Complex Fuzzy Logic (CFL) [2] that became the seminal works in the topic of *Complex Fuzzy Sets and Logic* (CFS&L). Additional early theoretical results appeared in [15, 16], and the first applications of CFS&L began appearing in 2007 [17, 18].

In this chapter, we review the truth valuation sets and complex fuzzy membership functions that have appeared in the literature, the existing results on complex fuzzy set operations and complex fuzzy relations, complex fuzzy logic and the existing applications of CFS&L.

2.2. Forms and Values of Membership functions

In this section, values and functional forms of complex fuzzy membership functions are studied.

2.2.1. Codomains of Complex Fuzzy Membership Functions

Perhaps the earliest form of CFS was proposed by Nguyen et al., in order to be able to use fuzzy logic in fields with true paradoxes such as humanity and philosophy [19]. Moses et al., then, introduced a two-dimensional membership grade for fuzzy subsets of the complex numbers, $C \rightarrow [0,1]\times[0,1]$ [12, 13]. [14] studied optimal selections of membership functions for fuzzy complex numbers.

Ramot et al. proposed a complex fuzzy membership degree as [1]:

$$\mu_s(x) = r_s(x).e^{(j\omega_s(x))}, \qquad j = \sqrt{-1} \tag{1}$$

Where $r_s(x)$ is the magnitude and $\omega_s(x)$ the phase of the complex fuzzy set *s* and x is drawn from a universe of discourse U. The grade is bounded to the unit disc in the complex plane with $r_s(x) \in [0,1]$ and any real value for the phase. He suggested that the CFS can be an effective model for problems whose semantics change over time (i.e, the phase represents a changing context). Moreover, this is a straightforward generalization of type-1 fuzzy sets; when $\omega_s(x) = 0$, the CFS and all operations in Ramot's paper reduce to their type-1 counterparts, with the membership magnitude playing the role of the type-1 codomain.

As the complex fuzzy membership grade is two-dimensional (amplitude and phase), a complex fuzzy set can be visually represented by a three-dimensional graph where the universe of discourse is the third axis. Figure 1 shows the complex fuzzy set:

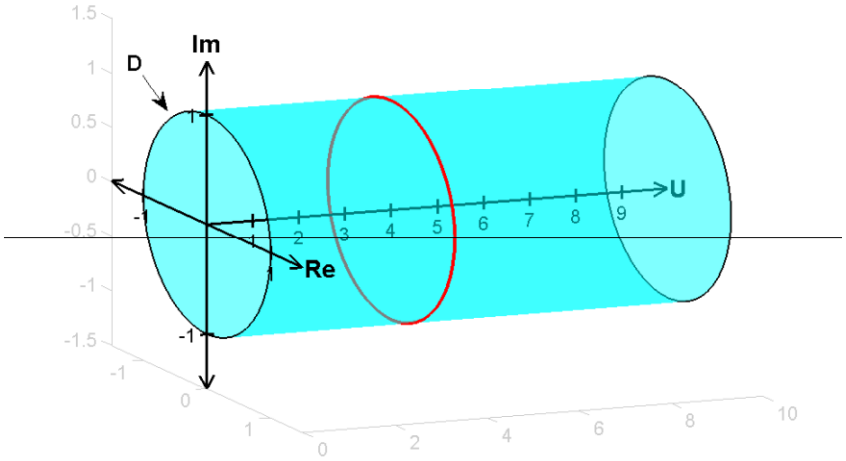


Figure 1: Complex fuzzy set defined in [15]

The unit disk is projected along the universe of discourse, U, giving a cylinder within which the complex fuzzy grades are bounded (the blue cylinder in Figure 1). The membership function itself is a trajectory within this cylinder. To obtain the membership degree of an object in the complex fuzzy set, the intersection of a unit disk centered at the object and the trajectory of the membership function in the cylinder is computed. For example, for x=4 in Figure 1, the intersection of the orthogonal unit disk (outlined in red) and a membership function gives the corresponding grade.

Tamir et al. defined complex fuzzy sets with membership degrees drawn from the unit square [0,1]×[0,1]. The memberships are given in Cartesian form, allowing both components of the membership to express type-1 fuzzy information [20], unlike Ramot's formulation. This characteristic makes Tamir's definition powerful to deal with problems with fuzzy cycle. The sets are termed "pure" complex fuzzy sets and the membership grades are defined as [20]:

$$\mu(V,z) = \mu_r(V) + j\mu_i(z)$$
$$\mu_r, \mu_i \in [0,1] \quad (2)$$

where $\mu_r(V)$ and $\mu_i(z)$ are the real and imaginary part of the complex fuzzy membership grade, $\mu(V,z)$, and can take any values from the interval [0,1]. Moreover, Polar presentation of pure complex fuzzy grade is defined as [20]:

$$\mu(V,z) = r(V)e^{j\sigma\phi(z)} \quad (3)$$

where r(V) and $\phi(z)$ are amplitude and phase of the pure complex fuzzy grade which take value from $[0,1]$, and σ is a scaling factor to keep the phase within the interval $(0,2\pi]$.

Tamir et al. interpreted pure complex fuzzy grades as a way to define pure fuzzy classes of order 1 [20]. General fuzzy classes are fuzzy sets that can contain fuzzy sets and objects. A pure fuzzy class of order M is a collection of pure fuzzy classes of order M-1; a fuzzy set is considered a pure fuzzy class of order 0. Thus, a pure fuzzy class of order 1 contains only fuzzy sets. A complex fuzzy class is defined as a pure fuzzy class of order 1. Assume that Γ is a complex fuzzy class, $\Gamma = \{V_i\}_{i=1}^{\infty}$, where V_i is a fuzzy set having objects on the universe of discourse, U, $z \in U$; a pure complex fuzzy grade, $\mu_\Gamma(V,z) = \mu_r(V) + j\mu_i(z)$, assigns the membership of an object, z, to a class, Γ through considering $\mu_i(z)$ as the grade of membership of z in V and $\mu_r(V)$ is the membership degree of V in Γ. The complex fuzzy class is also presented as [20]:

$$\Gamma = \{\mathrm{V}, \mathrm{z}, \mu_\Gamma(\mathrm{V},\mathrm{z}) | \mathrm{V} \in 2^{\mathrm{U}}, \mathrm{z} \in \mathrm{U}\} \quad (4)$$

Transformation between two proposed forms of pure complex fuzzy grade is defined as [20]:

$$\mu_\Gamma(V,z) = T_r\big(\mu_r(V) + j\mu_i(z)\big) = \mu_r(V)e^{j\theta\phi(z)} \quad (5)$$

$$\mu_\Gamma(V,z) = T_r\left(r(V)e^{j\theta\phi(z)}\right) = r(V) + j\phi(z) \quad (6)$$

where T_r is coordinate transformation function.

Yager defined Pythagorean membership grades as a pair of grade, $(A_Y(x), A_N(x))$, assigning membership and non-membership degree of each $x \in U$ to the fuzzy set A [21].

$$A_Y(x) = r(x)Cos\left(\theta(x)\right) \quad (7)$$

$$A_N(x) = r(x)Sin\left(\theta(x)\right) \quad (8)$$

where $r(x) = (1 - d(x))\frac{\pi}{2}$. $r(x)$ and $d(x)$ are the strength and direction of commitment at x, respectively, which are bounded in [0,1]. This new definition is considered in the family of non-standard membership degree along with intuitionistic fuzzy sets. The main difference between Pythagorean and intuitionistic fuzzy sets is in how the membership and non-membership components are constrained. Intuitionistic fuzzy sets must satisfy $A_Y(x) + A_N(x) \leq 1$, while Pythagorean fuzzy sets are constrained by $A_Y(x)^2 + A_N(x)^2 \leq 1$ [21]. Yager et al. showed that the Pythagorean membership grades, $\left(r(x)Cos\left(\theta(x)\right), r(x)Sin\left(\theta(x)\right)\right)$, are drawn from a subset of Ramot's complex fuzzy grades, $z = re^{i\theta}$ with the properties $r \in [0,1]$ and $\theta \in \left[0, \frac{\pi}{2}\right]$; Yager terms these the $\Pi - i$ numbers [22].

Salleh et al. defined the Complex Intuitionistic Fuzzy Set (CIFS) which is an extension of the intuitionistic fuzzy sets [23]. In CIFS, the membership and non-membership grade are drawn from a unit disk in the complex plane where $|\mu_A(x) + \gamma_A(x)| \leq 1$ [24].

In summation, four different codomains – the unit square, the unit circle, the unit positive quarter-circle, and the cross-product of the unit disc with itself – have been proposed in the CFS literature to date. There is very limited evidence on the properties of these codomains to date, and no more general conclusions about what other codomains might be useful can currently be drawn.

2.2.2. Forms of Complex Fuzzy Membership Functions

For CFS with the unit disc as codomain, two classes of complex fuzzy membership functions have been proposed: sinusoidal and Gaussian membership function. Sinusoidal membership function was introduced in [15, 25] as:

$$\mu(x) = r(x).e^{(j\omega(x))} \quad (9)$$

where

$$r(x) = d.sin(a(x) + b) + c \quad (10)$$

$$\theta = x$$

$$\omega(x) = \theta$$

where $r(\theta)$ is amplitude and θ is the phase of the membership grade of object *x.* a changes the frequency of the sine wave, b gives a phase shift whereas c shifts the wave vertically, and d changes the amplitude of the sine wave. Since the amplitude of complex fuzzy memberships is limited to [0,1], the parameters must satisfy the following conditions:

$$0 \leqslant d + c \leqslant 1, \qquad 1 \geqslant c \geqslant d \geqslant 0 \quad (11)$$

Four different forms of Gaussian membership functions have proposed [26-28]. The first function is described as [26]:

$$cGaussian(x,m,\sigma) = Re\big(cGaussian(x,m,\sigma)\big) + jIm\big(cGaussian(x,m,\sigma)\big) \quad (12)$$

where $Re(.)$ and $Im(.)$are real and imaginary parts of the membership grade which are defined as:

$$Re\big(cGaussian(x,m,\sigma)\big) = \exp[-0.5\left(\frac{x-m}{\sigma}\right)^2] \quad (13)$$

$$Im\big(cGaussian(x,m,\sigma)\big) = -\exp[-0.5\left(\frac{x-m}{\sigma}\right)^2]\times(\frac{x-m}{\sigma^2}) \quad (14)$$

where $\{m,\sigma\}$ are mean and spread of the Gaussian function, and x is the input. The second Gaussian function is [27]:

$$cGaussian(x,m,\sigma,\lambda) = r_s(x,m,\sigma)\exp(j\omega_s(x,m,\sigma,\lambda)) \quad (15)$$

where r_s and w_s are amplitude and phase of the complex fuzzy grade which are defined as:

$$r_s(x,m,\sigma) = Gaussian(x,m,\sigma) = \exp[-0.5\left(\frac{x-m}{\sigma}\right)^2] \quad (16)$$

$$\omega_s(x,m,\sigma,\lambda) = -\exp[-0.5\left(\frac{x-m}{\sigma}\right)^2]\times(\frac{x-m}{\sigma^2})\times\lambda \quad (17)$$

where $\{m,\sigma,\lambda\}$ are mean and spread and phase frequency factor for the complex fuzzy set, and x is the input. [29] proposed the third version of Gaussian membership function:

$$r_s(x,m,\sigma) = Gaussian(x,m,\sigma) = \exp[-0.5\left(\frac{x-m}{\sigma}\right)^2] \quad (18)$$

$$\omega_s(x,m,\sigma,\lambda) = -\exp[-0.5\left(\frac{x-m}{\sigma}\right)^2]\times(\frac{x-m}{\sigma^2}) \quad (19)$$

where {m,σ} are mean and spread of the Gaussian function, x is the input and r_s and w_s are amplitude and phase of the complex fuzzy grade.

Lastly, [28] proposed a Gaussian membership function as:

$$\mu(x) = A(x).e^{\mathrm{iP(x)}} = A(x)\angle P(x) \quad (20)$$

where A(x) and P(x) are amplitude and phase of the complex fuzzy membership function and defined as:

$$A(x) = \exp(-(\frac{x-c_A}{a_A})^2) \quad (21)$$

$$P(x) = 2\pi\exp((-(\frac{x-c_P}{a_P})^2) \quad (22)$$

CFS membership functions and their properties plainly remain a very young topic. Basic properties and manipulations (e.g., what are the core and support of a CFS? What is an α-cut?) have not yet been defined in general. Other functional forms have not yet been proposed (e.g. what are the equivalents to triangular or trapezoidal membership functions?) Furthermore, there has been almost no work in how we *interpret* a CFS. Sinusoids and Gaussian-type membership functions clearly have very different semantics. A Gaussian membership by its nature focuses on one segment of a universe of discourse; one region of feature space, one moment in time, etc. Sinusoids, by contrast, focus on recurring patterns. However, these general observations offer little

guidance in associating an intuitive, linguistic meaning to a specific CFS, or even a collection of them.

Linguistic variables – the association of a linguistic term with a type-1 fuzzy set – are a key part of the success of fuzzy logic, and we expect that the same will hold true for CFS. However, the two-dimensional nature of CFS membership functions has so far made interpreting them extremely difficult. We are aware of only three partial suggestions: first, Alkouri's definition of linguistic variables and hedges for CFS[30] ; in the paper, the linguistic variable for complex fuzzy sets is characterized as a sextuple in which different linguistic values are considered for uncertainty and periodicity. Also, six hedges are defined for CIFS: *very A, very-very A, indeed A, a little A, slightly A, more or less A and extremely A.* Second, Tamir et al.'s suggestion that a complex fuzzy class can be interpreted as a main term and a modifier [31]; it is able to describe propositions as "x …A…B" where A and B are linguistic values; e.g. "x is a volatile stock in a strong-portfolio" Finally, Dick et al.'s suggestion that non-membership and anti-membership could be used to model negations and antonyms [32]. In essence, the space of complex membership functions remains largely unexplored except for a few examples.

2.3. Complex Fuzzy Set Operations and Relations

In this section, we review complex fuzzy set operations as they apply equally to individual fuzzy sets and relations between fuzzy sets. Moreover, the topic of compositions and projections of complex fuzzy relations are studied.

2.3.1. Complex Fuzzy Set Operations

Ramot proposed union and intersection operations for his CFS, along with three unique operations manipulating the phase of the CFS membership grade [1]. Consider two complex fuzzy sets, *A* and *B*, with membership degrees of $\mu_A(x) = r_A(x).e^{j\omega_A(x)}$ and $\mu_B(x) = r_B(x).e^{j\omega_B(x)}$, respectively. The union and intersection of these two complex fuzzy sets, are defined as [1]:

$$\mu_{A\cup B}(x) = [r_A(x) \oplus r_B(x)]e^{j(\omega_{A\cup B})} \quad (23)$$

$$\mu_{A\cap B}(x) = [r_A(x) \star r_B(x)]e^{j(\omega_{A\cap B})} \quad (24)$$

where $\mu_{A\cup B}(x)$ is union and $\mu_{A\cap B}(x)$ is intersection of the complex fuzzy sets *A* and *B*, respectively. $\oplus$ can be any t-conorms and $\star$ is any t-norms. $\omega_{A\cup B}$ and $\omega_{A\cap B}$ are application-dependent, Equation (25) - (31) show the possible forms for $\omega_{A\cup B}$ that are also applicable for $\omega_{A\cap B}$ [1]:

$$\omega_{A\cup B} = \omega_A + \omega_B \quad (25)$$

$$\omega_{A\cup B} = \max(\omega_A, \omega_B) \quad (26)$$

$$\omega_{A\cup B} = \min(\omega_A, \omega_B) \quad (27)$$

$$\omega_{A\cup B} = \begin{cases} \omega_A & r_A > r_B \\ \omega_B & r_B < r_A \end{cases} \quad (28)$$

$$\omega_{A\cup B} = \frac{r_A.\omega_A + r_B.\omega_B}{r_A + r_B} \quad (29)$$

$$\omega_{A\cup B} = \frac{\omega_A + \omega_B}{2} \quad (30)$$

$$\omega_{A\cup B} = \omega_A - \omega_B \quad (31)$$

The three novel operations are reflection, rotation and directional complex (DC) fuzzy complement that are defined as [1]:

$$Ref\big(\mu_S(x)\big) = r_S(x).e^{-j\omega_S(x)} \quad (32)$$

where $Ref(.)$ indicates reflection of complex fuzzy set, S, with membership grade of $\mu_s(x) = r_S(x).e^{j\omega_S(x)}$. Equation (33) defines rotation of the complex fuzzy set, S, with θ radians [1]:

$$Rot_\theta\big(\mu_S(x)\big) = r_S(x).e^{j(\omega_S(x)+\theta)} \quad (33)$$

where θ is the rotation of the complex fuzzy set. Based on the rotation and traditional complement operations, Ramot et al. proposed directional complex (DC) fuzzy complement as[1]:

$$\mu_{\bar{S}^\theta}(x) = c\big(r_S(x)\big).e^{j(\omega_S(x)+\theta)} \quad (34)$$

where $c(.)$ is any fuzzy complement function.

Zhang et al. [16] studied different operations on Ramot's CFS [1] when the membership phase is restricted to $[0,2\pi]$. The union of two CFS A and B with $\mu_A(x) = r_A(x).e^{j\omega_A(x)}$ and $\mu_B(x) = r_B(x).e^{j\omega(x)}$, is defined as Equation (35) and proved to be an s-norm [16]:

$$\begin{aligned}\mu_{A\cup B}(x) &= r_{A\cup B}(x).e^{j\omega_{A\cup B}(x)} \\ &= \max\big(r_A(x), r_B(x)\big).e^{jmax(\omega_A(x),\omega_B(x))}\end{aligned} \quad (35)$$

The intersection between these two CFS was defined and proved to be a t-norm [16]:

$$\mu_{A\cap B}(x) = r_{A\cap B}(x).e^{j\omega_{A\cap B}(x)} = \min\big(r_A(x), r_B(x)\big).e^{jmin(\omega_A(x),\omega_B(x))} \quad (36)$$

The complement of a CFS, C, with $\mu_C(x) = r_C(x).e^{j\omega_C(x)}$ was proposed as [16]:

$$\mu_{\bar{C}}(x) = r_{\bar{C}}(x).e^{j\omega_{\bar{C}}(x)} = \left(1 - r_C(x)\right).e^{j(2\pi - \omega_C(x))} \quad (37)$$

It was shown that the involutive property for the complement function is also satisfied, $\bar{\bar{C}} = C$. Based on the definitions of union, intersection and complement proposed above, the following properties were obtained [16]:

$$\overline{A \cap B} = \bar{A} \cup \bar{B} \quad (38)$$

$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C) \quad (39)$$

$$(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$$

$$(A \cup B) \cap A = A \quad (40)$$

$$(A \cap B) \cup A = A$$

The following operations were also proposed [16]:

Complex fuzzy product (t-norm): (41)

$$\mu_{A \circ B}(x) = r_{A \circ B}.e^{j\omega_{A \circ B}(x)} = \left(r_A(x).r_B(x)\right).e^{j2\pi\left(\frac{\omega_A(x)}{2\pi}.\frac{\omega_B(x)}{2\pi}\right)}$$

Complex fuzzy Cartesian product: (42)

$$\mu_{A_1 \times A_2 \times \cdots \times A_N}(x) = r_{A_1 \times A_2 \times \cdots \times A_N}(x).e^{j\omega_{A_1 \times A_2 \times \cdots \times A_N}(x)}$$
$$= \min\left(r_{A_1}(x_1), r_{A_2}(x_2), \cdots, r_{A_N}(x_N)\right).e^{jmin\left(\omega_{A_1}(x_1), \omega_{A_2}(x_2), \cdots, \omega_{A_N}(x_N)\right)},$$
$$where\ x = (x_1, x_2, \cdots, x_N) \in \underbrace{U \times U \times \cdots \times U}_{N}.$$

Complex fuzzy probabilistic sum (s-norm): (43)

$$\mu_{A\hat{+}B}(x) = r_{A\hat{+}B}(x).\, e^{j\omega_{A\hat{+}B}(x)}$$
$$= \big(r_A(x) + r_B(x)$$
$$- r_A(x).\, r_B(x)\big).\, e^{j2\pi\left(\frac{\omega_A(x)}{2\pi}+\frac{\omega_B(x)}{2\pi}-\frac{\omega_A(x)}{2\pi}.\frac{\omega_B(x)}{2\pi}\right)}$$

Complex fuzzy bold sum (s-norm): (44)

$$\mu_{A\dot{\cup}B}(x) = r_{A\dot{\cup}B}(x).\, e^{j\omega_{A\dot{\cup}B}(x)}$$
$$= \min\big(1, r_A(x) + r_B(x)\big)\,.\, e^{jmin(2\pi,\omega_A(x)+\omega_B(x))}$$

Complex fuzzy bold intersection (t-norm): (45)

$$\mu_{A\dot{\cap}B}(x) = r_{A\dot{\cap}B}(x).\, e^{j\omega_{A\dot{\cap}B}(x)}$$
$$= \max(0, r_A(x) + r_B(x) - 1)\,.\, e^{jmax(0,\omega_A(x)+\omega_B(x)-2\pi)}$$

Complex fuzzy bounded difference: (46)

$$\mu_{A|-|B}(x) = r_{A|-|B}(x).\, e^{j\omega_{A|-|B}(x)} = \max(0, r_A(x) -$$
$$r_B(x)).\, e^{jmax(0,\omega_A(x)-\omega_B(x))}$$

Complex fuzzy symmetrical difference: (47)

$$\mu_{A\nabla B}(x) = r_{A\nabla B}(x).\, e^{j\omega_{A\nabla B}(x)} = |r_A(x) - r_B(x)|.\, e^{j|\omega_A(x)-\omega_B(x)|}$$

Complex fuzzy convex linear sum of min and max: (48)

$$\mu_{A||_\lambda B}(x) = r_{A||_\lambda B}(x).\, e^{j\omega_{A||_\lambda B}(x)} = \big[\lambda \min\big(r_A(x), r_B(x)\big) + (1 -$$
$$\lambda) \max\big(r_A(x), r_B(x)\big)\big].\, e^{j[\lambda \min(\omega_A(x),\omega_B(x))+(1-\lambda)\max(\omega_A(x),\omega_B(x))]}$$

where $(0 \le \lambda \le 1)$.

Moreover, a new definition for distance of complex fuzzy set was introduced by Zhang et al. as [16]:

$$d(A,B) = \max(sup_{X\in U}|r_A(X) - r_B(X)|, \frac{1}{2\pi} sup_{X\in U}|\omega_A(X) - \omega_B(X)|) \quad (49)$$

where $d(A,B)$ is distance of two complex fuzzy sets A and B. Then, based on this definition, δ-equalities of complex fuzzy sets were proposed; two complex fuzzy sets, A and B, are δ-equal, $A = (\delta)B$, if and only if $d(A,B) \leq 1-\delta, 0 \leq \delta \leq 1$. Properties of δ-equality of complex fuzzy sets were also discussed in the paper.

Two aggregation operations have been proposed in order to combine several complex fuzzy sets with unit circle codomain. Ramot et al. proposed an aggregation operation which is called vector aggregation and defined as [2]:

$$v: \{a|a \in \boldsymbol{C}, |a| \leq 1\}^n \rightarrow \{b|b \in \boldsymbol{C}, |b| \leq 1\} \quad (50)$$

$$\mu_A(x) = v\left(\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)\right) = \sum_{i=1}^{n} w_i \mu_{A_i}(x) \quad (51)$$

where $w_i \in \{a|a \in \boldsymbol{C}, |a| \leq 1\}$ for all i, and $\sum_{i=1}^{n}|w_i| = 1$. The operation is, in fact, a weighted vector sum considering effects of complex fuzzy phase on aggregation of complex fuzzy sets. Ma et al. introduced product-sum aggregation operation for complex fuzzy sets [33], that is an extension of the aggregation operation proposed by Ramot et al. [2]. Let $\tilde{A}_1, \tilde{A}_2, \dots, \tilde{A}_n$ be complex fuzzy sets defined on a universe of discourse, U, with $\mu_{\tilde{A}_i}(u) = x_i(u) + jy_i(u)$ and $c = (c_1, c_2, \dots, c_n)$ be a complex-valued vector in the unit disk of the complex plane with $c_i = x_i' + jy_i'$; the product-sum aggregation operation is defined as [33]:

$$p_c(u) = r\big(p_c(u)\big)e^{j\arg(p_c(u))} \quad (52)$$

where

$$r\left(p_c(u)\right) \quad (53)$$

$$= min\left\{1, \sqrt[2]{\left(\sum_{i=1}^{n}(x_i(u)x_i' - y_i(u)y_i')\right)^2 + \left(\sum_{i=1}^{n}(x_i(u)y_i' + y_i(u)x_i')\right)^2}\right\}$$

$$\arg\left(p_c(u)\right) = arctan\left(\frac{\sum_{i=1}^{n}(x_i(u)y_i' + y_i(u)x_i')}{\sum_{i=1}^{n}(x_i(u)x_i' - y_i(u)y_i')}\right) \quad (54)$$

Tamir et al. proposed union, intersection and complement operations for the pure complex fuzzy classes (unit square codomain) [20]. Assume a complex fuzzy class, $\Gamma = \{V, z, \mu_\Gamma(V, z) | V \in 2^U, z \in U\}$, defined by a pure complex fuzzy grade, $\mu(V, z) = \mu_r(V) + j\mu_i(z)$; the complement of this class is defined as [20]:

$$c(\mu_\Gamma(V, z)) = c(\mu_r(V)) + jc(\mu_i(z)) \quad (55)$$

where $c(.)$ is complement function and can be defined as standard fuzzy complement, $c(\mu_x(y)) = 1 - \mu_x(y)$. The union and intersection of two complex fuzzy classes defined as $\Gamma = \{V, z, \mu_\Gamma(V, z) | V \in 2^U, z \in U\}$ and $\Psi = \{T, z, \mu_\Psi(T, z) | T \in 2^U, z \in U\}$ are given by [20]:

$$\mu_{\Gamma \cup \Psi}(W, z) = (\mu_{\Gamma_r}(V) \oplus \mu_{\Psi_r}(T)) + j(\mu_{\Gamma_i}(z) \oplus \mu_{\Psi_i}(z)) \quad (56)$$

$$\mu_{\Gamma \cap \Psi}(W, z) = (\mu_{\Gamma_r}(V) \odot \mu_{\Psi_r}(T)) + j(\mu_{\Gamma_i}(z) \odot \mu_{\Psi_i}(z)) \quad (57)$$

where $\mu_{\Gamma \cup \Psi}(W, z)$ and $\mu_{\Gamma \cap \Psi}(W, z)$ are union and intersection functions, respectively; $W \in 2^U$ and $\oplus$ is any t-conorm and $\odot$ is any t-norm operation.

Yager et al. proposed union, intersection and complement operations for two Pythagorean fuzzy sets $A(x) = (A_Y(x), A_N(x))$ and $B(x) = (B_Y(x), B_N(x))$ as follows [22]:

$$A(x) \cup B(x) = (\max\left(A_Y(x), B_Y(x)\right), \min\left(A_N(x), B_N(x)\right)) \quad (58)$$

$$A(x) \cap B(x) = (min(A_Y(x), B_Y(x)), max(A_N(x), B_N(x))) \quad (59)$$

$$\bar{A}(x) = (A_N(x), A_Y(x)) \quad (60)$$

He also proposed the following function for mapping a $\Pi - i$ number to the unit interval [0, 1] as a way of ordering Pythagorean fuzzy values [22].

$$F(A) = \frac{1}{2} + r(x).(\frac{1}{2} - \frac{2.\theta(x)}{\pi}) \quad (61)$$

where $r = \sqrt{{A_Y}^2 + {A_N}^2}$ and $\theta = \tan^{-1}\frac{A_N}{A_Y}$. [22] also proposed an aggregation operation based on geometric mean for q different criteria that have Pythagorean fuzzy degrees, $C_j,\ j = 1, \dots, q$, as:

$$C(x) = \prod_{j=1}^{q} C_j(x)^{w_j} = \prod_{j=1}^{q} (r_j(x)e^{i\theta_j(x)})^{w_j} = \prod_{j=1}^{q} (r_j(x))^{w_j} e^{i\sum w_j\theta_j(x)} \quad (62)$$

where $\sum_{j=1}^{q} w_j = 1$.

Dick et al. proposed two new complex fuzzy conjunction and disjunction operators which form a triple DeMorgan with the complement operation in Equation. (60) [32]:

$$x \wedge y = (absmin\ (x_1, y_1), absmax(x_2, y_2)) \quad (63)$$

$$x \vee y = (absmax\ (x_1, y_1), absmin(x_2, y_2)) \quad (64)$$

where $x = (x_1, y_1)$, $y = (x_2, y_2) \in \boldsymbol{D}$ (unit disk), and *absmax* and *absmin* are defined as [32]:

$$absmax(x, y) = \begin{cases} x & if \quad |x| > |y| \\ y & if \quad |x| < |y| \\ |x| & if \quad |x| = |y| \wedge x \neq y \end{cases} \quad (65)$$

$$absmin(x, y) = \begin{cases} x & if \quad |x| < |y| \\ y & if \quad |x| > |y| \\ -|x| & if \quad |x| = |y| \wedge x \neq y \end{cases} \quad (66)$$

Salleh et al. in [24] extended the complement, union and intersection defined by [1, 16] for complex Atanassonov's intuitionistic fuzzy sets, and [34] extended the intuitionistic possibility and necessity measures defined by [23] for CAIFS.

2.3.2. Complex Fuzzy Relations

Ramot et al. presents complex fuzzy relations as the degree and phase of association, interaction and interconnectedness between elements of different universe of discourses [1]. Let U and V be two universes of discourse; the complex fuzzy relation between them is a complex fuzzy subset of the product space $U \times V$, that is characterized by a complex-valued membership function which is bounded in a unit disk, $\mu_R(x, y)$, and defined as [1]:

$$R(U,V) = \left\{ \left((x,y), \mu_R(x,y) \right) \middle| \, (x,y) \in U \times V \right\} \quad (67)$$

Composition of the complex fuzzy relations either defined on the same product spaces or on the different product spaces were proposed in [2]. In the case of the same product space, union and intersection operations defined for complex fuzzy sets are applicable otherwise the composition is defined as follows. Consider $R(U,V)$ with $\mu_R(x,y) = r_R(x,y).e^{j\omega_R(x,y)}$ and $S(V,W)$ with $\mu_S(y,z) = r_S(y,z).e^{j\omega_S(y,z)}$; the composition is determined as [2]:

$$\mu_{R \circ S}(x,z) = r_{R \circ S}(x,z).e^{j\omega_{R \circ S}(x,z)} \quad (68)$$

where

$$r_{R \circ S}(x,z) = \sup_{y \in V} [r_R(x,y) \star r_S(y,z)] \quad (69)$$

$$\omega_{R \circ S}(x,z) = f[g(\omega_R(x,y), \omega_S(y,z))] \quad (70)$$

where $\star$ and sup refer to a t-norm and supremum (least upper bound) operations, respectively, and g and f are the membership phase functions equivalent of those operations, respectively; [2] proposed possible choices for the sup operation as:

$$\omega_{R\circ S}(x,z) = \sup_{y\in V}[g(\omega_R(x,y),\omega_S(y,z))] \quad (71)$$

$$\omega_{R\circ S}(x,z) = \inf_{y\in V}[g(\omega_R(x,y),\omega_S(y,z))] \quad (72)$$

$$\omega_{R\circ S}(x,z) = g(\omega_R(x,y'),\omega_S(y',z)) \quad (73)$$

$$y' = \left\{ y \middle| \sup_{y\in V}[r_R(x,y)\star r_S(y,z)] \right\}$$

Complex fuzzy relation operations were introduced by Zhang et al. [35] for Ramot's complex fuzzy relations, again limiting membership phase to $[0,2\pi]$. Let A and B be two complex fuzzy sets on the universes of discourse U and V, respectively, with $\mu_A(x) = r_A(x).e^{j\omega_A(x)}$ and $\mu_B(y) = r_B(y).e^{j\omega_B(y)}$. The complex fuzzy union and intersection of A and B are defined as [35]:

Union: (74)

$$\mu_{A\oplus B}(x,y) = r_{A\oplus B}(x,y).e^{j\omega_{A\oplus B}(x,y)} = max\big(r_A(x),r_B(y)\big).e^{jmax(\omega_A(x),\omega_B(y))}$$

Intersection: (75)

$$\mu_{A\otimes B}(x,y) = r_{A\otimes B}(x,y).e^{j\omega_{A\otimes B}(x,y)} = min\big(r_A(x),r_B(y)\big).e^{jmin(\omega_A(x),\omega_B(y))}$$

Compositions of the complex fuzzy relations, $R(U,V)$ with $\mu_R(x,y) = r_R(x,y).e^{j\omega_R(x,y)}$ and $S(V,W)$ with $\mu_S(y,z) = r_S(y,z).e^{j\omega_S(y,z)}$, are defined as [35]:

Sup-min composition: (76)

$$\mu_{R\circ S}(x,z) = r_{R\circ S}(x,z).\, e^{j\omega_{R\circ S}(x,z)}$$
$$= \sup_{y\in V} min\big(r_R(x,y), r_S(y,z)\big).\, e^{j\sup_{y\in V} min\left(\omega_R(x,y),\omega_S(y,z)\right)}$$

Sup-product composition: (77)

$$\mu_{R\circ S}(x,z) = \big(r_R(x,y).\, r_S(y,z)\big).\, e^{j2\pi\left(\left(\frac{\omega_R(x,y)}{2\pi}.\frac{\omega_S(y,z)}{2\pi}\right)\right)}$$

Sup-Lukasiewicz composition (78)

$$\mu_{R\circ S}(x,z) = \big(r_R(x,y) * r_S(y,z)\big).\, e^{j\left(\omega_R(x,y)*\omega_S(y,z)\right)}$$

$$\omega_R(x,y) * \omega_S(y,z) = 2\pi\, max\left(0, \frac{\omega_R(x,y)}{2\pi} + \frac{\omega_S(y,z)}{2\pi} - 1\right)$$

For two complex fuzzy relations defined on the same product space, $\mu_A(x,y) = r_A(x,y).\, e^{j\omega_A(x,y)}$ and $\mu_B(x,y) = r_B(x,y).\, e^{j\omega_B(x,y)}$, the complex fuzzy union and intersection of the relations are defined as [35]:

Union: (79)

$$\mu_{A\cup B}(x,y) = r_{A\cup B}(x,y).\, e^{j\omega_{A\cup B}(x,y)}$$
$$= max\big(r_A(x,y), r_B(x,y)\big).\, e^{jmax\left(\omega_A(x,y),\omega_B(x,y)\right)}$$

Intersection: (80)

$$\mu_{A\cap B}(x,y) = r_{A\cap B}(x,y).\, e^{j\omega_{A\cap B}(x,y)}$$
$$= min\big(r_A(x,y), r_B(x,y)\big).\, e^{jmin\left(\omega_A(x,y),\omega_B(x,y)\right)}$$

Moreover, the distance of two complex fuzzy relations defined on the same product space was defined as [35]:

$$d(A,B) = max\left(\sup_{(x,y)\in U\times V} |r_A(x,y) - r_B(x,y)|, \frac{1}{2\pi}\sup_{(x,y)\in U\times V} |\omega_A(x,y) - \omega_B(x,y)|\right) \quad (81)$$

2.4. Complex Fuzzy Logic

Complex Fuzzy Logic (CFL) was first proposed by Ramot et al. in [2], using the same unit-disc codomain as his CFS in [1]. This yields an infinite-valued logic wherein the truth values are vectors from the unit disc. This makes CFL one of the very few vector-valued logics in existence. Vector logic, Boolean matrices and quantum logic also have multi-dimensional truth values [36] [37] [38]. [39] studied harmonic analysis on fuzzy sets by extending fuzzy logic to a unit circle $\{z : z \in \boldsymbol{C}, |z| = 1\}$.

Dick in 2005 showed that considering phase as a relative quantity in Ramot's papers [1, 2] can be interpreted as rotational invariance, meaning that if membership grades of two complex fuzzy sets are rotated with φ radians about the origin, the union, intersection and complement of the complex fuzzy sets are also rotated with the same phase, φ. He showed that algebraic product and traditional complement, $f(x) = -x$, cannot be considered as conjunction and negation operations with rotational invariance assumption [15]. He then studied amplitude and phase of complex fuzzy sets simultaneously and used vector logic proposed in [36, 40] as framework. It was shown that algebraic product is a possible candidate for conjunction operator, and existence of a dual disjunction operator was proved.

In [41], generalized propositional complex fuzzy logic was introduced based on Łukasiewicz logic system by considering Modus ponens as the rule inference and the following connectives:

Table 1: Basic complex propositional fuzzy logic connectives

Operation	Interpretation
Negation	$i('P) = 1 + j1 - i(p)$
Implication	$i(P \to Q) = min\big(1,1 - i(p_r) + i(q_r)\big) + j.min\big(1,1 - i(p_i) + i(q_i)\big)$
Conjunction	$i(P \otimes Q) = min\big(i(p_r), i(q_r)\big) + j.min\big(i(p_i), i(q_i)\big)$
Disjunction	$i(P \oplus Q) = max\big(i(p_r), i(q_r)\big) + j.max\big(i(p_i), i(q_i)\big)$

[42] proposed an extended complex Post logical system (ECPS) based on the extended Post system (EPS) of order $p > 2$ by DiZenzo [43]. One of the possible applications of the proposed system is in discrete processes such as digital signal processing (DSP), real-time applications and embedded systems. [44] gave a review on the axiomatic-based complex fuzzy logic and sets.

Dick et al. developed two complete and distributive lattices for Pythagorean fuzzy sets (PFS) based on Pythagorean fuzzy union and intersection (see Equations. (58) - (59)) and ranking function (Equation. (61)) proposed in [22]. He then developed two new complete and distributive lattices for complex fuzzy sets based on extending PFS to unit disk. This extension needs a definition for negations of the two axes in PFS ($\mu, \neg\mu$: membership and non-membership degree in PFS) to convert it to a complete unit disk; the negations are called anti-membership and anti-non-membership ($\bar{\mu}, \neg\bar{\mu}$). The first lattice is based on the assumption that $\mu = \neg\bar{\mu}$ and $\neg\mu = \bar{\mu}$. There is, thus, a mapping from unit disk to the PFS. The map is used to define a partial ordering over the unit disk leading to form the lattice. The second lattice is based on the assumption that $\mu \neq \neg\bar{\mu}$ and $\neg\mu \neq \bar{\mu}$; two new complex fuzzy union and intersection (see Equations. (63) - (64)) are defined and showed that they form the lattice. Moreover, it is shown that Pythagorean negation (Equation. (60)) forms a DeMorgan triple with the new union and intersection operators. This logic, also, gives interpretation for both fuzzy negation and antonym [32].

Ramot proposed a complex fuzzy implication for the unit-disc codomain in [2]. Consider the rule: *If x is A then y is B,* where x and y are variables taken from two different universe of discourses, *U* and *V*, respectively, and *A* and *B* are complex fuzzy sets defined on the corresponding universe of discourses; a complex fuzzy implication is characterized by the membership function $\mu_{A\to B}(x,y) = r_{A\to B}(x,y).e^{j\omega_{A\to B}(x,y)}$. The implication function is the algebraic product:

$$\mu_{A\to B}(x,y) = \mu_A(x).\mu_B(y) = r_A(x).r_B(y).e^{j(\omega_A(x)+\omega_B(y))} \quad (82)$$

Zhang et al. also studied more implication operations for CFS with the unit disk codomain, and with membership phase limited to $[0,2\pi]$, as follows [35]:

Dienes-Rescher implication operator: (83)

$$\mu_{A\to B}(x,y) = max\big(1-r_A(x),r_B(y)\big).e^{jmax\left(2\pi-\omega_A(x),\omega_B(y)\right)}$$

Lukasiewicz implication operator: (84)

$$\mu_{A\to B}(x,y) = \big(r_A(x)*r_B(y)\big).e^{j\left(\omega_A(x)*\omega_B(y)\right)}$$

$$\omega_A(x)*\omega_B(y) = 2\pi\, max\left(0,\frac{\omega_A(x)}{2\pi}+\frac{\omega_B(y)}{2\pi}-1\right)$$

Zadeh implication operator: (85)

$$\mu_{A\to B}(x,y) = max\Big(1 - r_A(x), min\big(r_A(x),r_B(y)\big)\Big).e^{jmax\left(2\pi-\omega_A(x),min\left(\omega_A(x),\omega_B(y)\right)\right)}$$

Godel implication operator: (86)

$$\mu_{A\to B}(x,y) = r_{A\to B}(x,y).e^{j\omega_{A\to B}(x,y)}$$

$$r_{A\to B}(x,y)=\begin{cases}1 & if\ \ r_A(x)\le r_B(y)\\ r_B(y) & otherwise\end{cases}$$

$$\omega_{A\to B}(x,y)=\begin{cases}2\pi & if\ \omega_A(x)\le \omega_B(y)\\ \omega_B(y) & otherwise\end{cases}$$

Mamdani implication operator: (87)

$$\mu_{A\to B}(x,y)=min\big(r_A(x),r_B(y)\big).e^{jmin\left(\omega_A(x),\omega_B(y)\right)}$$

Mamdani product implication operator: (88)

$$\mu_{A\to B}(x,y)=\big(r_A(x).r_B(y)\big).e^{j2\pi\left(\frac{\omega_A(x)}{2\pi}.\frac{\omega_B(y)}{2\pi}\right)}$$

Godel implication operator: (89)

$$\mu_{A\to B}(x,y)=r_{A\to B}(x,y).e^{j\omega_{A\to B}(x,y)}$$

$$r_{A\to B}(x,y)=\begin{cases}1 & if\ \ r_A(x)\le r_B(y)\\ max(1-r_A(x),r_B(y)) & otherwise\end{cases}$$

$$\omega_{A\to B}(x,y)=\begin{cases}2\pi & if\ \omega_A(x)\le \omega_B(y)\\ max(2\pi-\ \omega_A(x),\omega_B(y)) & otherwise\end{cases}$$

Reichenbach product implication: (90)

$$\mu_{A\to B}(x,y)=\big(1-r_A(x)+r_A(x).r_B(y)\big).e^{j\left(2\pi-\ \omega_A(x)+2\pi\frac{\omega_A(x)}{2\pi}.\frac{\omega_B(y)}{2\pi}\right)}$$

Ramot et al. used Generalized Modus Ponens for complex fuzzy inference rules. In the case of multi-complex fuzzy rules, he proposed using the vector aggregation operation (Equation. (51)) to combine the rule consequents into a final output. As Equation. (51) is a vector sum of the outputs, it considers phase terms of the outputs into account. If the phase terms are aligned, amplitude of the final output increases otherwise they have destructive effect on the amplitude of the output. In the defuzzification step, phase term of the final output is ignored and any of the

traditional defuzzification methods can be applied on the amplitude in order to obtain a crisp output [2].

Tamir et al. studied the axiomatization of complex fuzzy logic for the pure complex fuzzy sets defined in [20]. He developed propositional, first order predicate, generalized propositional complex fuzzy logic and a logical system for discrete complex fuzzy sets [31, 41, 42]. Tamir's works are different from the complex valued propositional logic (S-logic) introduced in [45] which is a six-valued logic containing imaginary logic variables $\{i, \neg i\}$, real logic variables $\{T, F\}$ and two basic states of S-logic $\{T \vee i, F \wedge \neg i\}$.

Similar to fuzzy propositions, complex fuzzy propositions are assigned a truth value, in this case from the unit square $[0,1]\times[0, i]$. A complex fuzzy proposition P can be defined as "*x* … *A* … *B*" where *A* and *B* are linguistic values, and … shows natural language constants; and it can be interpreted as $i(P) = i(p_r) + j.i(p_i)$ or $i(P) = i\big(r(p)\big).e^{j\sigma i(\theta(p))}$ (σ is a scaling factor in the interval $(0,2\pi]$) where $i(p_r)$ or $i\big(r(p)\big)$ denotes an interpretation of term *A*, and $i(p_i)$ or $i\big(\theta(p)\big)$ is an interpretation of term *B*.

Propositional and first order predicate complex fuzzy logic are defined in [31], and are denoted as ŁΠ CFL and ŁΠ∀ CFL, respectively. Propositional complex fuzzy logic uses a set of basic and derived connectives along with a set of axioms from [46] to combine complex fuzzy propositions; for two complex fuzzy propositions *P* and *Q,* Table 2 and Table 3 denote the connectives:

Table 2: Basic ŁΠ CFL connectives

Operation	Interpretation

Ł-Implication	$i(P \rightarrow_L Q) = min\big(1,1 - i(p_r) + i(q_r)\big) + j.min\big(1,1 - i(p_i) + i(q_i)\big)$
Π-Implication	$i(P \rightarrow_\Pi Q) = min\big(0, i(p_r)/i(q_r)\big) + j.min\big(0, i(p_i)/i(q_i)\big)$
Π-Conjunction	$i(P \otimes Q) = i(p_r).i(q_r) + j.\big(i(p_i).i(q_i)\big)$

Table 3: Derived ŁΠ CFL connectives

Operation	Interpretation
Ł-Negation	$i(\neg P) = 1 + j1 - i(p)$
Π-Delta	$\Delta(i(p)) = 1\ if\ i(p) = 1 + j1; else\ \Delta(i(p)) = 0 + j0$
Equivalence	$i(P \leftrightarrow Q) = i(P_r \rightarrow_L Q_r) \otimes i(Q_r \rightarrow_L P_r) + j.\big(i(P_i \rightarrow_L Q_i) \otimes i(Q_i \rightarrow_L P_i)\big)$
$P \ominus Q$	$i(P \ominus Q) = max\big(0, i(p_r) - i(q_r)\big) + j.max\big(0, i(p_i) - i(q_i)\big)$

Modus ponens and product necessitation are the rule inferences of ŁΠ CFL. The first order predicate complex fuzzy logic, ŁΠ∀ CFL, extends propositional complex fuzzy logic by including predicates, constant, variables, the quantifier ∀, and the identity sign =. It follows axioms defined in [46], and also adds generalization as one of the rule inferences. Tamir then extended the approach used by Běhounek et al. in [46] and proposed a formal definition for complex fuzzy class theory (CFCT) based on ŁΠ∀ CFL [31].

2.5. Applications of Complex Fuzzy Sets

In this section, we study the rationale for employing CFS&L and then examine how CFS have been operationalized; CFS (or indeed, fuzzy sets in general) are not directly "applied" to a practical problem, but are rather a fundamental concept in constructing an intelligent system to

solve a problem (e.g., a fuzzy inferential system, or a neuro-fuzzy network). Finally, we review the specific applications that have been proposed.

2.5.1. Rationale for Complex Fuzzy Sets

Several authors have suggested that certain classes of problems could be solved more efficiently and accurately by CFS-based approaches rather than type-1 fuzzy logic. Ramot et al. suggested that problems dealing with periodic or recurring phenomena, such as representing solar activity or the effect of financial indicators on each other, can be modeled more faithfully by using the phase component of CFS memberships [1, 2]. He also suggested that signal processing was another area of possible application for CFS [1]. Dick suggested that one of the possible applications of the complex fuzzy sets is in representing phenomena with approximately periodic behaviour, (which he terms *regularity*, following some of Zadeh's concepts in a 2001 keynote [47]). These are periodic phenomena that never repeat themselves exactly such as traffic congestion in a big city [15]. [26, 48]. Kosheleva et al. found that the complex-valued fuzzy sets are computationally efficient, and the approximate nature of t-norm and t-conorm fuzzy operations in expressing experts' beliefs leads to complex-valued degrees [49, 50]. There are also a few mathematical results pointing to CFS being an efficient and useful extension to type-1 fuzzy logic. Servin et al. showed that the only 2-D distributive extensions of fuzzy logic are interval-valued and complex-valued fuzzy logic [51]. [52] showed that if a negative value is selected for p in the Schweize-Sklar implication operator, $\mu_{A\to B}(x,y|p) = 1 - \left(\mu_A(x)^{-p} + \left(1-\mu_B(y)\right)^{-p} - 1\right)^{-\frac{1}{p}}$, and no exceptions put over the negative outputs of the implication operator, the negative output can be presented by complex numbers leading to complex truth values. That also makes it possible to have different ranges of selectivity and consensus in implication operators. [53] compared

complex fuzzy sets and type-2 fuzzy sets in terms of rationale, applications, definitions, structures and operations.

The measure of any new model, ultimately, is whether or not it provides superior results compared to existing ones. Several papers have now demonstrated the utility in time-series prediction, due to its power in capturing approximately periodic behaviour [25, 54, 55]. However, other papers demonstrate the power of complex fuzzy sets in time-series forecasting from a different point of view; they took advantages of two-dimensional membership degree in the complex fuzzy sets to obtain more information about a system in order to forecast it better [27, 48, 56, 57]. Moreover, function approximation, image restoration and knowledge discovery have all been efficiently accomplished using CFS-based algorithms [18, 26, 58-60]. CFS have also been employed as a very natural mechanism for modeling bivariate time-series; this is known as the dual-output property [26] [48]. Tamir et al. showed that events with fuzzy cyclic behavior, such as the stock market, can be captured by pure complex fuzzy sets [20]. Yager demonstrated an application of Pythagorean membership grade in multi-criteria decision making [21, 22]. Alkouri et al. showed that their CAIFS are also effective in multi-criteria decision making [61]. From this summary, it is plain that there are indeed practical problems where CFS-based approaches have outperformed all other existing models.

2.5.2. Operationalizations and Applications of Complex Fuzzy Sets

[17, 25] proposed the Adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS), the first, inductive, machine-learning realization of the complex fuzzy logic proposed by Dick [15]

and Ramot [2]. ANCFIS is a six-layered network based on the well-known ANFIS system proposed by Jang; it uses the sinusoidal membership function as defined in Equations. (9) - (11) [62]. We will study this system in detail in the Chapter 3.

[18] proposed a different variation of ANFIS called the Complex Neuro-Fuzzy System (CNFS) for function approximation. The system has a 6-layer network and uses a hybrid learning of particle swarm optimization (PSO) and recursive least-squares estimation (RLSE). The system uses the basic complex fuzzy membership function introduced by Ramot et al. [1]. Consequent and premise parameters are updated by RLSE and PSO, respectively. [26] extended the CNFS proposed in [18] by using the Gaussian-type complex fuzzy set defined in Equations. (12) - (14). This paper also proposed the "dual-output property," which refers to treating the real and imaginary components of the complex output (in Cartesian form) as separate variates. [59] applied the CNFS proposed in [26] for adaptive image noise cancellation, and [58] used the artificial bee colony (ABC) algorithm instead of PSO in the CNFS and applied it for adaptive image noise cancellation as well. [60] used CNFS with ABC-RLSE learning method [58] for knowledge discovery.

[27] replaced PSO in the CNFS proposed by [26] with the hierarchical multi-swarm particle swarm optimization (HMSPSO) algorithm. That paper uses the membership function defined in Equations. (15) - (17). The algorithm was implemented for time-series forecasting. [57] applied the system proposed in [27] for financial time-series forecasting. [63] devised a new hybrid learning algorithm based on PSO-GA (particle swarm optimization and genetic algorithm) and RLSE for the CNFS proposed in [27]. [56] proposed a CNFS using the Gaussian membership function introduced in [27] which updates the premise and consequent parameters based on PSO-

RLSE learning algorithm. To minimize the rulebase of CNFS, a clustering algorithm called FCM-Based Splitting Algorithm (FBSA) is employed [64]. [48] replaces the linear consequent function with an ARIMA model [56]. It took advantage of the "dual-output property" to naturally handle bi-variate time-series. [29] uses the membership functions defined in Equations. (18) - (19) and ABC-RLSE as learning method. The system is applied for adaptive image restoration.

[28] proposed a five-layer network based on ANFIS which is called adaptive complex neuro-fuzzy inferential system (ACNFIS). The network updates its premise and consequent parameters with least square estimation (LSE) and Levenberg-Marquardt algorithm, respectively; the membership functions used in the network are defined in Equations. (20) - (22); they applied ACNFIS for function approximation.

Ma et al. [33] developed a prediction method based on complex fuzzy sets in order to solve multiple periodic factor prediction problems in multisensory data fusion applications containing semantic uncertainty and periodicity. The method, first, represents observations of each factor by complex-valued membership grade; the values are assigned based on historical knowledge of the relationship between the factors and the event. Then the complex-valued membership grades are combined by the product sum aggregation proposed in the paper (Equation. (52)). In the prediction step, a product-sum aggregation of predicted factors is calculated and compared against the model; the test input is labeled to match the most-similar observation from the model.

Alkouri et al. defined linguistic variables for complex fuzzy sets by proposing different linguistic values for uncertainty and periodicity semantics. Linguistic hedges (as introduced by Zadeh [65]) were also extended to complex fuzzy sets. Hamming, Euclidean, Normalized

Hamming, and Normalized Euclidean distances and their boundaries were also obtained for complex fuzzy sets [30] . [66] introduced complex intuitionistic fuzzy soft sets which are an extension of intuitionistic fuzzy soft sets defined in [67]; different measurements of distance between these sets, and definition of entropy for them were developed based on [68]. [69] developed the idea of parameterized fuzzy soft sets introduced in [70] based on complex fuzzy sets. [71] proposed a conceptual method for modelling residential consumption utility based on complex fuzzy sets introduced in [1], and [72] proposed using complex fuzzy sets for modelling uncertainty and seasonal features of infrastructure utility consumption in data-driven forecasting models of regional infrastructure service demands. Finally, [73] designed a hardware implementation for the CFL proposed in [2].

Applications of pure complex fuzzy classes in disaster mitigation and management (DMM), and epidemical crisis prediction (ECP) were explored in [74, 75]; as each disaster unfolds, there are simultaneous problems of uncertainty and severity. Complex fuzzy logic was proposed as a means express these two features together. Karpenko et al. studied the existence of a solution for the Cauchy problem for fuzzy differential equations that is defined on pure complex fuzzy sets [76].

[77-79] applied t-norm and t-conorm operation of complex fuzzy sets to develop a complex-valued version of simplified fuzzy ARTMAP proposed in [78] and termed it as (CV-SFAM). [80] studies single input rule modules (SIRMs) connected fuzzy inference model [81] with complex fuzzy sets and termed it as CV-SIRM model; in this model, complex fuzzy sets are applied in the antecedent part. [44] also did a survey on complex fuzzy sets with focus on axiomatic-based fuzzy logic and application of CFL.

Chapter 3

Background

In this chapter, we review architecture of the ANCFIS and all the other approaches we use to compare to our systems throughout this dissertation including adaptive neuro-fuzzy inference system (ANFIS), radial basis function network (RBFN), support vector regression (SVR) and Auto-Regressive Integrated Moving Average (ARIMA). We also study delay embedding techniques for univariate and multivariate time-series. Univariate and multivariate time-series used in this dissertation and performance evaluation criteria are introduced in Section 3.4 and 3.5.

3.1. Approaches

3.1.1. ANFIS

ANFIS is a layered feed-forward network based on Takagi-Sugeno-Kang (TSK) fuzzy inferential system [62]. A hybrid of gradient descent and least-squares estimation is used to learn the network weights. Takagi-Sugeno rules have the following format:

$if\ x\ is\ A_1 and\ y\ is\ B_1 then\ f_1 = p_1 x + q_1 y + r_1$

$if\ x\ is\ A_2 and\ y\ is\ B_2 then\ f_2 = p_2 x + q_2 y + r_2$

where x and y are input variables, $A_i$and B_i are fuzzy sets, and f_i is a linear function of the input variables. Figure 2 shows ANFIS network structure [62] :

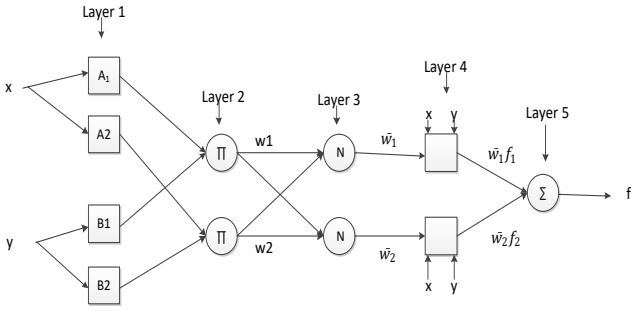


Figure 2: Equivalent ANFIS [62]

Each layer has a different transfer function, as follows:

- *Layer 1:* the output of *i*-th node of layer 1, $O_{1,i}$, is defined as the membership degree of an input in a specific fuzzy set:

$$O_{1,i} = \mu_{A_i} \quad (91)$$

where μ_{A_i}, is a membership function in one input dimension (usually we form a fuzzy partition of each input dimension).

- *Layer 2:* This layer implements the product t-norm, giving us the firing strength of the k-th rule w_i .

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), \quad i = 1, \dots, n \quad (92)$$

where n is number of rules.

- *Layer 3:* Nodes of this layer normalize the firing strength of each rule.

$$\bar{w}_i = \frac{w_i}{\sum_{j=1}^{n} w_j}, \qquad i = 1, .., n \quad (93)$$

where n is number of rules.

- *Layer 4:* This layer determines the output of each rule as a weighted linear function of the inputs.

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i) \quad (94)$$

where $\overline{w_i}$ is the output of layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set.

- *Layer 5:* This layer sums the output of each rule, yielding the network output.

$$O_{5,i} = overaloutput = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (95)$$

In the network, there are two kinds of parameters that can be updated. Antecedent parameters (parameters of membership functions) are found in Layer 1. For example, in the Gaussian membership function:

$$\mu_i(x) = \frac{-(x - c_i)^2}{2{\sigma_i}^2} \quad (96)$$

$\{c_i, \sigma_i\}$ are updatable parameters. Consequent parameters are found in Layer 4; these are the coefficients of the linear consequent function $\{p_i, q_i, r_i\}$. Jang et al. [62] determined that using only gradient descent in updating parameters is slow and susceptible to being trapped in local minima. He proposed implementing hybrid learning procedure in training network in order to solve the issues. Consequent parameters are updated in the forward pass by applying least square optimization, while membership function parameters (antecedent parameters) are updated in the backward pass by gradient descent.

3.1.2. RBFN

The Radial Basis Function Network (RBFN) is a single hidden layer neural network where the hidden layer has nonlinear transformation and output layer is linear:

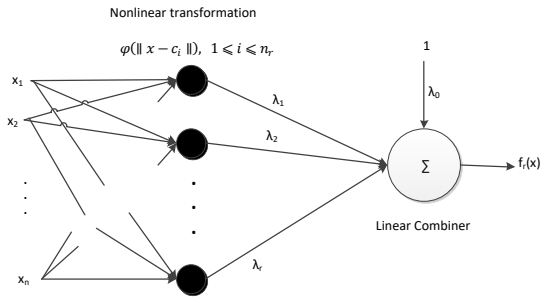


Figure 3: Schematic of RBF network[83]

The mapping used in the network from inputs to outputs is [83]:

$$f_r(x) = \lambda_0 + \sum_{i=1}^{n_r} \lambda_i \varphi(\| x - c_i \|) \quad (97)$$

where $x \in R^n$ is the input vector, $\varphi(.)$ is a given function from $\mathbb{R}^+$ to $\mathbb{R}$, $\|.\|$ denotes the Euclidean norm, λ_i are coefficients $0 \leqslant i \leqslant n_r$, $c_i \in R^n$ are the RBF centers, $1 \leqslant i \leqslant n_r$, $n_r$is the number of centers, and $\varphi(.)$ can be a Gaussian function [83]:

$$\varphi(x) = exp\left(\frac{-\| x - c_i \|^2}{\beta^2}\right) \quad (98)$$

where β is the spread parameter. We use MATLAB implementation (newrb.m) in which centers are selected iteratively from the given data set, and least square is used to determine the weights [83].

3.1.3. SVR

Support vector regression (SVR) estimates a function from training data, $\{(x_1, y_1), \dots, (x_l, y_l)\} \subset \mathbb{R}^d \times \mathbb{R}$, as [84]:

$$f(x) = \langle w, x \rangle + b \quad (99)$$

where $w \in \mathbb{R}^d$ indicates weights, $b \in \mathbb{R}$ is bias, and $\langle .,. \rangle$ is the dot product in $\mathbb{R}^d$. The aim is finding w and b such that the following constraint is satisfied [84, 85]:

$$minimize \ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} L(y(i), f(x(i), w)) \quad (100)$$

where the first part, $\|w\|^2 = \langle w, w \rangle$, satisfies flatness of the function. The second part assures that the error between the predicted values by the linear regression, $f(x)$, and actual ones is not more than ε, a user-defined parameter. $C > 0$ is also a user-defined parameter, and $y(i)$ is the desired

value. $L(.)$ is a loss function defined as [85]. We use the *e1071* implementation of SVR in the R environment [86].

3.1.4. ARIMA

The Auto-Regressive Integrated Moving Average (ARIMA) model is a combination of two models: the moving average and the autoregressive models, along with differencing of the data for stationarity. An Autoregressive model of order p, AR(p), is defined as [87].

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t \quad (101)$$

where e_t is white noise, c is a constant, and $\phi_i, i = 1,2, \dots, p$, are parameters of the model. AR(p) is a linear regression of p past values of the time-series. Moving average of order q, MA(q), is defined as [88]:

$$y_t = c + e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} \quad (102)$$

where c is a constant, e_{t-q} is the one-step forecast error for qth past value, and $\theta_i, i = 1,2, \dots, q$ indicate parameters of the MA model. The ARIMA (p,d,q) model is given by [87]:

$$\left(1 - \phi_1 B - \cdots - \phi_p B^p\right)(1 - B)^d y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q) e_t \quad (103)$$

where d is the order of differencing, and B is the backshift operator defined as $B^k y_t = y_{t-k}$.

3.1.5. ARFIMA

ARFIMA (Autoregressive Fractionally Integrated Moving Average) is a generalization of the ARIMA (Autoregressive Integrated Moving Average) model. An ARIMA (*p,d,q*) model can be represented as:

$$\left(1 - \sum_{i=1}^{p} \alpha_i L^i\right).(1 - L)^d X_t = \left(1 + \sum_{j=1}^{q} \beta_i L^i\right).\varepsilon_t \quad (104)$$

where p is the order of the auto-regression part of the model, α_i is the auto-regression coefficient of the i-th lag, L is the lag operator, d is the order of differencing, X_t is the t-th element of a time-series, q is the order of the moving-average portion of the model, β_i is the moving-average coefficient of the i-th lag, and ε_t is the t-th error term; errors are assumed to be independent, identically distributed random variables drawn from a zero-mean normal distribution [89]. In the ARIMA model, p, d, and q are integers, but in ARFIMA model, d is allowed to take non-integer values [90]. This makes ARFIMA models particularly useful for self-similar time-series, as the parameter d is directly related to the well-known Hurst parameter, which quantifies self-similarity.

3.2. ANCFIS

[17, 25] proposed the first inductive machine learning realization of the complex fuzzy logic proposed by Dick [15] and Ramot [2]. The ANCFIS architecture is a relative of Jang's well-known ANFIS [62]. The main differences are 1) ANCFIS uses a sinusoidal membership function as follows:

$$r(\theta) = dsin(a(\theta = x) + b) + c \quad (105)$$

where $r(\theta)$ is amplitude and θ is the phase of the membership grade of element x; 2) an additional layer implements rule interference, inspired by Ramot's vector aggregation (Equation. (51)) [2]; 3) network signals are complex-valued up through this rule interference layer; 4) the learning algorithm incorporates a derivative-free optimization component.

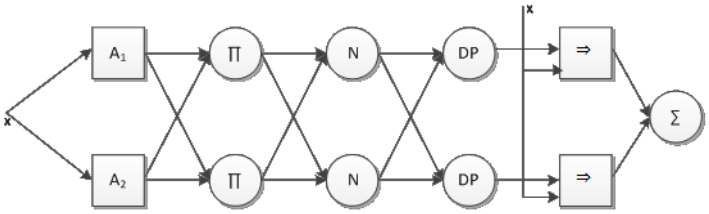


Figure 4: Two-rule ANCFIS architecture for univariate time-series problems [25]

As suggested by Dick [15], one possible application of complex fuzzy logic is capturing approximately periodic behaviour of phenomena; [25] suggested that time-series forecasting is a good example of such behaviours. Thus, sinusoidal functions are candidate complex fuzzy membership functions since a periodic function can be presented by a Fourier series, i.e. a sum of sin and cosine functions. In Equation. (105), the four parameters $\{a, b, c, d\}$ act as follows: a changes the frequency of the sine wave, b gives a phase shift whereas c shifts the wave vertically, and d changes the amplitude of the sine wave. Since the amplitude of complex fuzzy memberships is limited to [0,1], the parameters must satisfy the following conditions:

$$0 \leqslant d + c \leqslant 1, \qquad 1 \geqslant c \geqslant d \geqslant 0 \tag{106}$$

The use of a sinusoidal CFS in ANCFIS also implies an important operational difference between ANCFIS and ANFIS (and indeed most other machine learning algorithms). In using ANFIS and other algorithms for time-series forecasting, input vectors containing lagged values of a variate are presented to the network to predict next value of the variate. The components of the input vectors are considered orthogonal; thus to predict $f(t)$, the components $f(t-1), f(t-2), \dots, f(t-n)$ of an input vector are presented as separate inputs to the system. However, this cannot work in ANCFIS, because matching a sinusoidal membership function to an observation requires that we keep the phase information in our inputs. Orthogonal lagged inputs destroy this phase information by definition. Instead, in ANCFIS, we take a sliding window of the variate as a single input, $[f(t-1), f(t-2), \dots, f(t-n)]$, and then match that window to the membership functions. This implies that ANCFIS requires only a single input for each variate of a time-series, whereas systems using a lagged input require $\prod_{i=1}^{n} r_i$ inputs, where r_i is the number of lags for a

given variate, and n is the number of variates. Thus, ANCFIS significantly reduces the combinatorial explosion inherent in time-series forecasting. [25].

The ANCFIS architecture has 6 layers as follows [25]:

- *Layer 1*: In this layer, the input vector is convolved with the membership function. First, the membership function is sampled over one period by:

$$r_k(\theta_k) = dsin(a\theta_k + b) + c, \quad \theta_k = \frac{2\pi}{n}k, \quad k = 1,2,\dots,n \tag{107}$$

where n is the length of the input vector. Then, the sampled membership functions are convolved with the input vector:

$$conv = \sum_{k=1}^{2n-1} \sum_{j=\max(1,k+1-n)}^{\min(k,n)} f(j)g(k+1-j) \tag{108}$$

where $f(.)$ is the input vector, and $g(.)$ is the sampled membership function (in Cartesian coordinates). To ensure that the convolution sum remains within the unit disc, it is normalized using the Eliot function:

$$O_{1,i} = \frac{conv}{1+|conv|} \tag{109}$$

- *Layer 2*: In this layer, the firing strength of a fuzzy rule is calculated:

$$O_{2,i} = \prod_i O_{1.i}, i = 1,2,\dots,|O_1| \tag{110}$$

where $|O_1|$ is the number of nodes in layer 1. For univariate time-series, neurons in this layer reduce to identity function.

- *Layer 3*: The output of each node represents the normalized firing strength of a rule

$$O_{3,i} = \bar{w}_\iota = \frac{w_i}{\sum_{j=1}^{|O_2|} |w_j|}, \quad i = 1,2,\ldots,|O_2| \tag{111}$$

where $|O_2|$is the number of rules. This layer only normalizes the magnitude whereas phases are unchanged.

- *Layer 4*: This layer realizes the property of "rule interference" from [2], using the dot product.

$$O_{4,i} = w_i^{DP} = \bar{w}_\iota \cdot \sum_{i=1}^{|O_3|} \bar{w}_\iota \tag{112}$$

  where $|O_3|$ is the number of nodes in layer 3 and $\sum_{i=1}^{|O_3|} \bar{w}_\iota$ is the complex sum. Both constructive and destructive interference are possible.

- *Layer 5*: This layer implements the linear consequent function:

$$O_{5,i} = w_i^S = w_i^{DP}\left[\sum_{j=1}^{n} p_{i,j}x_j + r_i\right] \tag{113}$$

  where w_i^{DP} is the output of layer 4, x_j is the jth data point if the input vector, *n* is the length of the input vector, and $p_{i,j}$, r_i are the parameters of a linear function of x_j. $\{p_{i,j}, r_i\}$ are obtained in the forward pass by least squares estimation.

- *Layer 6*: This layer sums all incoming signals.

$$O_{6,i} = \sum_{i=1}^{N} w_i^S \tag{114}$$

Where *N* is number of rules.

As the network signals in ANCFIS are complex-valued, the backward-pass computations in the network must also be different from ANFIS. Like ANFIS, ANCFIS uses a hybrid learning rule where consequent parameters are updated on the forward pass, and antecedent parameters on

the backward pass. Indeed, as network signals are real-valued at the consequent layer (layer 5 in ANCFIS), we employ the same least-squares algorithm as ANFIS. However, the backward pass requires back-propagation of complex-valued signals; and ultimately there is no closed-form expression for the partial derivative of network error with respect to the CFS parameters in Equation. (105). As described in [25], we use gradient descent to determine the back-propagating error signals until Layer 1, and then use a derivative-free optimization technique (a variant of simulated annealing) to determine the update to the CFS parameters. This technique is Variable Neighbourhood Chaotic Simulated Annealing (VNCSA) algorithm. In chaotic simulated annealing (CSA), the generation of new candidate solutions is governed by a chaotic map instead of a random number generator. This potentially makes the algorithm faster, as we only search a fractal subset of the total solution space.

Calculating the partial derivatives of the membership parameters is done in two steps. First, the derivatives from $O_{1,i}$ to the sampled membership functions, $g(.)$ in Equation. (108) is calculated:

$$\frac{\partial O_{1,ji}}{\partial g(k)} \qquad k = 1,2,..,n \tag{115}$$

where n is the number of samples. Updates for the samples, $g(k)$, are calculated as:

$$g(k)_{new} = g(k)_{old} - \eta \frac{\partial E}{\partial g(k)} \tag{116}$$

where $\frac{\partial E}{\partial g(k)}$ is gradient descent of error with respect to the sampled membership functions, and η is the learning rate parameter. Then, the updated samples $g(k)_{new}$ are given as input to VNCSA to obtain the parameters $\{a, b, c, d\}$ based on the following optimization function:

$$E(a,b,c,d) = \sum_{m=1}^{r} [mag(g(k)_{new})_m - (d * \sin(a * phase(g(k)_{new})_m + b) + c)]^2 \quad (117)$$

where r is the number of sampled membership functions, and $mag(g(k)_{new})_m$ and $phase(g(k)_{new})_m$ are the amplitude and phase of the *m-th* updated sample.

VNCSA is a chaotic simulated annealing algorithm; like other algorithms in this class, its goal is to reduce the size of a search space by limiting the search to a fractal subset of the space. This is accomplished by replacing random number generation in classic simulated annealing with chaotic maps (which are deterministic but highly irregular functions). VNCSA is initialized using the logistic map [25]:

$$x_{i,j+1} = 4 \times x_{i,j} \times (1 - x_{i,j}) \quad (118)$$
$$x_{i,j} = x_{i,j+1} \qquad i = 1,2,3,4$$

where i is the number of variables in the objective function (Equation. (117)), and j is the number of iterations of the logistic map. The initial population is constrained to lie within the lower and upper bound of the variables as:

$$S_{i,j+1} = f_i + (k_i - f_i) \times x_{i,j+1} \quad (119)$$

where f_i and k_i are the lower and upper bound, respectively.

After initialization, at each temperature in the annealing schedule [82], L_{max} iterations of the following process are performed:

An Ulam-von Neumann map is used to generate candidate solutions based on the initial population as:

$$S_{ij+1}^{new} = S_i^{current} + D_i y_{i,j+1} \quad (120)$$

where $S_i^{current}$ is updated by S_{ij+1}^{new} based on simulated annealing (SA). $i = 1,2,3,4$ is the number of variables in the objective function, $j = 1,2, \dots, M$ is iterations of the Ulam-von Neumann map

and $S_i^{current}$ is initialized to the initial population. $y_{i,j}$ is the random numbers generated by the Ulam-von Neumann map and D_i is the neighbourhood defined as

$$y_{i,j+1} = 1 - 2{y_{i,j}}^2 \quad (121)$$

$$y_{i,j} = y_{i,j+1}$$

$$D_i = 0.1\times(k_i - f_i) \quad (122)$$

After the $L_{max}\times M$ iterations, the neighborhood and the temperature are updated as below, respectively,

$$D_i^{new} = (1-\alpha)D_i^{current} + \alpha\omega R_i \quad (123)$$

where R_i is the magnitude of the successful change made to the *i-th* variable, α is the damping constant controlling the rate at which information from R_i is folded into D_i with weighting ω.

$$T_{k+1} = T_k\beta \quad 0 < \beta < 1 \quad (124)$$

where β is a constant factor used to lower the temperature in each of the iterations.

3.3. Delay Embedding of a Time-series

A variety of authors have employed machine learning for time-series forecasting, e.g. [91-95]. All of these works share a common assumption: that the time-series is deterministic. Specifically, this means that evolution of the system observed to collect the time-series follows a single trajectory in state space, and knowledge of the past trajectory and current state allow the next state to be uniquely determined; each trajectory corresponds to one evolution of the system based on a given initial state. However, the given time-series does not provide information about either the system or the state space, but is merely a sequence of observations based on the system's evolution. Thus, to forecast time-series, we need a method to reconstruct the state space by using data points in the time-series [96].

Delay embedding is a common technique applied on a time-series to obtain delay vectors of it; each delay vector is a lagged representation of the time-series where the *dimension* and *delay* refer to the number of the previous observations in the vector, and the number of actual observations occurring between successive components of a delay vector. Each of the delay vectors is a point in a reconstructed state space. According to Takens' theorem [96], if the selected dimension is appropriate, the state space constructed by the delay vectors is equivalent to the original state space of the time-series, meaning that prediction of the time-series from a trajectory in the reconstructed state space is possible.

Delay vectors in a univariate time-series have the form [97]

$$S_m = (s_{m-(n-1)\tau}, s_{m-(n-2)\tau}, \dots, s_m) \quad (125)$$

where S_m is a delay vector, whose most recent component is the *m*-th element of the time-series S_m, s_i is the *i*-th element of the time-series, and the delay vector contains *n* components, each separated by τ-1 elements in the time-series.

Taken's embedding theorem does not, however, provide a constructive method for determining the parameters n and τ; instead we need to use heuristics to determine *adequate* values for both parameters. Mathematically, embeddings with different τ are equivalent to each other; however, in real-world data, the choice of the delay parameter has a significant influence on the utility of an embedding. Small values of τ generally lead to higher correlations between observations in each delay vector; and thus the distribution of delay vectors (and hence the apparent state-space trajectory) tend to be concentrated in a small region of the embedding space, potentially obscuring important features of the trajectory. On the other hand, large values of τ tend to make observations in a delay vector poorly correlated. This tends to result in the delay vectors

becoming a weakly differentiated cloud of points, with little apparent structure. For univariate data sets, commonly-used heuristics for determining the delay τ include the first zero of the autocorrelation function, or the first minimum of the time-delayed mutual information function [96]. Equations. (126) - (127) show autocorrelation function and time delayed mutual information, respectively [96]:

$$c_\tau = \frac{1}{\sigma^2}\langle (s_n - \langle s \rangle)(s_{n-\tau} - \langle s \rangle) \rangle \quad (126)$$

where c_τ is the autocorrelation between values of s_n and $s_{n-\tau}$ where there is a time lag of τ between them. $\langle . \rangle$ indicates average over time, and σ^2 denotes the variance.

$$I(\tau) = \sum_{i,j} p_{ij}(\tau) \ln p_{ij}(\tau) - 2\sum_i p_i \ln p_i \quad (127)$$

where $I(\tau)$ is the mutual information between s_n and $s_{n-\tau}$. By considering histogram of s_n, p_i is the probability that s_n has value in the ith interval, and p_{ij} is the joint probability that s_n has value in the ith interval and $s_{n-\tau}$ has values in the jth interval.

Kennel et.al [98] proposed applying the false nearest neighbors technique to determine dimension for univariate time-series. The idea is that if a delay vector is mapped to a neighborhood in the embedding space, its one-step-ahead evolution is also mapped to the one-step-ahead evolution of that neighborhood. If not, then that point only seemed to fall in the neighborhood due to unresolved projections of the true embedding space (it is a "false neighbor"). The Euclidean distance between one delay vector and its *r*th nearest neighbor in the embedding space of dimension *m* is given by [98]:

$$R_d^2(n,r) = \sum_{k=0}^{m-1} [s_{n-k\tau} - s_{n-k\tau}^r]^2 \quad (128)$$

where R_d is the Euclidean distance, and $s_{n-k\tau}$ are elements of the delay vector in the embedding space. When the dimension of the embedding space increases to m+1, the delay vectors have one more coordinate which is $s_{n-m\tau}$. The Euclidean distance between the delay vectors in the new embedding space is calculated as [98]:

$$R_{d+1}^2(n,r) = R_d^2(n,r) + [s_{n-m\tau} - s_{n-m\tau}^r]^2 \quad (129)$$

Thus the false nearest neighbor method can be stated as the following criterion [98]:

$$\frac{|s_{n-m\tau} - s_{n-m\tau}^r|}{R_d(n,r)} > R_{tol} \quad (130)$$

where R_{tol} is a threshold. That means increasing the embedding dimensionality must not increase the distance between two neighbors more than the given threshold. The estimated number of dimensions is determined by plotting the fraction of false nearest neighbors in the data set against the number of dimensions, for several different values of the threshold R_{tol}. When all of the curves saturate at a low value, we consider that to be the best estimate of the necessary embedding dimensionality for the data set.

Delay vectors of an M-variate time-series with length of N, $X_1, X_2, \ldots, X_N$ where $X_i = (x_{1,i}, x_{2,i}, \ldots, x_{M,i})$, are given by [99]:

$$V_n = \begin{matrix} (x_{1,n}, x_{1,n-\tau_1}, \cdots, x_{1,n-(d_1-1)\tau_1}, \\ x_{2,n}, x_{2,n-\tau_2}, \cdots, x_{2,n-(d_2-1)\tau_2}, \\ \cdots \quad \cdots, \\ x_{M,n}, x_{M,n-\tau_M}, \cdots, x_{M,n-(d_M-1)\tau_M}) \end{matrix} \quad (131)$$

where V_n is the delay vector having $= (x_{1,n}, x_{2,n}, \ldots, x_{M,n})$ as its most recent components from each variate, and the i-th variate has delay τ_i and dimension d_i $(i = 1, 2, \ldots, M)$, respectively. In other words, the delay vector for a multivariate time-series is formed by concatenating delay embeddings for each variate together. The same heuristics for the delays τ_i from univariate time-

series analysis are applied to each variate separately [99-101]. However, the dimensionality of the variates is perhaps better determined together. Cao et.al [99] proposed a nearest neighbor predictor to find dimensions, d_i, for a multivariate time-series. First, for a given set of dimensions and delays for each delay vector, $(\tau_i, d_i \quad i = 1,2, \dots, M)$, its nearest neighbors are determined using the Euclidean norm; second, their one-step prediction error is calculated. Then, the set of dimensions with the lowest prediction error is used as the embedding dimension. As we employ the KD-tree algorithm for finding nearest neighbors, we refer to Cao's method as "KDD" hereafter.

To obtain dimension for univariate time-series, we use TISEAN software package [102]. Dimension for multivariate time-series is calculated using Ruby code we have developed. Delay for both univariate and multivariate are calculated by Tisean.

3.4. Time-series

To divide time-series to training set and testing set, we follow a chronologically ordered single-split design in which observations in the training set occur earlier that those ones in the testing set.

3.4.1. Univariate Time-series

3.4.1.1. Solar Power Data Set

This data set was created in [55], as very few solar-power data sets are publicly available. It was developed from a public data set recording air temperature (℃) and total solar radiation (W/m^2) measured every one minute from May 30, 2008 to August 12, 2012 at the Lowry Range Solar Station. Total solar radiation is the sum of direct irradiance, diffuse irradiance, and ground-

reflected radiation, and is measured by a LICOR LI-200 Pyranometer mounted 7 feet above ground level on a Rotating Shadow Band Radiometer (RSR). Air temperature is measured by a thermometer mounted 5 feet above ground level inside a naturally aspirated radiation shield [103]. These two measurements are the principal variables affecting power output from a photovoltaic cell; we convert them to an estimated power output using the model proposed in [104], following the specifications of a Photowatt PW 2650-24V panel. The result is a new time-series recording solar-power production at one-minute intervals over a period of five years, giving over two million observations, 22122520 observations. For our experiments in this dissertation, we use data from July 31, 2012 to August 13, 2012, giving us 20000 measurements. The data set is split to 2/3 and 1/3 for training and testing set, respectively.

3.4.1.2. Santa Fe Laser A Data Set

The Santa Fe time-series forecasting competition held in 1991 has left us six data sets for use as benchmarks. The "Laser A" data set is frequently used, as it also exhibits chaotic behavior. This data set records the amplitude of an 81.5-micron 14NH3 cw (FIR) laser being controlled by the Lorenz system of equations for modeling turbulent flow; with appropriate choices of parameters, the Lorenz system is also chaotic. The data set has 1000 data points; we normalize it to the range of [0,1] and consider the first 900 data points as the training set, and the last 100 as the testing set

3.4.1.3. Stellar Data Set

This data set is also made up of observations of a physical system. The time-series records the daily observed brightness of a variable star on 600 successive midnights. We normalize the

data points to the range of [0 1] and consider the first 480 night measurements as the training set, and the remainder form the testing set.

3.4.1.4. Mackey-Glass Data Set

This data set is a realization of the Mackey-Glass differential equation, a frequently-used benchmark for testing time-series forecasting algorithms. The equation is given by [62]:

$$\dot{x}(t) = \frac{0.2x(t-\delta)}{1+x^{10}(t-\delta)} - 0.1x(t) \quad (132)$$

This equation is useful because it exhibits chaotic behavior for appropriate choices of the parameter δ. In particular, we follow the design in [62], where 1000 data points are generated for $124 < t < 1123$ and then normalized to the range of [0,1] with $x(0) = 1.2$, $\tau = 17$, and the time step = 0.1. The first 500 data points are used as the training set and the remaining data points as the testing set.

3.4.1.5. Sunspot Data Set

This data set consists of observations of a physical system: it is the average number of sunspots observed for each day in a calendar year, recorded from 1700-1979 (280 observations) [105]. We normalize the time-series to the range of [0 1] and consider the first 220 year measurements, years 1700-1920, as the training set and the remaining points as the testing set.

3.4.1.6. Wave Data Set

This data set also records observations of a physical system, but this time in a laboratory setting. The time-series measures the oscillation of a cylinder suspended in a tank of water every 0.15 s. There are a total of 320 data points of which the first 256 data points form the training set and the last 64 points are the testing set.

3.4.1.7. Software Reliability Growth Time-series

We study four univariate software reliability growth data sets. The time-series originally contain bug reports; however, we have converted them to interfailure times data sets in [10]. The time-series normalized to the range of [0 1] and split to 2/3 and 1/3 as training and testing sets, respectively.

3.4.1.7.1. Mozilla Data Set

Mozilla time-series was derived from the Bugzilla defect-tracking database used by the Mozilla project. The time-series has 86,077 data points recording bug reports from1999 to 2003 which gives 57,385 data points as training and 28,692 data points as testing set.

3.4.1.7.2. Android Data Set

Android time-series was extracted from bug reports and changes of 2012 Mining Software Repositories Challenge data set which has 20,168 data points. 13,457 data points are in the training set and 6,729 data points as the testing set.

3.4.1.7.3. ODC1 & ODC4 Data Sets

ODC1 and ODC4 are interfailure times data sets that we develop based on bug reports collected by IBM Corporation during their Orthogonal Defect Classification project. ODC1 and ODC4 have 1207 and 2008 bug reports, respectively [106]. ODC1 has 805 data points in the training set and 402 data points in the testing set. Training and testing sets in ODC4 have 1339 and 669 data points, respectively.

3.4.2. Multivariate Time-series

3.4.2.1. Motel Data Set

This bivariate time-series contains monthly records of occupancy of hotels, motels and guest room in Victoria, Australia over the period Jan 1980- June 1995 (186 data points) [107]. Its variates show the total number of room nights occupied and total revenues (thousands of dollars). We split the time-series into 2/3 and 1/3 for training and testing set giving 124 data points in the training and 62 data points in the testing sets and normalize them to the range of [0,1].

3.4.2.2. Precipitation Data Set

This trivariate time-series records the monthly precipitation of the east, middle and west regions of the state of Tennessee from 1895-1929 (420 data points) [108]. We consider data from Jan 1895 to Dec 1924 as the training set (360 data points) and data from Jan 1925 to Dec 1929 (60 data points) as the testing set and normalize them to the range of [0,1]; this split is same as [109] in order to compare their results.

3.4.2.3. Flour Data Set

This multivariate time-series has three variates recording the monthly average flour price for commodity exchanges in Buffalo, NY, USA, Minneapolis, MN, USA, and Kansas City, KS, USA over nine years, 1972-1980, (100 data points) [107]. We split the time-series into 90 data points in the training and 10 data points in the testing set and normalize them to the range of [0,1]; the split is the same as [110] in order to facilitate comparisons.

3.4.2.4. NASDAQ Data Set

The NASDAQ time-series records the daily opening and closing indices of the NASDAQ composite index from January 3, 2007 to December 20, 2010 (1000 data points) [111]; we split the time-series equally giving us 500 data points in the training and 500 data point in the testing set.

3.5. Performance Evaluation

Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) statistics are used to compare experiment results.

$$\mathrm{RMSE} = \sqrt{\frac{1}{K}\sum_{i=1}^{K} MSE_i} \qquad (133)$$

$$MSE_i = \frac{\sum_{\mathrm{j}=1}^{\mathrm{n}}\left(\mathrm{y_j} - \widehat{\mathrm{y}_1}\right)^2}{\mathrm{n}}$$

$$\mathrm{MAE} = \frac{1}{K}\sum_{i=1}^{K} MAE_i \qquad (134)$$

$$MAE_i = \frac{\sum_{\mathrm{j}=1}^{\mathrm{n}}\left|\mathrm{y_j} - \widehat{\mathrm{y}_1}\right|}{\mathrm{n}}$$

where K is the number of variates in the multivariate time-series and MSE_i and MAE_i are mean squared one-step-ahead error and mean absolute one-step-ahead of i-th variate with the length of n, where $\mathrm{y_j}$ shows actual value and $\widehat{\mathrm{y}_1}$ is the desired value.

3.5.1. Significant Test

Statistical test can be used to compare machine learning algorithms statistically. Demsar [112] recommended using non-parametric statistical tests for statistical comparison of machine

learning algorithms. He suggests using the Wilcoxon signed ranks test when comparing two machine learning algorithms and Friedman test when comparing performances of multiple machine learning algorithms on multiple data sets. In this dissertation, as we compare our designed machine learning algorithms with various well-known algorithms and apply them on several time-series, we use Friedman test as the statistical test.

3.5.1.1. Friedman Test

The Friedman statistic, *S*, is calculated as [113]:

$$S = \frac{12n}{k(k+1)} \sum_{j=1}^{k} \left(\bar{R}_j - \frac{k+1}{2} \right)^2 \quad (135)$$

where *k* is the number of approaches applied on *n* different time-series, and $\bar{R}_j$ is the average rank of the *j*-th method obtained by applying the method on the *n* different time-series. The statistic *S* tests the null hypothesis: $H_0: [\tau_1 = \cdots = \tau_k]$ against the alternative hypothesis: $H_1: [\tau_1, \ldots, \tau_k \; not \; all \; equal]$ at the α level of significance where τ_i is the *i*-th method effect. For either *n* or $k > 5$, *S* is approximately chi-square distributed with *k*-1 degree of freedom if H_0 is true; therefore, H_0 is rejected if $S \geq \chi^2_{k-1,\alpha}$where $\chi^2_{k-1,\alpha}$ is the upper α percentile point of a chi-square distribution with *k*-1 degrees of freedom [113, 114]. $\chi^2_{k-1,\alpha}$ is obtained by the *NSM3* package in the R environment [115].

If null hypothesis $H_0: [\tau_1 = \cdots = \tau_k]$ is rejected, the approaches are compared together using the Multiple Comparisons with the Best (MCB) method [116]. The null hypothesis is defined as:

$H_0: \tau_u = \tau_v$ where $1 \leq u < v \leq k$.

H_0 is rejected if $|\bar{R}_u - \bar{R}_v| \geq r_{\alpha,K,N}$ where $\bar{R}_u$ is the average rank of the u-th method, and for large-sample approximation $r_{\alpha,K,N} \approx q_\alpha \sqrt{\frac{k(k+1)}{12n}}$ where q_α is the α percentile point for the distribution of the range of k independent standard normal variables. q_α is estimated by the *NSM3* package in R environment [115].

Chapter 4

Input Representation for ANCFIS

ANCFIS is a neuro-fuzzy system that employs complex fuzzy sets for time-series forecasting. One of the particular advantages of this architecture is that each input to the network is a windowed segment of the time-series, rather than a single lag as in most other neural networks. This allows ANCFIS to predict even chaotic time-series very accurately, using a small number of rules. Some recent findings, however, indicate that published results on ANCFIS are sub-optimal; they could be improved by changing how we define an input window, or even using sub-sampled windows.

In this chapter, we compare the performance of ANCFIS using three different approaches to defining an input window, across six time-series data sets. These include chaotic data sets and time-series up to 20,000 observations in length.

4.1. Introduction

Time-series forecasting has emerged as the first major application of complex fuzzy sets and logic, which were first described by Ramot in [1]. Beginning in 2007, complex-valued neuro-fuzzy systems were developed to inductively learn forecasting models; these include the ANCFIS architecture [25], and the family of Complex Neuro-Fuzzy System (CNFS) architectures [18]. Both ANCFIS and CNFS are modifications of the well-known ANFIS architecture, in which complex fuzzy sets and complex-valued network signals are used. These architectures showed that complex fuzzy sets were naturally useful in creating very accurate forecasting models. ANCFIS

in particular is also very parsimonious; experiments in [25] showed that this architecture could forecast even chaotic systems with no more than three complex fuzzy rules.

One of the key reasons why ANCFIS is so parsimonious is its input format. Most generic machine learning algorithms must use lagged inputs in order to create a forecasting model. While this approach is mathematically sound, it means that the number of inputs to the learning algorithm has to be equal to the number of lags required to reconstruct the state space of the system that generated the time-series (i.e. to form a delay reconstruction in the sense of Takens [96]). This directly leads to a combinatorial explosion in the complexity of the model. However, due to the nature of complex fuzzy sets, ANCFIS does not use lagged inputs; rather, an entire windowed segment of the time-series is taken as a single input to the network, greatly reducing the curse of dimensionality. Recent experiments reported in [55] indicated that we might be able to further improve the accuracy of ANCFIS by sub-sampling the input windows. This is possible because in ANCFIS, we use sinusoidal membership functions for the complex fuzzy sets, which are sampled and convolved with the input window. Sub-sampling the input window simply implies that we also sample the complex fuzzy sets at a lower rate. Our goal in the current chapter is to determine if such sub-sampling generally leads to improved accuracy, or if this was a data set-specific effect.

We compare the forecast accuracy of ANCFIS using three different approaches to identifying and sampling input windows on six time-series data sets. Two of these (a realization of the Mackey-Glass map, and the Santa Fe Laser A data set) are known to be chaotic; the remainder are observations of physical processes (sunspots, stellar brightness, waves, solar power production). All but the last one have been previously studied in the forecasting community and in [25]; the solar power data set was developed in our laboratory, and is discussed in depth in [55]. In [25] the length of the input windows for each of the five data sets was set at one “period” in the

data set, as determined by ad-hoc inspection. We additionally explore the use of heuristics from [96] to construct two different delay embeddings of the time-series: one with an "optimal" delay between each lag, and one with the delays between each lag fixed at one. These input lags are concatenated together in chronological order to form our input windows.

4.2. Methodology

4.2.1. Experimental Design

The goal of this section is to evaluate alternative time-series representations in forecasting with ANCFIS. We explore three different approaches for setting the length of the input windows and sub-sampling them. The first is the approach used in [25], the second is the delay embedding technique from section 3.3, and the third is a hybrid of the two. Specifically:

- Method 1 is to make an ad-hoc determination of the length of one "period" in the data set. The input window is set to this length, and is not sub-sampled. As this method was used in [25] for five of the six data sets, we will use the same period lengths as in that paper. For the sixth data set (solar power forecasting) the length of a period is clearly 1 day (see our discussion in Section 4.2.1.1 for further details).
- Method 2 is to form a delay embedding, relying on the heuristics from Section 3.3 to guide our selection of the embedding dimensionality and delay. We will use the mutual-information heuristic to select the delay, and the false-nearest-neighbors technique to select the dimensionality. We can consider this a sub-sampling of an input window; for dimensionality m and delay τ, we select every τ-th sample from an input window of length $((m + 1) \cdot \tau) + 1$.

- Method 3 is to assume that the delay is always equal to 1, and to employ the false-nearest-neighbors technique for selecting the embedding dimensionality under that assumption. This will mean that the input window is again not sub-sampled.

  The embedding dimension and delay are determined from the training set only, and are then applied to both the training and testing sets. The results of the three input representations are compared in terms of RMSE.

4.2.1.1.Solar Power Data Set Representation

In this data set discussed in Section 3.4.1.1, the length of one period is clearly one day, or 1440 observations. When we attempted to take this entire period as an input window for the method 1, we found that the computation time is infeasibly long on our computer system (Intel(R) Core™ 2 Duo CPU E8400 @ 3.00GHz, 4GB of memory). We were thus forced to sub-sample this window; we take every tenth measurement, giving us 144 observations, which we concatenate together in chronological order. For Method 2, we plot the mutual information statistic versus delay in Figure 5. The first minimum of the curve occurs at $\tau = 370$, and so we adopt this value as our delay parameter. With this delay, the false-nearest-neighbour plot is given in Figure 6. For all values of R_{tol} examined (see Equation. (130)), the curves saturate at m = 12, and so we adopt this value as our embedding dimension. We can also view this input as a subsampled window of length $(m-1)\tau = 4070$ data points. For Method 3, we set the delay $\tau = 1$, and re-run the false-nearest-neighbour procedure. This time, the apparent minimum embedding dimension is 15, and so we adopt this as our window length.

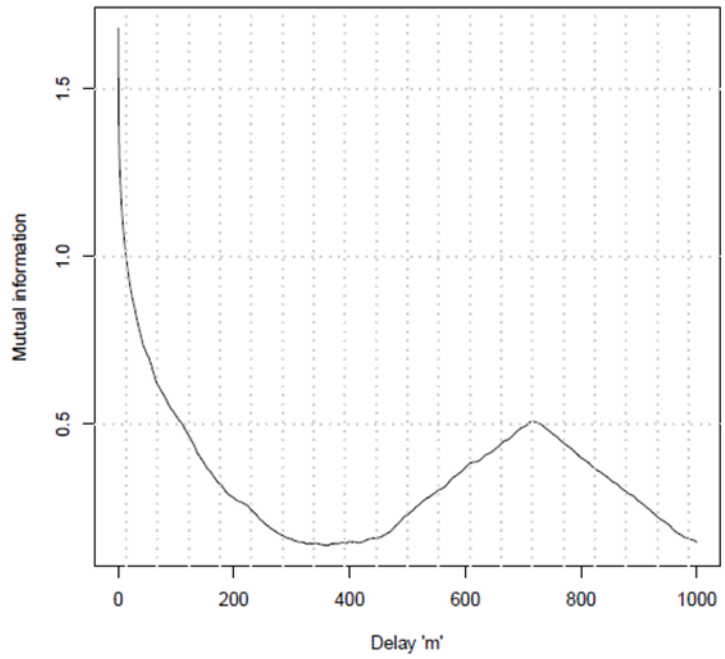


Figure 5: Mutual Information versus delay

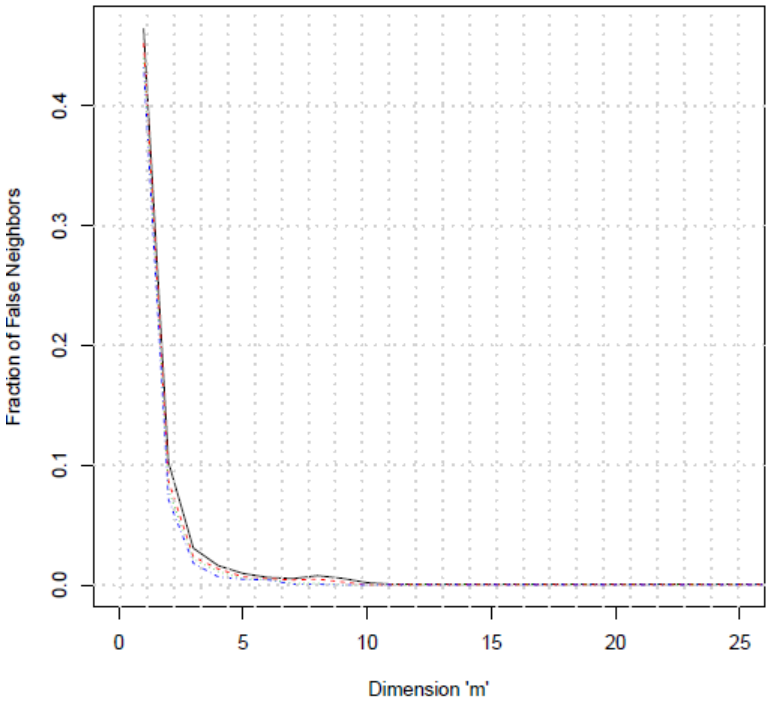


Figure 6: Fraction of false nearest neighbours versus dimensionality

4.2.2. Experimental Results

The following tables record the input window parameters and out-of-sample error for each of our input representations in Section 4.2.1, over each of the data sets described in Section 3.4.1.1 - 3.4.1.6. Table 4 presents our results for the solar power data set. Quite plainly, in the solar power data set, the traditional delay embedding was superior to the down-sampled "one period" input window, even though the smaller dimensionality provides far fewer tunable parameters in the consequent layer. It was also superior to the unit-delay input window created from Method 3, even though Method 2 again resulted in fewer dimensions.

Table 5 presents our results on the Mackey–Glass data set; the results for Method 1 are taken from [25]. Interestingly, this data set presents a completely different picture than the solar power data set. The traditional delay embedding gave—by two orders of magnitude—the least accurate predictions, even though the most accurate approach (Method 3) differed only in the delay length (reflecting what was stated in 3.3; all values of the delay parameter are theoretically equivalent, but in practice a good choice of the delay parameter can significantly impact the performance of a forecasting algorithm).

Table 6 presents our results for the Santa Fe LaserA data set. This time, Method 1 appears to be the best, while Method 2 yields the worst results. Table 7 presents our results for the sunspot data set. This time, the traditional delay embedding is somewhat worse than Methods 1 and 3; however, the difference is not very large. In addition, Methods 1 and 3 are nearly indistinguishable from one another.

Table 8 presents our results for the Stellar data set. This time, method 1 is substantially better than methods 2 or 3; furthermore, method 2 is slightly less accurate than method 3.

Table 9 presents our results for the Waves data set. Once again, method 1 proved to be the most accurate. However, this time method 3 was less accurate than method 2.

Table 4: Results for the Solar Power

Method	Input Vector Design		RMSE
	Dimension	Delay	
1	145	10	5.293
2	12	370	3.1057
3	15	1	4.847

Table 5: Results for the Mackey-Glass

Method	Input Vector Design		RMSE
	Dimension	Delay	
1 [25]	44	1	5.57e-4
2	9	11	0.015
3	9	1	5.29e-4

Table 6: Results for Santa Fe Laser A

Method	Input Vector Design		RMSE
	# Lags	Delay	
1 [25]	8	1	0.033
2	9	2	0.114
3	9	1	0.067

Table 7: Results for Sunspot

Method	Input Vector Design		RMSE
	# Lags	Delay	
1 [25]	12	1	0.091

2	5	4	0.103
3	6	1	0.089

Table 8: Results for Stellar

Method	Input Vector Design		RMSE
	# Lags	Delay	
1 [25]	27	1	7.49e-3
2	6	7	1.4e-2
3	6	1	1.3e-2

Table 9: Results for Waves

Method	Input Vector Design		RMSE
	# Lags	Delay	
1 [25]	12	1	0.0032
2	4	4	0.00866
3	4	1	0.0104

4.2.3. Discussion

As with many other experiments in pattern recognition, our general finding is that the "best" input representation for ANCFIS is data set-dependent. In five of our six data sets, the traditional delay embedding was clearly out-performed by the window-based approaches; but in our single largest data set, method 2 was clearly the best. Four times, method 1 was either the best approach or virtually identical to method 3 and superior to method 2.

Our findings do, however, suggest which methods seem more likely to succeed in future experiments. Method 3 was the better approach once, and essentially tied with method 1 on two

other data sets. This method also seems to lead to lower embedding dimensionalities than method 1. This still matters in ANCFIS, even though we have reduced the combinatorial explosion of rules seen in other machine learning methods using orthogonal lags. A complexity analysis carried out in [25] indicates that the running time of both the least-squares estimate of the consequent parameters, and the VNCSA optimization of the CFS parameters, depend linearly on the length of the input vector (this explains why running ANCFIS on the full input window for the solar-power data set took and infeasibly long time). Thus, with method 3 often providing strong results, and usually resulting in a significantly smaller input window, this seems to be the most effective initial approach to modeling a time-series with ANCFIS. We would recommend that method 1 be tried next, and finally the traditional delay embedding.

4.3. Conclusion

In this chapter, we have explored three different approaches for representing time-series inputs for the ANCFIS machine-learning algorithm. We compared input windows based on an ad-hoc determination of what constitutes one "period" in the data set; the traditional delay embedding, guided by the mutual-information and false-nearest-neighbour heuristics; and the use of only the false-nearest-neighbour heuristic, across six time-series data sets. While the "best" method appears to be data set-dependent, we found enough evidence that we recommend method 3 as the best combination of accuracy and expected computation time.

The three different time series representation presented in this chapter are used throughout the rest of this dissertation to create training and testing input vectors to train and evaluate the machine learning algorithms designed and implemented in this dissertation.

Chapter 5

Univariate Time-series Prediction by ANCFIS

Photovoltaic power is one of the most promising renewable energy sources. However, it is also intermittent, and thus short-term forecasts of photovoltaic power generation are needed to integrate this power source into the electricity grid. Parametric models of photovoltaic power generation have generally not been satisfactory, creating an interest in non-parametric approaches such as machine learning. In this chapter, we evaluate ANCFIS algorithm in the photovoltaic power forecasting task. We created a new, large-scale data set based on solar irradiance and temperature measurements at the Lowry Range Solar Station over a period of 5 years. We then used a simulated solar cell to convert these readings into an instantaneous power signal; this is the largest publicly-available solar power data set we are aware of (Section 3.4.1.1). We then apply three well-known machine learning algorithms (ANFIS, RBFN and SVR discussed in sections 3.1.1 - 3.1.3), ANCFIS (section 3.2) and ARIMA forecasting (section 3.1.4) in a one-minute-ahead forecasting problem.

5.1. Introduction

Fossil fuels are by their nature a finite and non-renewable resource. As they are also a principal energy source for the world, we will inevitably face major energy shortages in the future, unless the world transitions to other, renewable energy sources. Moreover, more than 90% of greenhouse gas emissions are associated with production and consumption of fossil fuels [117]. Renewable and clean energies such as solar, wind, biomass, and hydropower energy are promising

alternative energy sources (and all but biomass have been widely deployed). Solar photovoltaic (PV) energy production in particular has grown by 58% annually in recent years [118].

PV power, however, is intermittent; the power produced by PV plants varies with the instantaneous solar irradiance and outside air temperature; power output will drop whenever a cloud passes across the sun [119]. Furthermore, PV power is zero until dawn, increases during the day until reaching a maximum in the afternoon, and then decreases and returns to zero at night. Temperature has an inverse effect on the PV power; as temperature increases, the power drops [120]. These characteristics are a poor fit for the power grid, in which the instantaneous power supply must equal the instantaneous demand. This is an advantage of fossil fuels – the output of fossil-fuel plants is steady and relatively easy to adjust. If a PV plant is feeding energy to the grid, and its production drops significantly, that deficit must be made up in real-time with only the briefest of delays – otherwise the grid's protection circuits will activate, and customers could experience black outs [121]. Thus, in order to dispatch PV power to the electrical grid, an accurate prediction of PV power output is needed.

Development of photovoltaic technologies has been lead to much research in forecasting PV power. There are different ways of forecasting PV power. Some studies focus on forecasting of wide-area solar irradiance [122-131], which can be applied in PV power forecasting. However, some researches try to predict PV power directly from the time-series of past observations for a specific installation. [132] applied regression methods for this purpose. [121] used adaptive time-series models for on-line forecasting of PV power. [133] used a fuzzy forecasting system which works based on class labels assigned to input patterns. [134] proposed an on-line method based on Gaussian models, while [135] applied Bayesian methods to solve the problem. [136] developed a

Support Vector Machine forecasting model. Increasingly, artificial neural networks are used in the prediction of PV power. [137] implements a recurrent neural network (RNN), [138] developed neural network models for PV power forecasting in sunny and cloudy days, [139] applied multilayer perceptrons and ANFIS, [140] used multilayer perceptron network, [141] developed an online PV power prediction by RBF network, and [142] implemented dynamic neural networks, FTDNN and DTDNN, on power output data. [143] compares five forecasting model in prediction of PV power including Persistent model (PER), ARIMA, K-Nearest-Neighbors (KNN), neural networks and neural networks optimized with a Genetic algorithm. A review of research in this area may be found in [128, 144].

A key problem in PV power prediction study is that there is no public data set to work with. Most research is performed on proprietary data sets provided by private companies. For example, [132] employed data obtained from a photovoltaic system installed in the World Exhibition, Aichi, Japan in 2005. [121] was applied to data obtained during 2006 from 21 PV systems installed in Jutland, Denmark. [133] used data provided by an Italian private company, and [134] used data from a rooftop PV plant in Macau. [137] employed data from Okinawa Prefecture, Naha, Japan. [136] used data from a PV power plant situated in Kitakyushu, Japan. [138] used data gathered by Marmara University, Istanbul, Turkey during 2011. [139] used data from PV plants in the Czech Republic. [140] used data from PV plants installed at University of Jaén, Spain. [141] used data provided by PV systems installed in the Renewable Energy Research Center (RERS) of Huazhong University of Science and Technology (HUST), China. [142] was applied to power output data from a PV plant in MASDAR city, Abu Dhabi, U.A.E, and [143] used data from the city of Merced, California. None of these data sources are publicly available at this time.

For this chapter, we use a public-domain data set of solar irradiance and temperature. These two quantities are converted to PV power output by using a model presented in [104]. The resulting data set has over 2 million observations, sampled at a rate of once per minute. This data set is much larger, with a much higher sampling frequency, than the existing public-domain PV power data sets we are aware of. (We have made this unique data set freely available[1].) In a one-minute-ahead forecasting experiment on a subset of this data set, the SVR algorithm was the most accurate, followed by ANCFIS.

5.2. Methodology

5.2.1. Experimental Design

For input representation, we use delay embedding techniques discussed in Section 3.3. Based on the results from Section 4.2.1.1, a delay of 370 and 12 embedding dimensions is appropriate for the solar power data set.

For ANFIS, RBFN and SVR, we construct (separately for the training and testing sets) new data sets consisting of the delay vectors, plus the one-step-ahead value of the time-series at the moment of those delay vectors. This latter is the dependent variable, while the 12 delay vector components are the independent variables. For ANCFIS, we will treat the delay vector as a single input window, with the one-step-ahead value again being the dependent variable. In a previous study [4] (see Chapter 4), this input formulation was determined to be the be the best for ANCFIS among three different approaches.

[1] http://www.ualberta.ca/~yazdanba/

In ANFIS, the number of rules for a complete rulebase is determined by m^r where r is number of data points in the input vector and m is the number of membership functions defined per input. Having 12 input dimensions with two membership functions each yields $2^{12} = 4096$ rules. This is infeasible, and so we use fuzzy c-means clustering as a structure identification step. Training input vectors are clustered by FCM, and the centers of the clusters and membership grade matrix obtained by FCM are used to initialize the parameters of membership functions. We used MATLAB's genfis3 function for this step, which produces Gaussian membership functions. The number of clusters was determined via trial-and-error exploration.

We used MATLAB's built-in function newrb.m to carry out our RBFN experiments. The parameters explored in these experiments are the number of neurons in the hidden layer and the spread of each RBF. We implement SVR using the package *e1071* [145] in the R environment [146]. The best value of ε, C, and the spread of RBF kernel function are obtained by parameter exploration.

For ARIMA, we work with the training and testing data obtained from splitting the data set with 20000 data points to 2/3 and 1/3. Our data has an apparent seasonality since it measures solar power per minute, which follows changes of solar power during a day; solar power is zero until dawn, increases during day till reaching afternoon, and then decrease and return to zero at night. Figure 7 shows the training data.

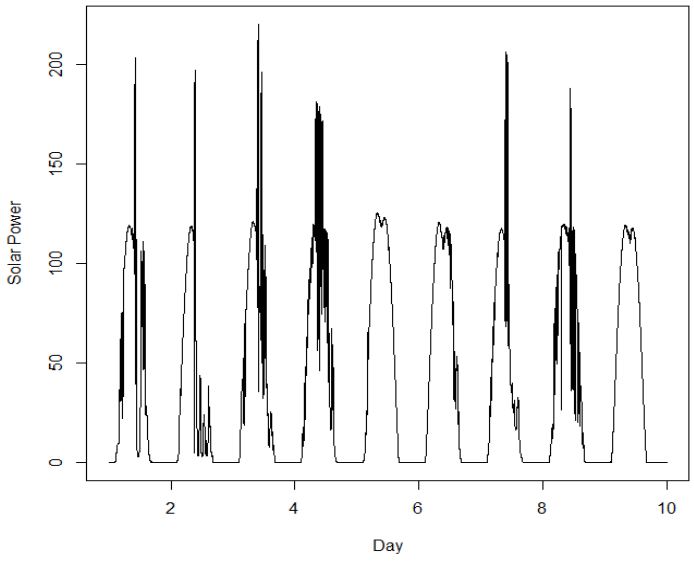


Figure 7: The training set containing solar power of 10 days

We use the Fourier transform to capture seasonality in the time-series [147, 148]; therefore, the ARIMA model is designed as [149]:

$$y_t = a + \sum_{k=1}^{K} S_k + N_t \quad (136)$$

where N_t is a non-seasonal ARIMA model, and $S_k = \alpha \sin\left(\frac{2\pi kt}{m}\right) + \beta\cos(\frac{2\pi kt}{m})$ is the Fourier transform. K is a user-defined parameter determining the number of harmonics required for modeling the seasonality. The ARIMA model is implemented by the "forecast" package in R environment [150]. Finally, the results of the five algorithms are compared in terms of RMSE.

5.2.2. Experimental Results

Table 10 shows the RMSE achieved by the five algorithms. The performance of the algorithms varies considerably, from 1.9701 for SVR to 6.8105 for RBFN. With over 6600 data points in the test set, the differences are all plainly statistically significant. ANCFIS is the most

accurate of the neural-network techniques, and the second-best overall. We examine the complexity of the neural-network techniques in Table 11; both ANFIS and ANCFIS performed well with 2 and three rules each, respectively. A much larger RBFN with 40 hidden nodes was the worst-performing algorithm in the entire group. The complexity of SVR and ARIMA models are not easily compared to neural networks, as SVR is a regression involving (usually) a large number of support vectors, while our ARIMA models also incorporate a Fourier transform of the data.

Table 10: RMSE of the five algorithms

Model	**RMSE**
ANCFIS	3.1057
ANFIS	3.7101
RBFN	6.8105
SVR	1.9701
ARIMA	4.822

Table 11: Model Complexity for Neural Networks

Model	**Complexity**
ANCFIS	3 Rules
ANFIS	2 Rules
RBFN	40 neurons

5.3. Conclusion

In this chapter, we have explored one-step-ahead forecasting in a subset of the solar power data set, using several different machine learning algorithms as well as ARIMA forecasting. We found that support vector regression was the most accurate approach on this data set, followed by ANCFIS.

The existing architecture of ANCFIS employed in this chapter has been designed for univariate time-series forecasting. To Apply ANCFIS for prediction of more general time series (e.g. multivariate time series), the architecture of ANCFIS needs to be changed. Chapter 6 extends the existing architecture of ANCFIS for multivariate time-series forecasting.

Chapter 6

Multivariate Time-series Prediction by ANCFIS

Multivariate time-series consist of sequential vector-valued observations of some phenomenon over time. Time-series forecasting (for both the univariate and the multivariate case) is a well-known, high-value machine learning problem, in which the goal is to predict future observations of the time-series based on prior ones. Several learning algorithms based on complex fuzzy logic have recently been shown to be very accurate and compact forecasting models. However, these models have only been tested on univariate and bivariate data sets. There has yet been no investigation of more general multivariate data sets.

We report on the extension of the ANCFIS learning architecture to the multivariate case. We investigate single-input-single-output, multiple-input-single-output, and multiple-input-multiple-output variations of the architecture, exploring their performances on four multi-variate time-series. We also explore modifications to the forward- and backward-pass computations in the architecture. We find that our best designs are superior to the published results on these data sets, and at least as accurate as kernel-based prediction algorithms.

6.1. Introduction

Time-series are observations recorded sequentially over time. Internet traffic, environmental sensor data, online transaction sequences and stock quotations from financial markets are examples of time-series. They are an important sub-category of data streams in which the data is not only temporally ordered, but the exact time of an observation is also recorded (explicitly or implicitly) [151]. Forecasting time-series is a high-value machine learning problem.

For example, internet traffic prediction aids network management; denial of service attacks and SPAM can also be detected as variations from the forecast [152, 153]. Precipitation forecasting is effective in water resource and drought management [154, 155]. Sales forecasts can be used to optimize inventory management and thus reduce costs [156, 157]. Stock market price forecasts can guide trading strategies covering tens of trillions of dollars' worth of assets [158].

Forecasting univariate time-series (those in which each observation is a single scalar) has been studied extensively, with numerous statistical and machine learning algorithms having been proposed, e.g. [159-163]. The more general version of the problem, multivariate time-series forecasting (in which each observation is a multi-dimensional vector), has also received considerable attention. Generally speaking, collecting a multivariate time-series only makes sense if there is some relationship between the data items comprising each component of the observation. The presence of this mutual information ought to make the tasks of noise reduction and forecasting easier [123, 164-167]. However, each component of an observation is itself a nonlinear projection of a complex state space down to a single scalar, and the relations between each component can thus be complex. Numerous algorithms have been proposed for multivariate time-series prediction, e.g. [99, 101, 168-170].

Machine learning algorithms based on CFS&L have shown promising results in time-series forecasting. CFS, as introduced by Ramot et al. [1, 2] are an extension of type-1 fuzzy sets in which the membership grades are complex numbers (drawn from the unit disc in the complex plane). The ANCFIS [25] was the first machine learning algorithm based on CFS whose main application is univariate time-series prediction [17, 25, 55, 171]. The family of Complex Neuro-Fuzzy System (CNFS) architectures are likewise machine learning algorithms based on CFS that have been applied to univariate and bivariate time-series prediction [27, 48, 56, 134] (as well as

other machine-learning problems, e.g. adaptive noise cancellation); it is worth noting that CNFS cannot be applied to time-series with more than two variables because of its structure. Currently, however, no system based on CFS&L exists for general multivariate time-series prediction.

In this chapter, we extend the ANCFIS architecture for general multivariate time-series forecasting. We explore three possible implementations: a set of single-input-single-output (SISO) networks, a set of multiple-input-single-output (MISO) networks, and a single multiple-input-multiple-output (MIMO) network. We also investigate modifications to the forward- and backward-pass computations, as well as the time-series embedding techniques we employ, in order to further improve the forecast accuracy of our algorithms. The proposed approaches are applied to three time-series (having two and three variates), and their results are compared against two well-known machine-learning algorithms RBFN and SVR (See sections 3.1.2 and 3.1.3) and published forecast results on these time-series. We use the Friedman test to compare the performance of the different algorithms and variants across the three data sets. Finally, our MIMO ANCFIS design is compared against the only current complex fuzzy algorithm for bivariate forecasting on a fourth data set.

6.2. Methodology

6.2.1. Network Design

In this section, three alternative designs for creating a multivariate ANCFIS system are studied, SISO ANCFIS, MISO ANCFIS and MIMO ANCFIS. In this case, “single” and “multiple” outputs refer to the number of variates of a time-series input to the network(s) as delay vectors, or the number predicted as outputs. In SISO ANCFIS, a separate univariate ANCFIS network is created for each variate, with the outputs of the networks concatenated into the final output vector.

In MISO ANCFIS, there is again one ANCFIS network for each variate and the outputs are treated the same as in SISO ANCFIS, but the input is the combined multivariate delay vector. In MIMO ANCFIS, only one ANCFIS network is created for the whole multivariate time-series. The input is the multivariate delay vector, and the network outputs the entire multivariate prediction vector. For example, for a two-variate time-series (x and y) with $\tau_1 = \tau_2 = 1$, and $d_1 = 3\ and\ d_2 = 2$, we have the following input vectors:

Table 12: Delay vectors for SISO ANCFIS

x_3, x_2, x_1, x_t	Input to the first ANCFIS system
y_2, y_1, y_t	Input to the second ANCFIS system

Table 13: Delay Vectors for MISO ANCFIS

$x_3, x_2, x_1, x_t, y_3, y_2, \boxed{y_t}$ $\uparrow$ $Target$	Input to the first ANCFIS system
$x_3, x_2, x_1, \boxed{x_t}, y_3, y_2, y_t$ $\uparrow$ $Target$	Input to the second ANCFIS system

Table 14: Delay Vectors for MIMO ANCFIS

$x_3, x_2, x_1, x_t, y_3, y_2, y_t$

Figure 8 shows MIMO ANCFIS architecture for a bi-variate time-series.

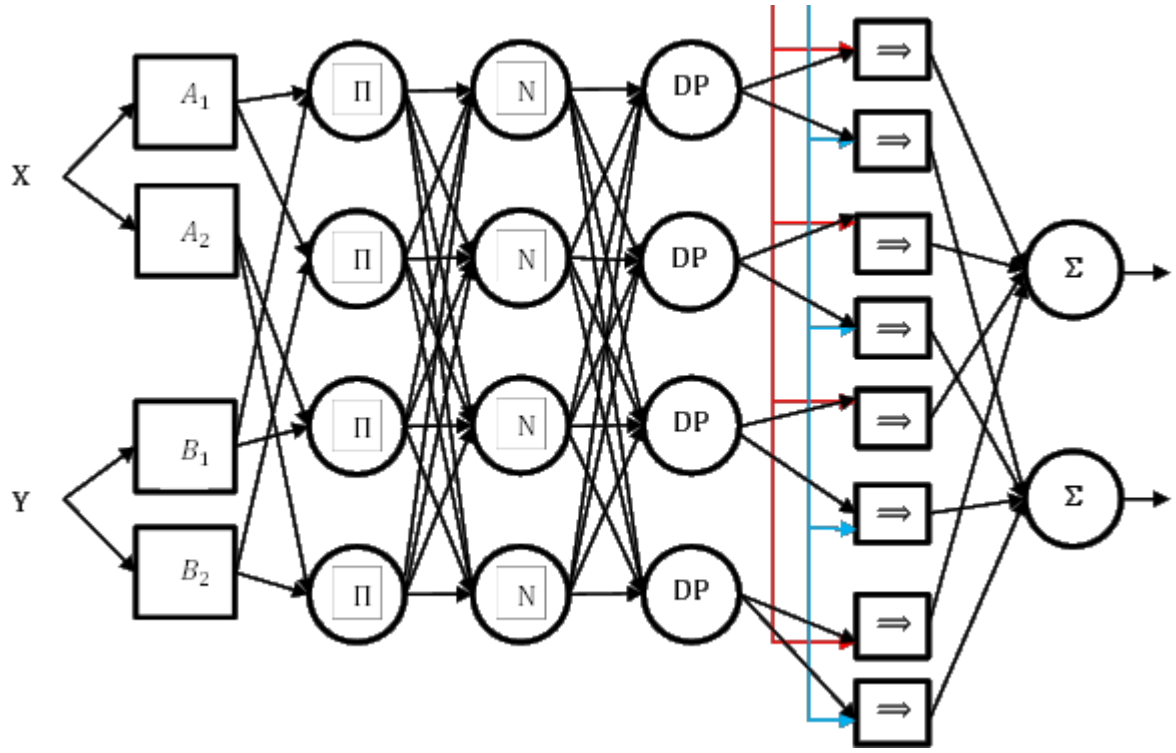


Figure 8: MIMO ANCFIS architecture for a bi-variate time-series problems with two membership functions for each variable.

Moreover, we investigate the effect of using different heuristics to determine the dimensionalities for each variate in our data sets. We contrast determining the dimensions separately via the false-nearest-neighbor technique [98] (FNN) with the combined approach proposed by Cao et al. [99] (KDD). The KDD algorithm is implemented here with three nearest neighbors; therefore, for each delay vector, the three nearest neighbors are obtained, and the root mean square of their prediction errors is used to select the set of dimensions. For both approaches, the time delay is calculated separately for each variable by the mutual information heuristic, and then manually cross-checked using the phase portrait (phase portraits are a plot of the variate $x(t)$ versus its delay $x(t\text{-}n)$ [96]) (see section 3.3). The forecast accuracies for these approaches are compared against each other, and against the well-known RBFN (see section 3.1.2) and SVR (see section 3.1.3) algorithms.

Beyond these network design changes, we also investigate two architectural changes that may improve the performance of MIMO ANCFIS. Firstly, we investigate three potential modifications to the rule interference operation. Secondly, we investigate a change to the VNCSA algorithm, altering the reinitialization of the algorithm at each epoch.

Our modifications to rule interference involve changing the Layer-3, Layer-4 and Layer-6 transfer functions in ANCFIS. Recall that Layer 3 normalizes the rule firing strengths using Equation (111), Layer-4 implements a dot product between the current rule's firing strength and the vector sum of all rule firing strengths using Equation (112), and Layer 6 is a summation of its inputs per Equation (114). We propose the following three variants of ANCFIS:

- In our first variant (1st ANCFIS), the normalization in layer 3 is applied on both amplitude and phase; in other words, the normalization is obtained by the complex sum of the node 2 outputs, $\sum_{j=1}^{|O_2|} O_{2,j}$, instead of the sum of output amplitudes of node 2, $\sum_{j=1}^{|O_2|} |O_{2,j}|$. In layer 4, meanwhile, we take the dot product of each Layer-3 output with the vector 1+0i (this was established as the lattice supremum for a complex fuzzy logic having the algebraic product as its conjunction in [15]). The new equations for Layers 3 and 4 are:

$$O_{3,i} = \bar{w}_\iota = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} O_{2,j}}, \quad i = 1,2,\ldots,|O_2| \qquad \text{(Layer 3)} \qquad (137)$$

$$O_{4,i} = w_i^{DP} = \bar{w}_\iota \cdot 1 = |\bar{w}_\iota|.\cos(\bar{\theta}_\iota) \qquad \text{(Layer 4)} \qquad (138)$$

where $|\bar{w}_\iota|$ and $\bar{\theta}_\iota$ are the amplitude and phase of the output of Layer 3, $\bar{w}_\iota$.

- In our second variant (2nd ANCFIS), the Layer 3 transfer function remains as above (Equation (137)). Now, however, we will define Layer 4 as the identity function, making all of the internal signals in ANCFIS complex-valued. In Layer 6, we take the dot product of the final output with 1+0i in order to obtain a real value. The main difference between this design and

the previous ones is that we work with complex values all over the network, and so rule interference occurs in the complex sum of Layer 6. (Note that this design is equivalent to eliminating Layer 4 from the design entirely.) The new equations are:

$$O_{3,i} = \bar{w}_\iota = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} O_{2,j}}, \quad i = 1,2,\ldots,|O_2| \qquad \text{(Layer 3)} \qquad (139)$$

$$O_{4,i} = w_i^{DP} = \bar{w}_\iota \qquad \text{(Layer 4)} \qquad (140)$$

$$O_{6,j} = \left(\sum_{i=1+(j-1)*N}^{j*N} w_i^S\right).1 \qquad \text{(Layer 6)} \qquad (141)$$

where j is the number of outputs, N is the number of rules.

- The third design (3rd ANCFIS) is based on the second one; the only difference is in the layer 6 where instead of the dot product, we use the amplitude of the complex sum as the final output, as below.

$$O_{6,j} = \left|\sum_{i=1+(j-1)*N}^{j*N} w_i^S\right| \qquad \text{(Layer 6)} \qquad (142)$$

In the last modification, we have changes in VNCSA algorithm and forward pass is kept same as original ANCFIS (New MIMO ANCFIS). Our modification to the VNCSA algorithm concerns the reinitialization of the algorithm at each learning epoch. As discussed, the VNCSA algorithm employs the Logistic map to generate an initial set of solutions with high variance. Then the Ulam-von Neumann map is used to generate new solutions from those initial ones. Therefore, the search space for $\{a,b,c,d\}$ is wide, and the initial solution set for VNCSA in one epoch might be completely independent of the previous one. The question we will investigate is whether it might be more advantageous to initialize each epoch with the final results of the previous epoch. We propose changing VNCSA as follows: for each epoch after the initial one, the initial population generated by the logistic map is replaced with the previous epoch values of the parameters $\{a,b,c,d\}$. Thus, the Ulam-von Neumann map generates new solutions based on the previous epoch

values. As the difference between sampled membership functions in two successive epochs is based on Equation (116)

$$\eta \frac{\partial E}{\partial g(k)} = g(k)_{old} - g(k)_{new}, \quad (143)$$

it is reasonable to limit the search space based on the parameters in the previous epoch. Naturally, in the first epoch, the original VNCSA is still used to explore the whole search space and find initial neighborhood for the parameters.

6.2.2. Experimental Design

Our experiments all follow a chronologically ordered single-split design, with all elements of the training set occurring earlier in time than testing set elements. We compare our results using the RMSE (Equation (133)) and MAE (Equation (134)) statistics. To determine if there is any significant difference between the results, Freidman statistic is calculated [113]. If null hypothesis $H_0: [\tau_1 = \cdots = \tau_k]$ is rejected, the approaches are compared together using the Multiple Comparisons with the Best (MCB) method [116] (see section 3.5.1).

6.2.3. Data Sets

We use Motel, Precipitation, Flour and NASDAQ time-series described in Section 3.4.2. The delay and dimensions obtained for the two different input vector sets built based on FNN and KDD are shown in Table 15-Table 18.

Table 15: Delay and dimension sets for Motel time-series

	Delay (τ_1, τ_2)	Dimension (FNN) (d_1, d_2)	Dimension (KDD) (d_1, d_2)
Motel	(6,6)	(6,6)	(2,2)

Table 16: Delay and dimension sets for Precipitation time-series

	Delay (τ_1, τ_2, τ_3)	Dimension (FNN) (d_1, d_2, d_3)	Dimension (KDD) (d_1, d_2, d_3)
Precipitation	(3,1,3)	(4,4,4)	(1,3,1)

Table 17: Delay and dimension sets for Flour time-series

	Delay (τ_1, τ_2, τ_3)	Dimension (FNN) (d_1, d_2, d_3)	Dimension (KDD) (d_1, d_2, d_3)
Flour	(1,1,1)	(3,3,3)	(1,3,2)

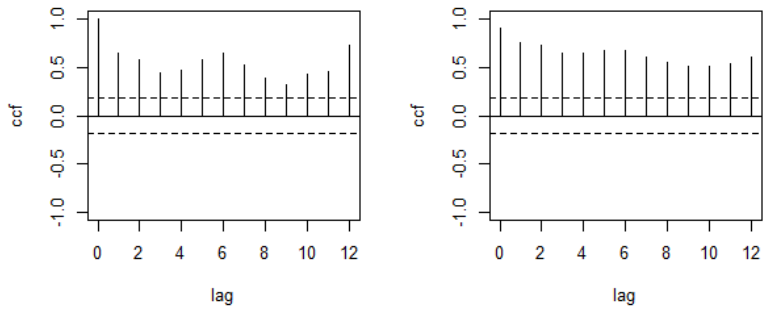
Table 18: Delay and dimension sets for NASDAQ time-series

	Delay (τ_1, τ_2)	Dimension (FNN) (d_1, d_2)	Dimension (KDD) (d_1, d_2)
NASDAQ	(1,1)	(1,3)	(1,3)

We also examine the autocorrelation structure of the time-series using the sample cross-correlation matrix. Each element of this matrix is the cross-correlation between two of the variates, defined as:

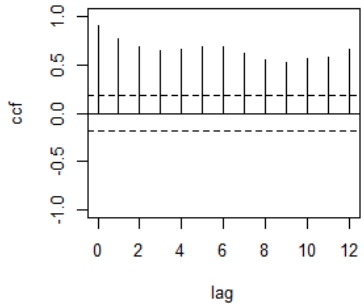
$$(f \otimes g)(\tau) = \sum_{i=-\infty}^{\infty} f_i^*(m) \cdot g(m+\tau) \quad (144)$$

where f and g are two variates in a multivariate time-series, τ is the lag, and f^* is the complex conjugate of f [172]. The ij-th entry in the matrix represents the cross-correlation between the i-th and j-th variates. The matrix is computed and presented by the 'ccm' routine in the "MTS" package in R. The cross-correlation matrices for the four time-series are presented in Figure 9-Figure 12:

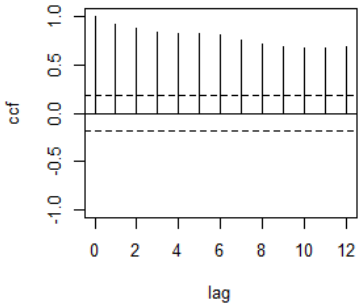


a. Auto-correlation of v_1

b. Cross-correlation of v_1 and v_2



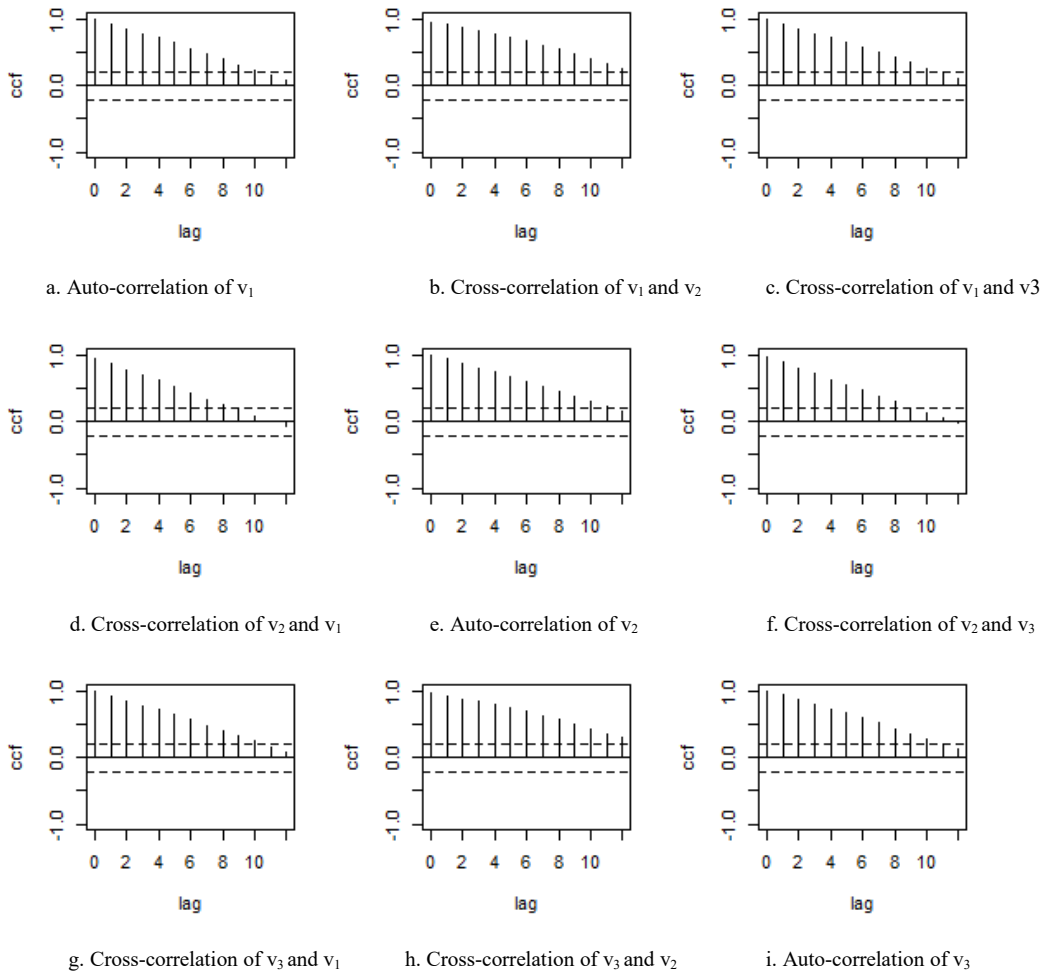
c. Cross-correlation of v_2 and v_1



d. Auto-correlation of v_2

Figure 9: Cross-Correlation Matrix for the Motel Data set with two variates (v_1 and v_2)

The cross-correlation matrix presented in Figure 9 do not appear to decaying, and so the time-series is not stationary. However, this dependence on previous values is the kind of non-stationarity that learning algorithms should excel at.



a. Auto-correlation of v_1 b. Cross-correlation of v_1 and v_2 c. Cross-correlation of v_1 and v3

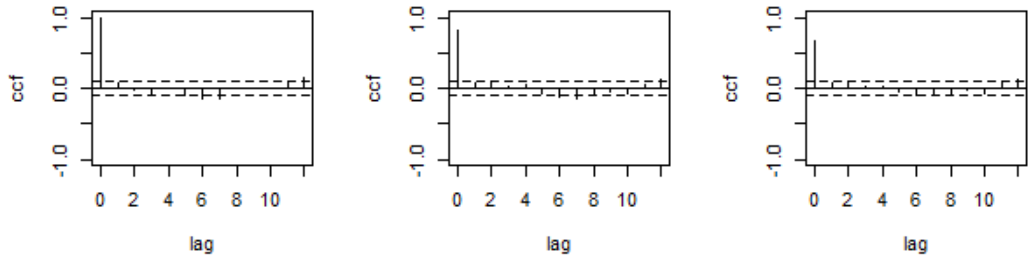
d. Cross-correlation of v_2 and v_1 e. Auto-correlation of v_2 f. Cross-correlation of v_2 and v_3

g. Cross-correlation of v_3 and v_1 h. Cross-correlation of v_3 and v_2 i. Auto-correlation of v_3

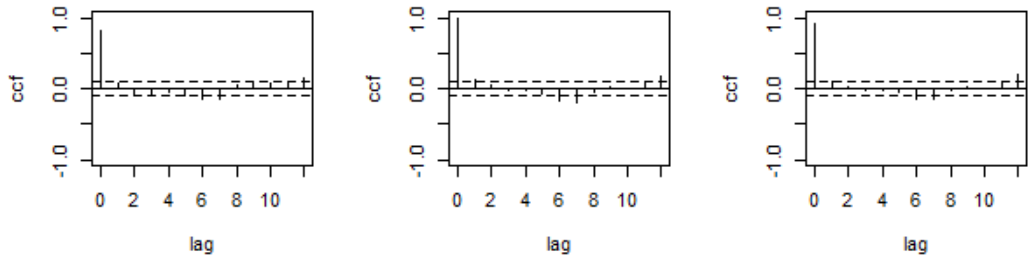
Figure 10: Cross-Correlation Matrix for the Flour Data set with three variates (v_1, v_2 and v_3)

The cross-correlation matrix presented in

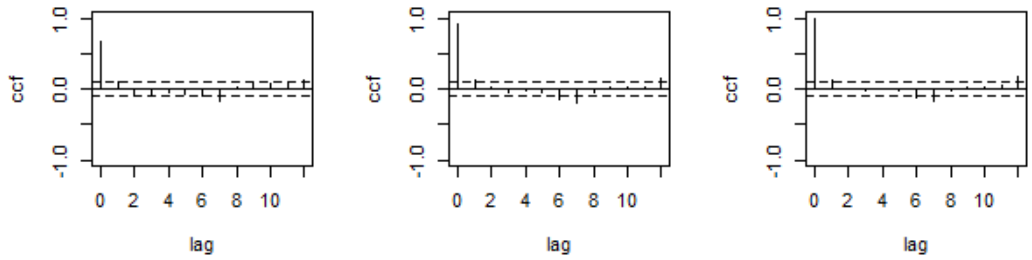
Figure 10 shows the correlations appearing to rapidly decay. This time-series appears stationary.



a. Auto-correlation of v_1 b. Cross-correlation of v_1 and v_2 c. Cross-correlation of v_1 and v3



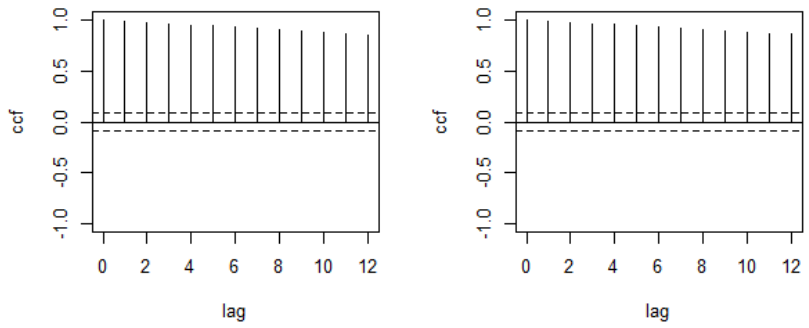
d. Cross-correlation of v_2 and v_1 e. Auto-correlation of v_2 f. Cross-correlation of v_2 and v_3



g. Cross-correlation of v_3 and v_1 h. Cross-correlation of v_3 and v_2 i. Auto-correlation of v_3

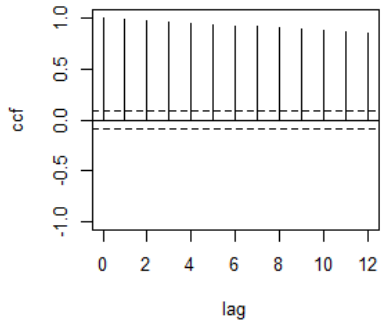
Figure 11: Cross-Correlation Matrix for the Precipitation Data set with three variates (v_1, v_2 and v_3)

The cross-correlation matrix presented in Figure 11 shows that the correlations decay swiftly, but there seems to be some remaining structure (possibly seasonality) for a longer period. So long as the period of this seasonality is much shorter than the length of the time-series (which appears to be true), machine learning algorithms should again be able to model this data set.

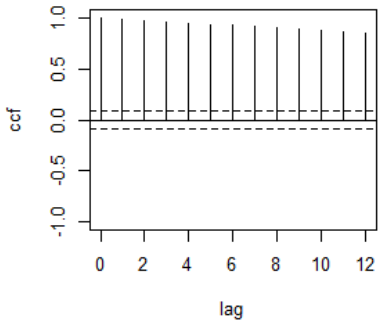


a. Auto-correlation of v_1

b. Cross-correlation of v_1 and v_2



c. Cross-correlation of v_2 and v_1



d. Auto-correlation of v_2

Figure 12: Cross-Correlation Matrix for the NASDAQ Data set with two variates (v_1 and v_2)

The cross-correlation matrix presented in Figure 12 is very similar to the matrix for the Motel data set, and so a learning algorithm should perform well on this data set as well.

6.2.4. Experimental Results

In this section, one-step-ahead predictions of the four multivariate time-series are examined. We first consider a classic Vector Auto-Regression Moving Average (VARMA) model, as an initial baseline (we specify the AR and MA orders in parentheses). We then examine the alternative designs of ANCFIS, RBFN, SVR, and finally compare against the other studies in the

literature for each time-series. Moreover, the impact of different approaches in selecting the dimension for the delay vectors on the multivariate time-series forecasting is discussed. Table 19, Table 20 and Table 21 indicate results of Motel, Flour, Precipitation time-series, based on RMSE, respectively.

Table 19: RMSE on the Motel time-series

Method	**RMSE**	
VARMA(2,2)	0.1849	
	KDD	FNN
RBFN SISO	0.19352	0.19638
RBFN MISO	0.18422	0.18601
RBFN MIMO	0.18362	0.18544
SVR SISO	0.19448	0.20530
SVR MISO	0.19436	0.20219
ANCFIS SISO	0.17875	0.18955
ANCFIS MISO	0.16136	0.21749
ANCFIS MIMO	0.16471	0.19393
1st ANCFIS	0.16357	0.23133
2nd ANCFIS	0.16433	0.21534
3rd ANCFIS	0.16129	0.17962
New MIMO ANCFIS	0.15457	0.23019

Table 20: RMSE on the Flour time-series

Method	**RMSE**	
VARMA(3,1)	0.3602	
	KDD	FNN
RBFN SISO	0.20737	0.1673
RBFN MISO	0.21672	0.28614

RBFN MIMO	0.17773	0.25094
SVR SISO	0.19878	0.18195
SVR MISO	0.23142	0.23270
ANCFIS SISO	0.20943	0.19357
ANCFIS MISO	0.19756	0.18650
ANCFIS MIMO	0.20457	0.19107
1st ANCFIS	0.19553	0.23986
2nd ANCFIS	0.19710	0.18211
3rd ANCFIS	0.20225	0.12871
New MIMO ANCFIS	0.19560	0.17705

Table 21: RMSE on the Precipitation time-series

Method	**RMSE**	
VARMA(6,0)	0.2098	
	KDD	FNN
RBFN SISO	0.2087	0.21021
RBFN MISO	0.207	0.19956
RBFN MIMO	0.2054	0.20068
SVR SISO	0.20965	0.21311
SVR MISO	0.20252	0.20464
ANCFIS SISO	0.20906	0.20791
ANCFIS MISO	0.20487	0.20339
ANCFIS MIMO	0.19758	0.19668
1st ANCFIS	0.20126	0.19142
2nd ANCFIS	0.19954	0.19253
3rd ANCFIS	0.19822	0.19530
New MIMO ANCFIS	0.20170	0.19505

In order to see if the 24 approaches (12 KDD and 12 FNN) applied on the 3 time-series are significantly different, the Freidman statistic, *S*, is calculated as Equation. (135); with *k*=24 and *n*=3, we obtain *S*=20.70. With $\alpha = 0.05$, $\chi^2_{k-1,\alpha} = 35.172$ (for 23 degree of freedom) and so we do not reject the null hypothesis. All of our ANCFIS variations, as well as RBFN and SVR exhibit no significant difference in prediction accuracy on these data sets. For simplicity, we thus choose the single MIMO approach, and the more automated KDD technique for finding dimensions, for further comparisons against the literature on these three data sets and the NASDAQ data set.

Table 22 - Table 24 shows the results of New MIMO ANCFIS KDD with the recent articles working on the same time-series used here. Note that we were unable to locate prior work in modeling the Motel time-series. In Table 24, we compare New MIMO ANCFIS KDD on the NASDAQ data set against the recently-published CNFS-ARIMA, which is a direct, complex-fuzzy-logic based competitor to ANCFIS.

Table 22: Flour time-series prediction based on RMSE

	Buffalo	Minneapolis	Kansas
New MIMO ANCFIS (KDD)	0.16043	0.21123	0.21078
DAN2 [110]	0.61725	0.22804	0.58052

Table 23: Precipitation time-series prediction based on MAE

	East	Middle	West
New MIMO ANCFIS (KDD)	0.15169	0.16329	0.15124
VTG Scheme[109] (Combined Model:	0.598	0.613	0.613

2nd Lagrange)			

Table 24: RMSE on NASDAQ time-series

Method	**RMSE**
VARMA(2,2)	77.00
New MIMO ANCFIS (KDD)	45.10
CNFS-ARIMA [48]	66.22

6.3. Conclusion

We have studied the extension of ANCFIS to multivariate time-series prediction. Three different ANCFIS architectural designs (SISO, MISO, MIMO) have been examined. A further exploration of alternative designs of MIMO ANCFIS is also performed. The designs were compared with well-known machine learning algorithms, RBFN and SVR. A Friedman test shows no significant difference between any of these methods. However, when we compare ANCFIS against the existing literature on these data sets, our method is clearly superior.

In future work, we will use the MIMO ANCFIS architecture as a testbed to explore different logical operations in complex fuzzy systems. Our prior work in [15] indicates that "complex fuzzy logic" is likely a family of multi-valued logics (much as type-1 fuzzy logic is). This means that the design space for complex fuzzy inferential systems is likely very large. However, we expect that not all complex fuzzy logics will be equally effective in any given learning problem. Thus, our testbed will be one method by which we can evaluate the utility of a new complex fuzzy logic.

However, the main disadvantages of ANCFIS is its slow learning algorithm which is a combination of forward pass and backward pass to update antecedent and consequent parameters. In the Chapter 7, a new machine learning algorithm based on randomized learning algorithms and complex fuzzy sets and logic is developed to solve the slow learning problem.

Chapter 7

RANCFIS: Randomized Adaptive Neuro-Complex Fuzzy Inference System

The ANCFIS was the first neuro-fuzzy system to combine complex fuzzy sets and rule interference for time-series forecasting. The induced forecasting models are accurate and parsimonious. However, the training algorithm (a hybrid of gradient descent and derivative-free optimization) is extremely slow. Thus, a different training algorithm is needed; one that is substantially faster, but also preserves the advantages of accuracy and parsimony.

In this chapter, we propose and evaluate a randomized-learning approach to train this neuro-fuzzy system. A number of recent results have shown that assigning fixed, random values to a subset of the adaptive parameters in a neural network model is an effective, simpler, and far faster alternative to optimizing those same parameters. We study mechanisms by which randomized learning may be combined with our system, and evaluate the system on both univariate and multivariate time-series. In general, we find that our proposed architecture is far faster than the original system, with no statistically significant difference in accuracy.

7.1. Introduction

Big Data, analytics, and business intelligence are today key elements of any major corporation's business strategy; they are either in place already, or a priority investment. Analysts expect total revenues in the Big Data / Analytics market were $122 *billion* USD in 2015, and this is projected to grow to $187 billion by 2019 [173]. Of this, the Business Intelligence (BI) /

Analytics software segment earned $17.9 billion in 2014, and this is expected to reach $26.8 billion in 2019 [174]. Notably, by 2020 40% of the BI/Analytics market will be for predictive and prescriptive analytics; this works out to $10.7 billion using the 2019 revenue forecast [175]. Of this, only $1.1 billion is expected to be spent on prescriptive analytics, and so the predictive analytics software market should be $9.6 billion by 2020. Thus, it is literal truth to say that forecasting in BI/Analytics will be a ten-billion-dollar industry within five years.

The focus of the current chapter is on designing forecasting algorithms based on CFS&L. Inductive forecasting algorithms such as the ANCFIS [7, 25, 54, 176], have proven to be accurate, parsimonious time-series forecasting algorithms. In a systematic review [177], we determined that these are currently the principal means by which CFS&L may be applied to real-world problems. The main alternative approach in fuzzy systems is to elucidate fuzzy rulebases from subject-matter experts. However, there has been very little progress in this direction in CFS&L (with the exception of [178]), likely because there are as yet no widely-accepted interpretations of complex fuzzy sets; and hence, linguistic models are very difficult to express. Thus, research into the applications of CFS&L must perforce follow the inductive learning approach, and are thus sensitive to the time and space complexity of those algorithms.

ANCFIS is based on the ANFIS architecture [62] (see section 3.1.1), and like its predecessor uses a hybrid learning algorithm combining least-mean-squares optimization in the forward pass with gradient descent in the backward pass, with the addition of a final derivative-free optimization step. This hybrid algorithm is relatively slow, which is a barrier to its more widespread use; slow algorithms have little future in a Big Data world. Thus, our goal is to develop a CFS&L-based algorithm that maintains the accuracy and parsimony of ANCFIS, but with dramatically reduced training times.

We propose a new training algorithm based on randomized learning for ANCFIS. We choose to randomly select and fix the parameters of our complex fuzzy set Membership Functions (MFs) in Layer 1 of ANCFIS, and optimize only the linear consequent function parameters. As these two are the only adaptive parameters in the network, and only the Layer-1 parameters are updated on the backward pass, randomizing them thus eliminates the backward pass in this network entirely, allowing for a very large speedup. The proposed approach is evaluated on three univariate and two multivariate time-series, and its results are compared against an earlier version of our current approach [176], ANCFIS, and other well-known approaches previously applied on these time-series.

Our contributions in this chapter are: 1) The design and implementation of a randomized-learning algorithm for the ANCFIS neuro-fuzzy system; 2) A comparative evaluation of this new training algorithm against other variants of ANCFIS and the existing literature.

7.2. Background

7.2.1. Randomized Learning Algorithms

From a theoretical point of view, randomized learning algorithms should be just as accurate in forecasting as conventional deterministic ones [179-182]. Consider a generic forecasting task of the form [183]:

$$x_1, x_2, \dots, x_d \mapsto f(x_1, x_2, \dots, x_d) \quad (145)$$

where $x_i \in \mathbb{R},\ i = 1, 2, .., d$, is the *i*-th observation in a time-series, and $f: \mathbb{R}^d \to \mathbb{R}$ is an unknown functional relation between those prior observations and the (presently unknown) next observation for which

$$f(x) = \sum_{i=1}^{\infty} c_i g(w_i^T x + b_i), \quad \sum_{j=1}^{\infty} c_j = 1, c_j \geq 0 \tag{146}$$

where $g()$ has a finite L_2 norm . Barron and Jones have shown that such original functions can be approximated with high accuracy through finite linear combinations of tunable basis functions $g(\cdot)$ [184, 185]:

$$f(x) \simeq f_K(x) = \sum_{k=1}^{K} \alpha_k \mathrm{g}(w_k{}^T x + b_k) \tag{147}$$

where w_{kj}, b_k, $\alpha_k \in \mathbb{R}$, j = 1,2,…,m are respectively the m weights, bias of one basis function, and the weighting of that basis function in the overall summation of K basis functions, K a finite number. Traditionally (for example, in a neural network model), determining the approximation involves fitting all of these parameters to the available data (e.g. via gradient descent). Plainly, however, this is time consuming for large data set, or large parameter vectors. Per [183, 186], however, the original function can also be approximated using linear combinations of functions with randomly selected parameters. In Equation. (146), let us consider the parameter set of f_K: $\omega_K = (K, \alpha_1, \dots, \alpha_k, \vec{w}_1, \dots, \vec{w}_k, b_1, \dots, b_k,)$. Choose a subset of these parameters to be set randomly; for simplicity, we choose the weights and biases of the basis functions, and denote this random-parameter set as $\lambda_K = (\vec{w}_1, \dots, \vec{w}_k, b_1, \dots, b_k)$. λ_K is a distribution defined on the probabilistic space $S_K(\Omega, \alpha)$ with probability measure $\mu_{K,\Omega,\alpha}$ and expectation E. Suppose that $S_K(\Omega, \alpha)$ and $\mu_{K,\Omega,\alpha}$ depend on the deterministic parameters (Ω, α), to be which should be determined in the learning stage. For any compact, $N, N \subset I^d, N \neq I^d$, where $I^d = [0,1]^d \subset \mathbb{R}^d$ is the unit hypercube, and any activation function of the form of Equation. (147) for which

$$0 < \int_{\mathbb{R}} \mathrm{g}^2(x) dx < \infty \tag{148}$$

there exists a sequence $f_K(x)$ and a sequence of probability measures $\mu_{K,\Omega,\alpha}$ such that the distance between the original $f(x)$and the approximated function $f_K(x)$ converges to zero for large *N* [183, 186]:

$$\rho_N(f, f_K) = \sqrt{E \int_N |f(x) - f_K(x)|^2 dx} \xrightarrow[k\to\infty]{} 0 \quad (149)$$

The key insight here is that the number of basis functions K is variable, and a sufficiently large number of randomly-chosen basis functions still approximates the target function *f*(*x*). Intuitively, one might expect that we would require a larger number of random basis functions than of optimally-chosen basis functions to achieve the same approximation error; however, such an analysis is beyond the scope of this article. Other authors have studied other classes of original functions as well [187, 188].

At a practical level, randomized learning has been shown to be an effective means of compensating for the slow convergence of existing neural network training algorithms. Randomization can be applied in a neural network by fixing the network configuration randomly; assigning random values to some of the adaptive parameters; and adding noise to input data or parameters in the training stage [189]. In the single hidden layer feed-forward network (SLFN) [190], the weights between the input and hidden layers are selected randomly, and the weights between the hidden and output layers are determined via linear regression. The well-known family of Extreme Learning Machines (ELM) are based on SLFN [191] [192-195]. Albers et al. [196] studied feed-forward neural networks in which the weights are selected randomly [196]. Functional link neural networks with random weights and biases were studied in [186, 197-199]. Maass et al. proposed the Liquid State Machine, which is a recurrent neural network with randomly chosen input and internal weights [200]. The Random Neural Network is a fully connected neural

network in which neurons exchange positive and negative signals; if an neuron is excited, its activation has a positive value, and it *may* fire (based on a draw from an exponential distribution) [201]. Vincent et al. proposed stacked denoising autoencoders for deep neural network learning; they add random noise to each incoming signal during training for a layer, and then train the neurons to remove that noise [202]. Random weights in back propagation for deep neural networks have been studied in [203]. Randomized learning of convolutional neural networks has been studied in [204-207]. More on randomized learning in neural networks can be found in[191, 192, 208-210].

The Random Vector Functional Link Network (RVFLN) applies a nonlinear transformation to its inputs in order to enhance them before feeding them to the input layer. The weights connecting the original to enhanced input nodes are selected randomly, and the output weights are optimized by minimizing the system error as [198]:

$$E = \frac{1}{2N}\sum_{i=1}^{N}(t^i - B^t d^i)^2 \quad (150)$$

where B^t is the vector of output weights, d is the enhanced input and there are N input data. A number of other studies have been conducted on RVFLN; recent examples include [210-212]. A Recurrent Neural Network (RNN) has at least one cyclic path in its connections. Echo-state networks [213] and Liquid State Machines[200] select the input and internal weights randomly, and calculate the output weights by the Recursive Least Squares (RLS) algorithm.

7.2.1.1. Extreme Learning Machines

The Extreme Learning Machine (ELM) algorithm defined in [191] is based on the SLFN architecture [190]. SLFN uses randomized learning, where the input weights are selected from

uniform random values in [-1, 1] and the output are determined by minimizing the following error [189, 190]:

$$\varepsilon^2 = \sum_{i=1}^{N} (y_i - \sum_{j=0}^{k} w_j f_{ij})^2 \quad (151)$$

where N is the number of data points, y_i is target, k is number of hidden neurons in the hidden layer, w_j is the output weight and f_{ij} is the activation value of the j-th hidden neuron on the i-th data point. The output weights (w) are obtained as [189, 190]:

$$W^{output} = R^{-1}P \quad (152)$$

$$R = \sum_{i=1}^{N} F_i F_i^T; \quad P = \sum_{i=1}^{N} y_i F_i^T; \quad F_i = [f_{i0}, f_{i1}, \dots, f_{ik}]^T$$

where f_{i0} is the bias term. In the ELM architecture, the output weights are calculated as:

$$\hat{\beta} = H^{\dagger} Y \quad (153)$$

where

$$H(w_1, \dots, w_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathrm{x}_1, \dots, \mathrm{x}_{\tilde{N}}) = \quad (154)$$

$$= \begin{bmatrix} g(w_1.\mathrm{x}_1 + b_1) & \cdots & g(w_{\tilde{N}}.\mathrm{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.\mathrm{x}_N + b_1) & \cdots & g(w_{\tilde{N}}.\mathrm{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad (155)$$

$$Y = \begin{bmatrix} \mathrm{y}_1^T \\ \vdots \\ \mathrm{y}_N^T \end{bmatrix}_{N \times m} \quad (156)$$

where $\tilde{N}$ is number of neurons in the hidden layer , N is the number of pairs of samples ($\mathrm{x}_i \in \mathcal{R}^n, \mathrm{y}_i \in \mathcal{R}^m$), ($\beta_i$) is the vector of output weights, and $H^{\dagger}$ is the Moore-Penrose generalized inverse of H. This network is a universal approximator if each neuron has an activation function, g, which is infinitely differentiable. Logistic, exponential, sine and cosine functions can all be considered candidate activation functions.

Online Sequential ELM (OS-ELM) is an incremental-learning variation on the ELM architecture, in which output weights (β_i) are estimated via the Recursive Least Square (RLS) algorithm [192]:

$$\hat{\beta} = (H^T H)^{-1} H^T Y \quad (157)$$

The OS-ELM incorporate a "boosting" step (parameter initialization for RLS), and a sequential learning step [192].

- *Boosting Step*: First a small initial training set $\{(\mathrm{x}_i, \mathrm{y}_i) | \mathrm{x}_i \in \mathcal{R}^n, \mathrm{y}_i \in \mathcal{R}^m, \ i = 1, \dots, \widetilde{N}\}$, selected, and random input weights and biases are drawn. Then $H_0 = [h_1, \dots, h_{\widetilde{N}}]^T$ is computed, where $h_i = [g(w_1.\mathrm{x}_i + b_1), \dots, g(w_{\widetilde{N}}.\mathrm{x}_i + b_{\widetilde{N}})]^T$. The initial output weights are estimated as: $\beta^{(0)} = M_0 H_0^T Y_0$ where $M_0 = (H_0^T H_0)^{-1}$ and $Y_0 = [y_1, \dots, y_{\widetilde{N}}]^T$.
- *Sequential learning Step*: For each sample $(\mathrm{x}_i, \mathrm{y}_i)$, $h_{k+1} = [g(w_1.\mathrm{x}_i + b_1), \dots, g(w_{\widetilde{N}}.\mathrm{x}_i + b_{\widetilde{N}})]^T$ is obtained and the output weights are calculated based on the RLS algorithm as:

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}} \quad (158)$$

$$\beta^{(k+1)} = \beta^{(k)} + M_{k+1} h_{k+1} (y_i^T - h_{k+1}^T \beta^{(k)}) \quad (159)$$

A review of ELM architectures, including the fully complex ELM [214], incremental ELM [215] and pruning ELM [216] can be found in [194].

7.3. Methodology

7.3.1. ANCFIS-ELM Network Design

ANCFIS-ELM is our first design in combining ANCFIS and randomized learning which is proposed in [176] as a five-layer network based on ANCFIS [25].

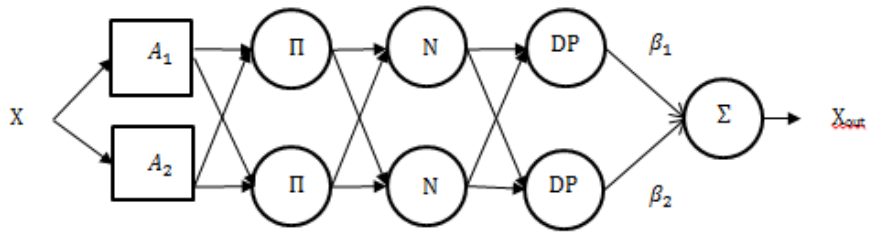


Figure 13: ANCFIS-ELM network for a univariate time-series with two membership functions

ANCFIS-ELM introduces a randomized learning algorithm ANCFIS. The sinusoidal MF parameters are selected randomly, thus eliminating the backward pass entirely. ANCFIS-ELM and ANCFIS are identical in structure for Layers 1-4: membership grades for an input are obtained, the firing strength of each rule is computed, then normalized, and rule interference is implemented. However, ANCFIS-ELM is based on the zero-order Takagi-Sugeno model (constant consequents); thus, it does not have ANCFIS's layer 5, and the outputs of layer 4 are directly connected to a weighted sum in the output layer. The output weights are obtained by the recursive least squares algorithm following ELM's sequential learning step [192] (Equation (158)- (159)).The initial output weights are determined by [217]:

$$\beta^{(0)} = 0 \qquad (160)$$

$$M_0 = \lambda^{-1} I, \qquad (161)$$

where λ is a small positive constant, and I is the identity matrix.

7.3.2. RANCFIS Network Design

The Randomized Adaptive Neuro-Complex Fuzzy Inference System (RANCFIS) is a six-layer feed-forward network using the same neuron transfer functions as ANCFIS [25], but with adaptive weights added on the connections between Layers 5 and 6.

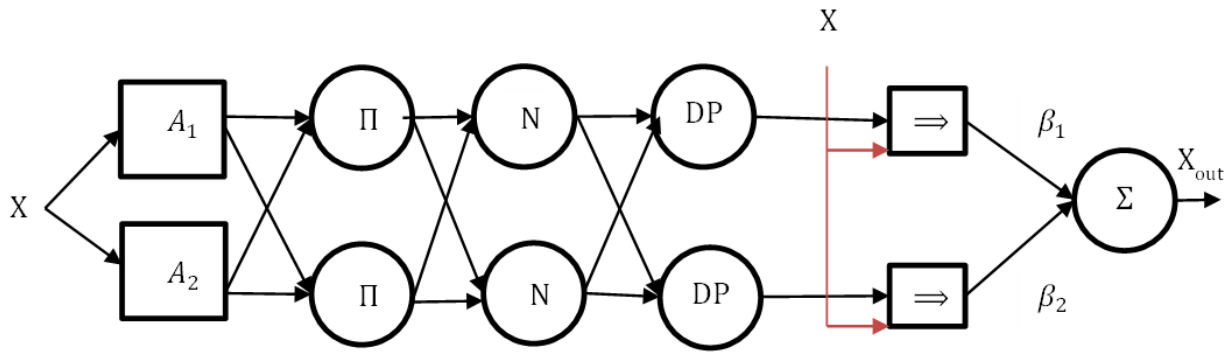


Figure 14: RANCFIS Architecture for a univariate time-series with two membership functions

RANCFIS randomly selects the sinusoidal membership function parameters $\{a, b, c, d\}$; as with ANCFIS-ELM, this eliminates the backward pass in training completely. Parameters are drawn from a uniform distribution, and then held constant throughout training and testing. In RANCFIS, however, Layer 5 is a linear function as in ANCFIS, rather than a constant as in ANCFIS-ELM. Layer 5 is fully connected to layer 6, with each connection weighted by the coefficient β_i. Thus, the i-th input to a Layer 6 neuron is:

$$I_{6,i} = w_i^{DP}\left[\sum_{k=1}^{j}\sum_{l=1}^{n} p_{i,kl}x_{kl} + r_i\right].\beta_i = w_i^{DP}\left[\sum_{k=1}^{j}\sum_{l=1}^{n} \gamma_{i,kl}x_{kl} + \eta_i\right] \quad (162)$$

where $\gamma_{i,kl} = p_{i,kl}.\beta_i$ and $\eta_i = r_i.\beta_i$. The output of the layer 6 is summation of the inputs to this layer (similar to ANCFIS) and defined as:

$$O_{6,j} = \sum_{i=1+(j-1)*N}^{j*N} I_{6,i} \quad (163)$$

where j is the number of outputs and N is the number of rules. This output can be reformulated as:

$$\mathrm{T} = \mathrm{H}.\beta \quad (164)$$

where

$$\mathrm{T}^T = [O_{6,1} \quad \cdots \quad O_{6,j}] \quad (165)$$

$$\mathrm{H}^T = \begin{bmatrix} w_i^{DP} x_{11} & w_i^{DP} x_{21} & \dots & w_i^{DP} x_{j1} \\ \vdots & \vdots & \vdots & \vdots \\ w_i^{DP} x_{1n} & w_i^{DP} x_{2n} & \dots & w_i^{DP} x_{jn} \\ w_i^{DP} & w_i^{DP} & \cdots & w_i^{DP} \end{bmatrix}$$

$$\beta^T = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1n} & \eta_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{j1} & \gamma_{j2} & \cdots & \gamma_j & \eta_j \end{bmatrix}$$

β is estimated as ANCFIS-ELM by using RLS algorithm [218] (Equation (158)and (159)) and initial weights are obtained by Equation (160) and (161).

7.3.3. Experimental Design

As discussed, ANCFIS is designed to accept an input window rather than lagged inputs of the time-series. However, in [171], we considered possibility of subsampling the input windows (see Chapter 5). The delay embedding methods (Section 3.3) were studied there as an approach to obtain subsamples of the input vectors (which we again refer to as delay vectors). Note, however, that we have not returned to the orthogonal-input interpretation of a delay vector; we still provide the entire vector as a single input to ANCFIS. The goal is merely to eliminate redundancy in the input vectors; the approach used in [25] was to take one entire approximate "period" (an approximately repeating subsequence) of the time-series as an input window. We found the subsampling approach to be effective, and to substantially speed up training. We will be using the subsampled input windows in our experiments in the current chapter; note that this also makes our

results directly comparable to standard machine learning algorithms using lagged inputs. As a cross-check, we will also use the full "one period" input windows for a subset of our data sets.

All of our experiments follow a chronologically ordered single-split design, with all the data points in the training set are earlier in time than the ones in the testing sets. For univariate time-series, the downsampling rate and the dimensionality of the delay vectors are obtained by the mutual information [96] and false nearest neighborhood (FNN) [98], respectively, and for multivariate time-series, mutual information [96] and the dimensionality-estimation approach presented in [99] (KDD) are used for delay and dimension, respectively, (see section (3.3)). The results are compared using the RMSE (Equation. (133)). To find out if there are significant differences between performances of RANCFIS, ANCFIS-ELM and ANCFIS, the Friedman test is applied. The Friedman statistic, *S*, is calculated as [113]. If null hypothesis H_0: $[\tau_1 = \cdots = \tau_k]$ is rejected, the approaches are compared together using the Multiple Comparisons with the Best (MCB) method [116] (see section 3.5.1.1).

Moreover, to compare learning speed of RANCFIS, ANCFIS-ELM and ANCFIS, we run an experiment with the same number of membership functions for each of the structures; two membership functions is considered for this experiment. To obtain the execution time for each of the architectures, we use *clock* function in the C language which returns the processor time consumed by the program. The function starts as the program runs and ends at the end of the training section before passing the testing set.

7.3.4. Data Sets

To study performance of RANCFIS and ANCFIS-ELM, ten time-series are considered including six univariate and four multivariate time-series. Solar Power, Santa Fe Laser A, Stellar, Mackey-Glass, Sunspot and wave are the univariate time-series discussed in Section 3.4.1.1 - 3.4.1.6. Motel, Precipitation, Flour and NASDAQ are multivariate time-series introduced in Section 3.4.2. Delays and dimensions for delay vectors are obtained as Table 25:

Table 25: Delays and dimensions for the univariate and multivariate time-series.

Time-series	Delay	Dimension
Solar Power	370	12
Santa Fe Laser A	2	9
Stellar	7	6
Mackey-Glass	11	9
Sunspot	4	5
Wave	4	4
Motel	[6 6]	[2 2]
Precipitation	[3 1 3]	[1 3 1]
Flour	[1 1 1]	[1 3 2]
NASDAQ	[1 1]	[1 3]

7.3.5. Experimental Results

In this section, we first compare one-step-ahead prediction by RANCFIS against ANCFIS-ELM and ANCFIS in Table 26:

Table 26: Comparing Results of RANCFIS, ANCFIS-ELM and ANCFIS in terms of RMSE

	RANCFIS	ANCFIS-ELM	ANCFIS
Solar Power	**1.949**	42.759	3.106 [171]
Santa Fe Laser A	**0.0523**	0.263	0. 114 [171]

Mackey-Glass	0.021	0.198	**0.015**[171]
Stellar	**0.0120**	0.216	0.014 [171]
Sunspot	0.112	0.205	**0.103**
Wave	0.121	0.225	**0.009**
Motel	**0.041**	0.197	0.155[7]
Flour	0.198	0.215	0.196
NASDAQ	**23.49**	87.285	23.50
Precipitation	0.206	0.204	**0.198**

In order to determine significant differences between the RANCFIS, ANCFIS-ELM and ANCFIS applied on the 10 time-series, the Friedman statistic, *S*, is calculated as Equation (135) with *k=3* and *n=10*, we obtain *S=15*. With $\alpha = 0.05$, $\chi^2_{k-1,\alpha} = 6.2$ (for 2 degree of freedom); that shows that the null hypothesis is rejected and there is significant difference between the approaches. Therefore, we compare the approaches using MCB method; for α=0.05, we obtain $q_\alpha = 3.315$ giving $r_{\alpha,K,N} = 1.048$. By considering $\overline{R_1}$, $\overline{R_2}$ and $\overline{R_3}$ as average rank of RANCFIS, ANCFIS and ANCFIS-ELM, respectively, we have:

$|\overline{R_1} - \overline{R_2}| = 0 < r_\alpha$
$|\overline{R_2} - \overline{R_3}| = 1.5 \geq r_\alpha$
$|\overline{R_1} - \overline{R_3}| = 1.5 \geq r_\alpha$

which clears that there is no significance difference between RANCFIS and ANCFIS; however RANCFIS and ANCFIS outperform ANCFIS-ELM substantially.

Table 27 shows the time spent in seconds for each algorithm to complete training with two membership functions, and the ratio between RANCFIS and the other structures.

Table 27: Comparing learning speed of RANCFIS, ANCFIS-ELM and ANCFIS in terms of second

	RANCFIS	ANCFIS-ELM	ANCFIS	$\frac{\text{RANCFIS}}{\text{ANCFIS-ELM}}$	$\frac{\text{ANCFIS}}{\text{RANCFIS}}$
Solar Power	2.222	0.699	5.388	3.179	2.425

Santa Fe Laser A	0.137	0.043	3.362	3.186	24.540
Mackey-Glass	0.070	0.024	3.161	2.917	45.157
Stellar	0.044	0.023	1.624	1.913	36.909
Sunspot	0.019	0.009	1.039	2.111	54.684
Wave	0.017	0.009	0.771	1.889	45.353
Motel	0.034	0.010	0.831	3.4	24.441
Flour	0.215	0.016	1.232	13.438	5.730
NASDAQ	0.126	0.027	0.255	4.667	2.024
Precipitation	0.634	0.042	1.341	15.095	2.115

Table 27 shows that the training execution time for RANCFIS in all of the univariate time-series except Solar Power (the largest one) is faster than ANCFIS by one order of magnitude; in the Solar Power data set, it is more than twice as fast as ANCFIS. In the multivariate time-series, RANCFIS is again faster than ANCFIS. As one would expect, RANCFIS is slower than the simpler ANCFIS-ELM model in all the time-series, but is also more accurate.

Table 28 compares RANCFIS against other well-known approaches were applied recently on each time-series in the literature. RANCFIS is superior to all others on 4/10 data sets.

Table 28: Comparing RANCFIS results with other approaches

	RANCFIS	Other Approaches
Solar Power	**1.949**	6.811[1]
Santa Fe Laser A	0.0523	**0.037**[2][219],
Mackey-Glass	0.021	**0.0047**[3][220]
Stellar	**0.0120**	0.018[4][220]
Sunspot	0.112	**0.036**[4][220]
Wave	0.121	**0.007**[171]

[1] RBFN
[2] Fuzzy Boolean Neural Network
[3] TSK-NFIS
[4] Neural Net

Motel	**0.041**	0.184[2][7]
Flour	0.198	**0.178**[2][9]
NASDAQ	**23.49**	33.11[1] [48]
Precipitation	0.206	**0.203** [2][9]

To see the effect of period-based input vectors, we apply RANCFIS to the Santa Fe Laser A and Mackey-Glass data sets, and compare the results to ANCFIS [25]. The input vector lengths for Mackey-Glass and Santa Fe Laser A are 44 and 8, respectively [25].

Table 29: Performances of RANCFIS and ANCFIS on period-based input vectors in terms of RMSE.

	RANCFIS	ANCFIS [25]
Santa Fe Laser A	0.082	0.033
Mackey Glass	0.00039	0.00055

7.4. Conclusion

In this chapter, we have proposed a new machine learning algorithm based on CFS&L. It is a feed-forward neuro-complex fuzzy systems which employs randomized learning. This modification eliminates the backpropagation learning step, and leads to a fast learning rate. Comparing the performance of RANCFIS with the other variations of neuro-complex fuzzy systems (ANCFIS-ELM and ANCFIS) in Table 26 shows that among the ten time-series under study, RANCFIS is superior in half of them and ANCFIS is better in the rest. The Friedman test confirms that there is no significant difference between the RANCFIS and ANCFIS, while both are clearly superior to ANCFIS-ELM.

[1] CNFS-ARIMA
[2] SVR

RANCFIS's speed and accuracy compared to the other neuro-complex fuzzy systems makes it a suitable choice for studying data stream mining. In comparison with other approaches that have been attempted on these time-series (Table 28), RANCFIS is the best on four out of ten time-series. We are, however, unable to comment on the comparative execution times of these algorithms. In future work, we will examine RANCFIS as a general stream-mining algorithm, for both forecasting and classification problems.

However, the randomized learning algorithms do not assure that randomly-selected parameter are optimal parameters for the system; in RANCFIS, this may manifest as a need for a large number of complex fuzzy sets. This would limit application of RANCFIS for large-scale learning. In the Chapter 8, we design a compact, fast and accurate learning algorithm suitable for data stream prediction.

Chapter 8

FANCFIS: Fast Adaptive Neuro-Complex Fuzzy Inference System

Large-scale time-series forecasting is an increasingly important problem, with sensor networks pouring out sampled data at unprecedented rates, petabytes of transactional data coming every day from giant stores such as Walmart, and terabytes of data generated over Facebook each day. While these are only examples of the much larger domain of general data streams, the uniformly-sampled time-series still remains a very large and important subdomain. Extensive research has shown that machine-learning algorithms can often be very effective forecasting models, but many of these algorithms do not scale well. The ANCFIS is one such approach; built to leverage complex fuzzy sets, it is both an accurate and parsimonious forecasting algorithm. However, its scaling is poor due to a relatively slow training algorithm (gradient descent hybridized with chaotic simulated annealing). Before the algorithm can be used for large-scale forecasting, a fast training algorithm that preserves the system's accuracy and compactness must be developed.

We propose the Fast Adaptive Neuro-Complex Fuzzy Inference System, which is designed for fast training of a compact, accurate forecasting model. We use the Fast Fourier Transform algorithm to identify the dominant frequencies in a time-series, and then create complex fuzzy sets to match them as the antecedents of a complex fuzzy rule. Consequent linear functions are then learned via recursive least-squares. We evaluate this algorithm on both univariate and multivariate time-series, finding that this incremental-learning algorithm is as accurate and compact as its slower predecessor, and can be trained much more quickly.

8.1. Introduction

Data streams are continuous flows of information arriving over time. These include sensed data, web clickstreams, stock market quotes, Internet traffic, and more. The data could be used for real-time decision applications, retrospective analyses, condition monitoring, entertainment, etc. Working with data streams is more challenging than conventional data sets because of their volume, velocity and volatility [221, 222]. Volume is the amount of data collected over time, the velocity of the data refers to the rate at which data is generated transmitted on the stream, and the volatility relates to changes in data distribution or data meaning over time. The volume and velocity of data in the modern world are famously huge and increasing rapidly. In 2012, over 2.8 ZB of data were generated and processed and this amount will increase by 15 times by 2020 [223]. Every minute, 2 million posts are shared on Facebook, 277,000 tweets are generated by Twitter, and 4 million search queries are submitted to Google [224]. Volatility, in the meantime, refers not only to a distributional shift in the data stream, but possibly to a change in its semantics as well. For example, a spam email can come to be considered a regular email over time due to changes in the content of the spam email, in the behaviour of the spammer and in the perspective of the receiver regarding a spam category [225]. The volume, velocity and volatility of stream data imply that models must be incrementally learned for a limited historical time window. The model must furthermore be accurate, and quick to train; the latter usually implies compactness as well, since smaller models can be trained faster (all things being equal).

A time-series is one particular case of streaming data. While the questions of volume, velocity and volatility remain the same, a time-series carries extra information in the form of timestamps for each observation; most commonly, these are the result of a sampled data-collection

process. This being the case, analysts can (and do) employ state-space and auto-regressive models for forecasting that might not be viable for more general streams. Both statistical models (e.g. FARIMA [226], GARCH [227], etc.) and machine-learning models [228-230] have been employed in forecasting, and found to be effective on a wide variety of data sets.

As one example, machine learning algorithms based on CFS&L have been shown to be very accurate and parsimonious forecasting models [25, 54, 55, 171]. The ANCFIS was the first machine learning algorithm based on Ramot's CFS&L [25] and Dick's findings in [15]; the system is based on Jang's ANFIS [231], modified to employ complex fuzzy sets in its rule antecedents and rule interference in its inferential process. Results in [25] showed that ANCFIS was very accurate even with a very small rulebase; univariate chaotic time-series were modeled well even with three or fewer rules. However, the backwards pass in ANCFIS is a hybrid of gradient descent and chaotic simulated annealing, which makes the algorithm relatively slow (See section 3.2)

The RANCFIS is a variant of ANCFIS that employs randomized learning; in which some of the adaptive parameters in the system are picked randomly and the other parameters are learned [6, 218] (see Chapter 7). RANCFIS was shown to learn much faster than other variations of ANCFIS, while being just as accurate [9]. The weakness of RANCFIS, however, is that (like other random learning algorithms) there is no way to assure that the randomly-chosen parameters are optimal for a given data set. We expect this would manifest as a need for a larger number of complex fuzzy sets (and therefore rules) than an optimal ANCFIS would require. Thus, RANCFIS models are quite likely to be less compact that ANCFIS models achieving the same accuracy. The added complexity of a RANCFIS model that matches the accuracy of an ANCFIS model is unlikely to lead to an overall increase in training time; however, additional complexity is

undesirable for large-scale learning. The question, therefore is: can a variant of ANCFIS with fast incremental learning be developed that retains the accuracy and compactness of the original?

In this chapter, we develop fast incremental algorithms for time-series forecasting based on ANCFIS [25]. The Fast Adaptive Neuro-Complex Fuzzy Inference System (FANCFIS) is a classic incremental-learning algorithm, drawing inspiration from radial basis function networks. In an initialization step, a Fast Fourier Transform (FFT) is used to determine the dominant frequencies in a time-series, and a CFS matching each one is added to the model. Then, the consequent parameters of the FANCFIS model are updated incrementally via recursive least squares. In our experiments, we find that both FANCFIS and RANCIS are as accurate as the ANCFIS technique, and significantly faster.

Our main contribution in this chapter is designing accurate and fast systems based on the ANCFIS architecture for data stream prediction. Our second contribution is to compare learning with randomized vs. induced complex fuzzy sets in the ANCFIS architecture, to provide new evidence on the advantages and disadvantages of randomized learning.

8.2. Background

8.2.1. Data Stream Mining

Data stream mining shares with all other data mining approaches the goal of turning vast amounts of data into useful, actionable knowledge. The problems of volume, velocity and volatility, combined with the permanent incompleteness of stream data, are the defining challenges of the stream mining field [221, 222]. The volume problem, for instance, directly implies that the entire historical data stream cannot be stored, and so any model trained on the data stream will have to be “single pass” [222]. The velocity problem indirectly reinforces this single pass

requirement, as training in a single pass over the data tends to be faster than multiple passes. The model must also be able to adapt to concept drifts in the data over time, which essentially means that older data must be "forgotten" in favor of new observations [232]. Again, a single pass algorithm generally lends itself to this adjustment. Beyond this, different techniques have been proposed to deal with the large volume of the data stream including load shedding, different sampling techniques, and windowing the data [233-236] [237] [238] [239, 240] [241] [242]. Various scaling techniques have been studied to address high volume and high velocity in stream data [243-246]. Volatility has been examined in many studies [247-250] [225, 251]; time windowing, instance weighting, and evolving classifier ensembles seem to be the three principal approaches to explicitly detecting and adapting to concept drifts [252].

Machine learning has of course been employing in data stream mining; plainly, existing algorithms need to be modified – or entirely new ones developed – to work with data streams. Various clustering, classification and prediction techniques have been developed for data streams [221]. A wide variety of specialized stream clustering algorithms have been developed over the years, with CluStream [242] being one of the earliest and best-known. The algorithm is a *k*-means clustering approach, applied over *microclusters*; small clusters of observations that are close in time as well as in feature space. The *k*-means algorithm then runs over the prototypes of the microclusters. More recently, algorithms such as STRAP [253] and AutoClust [254] have been developed. Other clustering techniques have been discussed in [255] [241] [256] [257] [242].

Stream classifiers are often developed from the familiar algorithms for tabular data. As these are not generally single-pass or incremental-learning algorithms, time windowing is used to temporarily retain a limited amount of historical data for training the classifiers. Numerous algorithms, including decision trees, neural networks, support vector machines and ensemble

learning have thus been adapted for the stream classification task [238] [258] [259] [239, 260]. Similarly, stream prediction algorithms are often built from classic tabular algorithms, again using the windowing approach [261] [262] [263] [264]. Surveys of machine learning in data streams mining may be found in [151, 237, 265-268].

8.2.2. Discrete Fourier Transform

According to the Fourier Theorem, any periodic signal can be represented by a (possibly infinite) sum of sine and cosine terms. Building on this result, the Fourier transform re-expresses a time-domain signal as a complex-valued function of frequency, with each frequency representing a different sinusoid in the sum above. The Fourier transform of a non-periodic signal can be obtained by considering the signal's period to be infinite [269]. In the case of discrete signals such as time-series with finite points, the discrete Fourier transform is used; only a limited number of sine and cosine terms is needed to represent the discrete signals precisely. The Fourier transform of a time-series with *N* data points, $A(k)\ k = 1, 2, \dots, N$, is calculated as [270]:

$$F_A(n) = \sum_{k=0}^{N-1} \frac{A(k)}{N} e^{-i2\pi nk/N} \quad (166)$$

where $F_A(n)$ is discrete Fourier transform (DFT) and is a complex number where its real and imaginary part are amplitude of sine and cosine wave, respectively. n is frequency and *N* is number of data points in the time-series $A(k)$. Inverse of the Fourier transform reconstructs time-series as [270]:

$$A(k) = \sum_{n=0}^{N-1} F_A(n) e^{\frac{i2\pi nk}{N}} \quad (167)$$

$$= \sum_{n=0}^{N-1} F(n)_{real} \cos(\frac{2\pi nk}{N}) - \sum_{n=0}^{N-1} F(n)_{imag} \sin(\frac{2\pi nk}{N})$$

where $F(n)_{real}$ and $F(n)_{imag}$ are real and imaginary parts of the Fourier transform, respectively. Plainly, a time-series with *N* data points needs *n*-1 waves with different frequencies to be reconstructed (*n*=0 is mean of the time-series).

Power spectrum of a frequency captures how much of the variance of a time-series is related to the given frequency (without consideration of phase) and is calculated as [270]:

$$|F_A(n)|^2 = [F(n)_{real}]^2 + \left[F(n)_{imag}\right]^2 \quad (168)$$

It should be noted that discrete Fourier transform assumes the full period of the time domain signal is presented; thus, the first and last point in the time-series should be almost similar. Otherwise the endpoint mismatch distorts power spectrum, creating a spurious spike at high frequencies. one solution to this problem is to multiply the time-series with a window function that smoothly decays to zero at the beginning and end of the time-series [271]. Alternatively, the end-to-end mismatch software in Tisean, instead of using a window function, finds a subsequent of the time-series whose endpoints more closely match [272].

Fast Fourier transform (FFT) is an efficient algorithm for calculating DFT. The complexity of regular DFT for *N* point is $O(N^2)$; however, FFT reduces it to $O(Nlog(N))$ [273]. Cooley et al. proposed the first FFT algorithm [274]; they showed that it is possible to divide a DFT computation of length N to smaller computations of size $N/2$ [273]:

(169)

$$F_A(n) = \sum_{k=0}^{N-1} a_n e^{-i2\pi nk/N}$$
$$= \sum_{k=0}^{N/2-1} a_{2n} e^{-i2\pi(2n)k/N} + \sum_{k=0}^{N/2-1} a_{2n+1} e^{-i2\pi(2n+1)k/N}$$
$$= \sum_{k=0}^{N/2-1} {a_n}^{even} e^{-i2\pi nk/(\frac{N}{2})}$$
$$+ e^{-i2\pi nk/N} \sum_{k=0}^{N/2-1} {a_n}^{odd} e^{-i2\pi nk/(\frac{N}{2})}$$

where the smaller parts are similar to a smaller DFT which can be divided to smaller parts as well.

DFT has been used for similarity search in a time-series. Sequences are transformed to the frequency domain; then the first few frequencies (ordered by the power spectrum) are compared against each other [275, 276]. [277] used DFT for similarity search over time-series stream. [278] proposed a monitoring system based on DFT for time-series which is able to detect similarity in multiple time-series stream in real time. Periodic pattern recognition has been investigated using autocorrelation function and fast Fourier transform [279] and power spectral density estimation using DFT and tree index [280]. Online clustering of data streams was performed in [281] by using a preprocessing step in which DFT is applied to obtain the distance between data streams. [282] used DFT in its structure in order to capture recurrent concepts in data streams; Decision Tree models are transformed by DFT when concept drift occurs.[283] proposed a system for anomaly detection in RFID tags which employs Fourier transform for outlier detection. [284] proposed a change detection algorithm for unsupervised data stream based on Fourier transform.

8.3. Methodology

8.3.1. FANCFIS

FANCFIS is a six-layered feed forward neural network inspired by RANCFIS; the latter's speed advantage over ANCFIS comes from eliminating the backwards learning pass in the network in favor of a random selection of CFS parameters. the fundamental difference between them is that FANCFIS does not employ randomized learning algorithm in its structure. Instead, the sinusoidal membership functions are determined by an FFT of a portion of the given time-series. We suspect that this would lead to a more compact network, as the CFS would be specifically tuned to the data set under analysis, and we will thus need fewer of them for the same level of accuracy. The actual neuron transfer functions in each layer are identical between RANCFIS and FANCIS.

The FANCFIS learning algorithm is a two-step process including an initialization step and an incremental learning step. Briefly, in the initialization step, membership functions parameters and the delay and dimension of the delay vector are determined. Then, the initial output weights of FANCFIS are calculated. In the incremental learning, the output weights are updated after each new data point is observed.

In the initialization step, a portion of the time-series is processed as a batch; We know that in the DFT, a time-series is represented by a limited number of coefficients in the frequency domain, and that frequencies with highest power spectrum give more information about the time-series [270, 285]. Thus, to create k CFS, we will select the k frequencies with the highest poser spectrum. We use the Tisean software [272] to identify a subsequent of the initialization data with a minimal endpoint mismatch. Then, the FFT of that subsequence is calculated and the k

frequencies with the highest power spectrum are identified. Membership functions are determined as:

$$r(\theta) = F(n_i)_{real}\cos(\theta = x) - F(n_i)_{imag}\sin(\theta = x) + DC \quad i = 1,2,\dots,N_{mf} \qquad (170)$$
$$\theta = \frac{2\pi n_i k}{N} \quad k = 1,2,\dots,N$$

where r and θ are amplitude and phase of the complex membership grade, respectively, $x \in X$ is an object of the universal set X, n_i indexes frequencies with the highest power spectrum, N_{mf} is the total number of membership functions,, $F(n_i)_{real}$ and $F(n_i)_{imag}$ are respectively the real and imaginary parts of the FFT for frequency of n_i, N is length of the length of the subsequence determined by Tisean and DC is zero frequency component, $F_A(0) = \sum_{k=0}^{N-1}\frac{A(k)}{N}$. We use the FFT implementation in [286].

We estimate the delay vectors' parameters (delay and dimension) using the whole of the initialization data. We use the time-delayed mutual information heuristic to find our delay τ [96], and the approach in [99] is used to determine the dimension m (see section 3.3). As FANCFIS has been written in C++ language, we have classes responsible for calculating delay and dimension and creating the delay vectors in the initialization learning step. The delay class is based on implementation of mutual information in Tisean software [102] and the dimension class finds the best dimension based on [99] and K-nearest neighbors (KNN) search implementation in [287]; we consider 15 as the maximum dimension to be considered.

After estimating the membership function parameters and the delay vectors, we can calculate the initial output weights for FANCFIS using the RLS algorithm (Equations. (158) - (161)). Thereafter, we keep a sliding window of delay vectors constructed from the last w observations in the time-series. In the incremental learning step, we first form a new delay vector

as a new observation arrives, and simultaneously drop the oldest delay vector from the sliding window. We then use the RLS algorithm to update the output weights on FANCFIS for the new delay vector.

8.3.2. Complexity Analysis

We will study the time complexity (order of growth) for FANCFIS and stream version of RANCFIS in this section. Table 30 and Table 31 show FANCFIS and RANCFIS algorithm, respectively.

Table 30: FANCFIS Algorithm

FANCFIS Algorithm
A. Initialization Step: 1. Normalize the training set as: $\frac{x_i - \max(X)}{\max(X) - \min(X)}$ (171) where X is a univariate time-series and x_i is the i-th data point in the time-series. To normalize a multivariate time-series, Equation (171) repeats for each variate. 2. Obtain delay and dimension for delay vectors a. Use the mutual information heuristic to estimate τ (Equation. (127)) b. Use the k-NN approach to estimate the dimension m as proposed in [99]. The nearest neighbor search is carried out by a KD-tree algorithm. 3. Form the delay vectors. Equation.(131) for multivariate time-series and Equation. (125) for univariate time-series

4. Calculate membership functions
    a. Perform an FFT of the training set.
    b. Select the frequencies with the highest power spectrum, and create a CFS for each following Equation (170)
5. For N epochs repeat:
    a. Calculate output weights (Equation (158)-(159))

B. Incremental Learning Step

6. For each new data point, delete the oldest data point and re-normalize the training set
7. Create a new delay vector for the new data point
8. Update the output weights

Table 31: RANCFIS Algorithm

RANCFIS Algorithm
A. Initialization Step: 1. Normalize the training set per Equation (171). 2. Obtain delay and dimension for delay vectors a. Use mutual information to estimate τ b. The approach from [99] to estimate *m* 3. Form the delay vectors. Equation.(131) for multivariate time-series and Equation. (125) for univariate time-series

4. Select random parameters for membership functions
5. For N epoch repeats:
    a. Calculate the output weights

B. Incremental Learning Step

6. For each new data point, delete the oldest data point and re-normalize the training set.
7. Create a new delay vector for the new data points
8. Update the output weights

We will study the running time relative to six different inputs: the number of variates in the time-series (NumVar), the number of membership functions (NumMF), the number of network outputs (NumofOut), the number of training data examples used for the initial learning step (NumTr), the maximum allowed dimension (from [99]) (DMax), and number of delays to find the optimum delay from them using mutual information (D) . We assume that NumMF is the same for all the variates in the time-series.

For FANCFIS, based on Table 30, in each step, we have the following maximum number of operations:

1. $NumTr * numVar$
2. $DMax^{numVar} * (NumDV * numVar * DMax) * KnnSearch + D * numVar$

where $(NumDV * numVar * DMax)$ is the number of operations to obtain delay vectors for $DMax^{numVar}$ sets of dimension. The running time of *KnnSearch* based on KD-Tree is estimated as $O(Knear.\log(NumDV))$ where $Knear$ is the number of the nearest neighbors considered in *Knn* search and *NumDV* is the number of delay vectors for a given

delay and dimension. By considering *DMax* and *D* as the dimension and delay for all the variates, $numDV = NumVar * (NumTr - D * (DMax - 1))$.

3. $NumDV * numVar * numDim$ where *numDim* is the dimension calculated in the initial learning step (step 2).
4. $numVar * Fourier$

   where running time of discrete Fourier transform is estimated as $O(NumTr.\log(NumTr))$
5. $epoch * (numVar * numMF * DMax^2 + numVar * numMF + numMF^{numVar} * numVar + numOut * (numMF^{numVar} * DMax^{numVar} + numMF^{numVar})^2)$

Where $numVar * numMF * DMax^2 + numVar * numMF$ shows number of operations for calculation of membership degrees, $numMF^{numVar} * numVar$ is the number of operations for obtaining firing strength of the rules and $numOut * (numMF^{numVar} * DMax^{numVar} + numMF^{numVar})^2$is the maximum number of operations for calculating the output weights.

At the end of the initial learning, we have the following number of operations:

$$NumTr * numVar + D * numVar + DMax^{numVar} * (NumDV * numVar * DMax) * KnnSearch + NumDV * numVar * numDim + numVar * Fourier + epoch * (numVar * numMF * DMax^2 + numVar * numMF + numMF^{numVar} * numVar + numOut * (numMF^{numVar} * DMax^{numVar} + numMF^{numVar})^2)$$

The most time-consuming part is calculation of Fourier transform ($numVar * Fourier$). Thus, the complexity is: $O(NumTr.\log(NumTr))$. For incremental learning, we have the following number of operations for each step.

6. $NumTr * numVar$
7. $numVar * numDim$ where *numDim* is the dimension calculated in the initial learning step (step 2).
8. $numVar * numMF * DMax^2 + numVar * numMF + numMF^{numVar} * numVar + numOut * (numMF^{numVar} * DMax^{numVar} + numMF^{numVar})^2$

Thus, the operations for the step are:

$$NumTr * numVar + numVar * numDim + numVar * numMF * DMax^2 + numVar * numMF + numMF^{numVar} * numVar + numOut * (numMF^{numVar} * DMax^{numVar} + numMF^{numVar})^2$$

the most time-consuming part is normalizing the time-series for the new data points; thus the running time for the incremental learning is estimated as $O(NumTr)$.

For RANCFIS-Stream, based on Table 31, all the steps are same as the steps in DS-ANCFIS except the step four (determining membership functions); in this step, complexity for choosing random values for membership function is O(1). Thus, the complexity in the initial step is equal to the calculation of dimension for the delay vectors which is the most time-consuming section, $O(Knear.\log(NumDV))$, and for the incremental learning, the running time is estimated as $O(NumTr)$.

8.3.3. Experimental Design

In all of our experiments, we follow a chronologically ordered single-split design in which observations in the training set occur earlier that those ones in the testing set. To simulate data streams, for the initialization step, we use the chronologically earliest portion of the training set and the remaining observations are passed one by one to the system in the online learning. We do

explore the size of the initialization set; 10%, 20% and 50% of the training set are considered in our experiments. The explorations also consider different number of membership functions; we examine 1 to 5 membership functions for each variate in the time-series.

Our results in RANCFIS and FANCFIS are compared by RMSE and MAE statistics (Equations. (133) and (134)). To determine if there is significant difference between performances RANCFIS and FANCFIS, the Friedman test is applied. The Friedman statistic, *S*, is calculated as [113] (see section 3.5.1.1).

8.3.4. Data Sets

To study FANCFIS and RANCFIS performance in time-series stream prediction, 14 time-series are considered: ten univariate time-series (including four software reliability growth data sets) and four multivariate time-series (section **Error! Reference source not found.**)

8.3.5. Experimental Results

The one-step ahead prediction of time-series stream by FANCFIS and RANCFIS are compared against each other. Table 32 shows the best parameter set identified for each algorithm on each data set: the number of membership functions, delay and dimension of delay vectors and portion of data saved in the initial learning set. Table 33 shows the results of comparing of these two methods

Table 32: Number of membership functions, delay vector's parameters and portion of data saved

	FANCFIS				RANCFIS			
	Num of MF	Delay	Dimension	Portion	Num of MF	Delay	Dimension	Portion

Solar Power	3	0	1	50%	1	0	1	50%
Santa Fe	4	3	4	10%	1	3	4	10%
Mackey-Glass	2	9	3	50%	1	6	2	10%
Stellar	3	5	3	50%	1	5	3	50%
Sunspot	5	1	8	10%	1	1	8	10%
Wave	5	3	13	50%	2	3	13	50%
Mozilla	4	6	12	50%	1	6	12	50%
Android	5	2	15	50%	3	2	12	10%
ODC1	5	2	5	20%	3	2	5	20%
ODC4	1	2	8	50%	3	2	8	50%
Motel	2	[2 2]	[10 3]	50%	2	[3 2]	[2 1]	20%
Flour	1	[2 3]	[3 2]	20%	3	[3 1]	[1 4]	10%
NASDAQ	3	[6 7]	[5 7]	50%	1	[6 7]	[5 7]	50%
Precipitation	1	[1 1]	[6 15]	50%	1	[1 1]	[6 15]	50%

Table 33: Comparing FANCFIS and RANCFIS

	FANCFIS	RANCFIS
Solar Power	4.8948	4.8960
Santa Fe Laser A	0.0557	0.0648
Mackey-Glass	0.0136	0.0184
Stellar	0.01560	0.01563
Sunspot	0.0988	0.1031
Wave	0.0622	0.0741

Mozilla	0.0273	0.0274
Android	0.0639	0.0624
ODC1	0.0299	0.0304
ODC4	0.0325	0.0322
Motel	0.1928	0.2433
Flour	0.1447	0.1697
NASDAQ	23.957	24.027
Precipitation	0.20206	0.20206

In order to determine significant differences between FANCFIS and RANCFIS in time-series stream prediction, we calculate the Friedman statistic, *S*, in Equation. (135) with *k=2* and *n=14*. We obtain *S=6.3* with $\alpha = 0.05$, $\chi^2_{k-1,\alpha} = 4.57$ (for 1 degree of freedom); this shows that the null hypothesis is rejected and there is a significant difference between RANCFIS and FANCFIS.

To compare the execution time of FANCFIS and RANCFIS, Table 34 indicates the time taken for the initialization step in terms of second with 10% of portion of time-series saved in and by using two membership functions, respectively. We use Clock function in the C language to obtain the processor time consumed by the program.

Table 34: Time Taken for FANCFIS and RANCFIS in the initialization step for 2 membership functions and 10% of the data (seconds)

	FANCFIS	RANCFIS

Solar Power	122.028	121.255
Santa Fe Laser A	0.333488	0.323909
Mackey-Glass	0.030464	0.027915
Stellar	0.036497	0.037076
Sunspot	0.0182	0.018151
Wave	0.012745	0.013863
Mozilla	3965.68	3785.41
Android	199.018	197.41
ODC1	0.388585	0.373142
ODC4	1.2848	1.27255
Motel	0.010225	0.013563
Flour	0.07884	0.051655
NASDAQ	2.34974	1.63312
Precipitation	8.26981	8.334

Table 35 shows performance of FANCFIS against batch learning algorithms including batch learning version of RANCFIS [218] (see Chapter 7), ANCFIS, and other approaches in the literature.

Table 35: Performance of FANCFIS against other approaches

	FANCFIS	RANCFIS [218]	ANCFIS	Other Approaches
Solar Power	4.8948	1.949	3.106 [171]	6.811[1][6]
Santa Fe Laser A	0.0557	0.0523	0. 114 [171]	0.037[2][219],
Mackey-Glass	0.0136	0.021	0.015 [171]	0.0047[3][220]

[1] RBFN
[2] Fuzzy Boolean Neural Network
[3] TSK-NFIS

Stellar	0.01560	0.0120	0.014[171]	0.018[1][220]
Sunspot	0.0988	0.112	0.103[171]	0.036[4][171]
Wave	0.0622	0.121	0.009 [171]	0.007[171]
Mozilla	0.0273	-	-	0.0267[2][9]
Android	0.0639	-	-	0.066[5] [9]
ODC1	0.0299	-	-	0.028[5][9]
ODC4	0.0325	-	-	0.033[5][9]
Motel	0.1928	0.041	0.155[7]	0.1841[1][7]
Flour	0.1447	0.198	0.196[7]	0.178[1][7]
NASDAQ	23.957	23.49	23.50[7]	33.11[3] [48]
Precipitation	0.20206	0.206	0.198[7]	0.203 [4][7]

[1] Neural Net
[2] ANCFIS-ELM
[3] CNFS-ARIMA
[4] SVR

8.4. Conclusion

In this chapter, we have proposed FANCFIS, a fast learning variant of the ANCFIS neuro-complex -fuzzy system. The system first uses an FFT on an initialization window to determine optimal CFS membership functions for the complex fuzzy rule antecedents, then recursive least squares to update the consequent parameters. The performance of the system was compared against a stream version of RANCFIS (see Chapter 7). Our experimental results in Table 33 show that there is a significant difference between FANCFIS and RANCFIS performance, with FANCFIS being more accurate. However, FANCFIS will be asymptotically slower than RANCFIS based on the complexity study in Section 8.3.2.

The proposed systems are able to deal with volume and velocity in the data stream as it works on a window of data; thus, there is no need to save the whole data set. Moreover, the complexity study shows that they are fast enough to work with high velocity data stream. In the future work, we will modify the system to be able to deal with concept drifts in a data stream.

Chapter 9

Conclusions and Future Directions

9.1. Conclusions

Complex fuzzy sets and logics are developing in the recent years and more researchers are working in this filed. Machine learning algorithms based on complex fuzzy sets and logic are showing promising results in different areas specifically time-series prediction. In this dissertation,

we first reviewed the latest research in the area and then designed and developed machine learning algorithms based on CFS&L.

Machine learning algorithms used for time-series prediction mostly work with input windows of time-series which is lagged representation of the time-series. In this dissertation, we compared three different input representations for machine learning algorithms based on CFS&L. Input windows based on an ad-hoc determination of what constitutes one "period" in the data set, the traditional delay embedding, guided by the mutual-information and false-nearest-neighbour heuristics, and the use of only the false-nearest-neighbour heuristic were studied and compared against each other by applying on six univariate time-series data sets. While the "best" method appears to be data set-dependent, we found enough evidence that we recommend the latter method as the best combination of accuracy and expected computation time.

Before designing any new algorithm, we have evaluated the performance of the existence machine learning algorithms based on CFS&L. ANCFIS was designed to work on univariate time-series; thus, we have explored one-step-ahead forecasting ability of ANCFIS on photovoltaic power data set. Its results are compared against several different machine learning algorithms as well as ARIMA forecasting. We found that support vector regression was the most accurate approach on this data set, followed by ANCFIS.

In the next step, we have developed ANCFIS architecture for multivariate time-series prediction. Three different ANCFIS architectural designs (SISO, MISO, MIMO) have been examined. A further exploration of alternative designs of MIMO ANCFIS is also performed. The designs were compared with well-known machine learning algorithms, RBFN and SVR. A

Friedman test shows no significant difference between any of these methods. However, when we compare ANCFIS against the existing literature on these data sets, our method is clearly superior.

The main disadvantage of ANCFIS architecture is its slow learning algorithm. To speed up this neuro-fuzzy system, we have designed a new algorithm (RANCFIS). The algorithm is a feed-forward neuro-complex fuzzy system which has taken advantages of randomized learning algorithm in its structure. This modification in the structure eliminates the back propagation learning and leads to a fast learning rate. Our results showed that RANCFIS learning rate is faster than ANCFIS, and RANCFIS is slower that ANCFIS-ELM. Comparing performance of RANCFIS with the other variations of neuro-complex fuzzy systems (ANCFIS-ELM and ANCFIS) showed that among ten understudy time-series, RANCFIS outperforms in half of them and ANCFIS is better in the rest. However, the statistical significant tests cleared that there is no significant difference between RANCFIS and ANCFI. ANCFIS-ELM cannot compete with these two architectures.

In our last design, we have designed a fast and compact learning algorithm based on CFS&L suitable for data stream mining. The FANCFIS has a two-step learning step: initialization step and incremental step. The membership functions in this design is determined using FFT. Our experimental results showed that there is a significant difference between the FANCFIS and the stream version of RANCFIS performance, with FANCFIS being more accurate. However, FANCFIS will be asymptotically slower than RANCFIS. The FANCFIS is able to deal with volume and velocity in the data stream.

9.2. Future Directions

The FANCFIS design can be improved. The system is currently able to deal with volume and velocity in data streams but it does not have any structure to detect concept of Drift. Comparing two window of a time-series, implementing instance weighting or evolving the FANCFIS over time can be the solution to this problem.

Moreover, the machine learning algorithms designed and developed in this dissertation have been applied on time-series prediction. Applications of the algorithms in other areas need to be investigated such as classification, clustering and image processing.

Bibliography

[1] D. Ramot, *et al.*, "Complex fuzzy sets," *Fuzzy Systems, IEEE Transactions on,* vol. 10, pp. 171-186, 2002.

[2] D. Ramot, *et al.*, "Complex fuzzy logic," *Fuzzy Systems, IEEE Transactions on,* vol. 11, pp. 450-461, 2003.

[3] O. Yazdanbakhsh and S. Dick, "A Systematic Review of Complex Fuzzy Sets and Logic," *Fuzzy Sets and Systems,* 2017.

[4] O. Yazdanbakhsh and S. Dick, "Time-Series Forecasting via Complex Fuzzy Logic," in *Frontiers of Higher Order Fuzzy Sets*, A. Sadeghian and H. Tahayori, Eds., ed Heidelberg, Germany: Springer, 2015.

[5] O. Yazdanbakhsh, *et al.*, "Predicting Solar Power Output using Complex Fuzzy Logic," presented at the Fuzzy Information Processing Society (NAFIPS), 2013 Annual Meeting of the North American Edmonton, Alberta, 2013.

[6] O. Yazdanbakhsh and S. Dick, "Forecasting of Photovoltaic Power Output via Complex Fuzzy Logic," *Journal of Multiple-Valued Logic and Soft Computing,* Submitted.

[7] O. Yazdanbakhsh and S. Dick, "Multi-variate timeseries forecasting using complex fuzzy logic," in *Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American*, 2015, pp. 1-6.

[8] O. Yazdanbahksh and S. Dick, "Forecasting of multivariate time-series via complex fuzzy logic," *IEEE Transactions on Systems, Man and Cybernetics: Systems* Under Review.

[9] O. Yazdanbakhsh and S. Dick, "ANCFIS-ELM: A Machine Learning Algorithm based on Complex Fuzzy Sets," presented at the World Congress on Computational Intelligence, Vancouver, Canada, 2016.

[10] O. Yazdanbakhsh, *et al.*, "On Deterministic Chaos in Software Reliability Growth Models," *Applied Soft Computing,* Submitted.

[11] J. Buckley, "Fuzzy complex numbers," *Fuzzy Sets and Systems,* vol. 33, pp. 333-345, 1989.

[12] D. Moses, *et al.*, "Linguistic coordinate transformations for complex fuzzy sets," in *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International*, 1999, pp. 1340-1345.

[13] D. Moses, *et al.*, "Complex membership grades with an application to the design of adaptive filters," *Computer Science Journal of Moldova,* vol. 7, pp. 253-283, 1999.

[14] H. T. Nguyen, *et al.*, "Complex fuzzy sets: towards new foundations," in *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, 2000, pp. 1045-1048.

[15] S. Dick, "Toward complex fuzzy logic," *Fuzzy Systems, IEEE Transactions on,* vol. 13, pp. 405-414, 2005.

[16] G. Zhang, *et al.*, "Operation properties and δ-equalities of complex fuzzy sets," *International journal of approximate reasoning,* vol. 50, pp. 1227-1249, 2009.

[17] J. Man, *et al.*, "Towards inductive learning of complex fuzzy inference systems," in *Fuzzy Information Processing Society, 2007. NAFIPS'07. Annual Meeting of the North American*, 2007, pp. 415-420.

[18] C. Li and T.-W. Chiang, "Complex neuro-fuzzy self-learning approach to function approximation," in *Intelligent Information and Database Systems*, ed: Springer, 2010, pp. 289-299.
[19] H. T. Nguyen*, et al.*, "On the possibility of using complex values in fuzzy logic for representing inconsistencies," *International Journal of Intelligent Systems,* vol. 13, pp. 683-714, 1998.
[20] D. E. Tamir*, et al.*, "A new interpretation of complex membership grade," *International Journal of Intelligent Systems,* vol. 26, pp. 285-312, 2011.
[21] R. Yager, "Pythagorean membership grades in multi-criteria decision making," *Fuzzy Systems, IEEE Transactions,* vol. 22, pp. 958-965, 2014.
[22] R. R. Yager and A. M. Abbasov, "Pythagorean membership grades, complex numbers, and decision making," *International Journal of Intelligent Systems,* 2013.
[23] K. T. Atanassov, "Intuitionistic fuzzy sets," *Fuzzy Sets and Systems,* vol. 20, pp. 87-96, 1986.
[24] A. R. Salleh, "Complex intuitionistic fuzzy sets," in *AIP Conference Proceedings*, 2012, p. 464.
[25] Z. Chen*, et al.*, "ANCFIS: A neurofuzzy architecture employing complex fuzzy sets," *Fuzzy Systems, IEEE Transactions on,* vol. 19, pp. 305-322, 2011.
[26] C. Li and T.-W. Chiang, "Function approximation with complex neuro-fuzzy system using complex fuzzy sets–a new approach," *New generation computing,* vol. 29, pp. 261-276, 2011.
[27] C. Li and T.-W. Chiang, "Complex Fuzzy Computing to Time-series Prediction A Multi-Swarm PSO Learning Approach," in *Intelligent Information and Database Systems*, ed: Springer, 2011, pp. 242-251.
[28] R. Shoorangiz and M. H. Marhaban, "Complex Neuro-Fuzzy System for Function Approximation," *International Journal of Applied Electronics in Physics & Robotics,* vol. 1, pp. 5-9, 2013.
[29] C. Li, "Adaptive image restoration by a novel neuro-fuzzy approach using complex fuzzy sets," *International Journal of Intelligent Information and Database Systems,* vol. 7, pp. 479-495, 2013.
[30] A. U. M. Alkouri and A. R. Salleh, "Linguistic variables, hedges and several distances on complex fuzzy sets," *Journal of Intelligent and Fuzzy Systems: Application in Engineering and Technology,* vol. 26, pp. 2527-2535, 2014.
[31] D. E. Tamir and A. Kandel, "Axiomatic theory of complex fuzzy logic and complex fuzzy classes," *International Journal of Computers, Communications & Control VI (3),* 2011.
[32] S. Dick*, et al.*, "On Pythagorean and Complex Fuzzy Set Operations," *IEEE Transactions on fuzzy systems,* vol. in press, 2015.
[33] J. Ma*, et al.*, "A method for multiple periodic factor prediction problems using complex fuzzy sets," *Fuzzy Systems, IEEE Transactions on,* vol. 20, pp. 32-45, 2012.
[34] A. U. M. Alkouri and A. R. Salleh, "Some operations on complex Atanassov's intuitionistic fuzzy sets," in *THE 2013 UKM FST POSTGRADUATE COLLOQUIUM: Proceedings of the Universiti Kebangsaan Malaysia, Faculty of Science and Technology 2013 Postgraduate Colloquium*, 2013, pp. 987-993.

[35] G. Zhang*, et al.*, "Delta-equalities of Complex Fuzzy Relations," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 1218-1224.
[36] E. Mizraji, "Vector logics: the matrix-vector representation of logical calculus," *Fuzzy Sets and Systems,* vol. 50, pp. 179-185, 1992.
[37] V. V. Shende*, et al.*, "Synthesis of quantum-logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 25, pp. 1000-1010, 2006.
[38] C. Edwards, "The logic of Boolean matrices," *The Computer Journal,* vol. 15, pp. 247-253, 1972.
[39] T. KARAÇAY, "HARMONIC ANALYSIS IN FUZZY SYSTEMS," *International Journal of Science and Research (IJSR),* vol. 3, pp. 1300-1306.
[40] E. Mizraji, "The operators of vector logic," *Mathematical Logic Quarterly,* vol. 42, pp. 27-40, 1996.
[41] D. E. Tamir*, et al.*, "The Theory and Applications of Generalized Complex Fuzzy Propositional Logic," in *Soft Computing: State of the Art Theory and Novel Applications*, ed: Springer, 2013, pp. 177-192.
[42] D. E. Tamir*, et al.*, "Discrete complex fuzzy logic," in *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, 2012, pp. 1-6.
[43] S. D. Zenzo, "A many-valued logic for approximate reasoning," *IBM journal of research and development,* vol. 32, pp. 552-565, 1988.
[44] D. E. Tamir*, et al.*, "Complex Fuzzy Sets and Complex Fuzzy Logic an Overview of Theory and Applications," in *Fifty Years of Fuzzy Logic and its Applications*, ed: Springer, 2015, pp. 661-681.
[45] V. Sgurev, "Features of Disjunction and Conjunction in the Complex Propositional S-logic," *Comptes rendus de l'Academie bulgare des sciences,* pp. 1491-1502, 2014.
[46] L. Běhounek and P. Cintula, "Fuzzy class theory," *Fuzzy Sets and Systems,* vol. 154, pp. 34-55, 2005.
[47] L. A. Zadeh, "Probability theory and fuzzy logic," *on the Internet at: www. Ieeesmc. org/dec2002/Probability% 20Theory% 20and% 20Fuzzy% 20Logic. Pdf,* 2002.
[48] C. Li and T. Chiang, "Complex Neuro-Fuzzy ARIMA Forecasting—A New Approach Using Complex Fuzzy Sets," 2011.
[49] O. Kosheleva*, et al.*, "Why complex-valued fuzzy? Why complex values in general? A computational explanation," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 2013, pp. 1233-1236.
[50] O. Kosheleva and V. Kreinovich, "Approximate nature of traditional fuzzy methodology naturally leads to complex-valued fuzzy degrees," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, 2014, pp. 1475-1479.
[51] C. Servin*, et al.*, "From 1-D to 2-D fuzzy: A proof that interval-valued and complex-valued are the only distributive options," Departmental Technical Reports (CS)2015.
[52] h. Whalen, "Real and Imaginary Truth in Complex Fuzzy Implication," in *NAFIPS 2015*, Redmond, Washington, 2015.
[53] S. Greenfield and F. Chiclana, "Fuzzy in 3-D: Contrasting complex fuzzy sets with type-2 fuzzy sets," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 2013, pp. 1237-1242.

[54] S. Aghakhani and S. Dick, "An on-line learning algorithm for complex fuzzy logic," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1-7.
[55] O. Yazdanbaksh*, et al.*, "Predicting solar power output using complex fuzzy logic," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 2013, pp. 1243-1248.
[56] C. Li*, et al.*, "A novel self-organizing complex neuro-fuzzy approach to the problem of time-series forecasting," *Neurocomputing,* 2012.
[57] C. Li and T.-W. Chiang, "Intelligent financial time-series forecasting: A complex neuro-fuzzy approach with multi-swarm intelligence," *International Journal of Applied Mathematics and Computer Science,* vol. 22, pp. 787-800, 2012.
[58] C. Li and F. Chan, "Complex-Fuzzy Adaptive Image Restoration–An Artificial-Bee-Colony-Based Learning Approach," in *Intelligent Information and Database Systems*, ed: Springer, 2011, pp. 90-99.
[59] C. Li*, et al.*, "Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling," *Neurocomputing,* vol. 94, pp. 121-139, 2012.
[60] C. Li and F.-T. Chan, "Knowledge discovery by an intelligent approach using complex fuzzy sets," in *Intelligent Information and Database Systems*, ed: Springer, 2012, pp. 320-329.
[61] A. U. M. Alkouri and A. R. Salleh, "Complex Atanassov's Intuitionistic Fuzzy Relation," in *Abstract and Applied Analysis*, 2013.
[62] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 23, pp. 665-685, 1993.
[63] P. Thirunavukarasu*, et al.*, "Complex Neuro Fuzzy System Using Complex Fuzzy Sets and Update the Parameters by PSO-GA and RLSE Method."
[64] H. Sun*, et al.*, "FCM-based model selection algorithms for determining the number of clusters," *Pattern Recognition,* vol. 37, pp. 2027-2037, 2004.
[65] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—I," *Information sciences,* vol. 8, pp. 199-249, 1975.
[66] T. Kumar and R. K. Bajaj, "On Complex Intuitionistic Fuzzy Soft Sets with Distance Measures and Entropies," *Journal of Mathematics,* vol. 2014, 2014.
[67] P. K. Maji, "More on intuitionistic fuzzy soft sets," in *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, ed: Springer, 2009, pp. 231-240.
[68] Y. Jiang*, et al.*, "Entropy on intuitionistic fuzzy soft sets and on interval-valued fuzzy soft sets," *Information sciences,* vol. 240, pp. 95-114, 2013.
[69] S. Rengarajulu, "Parameterized Soft Complex Fuzzy Sets," *Journal of Progressive Research in Mathematics,* vol. 4, pp. 303-308, 2015.
[70] N. Çağman*, et al.*, "Fuzzy parameterized fuzzy soft set theory and its applications."
[71] J. Ma*, et al.*, "A conceptual method for modeling residential utility consumption using complex fuzzy sets," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 2013, pp. 1227-1232.
[72] J. Ma*, et al.*, "Data-Driven Forecasts of Regional Demand for Infrastructure Services," 2013.

[73] A. Deshmukh, *et al.*, "Implementation of complex fuzzy logic modules with VLSI approach," *International Journal on Computer Science and Network Security,* vol. 8, pp. 172-178, 2008.
[74] D. E. Tamir, *et al.*, "Soft Computing Based Epidemical Crisis Prediction," in *Intelligent Methods for Cyber Warfare*, ed: Springer, 2015, pp. 43-67.
[75] A. Kandel, *et al.*, "Fuzzy logic and data mining in disaster mitigation," in *Improving Disaster Resilience and Mitigation-IT Means and Tools*, ed: Springer, 2014, pp. 167-186.
[76] D. Karpenko, *et al.*, "The Cauchy problem for complex fuzzy differential equations," *Fuzzy Sets and Systems,* vol. 245, pp. 18-29, 2014.
[77] C. K. Loo, *et al.*, "A Novel Complex-Valued Fuzzy ARTMAP for Sparse Dictionary Learning," in *Neural Information Processing*, 2013, pp. 360-368.
[78] T. Kasuba, "Simplified fuzzy ARTMAP," *AI EXPERT,* vol. 8, pp. 18-18, 1993.
[79] M.-T. Vakil-Baghmisheh and N. Pavešić, "A fast simplified fuzzy ARTMAP network," *Neural processing letters,* vol. 17, pp. 273-316, 2003.
[80] H. Seki and T. Nakashima, "Complex-valued SIRMs connected fuzzy inference model," in *Granular Computing (GrC), 2014 IEEE International Conference on*, 2014, pp. 250-253.
[81] N. Yubazaki, *et al.*, "SIRMs (Single Input Rule Modules) Connected Fuzzy Inference Model," *JACIII,* vol. 1, pp. 23-30, 1997.
[82] E. Aarts and J. Korst, "Simulated annealing and Boltzmann machines," 1988.
[83] S. Chen, *et al.*, "Orthogonal least squares learning algorithm for radial basis function networks," *Neural Networks, IEEE Transactions on,* vol. 2, pp. 302-309, 1991.
[84] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing,* vol. 14, pp. 199-222, 2004.
[85] N. Sapankevych and R. Sankar, "Time-series prediction using support vector machines: a survey," *Computational Intelligence Magazine, IEEE,* vol. 4, pp. 24-38, 2009.
[86] D. Meyer, *et al.*, "Misc Functions of the Department of Statistics (e1071)," *TU Wien URL: http://cran.rproject.org/web/packages/e1071/e1071.pdf,* 2012.
[87] J. Hyndman and G. Athanasopoulos. (2013). *Forecasting: principles and practice*. Available: https://www.otexts.org/book/fpp
[88] D. Peña, *et al.*, *A course in time-series analysis* vol. 322: John Wiley & Sons, 2011.
[89] T. C. Mills, *Time-series Techniques for Economists*. Boston, MA, USA: Cambridge University Press, 1990.
[90] C. W. Granger and R. Joyeux, "An introduction to long-memory time-series models and fractional differencing," *Journal of time-series analysis,* vol. 1, pp. 15-29, 1980.
[91] R. J. Frank, *et al.*, "Time-series prediction and neural networks," *Journal of intelligent and robotic systems,* vol. 31, pp. 91-103, 2001.
[92] C. Wu and K. Chau, "Data-driven models for monthly streamflow time-series prediction," *Engineering Applications of Artificial Intelligence,* vol. 23, pp. 1350-1367, 2010.
[93] C. Wu, *et al.*, "Prediction of rainfall time-series using modular artificial neural networks coupled with data-preprocessing techniques," *Journal of hydrology,* vol. 389, pp. 146-167, 2010.
[94] F. Zhao, *et al.*, "Neuro-fuzzy based condition prediction of bearing health," *Journal of Vibration and Control,* 2009.

[95] R. Jursa and K. Rohrig, "Short-term wind power forecasting using evolutionary algorithms for the automated specification of artificial intelligence models," *International journal of forecasting,* vol. 24, pp. 694-709, 2008.
[96] H. Kantz and T. Schreiber, *Nonlinear time-series analysis*. Cambridge ; New York: Cambridge University Press, 1997.
[97] R. Hegger*, et al.*, "Practical implementation of nonlinear time-series methods: The TISEAN package," *arXiv preprint chao-dyn/9810005,* 1998.
[98] M. B. Kennel*, et al.*, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Physical Review A,* vol. 45, p. 3403, 1992.
[99] L. Cao*, et al.*, "Dynamics from multivariate time-series," *Physica D: Nonlinear Phenomena,* vol. 121, pp. 75-88, 1998.
[100] S. Boccaletti*, et al.*, "Reconstructing embedding spaces of coupled dynamical systems from multivariate data," *Physical Review E,* vol. 65, p. 035204, 2002.
[101] L.-y. Su, "Prediction of multivariate chaotic time-series with local polynomial fitting," *Computers & Mathematics with Applications,* vol. 59, pp. 737-744, 2010.
[102] R. Hegger*, et al.*, "Practical implementation of nonlinear time-series methods: The TISEAN package," *Chaos,* vol. 9, pp. 413-435, Jun 1999.
[103] N. R. E. L. (NREL). *Lowry Range Solar Station (LRSS)*. Available: http://www.nrel.gov/midc/lrss/
[104] A. Bellini*, et al.*, "Simplified model of a photovoltaic module," in *Applied Electronics, 2009. AE 2009*, 2009, pp. 47-51.
[105] M. Li*, et al.*, "Sunspot numbers forecasting using neural networks," in *Intelligent Control, 1990. Proceedings., 5th IEEE International Symposium on*, 1990, pp. 524-529.
[106] M. R. Lyu, *Handbook of Software Reliabiity Engineering*. New York, NY: McGraw-Hill, 1996.
[107] R. J. Hyndman and M. Akram, "Time-series data library," ed, 2006.
[108] T. Masters, *Neural, novel and hybrid algorithms for time-series prediction*: John Wiley & Sons, Inc., 1995.
[109] T. Jo, "VTG schemes for using back propagation for multivariate time-series prediction," *Applied Soft Computing,* vol. 13, pp. 2692-2702, 2013.
[110] M. Ghiassi and S. Nangoy, "A dynamic artificial neural network model for forecasting nonlinear processes," *Computers & Industrial Engineering,* vol. 57, pp. 287-297, 2009.
[111] NASDAQ Composite Index. (2014). *Yahoo Finance*. Available: http://finance.yahoo.com/q?s=^IXIC
[112] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research,* vol. 7, pp. 1-30, 2006.
[113] M. Hollander*, et al.*, *Nonparametric statistical methods*: John Wiley & Sons, 2013.
[114] D. Wackerly*, et al.*, *Mathematical statistics with applications*: Cengage Learning, 2007.
[115] G. Schneider*, et al.*, "NSM3: Functions and Data sets to Accompany Hollander, Wolfe, and Chicken - Nonparametric Statistical Methods, Third Edition," 1.3 ed, 2015.
[116] A. J. Koning*, et al.*, "The M3 competition: Statistical tests of the results," *International journal of forecasting,* vol. 21, pp. 397-409, 2005.
[117] Staff, "Inventory of U.S. greenhouse gas emission and sinks: 1990-2010," U.S. Environmental Protection Agency, Washington, DC2012.

[118] J. L. Sawin, *et al.*, "Renewables 2012 global status report," REN21 Secretariat, Paris, France2012.
[119] Y. Huang, *et al.*, "Comparative study of power forecasting methods for PV stations," in *2010 International Conference on Power System Technology*, 2010, pp. 24-28.
[120] V. Omubo-Pepple, *et al.*, "Effects of temperature, solar flux and relative humidity on the efficient conversion of solar energy to electricity," *European Journal of Scientific Research,* vol. 35, pp. 173-180, 2009.
[121] P. Bacher, *et al.*, "Online short-term solar power forecasting," *Solar Energy,* vol. 83, pp. 1772-1783, 2009.
[122] B. H. Chowdhury and S. Rahman, "Forecasting sub-hourly solar irradiance for prediction of photovoltaic output," in *19th IEEE Photovoltaic Specialists Conference*, 1987, pp. 171-176.
[123] A. Sfetsos and A. Coonick, "Univariate and multivariate forecasting of hourly solar radiation with artificial intelligence techniques," *Solar Energy,* vol. 68, pp. 169-178, 2000.
[124] L. Hontoria, *et al.*, "Generation of hourly irradiation synthetic series using the neural network multilayer perceptron," *Solar Energy,* vol. 72, pp. 441-446, 2002.
[125] S. Cao and J. Cao, "Forecast of solar irradiance using recurrent neural networks combined with wavelet analysis," *Applied Thermal Engineering,* vol. 25, pp. 161-172, 2005.
[126] A. Mellit, *et al.*, "An adaptive wavelet-network model for forecasting daily total solar-radiation," *Applied Energy,* vol. 83, pp. 705-722, 2006.
[127] F. O. Hocaoğlu, *et al.*, "Hourly solar radiation forecasting using optimal coefficient 2-D linear filters and feed-forward neural networks," *Solar Energy,* vol. 82, pp. 714-726, 2008.
[128] A. Mellit, *et al.*, "Methodology for predicting sequences of mean monthly clearness index and daily solar radiation data in remote areas: Application for sizing a stand-alone PV system," *Renewable Energy,* vol. 33, pp. 1570-1590, 2008.
[129] J. Cao and X. Lin, "Study of hourly and daily solar irradiation forecast using diagonal recurrent wavelet neural networks," *Energy Conversion and Management,* vol. 49, pp. 1396-1406, 2008.
[130] G. Reikard, "Predicting solar radiation at high resolutions: A comparison of time-series forecasts," *Solar Energy,* vol. 83, pp. 342-349, 2009.
[131] A. Moreno-Munoz, *et al.*, "Short term forecasting of solar radiation," in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, 2008, pp. 1537-1541.
[132] M. Kudo, *et al.*, "Forecasting electric power generation in a photovoltaic power system for an energy network," *Electrical Engineering in Japan,* vol. 167, pp. 16-23, 2009.
[133] E. D'Andrea and B. Lazzerini, "Fuzzy forecasting of energy production in solar photovoltaic installations," in *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, 2012, pp. 1-8.
[134] Y. Su, *et al.*, "Real-time prediction models for output power and efficiency of grid-connected solar photovoltaic systems," *Applied Energy,* vol. 93, pp. 319-326, 2012.
[135] A. Bracale, *et al.*, "A Bayesian method for short-term probabilistic forecasting of photovoltaic generation in smart grid operation and control," *Energies,* vol. 6, pp. 733-747, 2013.
[136] J. G. Silva Fonseca, *et al.*, "Use of support vector regression and numerically predicted cloudiness to forecast power output of a photovoltaic power plant in Kitakyushu, Japan," *Progress in Photovoltaics: Research and Applications,* vol. 20, pp. 874-882, 2012.

[137] A. Yona, *et al.*, "Application of recurrent neural network to short-term-ahead generating power forecasting for photovoltaic system," in *Power Engineering Society General Meeting, 2007. IEEE*, 2007, pp. 1-6.
[138] A. Mellit, *et al.*, "Artificial neural network-based model for estimating the produced power of a photovoltaic module," *Renewable Energy,* vol. 60, pp. 71-78, 2013.
[139] L. Prokop, *et al.*, "Photovoltaic power plant output estimation by neural networks and fuzzy inference," in *Intelligent Data Engineering and Automated Learning-IDEAL 2012*, ed: Springer, 2012, pp. 810-817.
[140] F. Almonacid, *et al.*, "Estimation of the energy of a PV generator using artificial neural network," *Renewable Energy,* vol. 34, pp. 2743-2750, 2009.
[141] C. Chen, *et al.*, "Online 24-h solar power forecasting based on weather type classification using artificial neural network," *Solar Energy,* vol. 85, pp. 2856-2870, 2011.
[142] N. Al-Messabi, *et al.*, "Forecasting of photovoltaic power yield using dynamic neural networks," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1-5.
[143] H. T. Pedro and C. F. Coimbra, "Assessment of forecasting techniques for solar power production with no exogenous inputs," *Solar Energy,* vol. 86, pp. 2017-2028, 2012.
[144] M. Paulescu, *et al.*, "Forecasting the Power Output of PV Systems," in *Weather Modeling and Forecasting of PV Systems Operation*, ed: Springer, 2013, pp. 325-345.
[145] E. Dimitriadou, *et al.*, "Misc functions of the Department of Statistics (e1071), TU Wien," *R package,* pp. 1.5-24, 2008.
[146] R. D. C. Team, "R: A language and environment for statistical computing," ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2013. url: http://www. R-project. org2005.
[147] M. West, "Bayesian Forecasting," 1996.
[148] A. M. De Livera, *et al.*, "Forecasting time-series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association,* vol. 106, pp. 1513-1527, 2011.
[149] J. Hyndman. (2010). *Forecasting with long seasonal period*. Available: http://robjhyndman.com/hyndsight/longseasonality/
[150] R. J. Hyndman and Y. Khandakar, "Automatic time-series for forecasting: the forecast package for R," 2007.
[151] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys (CSUR),* vol. 45, p. 12, 2012.
[152] P. Cortez, *et al.*, "Multi-scale Internet traffic forecasting using neural networks and time-series methods," *Expert Systems,* vol. 29, pp. 143-155, 2012.
[153] S. Chabaa, *et al.*, "Identification and prediction of internet traffic using artificial neural networks," *Journal of Intelligent Learning Systems and Applications,* vol. 2, p. 147, 2010.
[154] V. Nourani, *et al.*, "A combined neural-wavelet model for prediction of Ligvanchai watershed precipitation," *Engineering Applications of Artificial Intelligence,* vol. 22, pp. 466-472, 2009.
[155] W.-C. Wang, *et al.*, "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time-series," *Journal of hydrology,* vol. 374, pp. 294-306, 2009.

[156] R. Kuo and K. Xue, "Fuzzy neural networks with application to sales forecasting," *Fuzzy Sets and Systems,* vol. 108, pp. 123-143, 1999.
[157] Z.-L. Sun*, et al.*, "Sales forecasting using extreme learning machine with applications in fashion retailing," *Decision Support Systems,* vol. 46, pp. 411-419, 2008.
[158] M. A. Boyacioglu and D. Avci, "An adaptive network-based fuzzy inference system (ANFIS) for the prediction of stock market return: the case of the Istanbul stock exchange," *Expert Systems with Applications,* vol. 37, pp. 7908-7912, 2010.
[159] J. G. De Gooijer and R. J. Hyndman, "25 years of time-series forecasting," *International journal of forecasting,* vol. 22, pp. 443-473, 2006.
[160] G. Zhang*, et al.*, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting,* vol. 14, pp. 35-62, 1998.
[161] G. P. Zhang, "Time-series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing,* vol. 50, pp. 159-175, 2003.
[162] F. E. Tay and L. Cao, "Application of support vector machines in financial time-series forecasting," *Omega,* vol. 29, pp. 309-317, 2001.
[163] C. Chatfield, *Time-series forecasting*: CRC Press, 2000.
[164] A. A. Santos*, et al.*, "Comparing univariate and multivariate models to forecast portfolio value-at-risk," *Journal of Financial Econometrics,* vol. 11, pp. 400-441, 2013.
[165] M. Han and Y. Wang, "Analysis and modeling of multivariate chaotic time-series based on neural network," *Expert Systems with Applications,* vol. 36, pp. 1280-1290, 2009.
[166] S. S. Jones*, et al.*, "A multivariate time-series approach to modeling and forecasting demand in the emergency department," *Journal of biomedical informatics,* vol. 42, pp. 123-139, 2009.
[167] J. Ma and L. Liu, "Multivariate nonlinear analysis and prediction of Shanghai stock market," *Discrete Dynamics in Nature and Society,* vol. 2008, 2008.
[168] K. Chakraborty*, et al.*, "Forecasting the behavior of multivariate time-series using neural networks," *Neural networks,* vol. 5, pp. 961-970, 1992.
[169] G. C. Reinsel, *Elements of multivariate time-series analysis*: Springer Science & Business Media, 2003.
[170] S.-M. Chen and K. Tanuwijaya, "Multivariate fuzzy forecasting based on fuzzy time-series and automatic clustering techniques," *Expert Systems with Applications,* vol. 38, pp. 10594-10605, 2011.
[171] O. Yazdanbakhsh and S. Dick, "Time-series forecasting via complex fuzzy logic," in *Frontiers of Higher Order Fuzzy Sets*, ed: Springer, 2015, pp. 147-165.
[172] J. Y. Campbell*, et al.*, *The Econometrics of Financial Markets*. Princeton, NJ, USA: Princeton University Press, 1996.
[173] Staff, "Worldwide Semiannual Big Data and Analytics Spending Guide," IDC Research, Inc., Framingham, MA, USA2016.
[174] Staff, "Business Intelligence and Analytics Software Market by Segment (BI platforms, CPM Suite, Advanced and Predictive Analytics, Content Analytics, Analytics Application), by Services, by Deployment Mode, by Org. Size, by Verticals, by Regions - Global Forecast to 2020," MarketsandMarkets, Pune, India2015.
[175] D. Laney and A. Jain, "100 Data and Analytics Predictions Through 2020," Gartner, Inc., Stamford, CT, USA2016.

[176] O. Yazdanbakhsh and S. Dick, "ANCFIS-ELM: A Machine Learning Algorithm based on Complex Fuzzy Sets," presented at the World Congress on Computational Intelligence (WCCI), Vancouver, Canada, 2016.
[177] O. Yazdanbakhsh and S. Dick, "A Systematic Review of Complex Fuzzy Sets and Logic," *Fuzzy Sets & Systems,* Submitted.
[178] J. Ma*, et al.*, "A method for multiple periodic factor prediction problems using complex fuzzy sets," *IEEE T. Fuzzy Syst.,* vol. 20, pp. 32-45, 2012.
[179] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural computation,* vol. 9, pp. 1545-1588, 1997.
[180] F. Moosmann*, et al.*, "Randomized clustering forests for building fast and discriminative visual vocabularies," 2006.
[181] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2007, pp. 1177-1184.
[182] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of computer and system sciences,* vol. 69, pp. 593-616, 2004.
[183] A. N. Gorban*, et al.*, "Approximation with random bases: Pro et Contra," *Information Sciences,* 2015.
[184] L. K. Jones, "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training," *The annals of Statistics,* pp. 608-613, 1992.
[185] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *Information Theory, IEEE Transactions on,* vol. 39, pp. 930-945, 1993.
[186] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *Neural Networks, IEEE Transactions on,* vol. 6, pp. 1320-1329, 1995.
[187] A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, 2008, pp. 555-561.
[188] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in neural information processing systems*, 2009, pp. 1313-1320.
[189] L. Zhang and P. Suganthan, "A Survey of Randomized Algorithms for Training Neural Networks," *Information Sciences,* 2016.
[190] W. F. Schmidt*, et al.*, "Feedforward neural networks with random weights," in *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, 1992, pp. 1-4.
[191] G.-B. Huang*, et al.*, "Extreme learning machine: theory and applications," *Neurocomputing,* vol. 70, pp. 489-501, 2006.
[192] G.-B. Huang*, et al.*, "On-Line Sequential Extreme Learning Machine," *Computational Intelligence,* vol. 2005, pp. 232-237, 2005.
[193] H.-J. Rong*, et al.*, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 39, pp. 1067-1072, 2009.
[194] G.-B. Huang*, et al.*, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics,* vol. 2, pp. 107-122, 2011.

[195] W. Zhang and H. Ji, "Fuzzy extreme learning machine for classification," *Electronics Letters,* vol. 49, pp. 448-450, 2013.
[196] D. Albers*, et al.*, "Dynamical behavior of artificial neural networks with random weights," *Intelligent Engineering Systems Through Artificial Neural Networks,* vol. 6, pp. 17-22, 1996.
[197] A. Elisseeff and H. Paugam-Moisy, "JNN, a randomized algorithm for training multilayer networks in polynomial time," *Neurocomputing,* vol. 29, pp. 3-24, 1999.
[198] Y.-H. Pao*, et al.*, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing,* vol. 6, pp. 163-180, 1994.
[199] Y.-H. Pao and S. M. Phillips, "The functional link net and learning optimal control," *Neurocomputing,* vol. 9, pp. 149-164, 1995.
[200] W. Maass*, et al.*, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation,* vol. 14, pp. 2531-2560, 2002.
[201] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation,* vol. 1, pp. 502-510, 1989.
[202] P. Vincent*, et al.*, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research,* vol. 11, pp. 3371-3408, 2010.
[203] T. P. Lillicrap*, et al.*, "Random feedback weights support learning in deep neural networks," *arXiv preprint arXiv:1411.0247,* 2014.
[204] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *Advances in Neural Information Processing Systems*, 2011, pp. 2528-2536.
[205] K. Jarrett*, et al.*, "What is the best multi-stage architecture for object recognition?," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 2146-2153.
[206] A. Saxe*, et al.*, "On random weights and unsupervised feature learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 1089-1096.
[207] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557,* 2013.
[208] S. Timotheou, "The random neural network: a survey," *The computer journal,* vol. 53, pp. 251-267, 2010.
[209] H. Bakırcıoğlu and T. Koçak, "Survey of random neural network applications," *European journal of operational research,* vol. 126, pp. 319-330, 2000.
[210] L. Zhang and P. N. Suganathan, "A comprehensive evaluation of random vector functional link networks," *Information Sciences,* vol. 367-368, pp. 1094-1105, 2016.
[211] Scardapane*, et al.*, "A semi-supervised random vector functional-link network based on the transductive framework," *Information Sciences,* vol. 364-365, pp. 156-166, 2016.
[212] Y. Ren*, et al.*, "Random vector functional link network for short-term electricity load demand forecasting," *Information Sciences,* vol. 367-368, pp. 1078-1093, 2016.
[213] H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," presented at the NIPS, 2002.
[214] G.-B. Huang*, et al.*, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing,* vol. 71, pp. 576-583, 2008.
[215] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing,* vol. 71, pp. 3460-3468, 2008.

[216] H.-J. Rong, *et al.*, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing,* vol. 72, pp. 359-366, 2008.
[217] S. S. Haykin, *Neural networks and learning machines* vol. 3: Pearson Education Upper Saddle River, 2009.
[218] O. Yazdanbakhsh and S. Dick, "Induction of Complex Fuzzy Infernetial System via Randomized Learning," 2016.
[219] J. A. B. Tomé and J. P. Carvalho, "One step ahead prediction using Fuzzy Boolean Neural Networks," in *EUSFLAT Conf.*, 2005, pp. 500-505.
[220] D. Graves and W. Pedrycz, "Fuzzy prediction architecture using recurrent neural networks," *Neurocomputing,* vol. 72, pp. 1668-1678, 2009.
[221] M. M. Gaber, *et al.*, "Data stream mining," in *Data Mining and Knowledge Discovery Handbook*, ed: Springer, 2009, pp. 759-787.
[222] S. Muthukrishnan, *Data streams: Algorithms and applications*: Now Publishers Inc, 2005.
[223] G. Krempl, *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter,* vol. 16, pp. 1-10, 2014.
[224] A. Jha, *et al.*, "A Review on the Study and Analysis of Big Data using Data Mining Techniques," *International Journal of Latest Trends inEngineering and Technology (IJLTET),* vol. 6, 2016.
[225] J. Gama, *et al.*, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR),* vol. 46, p. 44, 2014.
[226] S. Dick, *et al.*, "An empirical investigation of Web session workloads: Can self-similarity be explained by deterministic chaos?," *Information Processing & Management,* vol. 50, pp. 41-53, 2014.
[227] R. C. Garcia, *et al.*, "A GARCH forecasting model to predict day-ahead electricity prices," *IEEE transactions on power systems,* vol. 20, pp. 867-874, 2005.
[228] G. Zhiqiang, *et al.*, "Financial time-series forecasting using LPP and SVM optimized by PSO," *Soft Computing,* vol. 17, pp. 805-818, 2013.
[229] L. Wang, *et al.*, "Back propagation neural network with adaptive differential evolution algorithm for time-series forecasting," *Expert Systems with Applications,* vol. 42, pp. 855-863, 2015.
[230] T. Kuremoto, *et al.*, "Time-series forecasting using a deep belief network with restricted Boltzmann machines," *Neurocomputing,* vol. 137, pp. 47-56, 2014.
[231] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 23, pp. 665-685, 1993.
[232] D. Brzeziński, "Mining data streams with concept drift," Master's thesis, Poznan University of Technology, 2010.
[233] B. Babcock, *et al.*, "Load shedding techniques for data stream systems," in *Proc. Workshop on Management and Processing of Data Streams*, 2003.
[234] Y. Chi, *et al.*, "Loadstar: load shedding in data stream mining," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 1302-1305.
[235] X. Wu, *et al.*, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, pp. 97-107, 2014.
[236] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future," *ACM SIGKDD Explorations Newsletter,* vol. 14, pp. 1-5, 2013.

[237] H.-L. Nguyen, *et al.*, "A survey on data stream clustering and classification," *Knowledge and information systems,* vol. 45, pp. 535-569, 2015.
[238] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 71-80.
[239] H. Wang, *et al.*, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226-235.
[240] A. Bifet, "Adaptive stream mining: Pattern learning and mining from evolving data streams," in *Proceedings of the 2010 conference on adaptive stream mining: Pattern learning and mining from evolving data streams*, 2010, pp. 1-212.
[241] F. Cao, *et al.*, "Density-Based Clustering over an Evolving Data Stream with Noise," in *SDM*, 2006, pp. 328-339.
[242] C. C. Aggarwal, *et al.*, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*, 2003, pp. 81-92.
[243] S. Kaisler, *et al.*, "Big data: issues and challenges moving forward," in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, 2013, pp. 995-1004.
[244] T. Heinze, *et al.*, "Auto-scaling techniques for elastic data stream processing," in *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, 2014, pp. 296-302.
[245] D. Puthal, *et al.*, "A Dynamic Key Length Based Approach for Real-Time Security Verification of Big Sensing Data Stream," in *International Conference on Web Information Systems Engineering*, 2015, pp. 93-108.
[246] J. M. Tien, "Big data: Unleashing information," *Journal of Systems Science and Systems Engineering,* vol. 22, pp. 127-151, 2013.
[247] M. Spiliopoulou, *et al.*, "MONIC and Followups on Modeling and Monitoring Cluster Transitions," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 622-626.
[248] M. F. Uddin and N. Gupta, "Seven V's of Big Data understanding Big Data to extract value," in *American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the*, 2014, pp. 1-5.
[249] D. T. J. Huang, *et al.*, "Detecting volatility shift in data streams," in *2014 IEEE International Conference on Data Mining*, 2014, pp. 863-868.
[250] S. Chandrasekaran, *et al.*, "TelegraphCQ: continuous dataflow processing," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 668-668.
[251] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784,* 2010.
[252] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin,* vol. 106, 2004.
[253] X. Zhang, *et al.*, "Data stream clustering with affinity propagation," *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, pp. 1644-1656, 2014.
[254] E. Lughofer and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing split-and-merge concepts–towards a plug-and-play approach," *Information sciences,* vol. 304, pp. 54-79, 2015.

[255] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 133-142.
[256] L. Wan*, et al.*, "Density-based clustering of data streams at multiple resolutions," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 3, p. 14, 2009.
[257] L. O'callaghan*, et al.*, "Streaming-data algorithms for high-quality clustering," in *icde*, 2002, p. 0685.
[258] D. Leite*, et al.*, "Evolving granular neural network for semi-supervised data stream classification," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 2010, pp. 1-8.
[259] I. W. Tsang*, et al.*, "Simpler core vector machines with enclosing balls," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 911-918.
[260] N. C. Oza, "Online bagging and boosting," in *Systems, man and cybernetics, 2005 IEEE international conference on*, 2005, pp. 2340-2345.
[261] M. Kontaki*, et al.*, "Adaptive similarity search in streaming time-series with sliding windows," *Data & Knowledge Engineering,* vol. 63, pp. 478-502, 2007.
[262] G. Cormode*, et al.*, "Conquering the divide: Continuous clustering of distributed data streams," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, 2007, pp. 1036-1045.
[263] U. Y. Ogras and H. Ferhatosmanoglu, "Online summarization of dynamic time-series data," *The VLDB Journal,* vol. 15, pp. 84-98, 2006.
[264] N. Wagner*, et al.*, "Time-series forecasting for dynamic environments: the DyFor genetic program model," *Evolutionary Computation, IEEE Transactions on,* vol. 11, pp. 433-452, 2007.
[265] M. M. Gaber*, et al.*, "Mining data streams: a review," *ACM Sigmod Record,* vol. 34, pp. 18-26, 2005.
[266] J. Gama and M. M. Gaber, *Learning from data streams*: Springer, 2007.
[267] M. M. Gaber*, et al.*, "A survey of classification methods in data streams," in *Data Streams*, ed: Springer, 2007, pp. 39-59.
[268] S. Guha*, et al.*, "Clustering data streams: Theory and practice," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 15, pp. 515-528, 2003.
[269] S. V. Vaseghi, *Advanced digital signal processing and noise reduction*: John Wiley & Sons, 2008.
[270] R. B. Stull, *An introduction to boundary layer meteorology* vol. 13: Springer Science & Business Media, 2012.
[271] A. Galka, *Topics in nonlinear time-series analysis: with implications for EEG analysis* vol. 14: World Scientific, 2000.
[272] R. Hegger*, et al.*, "Practical implementation of nonlinear time-series methods: The TISEAN package," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 9, pp. 413-435, 1999.
[273] E. W. Weisstein, "Fast fourier transform," 2015.
[274] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation,* vol. 19, pp. 297-301, 1965.
[275] C. Faloutsos*, et al.*, *Fast subsequence matching in time-series databases* vol. 23: ACM, 1994.

[276] R. Agrawal*, et al.*, "Efficient similarity search in sequence databases," in *International Conference on Foundations of Data Organization and Algorithms*, 1993, pp. 69-84.
[277] X. Lian and L. Chen, "Efficient similarity search over future stream time-series," *IEEE Transactions on Knowledge and Data Engineering,* vol. 20, pp. 40-54, 2008.
[278] Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in *Proceedings of the 28th international conference on Very Large Data Bases*, 2002, pp. 358-369.
[279] C. Berberidis*, et al.*, "On the discovery of weak periodicities in large time-series," in *European Conference on Principles of Data Mining and Knowledge Discovery*, 2002, pp. 51-61.
[280] M. Vlachos*, et al.*, "Identifying similarities, periodicities and bursts for online search queries," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 131-142.
[281] J. Beringer and E. Hüllermeier, "Online clustering of parallel data streams," *Data & Knowledge Engineering,* vol. 58, pp. 180-204, 2006.
[282] S. Sripirakas and R. Pears, "Mining recurrent concepts in data streams using the discrete fourier transform," in *International Conference on Data Warehousing and Knowledge Discovery*, 2014, pp. 439-451.
[283] E. Masciari, "SMART: Stream monitoring enterprise activities by RFID Tags," *Information sciences,* vol. 195, pp. 25-44, 2012.
[284] R. M. Vallim and R. F. De Mello, "Proposal of a new stability concept to detect changes in unsupervised data streams," *Expert Systems with Applications,* vol. 41, pp. 7350-7360, 2014.
[285] B. D. Storey, "Computing Fourier series and power spectrum with Matlab."
[286] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, 1998, pp. 1381-1384.
[287] M. Muja and D. Lowe, "Flann-fast library for approximate nearest neighbors user manual."