Forecasting of Multivariate Time Series via Complex Fuzzy Logic

Omolbanin Yazdanbakhsh, *Member, IEEE*, and Scott Dick, *Member, IEEE*

***Abstract*—Multivariate time series consist of sequential vector-valued observations of some phenomenon over time. Time series forecasting (for both univariate and 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 datasets. There has as yet been no investigation of more general multivariate datasets. We report on the extension of the adaptive neuro-complex-fuzzy inferential system 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 performance on four multivariate 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 datasets, and at least as accurate as kernel-based prediction algorithms.**

***Index Terms*—Complex fuzzy logic, complex fuzzy sets (CFSs), neuro-fuzzy systems, time series forecasting.**

I. Introduction

TIME series are observations recorded sequentially over time, such as Internet traffic, environmental sensor data, online transaction sequences, and stock quotations from financial markets. They are an important subcategory 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) [23]. 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 [11], [17]. Precipitation forecasting is effective in water resource and drought management [59], [83]. Sales forecasts can be used to optimize inventory management and thus reduce costs [39], [72]. Stock market price forecasts can guide trading strategies covering tens of trillions of dollars' worth of assets [6].

Manuscript received May 19, 2016; revised September 14, 2016; accepted November 4, 2016. Date of publication February 9, 2017; date of current version July 17, 2017. This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN 262151. This paper was recommended by Associate Editor F. Chiclana.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: yazdanba@ualberta.ca; dick@ece.ualberta.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2630668

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 (see [13], [18], [79], [91], [92]). 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 [26], [33], [52], [64], [67]. 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 (see [10], [12], [14], [62], [70]).

Machine learning algorithms based on complex fuzzy sets and logic (CFSs&L) have shown promising results in time series forecasting. Complex fuzzy sets (CFSs), as introduced by Ramot *et al.* [60], [61] 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 adaptive neuro-complex-fuzzy inferential system (ANCFIS) [16] was the first machine learning algorithm based on CFS whose main application is univariate time series prediction [16], [55], [87], [88]. 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 [44], [47]–[49] (as well as other machine-learning problems, e.g., adaptive noise cancellation). Currently, however, no system based on CFS&L exists for general multivariate time series prediction.

In this paper, we extend the ANCFIS architecture for general multivariate time series forecasting. We explore three possible implementations: 1) a set of single-input-single-output (SISO) networks; 2) a set of multiple-input-single-output (MISO) networks; and 3) 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 [radial basis function network (RBFN) and support vector regression (SVR)], 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 datasets. Finally, our MIMO ANCFIS design is compared against the only current complex fuzzy algorithm for bivariate forecasting on a fourth dataset.

Our first contribution in this paper is to expand the known design space for machine learning 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.

The remainder of this paper is organized as follows. In Section II, we provide background on CFSs&L, the ANCFIS system and delay embedding techniques. Our methodology is presented in Section III, and our experimental results are presented in Section IV. We close with a summary and discussion of future work in Section V.

II. Literature Review

We review the development of CFSs&L, their application to time series forecasting, the univariate ANCFIS architecture, and delay embeddings of a time series based on Takens' theorem (also known as the method of lags).

A. Complex Fuzzy Sets

Ramot *et al.* [61] proposed CFSs having membership functions as

$$\mu_s(x) = r_s(x).e^{(jw_s(x))} \tag{1}$$

where r_s and w_s are the magnitude and phase of the complex membership grade, respectively, with the magnitude constrained to $r_s \leq 1$, and $j = \sqrt{-1}$. Complex fuzzy intersections and unions are defined as t-norms and t-conorms over the magnitude, respectively; complex fuzzy complements are also the type-1 complements taken only over the magnitude. The first complex fuzzy logic was proposed by Ramot *et al.* [60] who suggested algebraic product as a complex fuzzy implication. He also proposed "vector aggregation" for combining rules. This is a weighted complex sum that permits constructive and destructive interference between rules, defined as

$$v : \{a|a \in \mathbf{C}, |a| \leq 1\}^n \rightarrow \{b|b| \in \mathbf{C}, |b| \leq 1\} \tag{2}$$

$$\mu_A(x) = v\big(\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)\big) = \sum_{i=1}^{n} w_i \mu_{A_i}(x) \tag{3}$$

where $w_i \in \{a|a\ a \in \mathbf{C}, |a| \leq 1\}$ for all i, and $\sum_{i=1}^{n} |w_i| = 1$. Dick [21] interpreted Ramot's definition of phase as a relative quantity in [60] and [61] as rotational invariance; he proved that the algebraic product, among others, cannot be used as the conjunction operation in complex fuzzy logics with rotational invariance. He then proposed studying the amplitude and phase of CFSs simultaneously, and showed that in these circumstances algebraic product can be a complex fuzzy conjunction. Various complex fuzzy operations and relations are proposed in [54], [60], [61], [89], and [90].

Tamir *et al.* [74] proposed CFSs with memberships drawn from the set $[0, 1]\times[0, j]$. These sets are called "pure" CFSs and defined as

$$\mu(V, z) = \mu_r(V) + j\mu_i(z)\mu_r, \mu_i \in [0, 1] \tag{4}$$

where $\mu_r(V)$ and $\mu_i(z)$ are the real and imaginary part of the pure complex fuzzy membership grade, $\mu(V, z)$. He showed that pure CFSs can represent pure fuzzy classes of order 1; Γ is a pure fuzzy class of order 1 defined over a universe of discourse (T) iff $\Gamma = \{V|V \in F(T)\}$, with $F(T)$ the set of all possible fuzzy subsets of T [74]. He also defined union, intersection and complement operations for pure CFSs. *Pure complex fuzzy logics* were studied in [75], [76], and [78].

Alkouri and Salleh [63] proposed complex Atanassov's intuitionistic fuzzy sets (CAIFSs), which are an extension of intuitionistic fuzzy sets [4] (sets having both a membership function, $\mu_A(x) : U \rightarrow [0, 1]$, and a nonmembership function, $\gamma_A(x) : U \rightarrow [0, 1]$, where $0 \leq \mu_A(x)+\gamma_A(x) \leq 1$). In CAIFS, both $\mu_A(x)$ and $\gamma_A(x)$ are complex fuzzy membership functions as in (1). Complex fuzzy operations were also extended for CAIFS based on [61] and [90]. Yager *et al.* [84], [85] defined Pythagorean membership grades as $\mu = re^{\theta}$, where $r \in [0, 1]$ and $\theta \in [0, (\pi/2)]$. For any object $x \in U$ a Pythagorean fuzzy set is given by $A = \{\langle x, A_Y, A_N\rangle\ |x \in U\}$, where $A_Y : U \rightarrow [0, 1]$ and $A_N : U \rightarrow [0, 1]$ are membership and nonmembership grades, respectively, and defined as

$$A_Y = r(x)\mathrm{Cos}(\theta(x)) \tag{5}$$

$$A_N = r(x)\mathrm{Sin}(\theta(x)) \tag{6}$$

where $r(x)$ and $\theta(x)$ are, respectively, the magnitude and phase of a complex number (defined as functions of the object x), and $(A_Y)^2 + (A_N)^2 \leq 1$. Plainly, a Pythagorean fuzzy set can be considered a special case of a CFS, and also a generalization of intuitionistic fuzzy sets. A conjunction operator and an aggregation operator based on the geometric mean were also proposed for Pythagorean memberships [85]. Dick *et al.* [22] studied the relationships between Pythagorean and CFSs, ultimately proposing two new complex fuzzy logics based on this analysis. They suggested that one of these logics could offer a unified representation of negation and antonym on fuzzy logic.

a) ANCFIS: The first machine learning realization of the complex fuzzy logics introduced in [21] and [60] was proposed in [16]. ANCFIS (see Fig. 1) is a layered feed-forward neural network, based on the well-known adaptive neuro fuzzy inference system (ANFIS) [31], although there are fundamental differences between them. The first difference is in the membership functions used by these two systems; ANFIS applies traditional bell-shaped, triangular-shaped, and trapezoidal-shaped membership functions [31], but ANCFIS employs sinusoidal membership functions [16]:

$$r(\theta) = \mathrm{dsin}(a(\theta = x) + b) + c \tag{7}$$

where $r(\theta)$ is amplitude, θ is phase, and $x \in X$ is an object belonging to the universal set X. The membership function parameters $\{a, b, c, d\}$ modify the sine wave; a changes the frequency of the sine wave, b creates a phase shift, c shifts the wave vertically and d changes its amplitude. However, there are two conditions that must be satisfied in order to keep the complex fuzzy membership degree within the unit disc of the complex plane

$$0 \leq d + c \leq 1, \qquad 1 \geq c \geq d \geq 0. \tag{8}$$

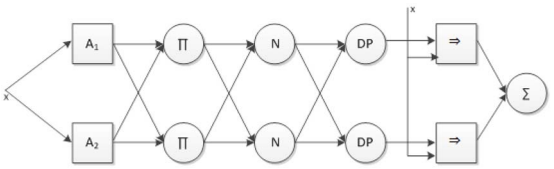

Fig. 1. ANCFIS architecture for univariate time series problems with two membership functions.

The second difference is in the way that input is fed to the systems; input vectors having lagged data points (historical data points) go to both of the systems; however, in ANFIS, the lagged data points in the input vector are presented to the system as separate inputs belonging to orthogonal dimensions of a feature space. On the other hand, in ANCFIS, the entire input vector is fed to the system as one input. This retains the relationship between the lagged data points, while reducing the number of data points and thus the complexity of the system when compared with simply windowing the time series [87]. The input vectors are fuzzified by convolving them with the membership functions of (7). The convolution operation used here can be considered as a similarity measure between the input vector and the membership function. Network signals on the bulk of the connections in ANCFIS are complex-valued.

The third difference is the addition of a new layer to ANCFIS in order to implement rule inference in complex fuzzy logic. Rather than using the vector aggregation of (2) and (3) as proposed by Ramot *et al.* [60], we implement rule interference via the dot product between the current rule's firing strength and the complex sum of all firing strengths. This allows constructive and destructive interference based on the phase of the firing strengths of the rules.

The fourth difference is in the learning algorithms. Both ANFIS and ANCFIS employ a hybrid learning scheme, in which parameters are updated in both the forward and backward passes. In the forward pass for both, consequent parameters (layer 4 in ANFIS, layer 5 in ANCFIS) are updated by Kalman filters. Antecedent parameters (layer 1 in both networks) are updated in the backward pass. ANFIS uses gradient descent for this purpose, while ANCFIS uses a hybrid of gradient descent and derivative-free optimization.

The ANCFIS network structure is composed of six layers as follows [16].

Layer 1: In this layer, complex fuzzy grades are assigned to input vectors by convolving them with membership functions of the form of (7). First, for each input vector, a given membership function is sampled over one period where the numbers of samples is equal to the length of the input vector

$$r_k(\theta_k) = dsin(a\theta_k + b) + c,\ \theta_k = \frac{2\pi}{n}k \quad k = 1, 2, \ldots, n \tag{9}$$

where n is the length of the input vector. The resulting polar coordinates for each sample are treated as a complex number, and converted to a Cartesian representation via the well-known equations $x = r \times \cos\theta$ and $y = r \times \sin\theta$. Then, the complex-valued convolution between the sampled membership function and the input vector is obtained by

$$\text{conv} = \sum_{k=1}^{2n-1} \sum_{j=\max(1,k+1-n)}^{\min(k,n)} f(j)g(k+1-j) \tag{10}$$

where $f(.)$ is a data point in the input vector, and $g(.)$ is the sampled membership function. The convolution sum is then limited to a unit disk by applying the Elliot function, yielding the transfer function

$$O_{1,ji} = \frac{\text{conv}}{1 + |\text{conv}|}\ i = 1, 2, \ldots, m, j = 1, 2, \ldots, n \tag{11}$$

where m is the number of membership functions, and $O_{1,ji}$ is the output of the ith node in layer 1 for the jth input vector.

Layer 2: The firing strength of each complex fuzzy rule is obtained by taking the conjunction of outputs from selected layer 1 nodes representing the antecedent clauses in a rule (i.e., layers 1 and 2 are not fully connected). In [16], complex fuzzy conjunctions are implemented by the algebraic product, yielding the transfer function

$$O_{2,i} = \prod_{\substack{j=1 \\ k_j \in K_j}}^{n} O_{1,jk_j}\ i = 1, 2, \ldots, m^n \tag{12}$$

where $O_{2,i}$ is the firing strength of the ith rule, k_j is the membership grade of the jth input vector (selected from the set of membership grades for the jth input vector, K_j, which has a cardinality of m), and n is the number of input vectors.

Layer 3: In this layer, the firing strength of each rule is normalized by

$$O_{3,i} = \widehat{w}_i = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} |O_{2,j}|}, i = 1, 2, \ldots, |O_2| \tag{13}$$

where $|O_2|$ is the number of rules. This normalization only affects the amplitude of a complex value; phase is unchanged.

Layer 4: In this layer, the operation of rule interference is implemented via the dot product

$$O_{4,i} = w_i^{\text{DP}} = \widehat{w}_i \cdot \sum_{j=1}^{|O_3|} \widehat{w}_j \tag{14}$$

where $|O_3|$ is the number of nodes in layer 3 and $\sum_{i=1}^{|O_3|} \widehat{w}_j$ is the complex sum of all layer 3 outputs.

Layer 5: In this layer, a linear consequent function is learned for each rule

$$O_{5,i} = w_i^s = w_i^{\text{DP}} \left[\sum_{k=1}^{j} \sum_{l=1}^{n} p_{i,kl} x_{kl} + r_i \right] \tag{15}$$

where w_i^{DP} is the output of layer 4, j is the numbers of input vectors, n is the corresponding length of each input vector, x_{kl} is the lth data point of the kth input vector, and $p_{i,kl}$, r_i are the parameters of the linear function, obtained in the forward pass by a Kalman filter.

Layer 6: All the incoming signals are summed here to obtain final outputs of ANCFIS

$$O_{6,j} = \sum_{i=1+(j-1)*N}^{j*N} w_i^s \tag{16}$$

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

The backward pass in ANCFIS is a combination of gradient descent and a derivative-free optimization technique called variable-neighborhood chaotic simulated annealing (VNCSA). From layers 6 to 1, gradient descent is applied to propagate

the network errors through the hidden layers; the approach in [40] is implemented from layers 4 to 1 to calculate the partial derivatives of complex variables. However, there is no closed-form solution for the partial derivatives of the layer 1 outputs with respect to the membership function parameters in (7), and so at this point VNCSA is invoked.

The updates of the membership parameters are calculated in two steps. First, the derivatives from $O_{1,ji}$ to the sampled membership functions, $g(.)$ in (10), is calculated

$$\frac{\partial O_{1,ji}}{\partial g(k)} \; k = 1, 2, \ldots, n \tag{17}$$

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

$$g(k)_{\text{new}} = g(k)_{\text{old}} - \eta \frac{\partial E}{\partial g(k)} \tag{18}$$

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

$$J(a,b,c,d) = \sum_{m=1}^{r} \Big[\text{mag}(g(k)_{\text{new}})_m - \big(d * \sin\big(a * \text{phase}(g(k)_{\text{new}})_m + b\big) + c\big)\Big]^2 \tag{19}$$

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

VNCSA is a chaotic simulated annealing (SA) 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 SA with chaotic maps (which are deterministic but highly irregular functions). VNCSA is initialized using the logistic map [16]

$$x_{i,j+1} = 4 \times x_{i,j} \times \big(1 - x_{i,j}\big) x_{i,j} = x_{i,j+1} \;\; i = 1, 2, 3, 4 \tag{20}$$

where i is the number of variables in the objective function (19), 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

$$S_{i,j+1} = f_i + (k_i - f_i) \times x_{i,j+1} \tag{21}$$

where f_i and k_i are the lower and upper bound, respectively. After initialization, at each temperature in the annealing schedule [1], $L_{\max}$ iterations of the following process are performed.

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

$$S_{ij+1}^{\text{new}} = S_i^{\text{current}} + D_i y_{i,j+1} \tag{22}$$

where S_i^{current} is updated by S_{ij+1}^{new} based on SA. $i = 1, 2, 3, 4$ is the number of variables in the objective function, $j = 1, 2, \ldots, M$ is the current iteration of the Ulam-von Neumann map, S_i^{current} is initialized to the initial population. $y_{i,j}$ is the value of the Ulam-von Neumann map and D_i is the neighborhood defined as

$$y_{i,j+1} = 1 - 2{y_{i,j}}^2 y_{i,j} = y_{i,j+1} \tag{23}$$

$$D_i = 0.1 \times (k_i - f_i). \tag{24}$$

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

$$D_i^{\text{new}} = (1 - \alpha) D_i^{\text{current}} + \alpha \omega R_i \tag{25}$$

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

$$T_{k+1} = T_k \beta \quad 0 < \beta < 1 \tag{26}$$

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

Other complex fuzzy machine learning systems have been proposed. An online variation of ANCFIS was proposed in [2] which uses recursive least-squares instead of a Kalman filter in the forward pass, and the downhill-simplex optimization algorithm instead of VNCSA in the backward pass. Li and Chiang [45] proposed another family of complex fuzzy machine learning system based on ANFIS, the CNFS. The system is a six-layer network and is trained by a hybrid of particle swarm optimization (PSO) and recursive least-square estimation (RLSE). Li and Chiang [46] extended the CNFS by adding Gaussian CFSs

$$\begin{aligned}
\text{cGaussian}(h, m, \sigma) &= \text{Re}(\text{cGaussian}(h, m, \sigma)) \\
&\quad + j\text{Im}(\text{cGaussian}(h, m, \sigma)) \\
\text{Re}(\text{cGaussian}(h, m, \sigma)) &= \exp\left(-0.5\left(\frac{h-m}{\sigma}\right)^2\right) \\
\text{Im}(\text{cGaussian}(h, m, \sigma)) &= -\exp\left(-0.5\left(\frac{h-m}{\sigma}\right)^2\right) \\
&\quad \times \left(\frac{h-m}{\sigma^2}\right)
\end{aligned} \tag{27}$$

where $\{m, \sigma\}$ are the mean and spread of the Gaussian function, and h is the input. The CNFS proposed in [46] was applied for adaptive image noise cancelation in [50]. Li and Chan [43] modified the CNFS learning algorithm to use the artificial bee colony (ABC)-RLSE learning algorithm and also applied it for adaptive image noise cancelation; this system was also applied for knowledge discovery in [42].

Reference [44] is another variation on CNFS that uses hierarchical multiswarm PSO (RLSE) as its learning algorithm. The CFS membership functions in this variant have the following form:

$$\begin{aligned}
\text{cGaussian}(x, m, \sigma, \lambda) &= r_s(x, m, \sigma) \exp(j\omega_s(x, m, \sigma, \lambda)) \\
r_s(x, m, \sigma) &= \text{Gaussian}(x, m, \sigma) \\
&= \exp\left(-0.5\left(\frac{x-m}{\sigma}\right)^2\right) \omega_s(x, m, \sigma, \lambda) \\
&= -\exp\left(-0.5\left(\frac{x-m}{\sigma}\right)^2\right) \\
&\quad \times \left(\frac{x-m}{\sigma^2}\right) \times \lambda
\end{aligned} \tag{28}$$

where $\{m, \sigma, \lambda\}$ are the mean, spread, and phase frequency factor for the CFS, respectively, and x is the input. This variant was applied to time series prediction in [47]. A variant

in [80] applied PSO and genetic algorithm and RLSE as the hybrid learning algorithm, and used the membership function proposed in (28). Li *et al.* [48] proposed another CNFS variant, using the membership function in (28), with the PSO-RLSE learning algorithm. To minimize the number of rules in the CNFS, a clustering algorithm called FCM-based splitting algorithm is employed in [71]. Reference [49] was based on the CFS and learning algorithms of [48], but replaced the linear consequent function with an ARIMA model. Li *et al*. [48] and Shoorangiz and Marhaban [68] proposed other variations of machine for complex fuzzy systems. Other works on CFS&L can be found in [3], [20], [25], [34], [36], [51], [53], [54], and [77].

The "dual-output property" was proposed as a signature characteristic of the CNFS family of architectures. This refers to the treatment of a complex-valued output as two independent real values. The property is used in [46] to approximate two functions simultaneously, and in [49], to predict bivariate time series; the real and imaginary parts of the complex-valued output each represent one function or one variate.

At this point, it is necessary to distinguish between time-series forecasting using neuro-fuzzy methods, and fuzzy time series (FTS) forecasting. The former employs neuro-fuzzy systems to forecast a crisp time series, while the latter first "fuzzifies" the time series (using various approaches) before forecasting. Examples of this latter approach may be found in [7] and [8]. The research reported in the current paper concerns only crisp time series forecasting; no transformation of our datasets to an FTS is performed.

B. Delay Embedding of Time Series

A variety of authors have employed machine learning for time series forecasting (see [7], [8], [22], [30], [73]). 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 [35].

One approach to state space reconstruction, the technique of *delay embeddings*, is based on Takens' theorem in [73], which showed that a sufficient number of the present and earlier values of a time series (*lags*) can form a state space that is equivalent to the original, unrecoverable one (in the sense that they are related by a smooth, invertible mapping). The *delay vectors* thus constructed are samples of a state trajectory, which should in theory be sufficient *in themselves* to predict the next point(s) on that trajectory. Thus, a collection of delay vectors formed from a time series can be treated as a set of instance records for machine learning; the lags are the independent variables, and the predicted next value(s) are the dependent variable(s). The key parameters for a delay embedding are the dimensionality of a delay vector, and the time delay between successive components of the delay vector [35].

Delay vectors in a univariate time series have the form [27]

$$S_m = \left(s_{m-(n-1)\tau}, s_{m-(n-2)\tau}, \ldots, s_m\right) \tag{29}$$

where S_m is a delay vector, whose most recent component is the mth element of the time series s_m, s_i is the ith element of the time series, and the delay vector contains n components, each separated by $\tau - 1$ elements in the time series. For univariate datasets, 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 [35]. Kennel *et al.* [37] proposed the false nearest neighbors (FNNs) heuristic for the dimension n. The main idea of this heuristic is that if a delay vector (a point in the state space) is in a neighborhood in the state space, its one step-ahead evolution must remain in the one-step ahead evolution of the neighborhood; otherwise, the point only seems to fall in the neighborhood because of unresolved projections of the actual state space.

Delay vectors of an M-variate time series $X_1, X_2, \ldots, X_N$, where $X_i = (x_{1,i}, x_{2,i}, \ldots, x_{M,i})$, are given by [10]

$$V_n = \begin{matrix} \big(x_{1,n}, x_{1,n-\tau_1}, \ldots, x_{1,n-(d_1-1)\tau_1} \\ x_{2,n}, x_{2,n-\tau_2}, \ldots, x_{2,n-(d_2-1)\tau_2} \\ \cdots \quad \cdots \\ x_{M,n}, x_{M,n-\tau_M}, \ldots, x_{M,n-(d_M-1)\tau_M}\big) \end{matrix} \tag{30}$$

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 ith 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 [5], [10], [70]. However, the dimensionality of the variates is perhaps better determined together. Cao *et al.* [10] 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, $(\tau_i, d_i\ i = 1, 2, \ldots, M)$, its nearest neighbors are determined using the Euclidean norm; second, their one-step-ahead prediction error is calculated. Then, the set of dimensions with the lowest prediction error is used as the embedding dimension.

C. RBFN and SVR

An RBFN is a two layer neural network consisting of a hidden layer implementing a nonlinear transformation and a linear output layer, where the input–output mapping is defined as [15]

$$f_r(x) = \lambda_0 + \sum_{i=1}^{n_r} \lambda_i \varphi(\| x - c_i \|) \tag{31}$$

where $x \in R^n$ is the input vector, $\varphi(.)$ is a given function from $\mathbb{R}^+$ to $\mathbb{R}$, $\|\cdot\|$ denotes the Euclidean norm, λ_i are the weights $0 \le i \le n_r$, $c_i \in R^n$ are the RBF centers, $1 \le i \le n_r$, n_r is the number of centers, and $\varphi(.)$ can be a Gaussian function [15]

$$\varphi(x) = \exp\left(\frac{-\| x - c_i \|^2}{\beta^2}\right) \tag{32}$$

where β is the spread parameter.

SVR estimates a function based on training data, $\{(x_1, y_1), \ldots, (x_l, y_l)\} \subset \mathbb{R}^d \times \mathbb{R}$, as [69]

$$f(x) = \langle w, x \rangle + b \tag{33}$$

where $w \in \mathbb{R}^d$ are the weights, $b \in \mathbb{R}$ is the bias, and $\langle .,. \rangle$ is the dot product in $\mathbb{R}^d$. The goal is to find w and b such that the following constraint is satisfied [65], [69]:

$$\text{minimize} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} L(y(i), f(x(i), w)) \tag{34}$$

where the first part, $\|w\|^2 = \langle w, w \rangle$, biases the optimization toward smooth functions, and the second part assures that the error between the values predicted by the linear regression, $f(x)$, and the actual observations is not more than ε, a user-defined parameter. $C > 0$ is a user-defined parameter, $y(i)$ is the desired value, and $L(.)$ is a loss function; the ε-insensitive loss function [65]

$$L(y(i), f(x(i), w)) = \begin{cases} |y(i) - f(x(i), w)| - \varepsilon & \text{if } |y(i) - f(x(i), w)| \geq \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

is commonly used. Moreover, minor violations of the threshold ε are accommodated by slack variables, ξ, ξ^*. Thus, the regression problem can be formulated as [69]

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\left(\xi_i + \xi_i^*\right) \\ &\text{subject to} \quad \begin{cases} y_i - f(x_i, w) \leq \varepsilon + \xi_i^* \\ f(x_i, w) - y_i \leq \varepsilon + \xi_i \\ \xi_i, \xi_i^* \geq 0, i = 1, \ldots, n. \end{cases} \end{aligned} \tag{36}$$

The problem can be solved by Lagrange multiplier techniques that give the following results [69]:

$$\begin{aligned} w &= \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right)x_i \\ f(x) &= \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right)\langle x_i, x \rangle + b \end{aligned} \tag{37}$$

where α_i and α_i^* are Lagrange multipliers associated with each data point in the training set. Data points with non-zero Lagrange multipliers are involved in the regression, and are called support vectors.

When the relation between the training data cannot be formulated by linear regression, the data points are mapped into a higher dimension feature space, $\varphi(x)$, where their relation is linear. The mapping is done by a kernel function, K, yielding [69]

$$f(x) = \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right) \times K(x_i, x) + bK(x_i, x) = \langle \varphi(x_i), \varphi(x) \rangle. \tag{38}$$

The Gaussian RBF kernel is one of the most commonly used kernel functions, and is defined as [81]

$$K(x_i, x) = \exp\left(-0.5\frac{\|x - x_i\|^2}{\sigma^2}\right) \tag{39}$$

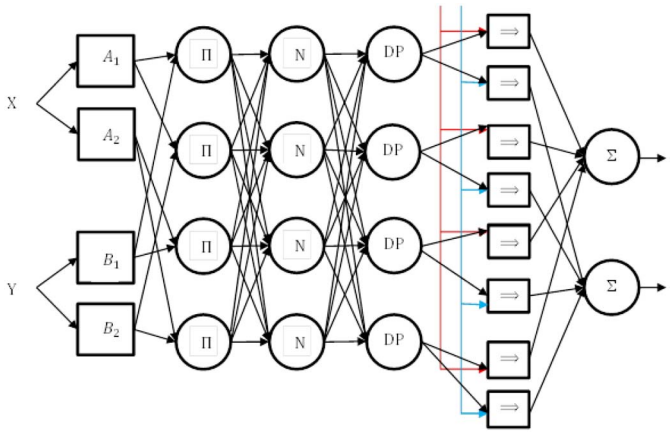

Fig. 2. MIMO ANCFIS architecture for a bivariate time series problem with two membership functions for each variate.

where σ defines the width of the kernel function.

We study three alternative designs of RBFN in dealing with multivariate time series, SISO, MISO, and MIMO. For SVR, only SISO and MISO designs are investigated. We use the *e*1071 implementation of SVR in the *R* environment [57], and the newrb.m implementation of RBFN in MATLAB [19].

III. Methodology

A. Network Design

In [86], we studied three alternative designs for creating a multivariate ANCFIS system: 1) SISO ANCFIS; 2) MISO ANCFIS; and 3) 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, and the number of variates 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. Fig. 2 shows the MIMO ANCFIS architecture for a bivariate time series.

In this paper, we refine and expand upon the preliminary results in [86]. Our new results supersede those earlier findings. Moreover, we investigate the effect of using different heuristics to determine the dimensionalities for each variate in our datasets. We contrast determining the dimensions separately via the FNN technique [37] with the k-NN prediction error approach proposed by Cao *et al*. [10]. We fix k at 3 in our experiments; 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 $x(t\text{-}n)$ [35]). The forecast accuracies for these approaches are

compared against each other, and against the well-known RBFN and SVR algorithms.

Beyond these network design changes, we also investigate two architectural changes that may improve the performance of MIMO ANCFIS. First, we investigate three potential modifications to the rule interference operation. Second, 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 layers 3, 4, and 6 transfer functions. Recall that layer 3 normalizes the rule firing strengths using (13), layer 4 implements a dot product between the current rule's firing strength and the vector sum of all rule firing strengths using (14), and layer 6 is a summation of its inputs per (16). We propose the following three variants of ANCFIS.

1) In our first variant (first 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 [21]). The new equations for layers 3 and 4 are

$$O_{3,i} = \widehat{w}_i = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} O_{2,j}},\ i = 1, 2, \ldots, |O_2| \text{(layer 3)} \tag{40}$$

$$O_{4,i} = w_i^{\mathrm{DP}} = \widehat{w}_i \cdot 1 = |\widehat{w}_i|.\cos\left(\widehat{\theta}_i\right) \text{(layer 4)} \tag{41}$$

where $|\widehat{w}_i|$ and $\widehat{\theta}_i$ are the amplitude and phase of the output of layer 3, $\widehat{w}_i$.

2) In our second variant (second ANCFIS), the layer 3 transfer function remains (49) as above. 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 throughout 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

$$O_{3,i} = \widehat{w}_i = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} O_{2,j}},\ i = 1, 2, \ldots, |O_2| \quad \text{(layer 3)} \tag{42}$$

$$O_{4,i} = w_i^{\mathrm{DP}} = \widehat{w}_i \quad \text{(layer 4)} \tag{43}$$

$$O_{6,j} = \left(\sum_{i=1+(j-1)*N}^{j*N} w_i^s \right).1 \quad \text{(layer 6)}. \tag{44}$$

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

$$O_{6,j} = \left| \sum_{i=1+(j-1)*N}^{j*N} w_i^s \right|. \quad \text{(layer 6)}. \tag{45}$$

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 (18)]

$$\eta \frac{\partial E}{\partial g(k)} = g(k)_{\mathrm{old}} - g(k)_{\mathrm{new}} \tag{46}$$

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. We explore this variation only in the MIMO design, and so we will refer to it as "new MIMO ANCFIS" in our experiments.

B. 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 root mean square error (RMSE) and mean absolute error (MAE) statistics

$$\begin{aligned} \mathrm{RMSE} &= \sqrt{\frac{1}{K}\sum_{i=1}^{K} \mathrm{MSE}_i} \\ \mathrm{MSE}_i &= \frac{\sum_{j=1}^{n} \left(y_j - \widehat{y}_i\right)^2}{n} \end{aligned} \tag{47}$$

$$\begin{aligned} \mathrm{MAE} &= \frac{1}{K}\sum_{i=1}^{K} \mathrm{MAE}_i \\ \mathrm{MAE}_i &= \frac{\sum_{j=1}^{n} \left|y_j - \widehat{y}_i\right|}{n} \end{aligned} \tag{48}$$

where K is the number of variates in the multivariate time series, MSE_i and MAE_i are the mean squared one-step-ahead error and mean absolute one-step-ahead error of the ith variate, y_j is the actual value, and $\widehat{y}_i$ is the predicted value. Both RMSE and MAE are scale-dependent measures, meaning that their interpretation depends upon the range of the dependent variable(s) being measured, and thus they cannot easily be used to compare errors on different datasets. However, they are suitable for comparing and ranking different prediction

algorithms on a given dataset [30], and that is the purpose we are using them for.

To determine if any differences observed are significant, we apply the Friedman test. The Friedman statistic, S, is calculated as [28]

$$S = \frac{12n}{k(k+1)} \sum_{j=1}^{k} \left(\overline{R_j} - \frac{k+1}{2} \right)^2 \tag{49}$$

where k is the number of approaches applied on n different time series, and $\overline{R_j}$ is the average rank of the jth 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 ith 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 [28], [82]. $\chi^2_{k-1,\alpha}$ is obtained by the NSM3 package in the *R* environment [66].

Finally, if H_0 is rejected, the approaches are compared together using the multiple comparisons with the best method [38]. The null hypothesis is defined as

$$H_0 : \tau_u = \tau_v, \quad \text{where} \quad 1 \leq u < v \leq k.$$

H_0 is rejected if $|\overline{R}_u - \overline{R}_v| \geq r_{\alpha,K,N}$, where $\overline{R}_u$ is the average rank of the uth method, and for large-sample approximation $r_{\alpha,K,N} \approx q_\alpha \sqrt{(k(k+1))/(12n)}$, 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 the *R* environment [66]. To visualize the results of the comparisons, a plot is drawn where each method has an interval with the center as the corresponding average rank, $\overline{R}_u$, and the length of $r_{\alpha,K,N}$. The best method is the one with the lowest upper boundary, which is considered as a reference line. If the rank interval of any of the method is above the reference line, we can conclude that the method is significantly worse than the best method. If any of the methods do not overlap, H_0 is rejected for them [38].

C. Datasets

For each of our datasets, we define what each variate represents, specify the split point between training samples and the out-of-sample test data, and the delay and dimension parameters for a delay embedding (the latter determined by both techniques). Finally, we 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

$$(f \otimes g)(\tau) = \sum_{i=-\infty}^{\infty} f_i^*(m) \cdot g(m+\tau) \tag{50}$$

where f and g are two variates in a multivariate time series, τ is the lag, and f^* is the complex conjugate of f [9]. The ijth entry in the matrix represents the cross-correlation between the ith and jth variates. The matrix is computed and presented by the "ccm" routine in the "MTS" package in *R*.

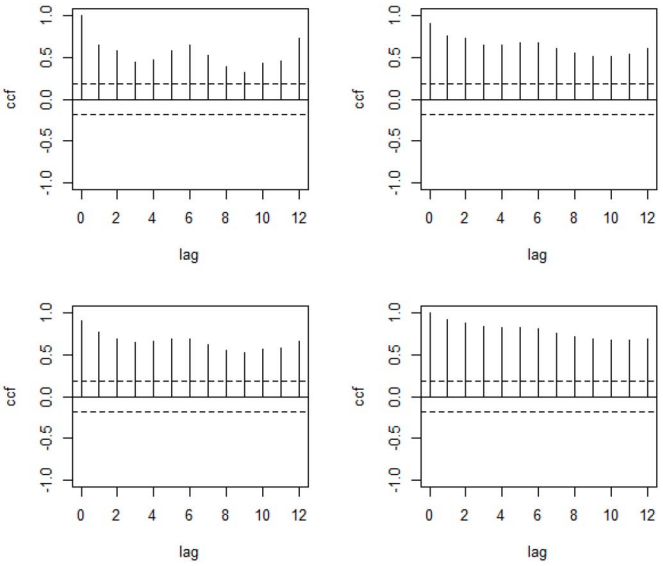

Fig. 3. Cross-correlation matrix for the motel dataset.

TABLE I
DELAY AND DIMENSION SETS FOR MOTEL TIME SERIES

	Delay (τ_1, τ_2)	Dimension (FNN) (d_1, d_2)	Dimension (k-NN Prediction Error) (d_1, d_2)
Motel	(6,6)	(6,6)	(2,2)

1) Motel: This bivariate time series contains monthly records of occupancy of hotels, motels, and guest room in Victoria, Australia over the period January 1980–June 1995 (186 data points) [29]. Its variates show the total number of room nights occupied and total revenues (thousands of dollars), respectively. The first 2/3 of the time series are used as the training set, with the final 1/3 reserved for the out-of-sample testing set. This yields 124 data points in the training and 62 data points in the testing sets. We normalize the data to the range [0, 1]. The cross-correlation matrix is presented in Fig. 3; the correlations do not appear to decaying, and so the time series is not stationary. However, this dependence on previous values is the kind of nonstationarity that learning algorithms should excel at, and so we will continue with our experiments. The delay and dimensions obtained for the two different input vector sets are determined in Table I.

2) Flour Price: 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) [29]. We split the time series into 90 data points in the training and ten data points in the testing set and normalize them to the range of [0, 1]; the split is the same as [24] in order to facilitate comparisons. The cross-correlation matrix is presented in Fig. 4, with the correlations appearing to rapidly decay. This time series appears stationary. The delay and dimensions are determined in Table II.

3) Precipitation: 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) [56]. We consider data from January 1895 to December 1924 as the

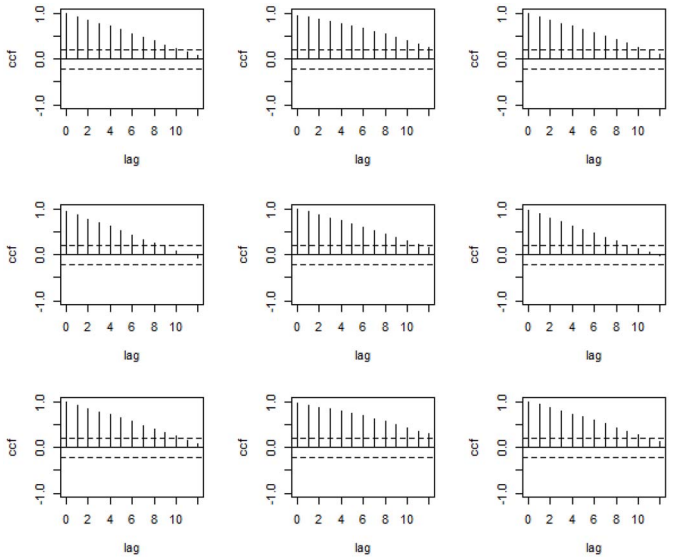

Fig. 4. Cross-correlation matrix for the flour dataset.

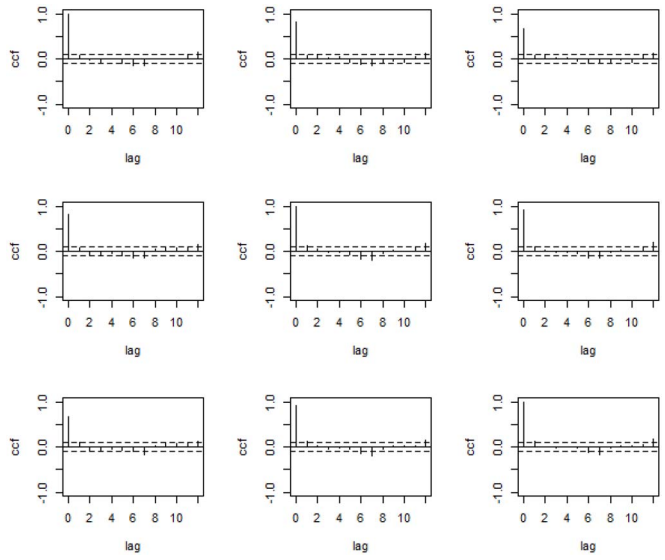

Fig. 5. Cross-correlation matrix for the precipitation dataset.

TABLE II
DELAY AND DIMENSION SETS FOR FLOUR TIME SERIES

	Delay (τ_1, τ_2, τ_3)	Dimension (FNN) (d_1, d_2, d_3)	Dimension (k-NN Prediciton Error) (d_1, d_2, d_3)
Flour	(1,1,1)	(3,3,3)	(1,3,2)

TABLE III
DELAY AND DIMENSION SETS FOR PRECIPITATION TIME SERIES

	Delay (τ_1, τ_2, τ_3)	Dimension (FNN) (d_1, d_2, d_3)	Dimension (k-NN Prediction Error) (d_1, d_2, d_3)
Precipitation	(3,1,3)	(4,4,4)	(1,3,1)

training set (360 data points) and data from January 1925 to December 1929 (60 data points) as the testing set and normalize them to the range of [0, 1]; this split is same as [32] in order to compare their results. The cross-correlation matrix is presented in Fig. 5. 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 dataset. The delay and dimension are determined in Table III.

4) NASDAQ: 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) [58]; we split the time series equally giving us 500 data points in the training and 500 data point in the testing set. We use this time series to compare our ANCFIS results with the CNFS-ARIMA learning algorithm proposed by Li and Chiang [49]. The cross-correlation matrix is presented in Fig. 6; it is very similar to the matrix for the motel dataset, and so a learning algorithm should perform well on this dataset as well. The dimensions and delay we obtain are shown in Table IV.

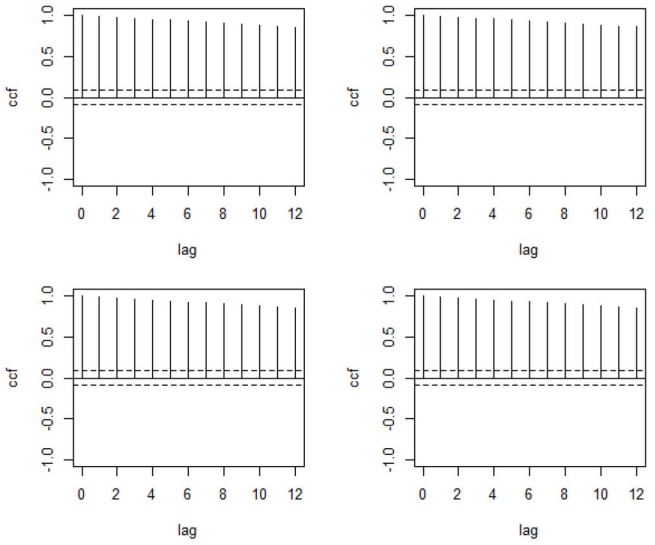

Fig. 6. Cross-correlation matrix for the NASDAQ dataset.

TABLE IV
DELAY AND DIMENSION SETS FOR NASDAQ TIME SERIES

	Delay (τ_1, τ_2)	Dimension (FNN) (d_1, d_2)	Dimension (k-NN Prediction Error) (d_1, d_2)
NASDAQ	(1,1)	(1,3)	(1,3)

IV. 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 V
RMSE OF THE MOTEL TIME SERIES

Method	**RMSE**	
VARMA(2,2)	0.1849	
	k-NN Prediction Error	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 VI
RMSE OF THE FLOUR TIME SERIES

Method	**RMSE**	
VARMA(3,1)	0.3602	
	k-NN Prediction Error	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 VII
RMSE OF THE PRECIPITATION TIME SERIES

Method	**RMSE**	
VARMA(6,0)	0.2098	
	k-NN Prediction Error	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

Tables V–VII indicate the RMSE out-of-sample forecast errors on the motel, flour, and precipitation time series, respectively.

In order to see if the 24 approaches (12 for each dimensionality estimator) applied on the three time series are significantly different, the Freidman statistic, S, is calculated as (58); 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 datasets. For simplicity, we thus choose the single MIMO approach, and the more automated k-NN prediction error technique for finding dimensions, for further comparisons against the literature on these three datasets and the NASDAQ dataset.

TABLE VIII
COMPARATIVE RMSE OF THE FLOUR TIME SERIES

	Buffalo	Minneapolis	Kansas
New MIMO ANCFIS (*k*-NN Prediction Error)	0.16043	0.21123	0.21078
DAN2 [24]	0.61725	0.22804	0.58052

TABLE IX
COMPARATIVE MAE OF THE PRECIPITATION TIME SERIES

	East	Middle	West
New MIMO ANCFIS (*k*-NN Prediction Error)	0.15169	0.16329	0.15124
VTG Scheme[32] (Combined Model: 2nd Lagrange)	0.598	0.613	0.613

TABLE X
COMPARATIVE RMSE OF THE NASDAQ TIME SERIES

Method	**RMSE**
VARMA(2,2)	77.00
New MIMO ANCFIS (*k*-NN Prediction Error)	45.10
CNFS-ARIMA [49]	66.22

Tables VIII and IX compare the results of the New MIMO ANCFIS variant 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 X, we compare new MIMO ANCFIS on the NASDAQ dataset against the recently-published CNFS-ARIMA, which is a direct, complex-fuzzy-logic based competitor to ANCFIS.

V. CONCLUSION

We have studied the extension of ANCFIS to multivariate time series prediction. Three different ANCFIS architectural designs (SISO, MISO, and MIMO) have been examined. A further exploration of alternative designs of MIMO ANCFIS is also performed. The designs were compared with two 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 datasets, 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 [35] 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.

REFERENCES

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*. Chichester, U.K.: Wiley, 1988.

[2] S. Aghakhani and S. Dick, "An on-line learning algorithm for complex fuzzy logic," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ)*, Barcelona, Spain, 2010, pp. 1–7.

[3] A. U. M. Alkouri and A. R. Salleh, "Linguistic variables, hedges and several distances on complex fuzzy sets," *J. Intell. Fuzzy Syst.*, vol. 26, no. 5, pp. 2527–2535, 2014.

[4] K. T. Atanassov, "Intuitionistic fuzzy sets," *Fuzzy Sets Syst.*, vol. 20, no. 1, pp. 87–96, 1986.

[5] S. Boccaletti, D. L. Valladares, L. M. Pecora, H. P. Geffert, and T. Carroll, "Reconstructing embedding spaces of coupled dynamical systems from multivariate data," *Phys. Rev. E*, vol. 65, no. 3, 2002, Art. no. 035204.

[6] 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 Syst. Appl.*, vol. 37, no. 12, pp. 7908–7912, 2010.

[7] E. Bulut, "Modeling seasonality using the fuzzy integrated logical forecasting (FILF) approach," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1806–1812, 2014.

[8] E. Bulut, O. Duru, and S. Yoshida, "A fuzzy integrated logical forecasting (FILF) model of time charter rates in dry bulk shipping: A vector autoregressive design of fuzzy time series with fuzzy c-means clustering," *Marit. Econ. Logistics*, vol. 14, no. 3, pp. 300–318, 2012.

[9] J. Y. Campbell, A. W. Lo, and A. C. MacKinlay, *The Econometrics of Financial Markets*. Princeton, NJ, USA: Princeton Univ. Press, 1996.

[10] L. Cao, A. Mees, and K. Judd, "Dynamics from multivariate time series," *Phys. D Nonlin. Phenom.*, vol. 121, nos. 1–2, pp. 75–88, 1998.

[11] S. Chabaa, A. Zeroual, and J. Antari, "Identification and prediction of Internet traffic using artificial neural networks," *J. Intell. Learn. Syst. Appl.*, vol. 2, no. 3, pp. 147–155, 2010.

[12] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka, "Forecasting the behavior of multivariate time series using neural networks," *Neural Netw.*, vol. 5, no. 6, pp. 961–970, 1992.

[13] C. Chatfield, *Time-Series Forecasting*. Boca Raton, FL, USA: CRC Press, 2000.

[14] S.-M. Chen and K. Tanuwijaya, "Multivariate fuzzy forecasting based on fuzzy time series and automatic clustering techniques," *Expert Syst. Appl.*, vol. 38, no. 8, pp. 10594–10605, 2011.

[15] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[16] Z. Chen, S. Aghakhani, J. Man, and S. Dick, "ANCFIS: A neurofuzzy architecture employing complex fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 2, pp. 305–322, Apr. 2011.

[17] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Multi−scale Internet traffic forecasting using neural networks and time series methods," *Expert Syst.*, vol. 29, no. 2, pp. 143–155, 2012.

[18] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *Int. J. Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[19] H. Demuth, M. Beale, and M. Hagan, *Neural Network Toolbox for Use With MATLAB*. Natick, MA, USA: The MatWorks, Inc., 2007.

[20] A. Y. Deshmukh, A. B. Bavaskar, P. R. Bajaj, and A. G. Keskar, "Implementation of complex fuzzy logic modules with VLSI approach," *Int. J. Comput. Sci. Netw. Security*, vol. 8, no. 9, pp. 172–178, 2008.

[21] S. Dick, "Toward complex fuzzy logic," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 3, pp. 405–414, Jun. 2005.

[22] S. Dick, R. R. Yager, and O. Yazdanbakhsh, "On pythagorean and complex fuzzy set operations," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 5, pp. 1009–1021, Oct. 2016.

[23] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–34, 2012.

[24] M. Ghiassi and S. Nangoy, "A dynamic artificial neural network model for forecasting nonlinear processes," *Comput. Ind. Eng.*, vol. 57, no. 1, pp. 287–297, 2009.

[25] S. Greenfield and F. Chiclana, "Fuzzy in 3-D: Contrasting complex fuzzy sets with type-2 fuzzy sets," in *Proc. Joint IFSA World Congr. NAFIPS Annu. Meeting (IFSA/NAFIPS)*, Edmonton, AB, Canada, 2013, pp. 1237–1242.

[26] M. Han and Y. Wang, "Analysis and modeling of multivariate chaotic time series based on neural network," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 1280–1290, 2009.

[27] R. Hegger, H. Kantz, and T. Schreiber, "Practical implementation of nonlinear time series methods: The TISEAN package," *Chaos*, vol. 9 no. 2, pp. 413–435, 1999.

[28] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*. New York, NY, USA: Wiley, 2013.

[29] R. J. Hyndman and M. Akram. *Time Series Data Library*. Accessed on Sep. 1, 2016. [Online]. Available: https://datamarket.com/data/list/?q=provider:tsdl

[30] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[31] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[32] T. Jo, "VTG schemes for using back propagation for multivariate time series prediction," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2692–2702, 2013.

[33] S. S. Jones *et al.*, "A multivariate time series approach to modeling and forecasting demand in the emergency department," *J. Biomed. Informat.*, vol. 42, no. 1, pp. 123–139, 2009.

[34] A. Kandel, D. Tamir, and N. D. Rishe, "Fuzzy logic and data mining in disaster mitigation," in *Improving Disaster Resilience and Mitigation-IT Means and Tools*, H.-N. Teodorescu, A. Kirschenbaum, S. Cojocaru, and C. Bruderlein, Eds. Amsterdam, The Netherlands: Springer, 2014, pp. 167–186.

[35] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[36] D. Karpenko, R. A. Van Gorder, and A. Kandel, "The Cauchy problem for complex fuzzy differential equations," *Fuzzy Sets Syst.*, vol. 245, pp. 18–29, Jun. 2014.

[37] M. B. Kennel, R. Brown, and H. D. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A*, vol. 45, no. 6, pp. 3403–3411, 1992.

[38] A. J. Koning, P. H. Franses, M. Hibon, and H. O. Stekler, "The M3 competition: Statistical tests of the results," *Int. J. Forecasting*, vol. 21, no. 3, pp. 397–409, 2005.

[39] R. J. Kuo and K. C. Xue, "Fuzzy neural networks with application to sales forecasting," *Fuzzy Sets Syst.*, vol. 108, no. 2, pp. 123–143, 1999.

[40] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2101–2104, Sep. 1991.

[41] C. Li, "Adaptive image restoration by a novel neuro-fuzzy approach using complex fuzzy sets," *Int. J. Intell. Inf. Database Syst.*, vol. 7, no. 6, pp. 479–495, 2013.

[42] C. Li and F.-T. Chan, "Knowledge discovery by an intelligent approach using complex fuzzy sets," in *Intelligent Information and Database Systems*. Heidelberg, Germany: Springer, 2012, pp. 320–329.

[43] C. Li and F. Chan, "Complex-fuzzy adaptive image restoration—An artificial-bee-colony-based learning approach," in *Intelligent Information and Database Systems*. Heidelberg, Germany: Springer, 2011, pp. 90–99.

[44] 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*. Heidelberg, Germany: Springer, 2011, pp. 242–251.

[45] C. Li and T.-W. Chiang, "Complex neuro-fuzzy self-learning approach to function approximation," in *Intelligent Information and Database Systems*. Heidelberg, Germany: Springer, 2010, pp. 289–299.

[46] C. Li and T.-W. Chiang, "Function approximation with complex neuro-fuzzy system using complex fuzzy sets—A new approach," *New Gener. Comput.*, vol. 29, no. 3, pp. 261–276, 2011.

[47] C. Li and T.-W. Chiang, "Intelligent financial time series forecasting: A complex neuro-fuzzy approach with multi-swarm intelligence," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 4, pp. 787–800, 2012.

[48] C. Li, T.-W. Chiang, and L.-C. Yeh, "A novel self-organizing complex neuro-fuzzy approach to the problem of time series forecasting," *Neurocomputing*, vol. 99, pp. 467–476, Jan. 2013.

[49] C. Li and T.-W. Chiang, "Complex neurofuzzy ARIMA forecasting—A new approach using complex fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 567–584, Jun. 2013.

[50] C. Li, T. Wu, and F.-T. Chan, "Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling," *Neurocomputing*, vol. 94, pp. 121–139, Oct. 2012.

[51] C. K. Loo, A. Memariani, and W. S. Liew, "A novel complex-valued fuzzy ARTMAP for sparse dictionary learning," in *Proc. Neural Inf. Process.*, Daegu, South Korea, 2013, pp. 360–368.

[52] J. Ma and L. Liu, "Multivariate nonlinear analysis and prediction of Shanghai stock market," *Discrete Dyn. Nat. Soc.*, vol. 2008, pp. 1–8, Jan. 2008.

[53] J. Ma, R. Wickramasuriya, M. Safari, T. Davies, and P. Perez, "A conceptual method for modeling residential utility consumption using complex fuzzy sets," in *Proc. Joint IFSA World Congr. NAFIPS Annu. Meeting (IFSA/NAFIPS)*, Edmonton, AB, Canada, 2013, pp. 1227–1232.

[54] J. Ma, G. Zhang, and J. Lu, "A method for multiple periodic factor prediction problems using complex fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 1, pp. 32–45, Feb. 2012.

[55] J. Y. Man, Z. Chen, and S. Dick, "Towards inductive learning of complex fuzzy inference systems," in *Proc. Annu. Meeting North Amer. Fuzzy Inf. Process. Soc. (NAFIPS)*, San Diego, CA, USA, 2007, pp. 415–420.

[56] T. Masters, *Neural, Novel and Hybrid Algorithms for Time Series Prediction*. New York, NY, USA: Wiley, 1995.

[57] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *Misc Functions of the Department of Statistics (e1071)*, TU Wien, Vienna, Austria, 2012. [Online]. Available: http://cran.rproject.org/web/packages/e1071/e1071.pdf

[58] NASDAQ Composite Index. (2014). *Yahoo Finance*. [Online]. Available: http://finance.yahoo.com/q?s=^IXIC

[59] V. Nourani, M. T. Alami, and M. H. Aminfar, "A combined neural-wavelet model for prediction of Ligvanchai watershed precipitation," *Eng. Appl. Artif. Intell.*, vol. 22, no. 3, pp. 466–472, 2009.

[60] D. Ramot, M. Friedman, G. Langholz, and A. Kandel, "Complex fuzzy logic," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 4, pp. 450–461, Aug. 2003.

[61] D. Ramot, R. Milo, M. Friedman, and A. Kandel, "Complex fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 171–186, Apr. 2002.

[62] G. C. Reinsel, *Elements of Multivariate Time Series Analysis*. New York, NY, USA: Springer-Verlag, 2003.

[63] A. S. Alkouri and A. R. Salleh, "Complex intuitionistic fuzzy sets," in *Proc. AIP Conf.*, Kuala Lumpur, Malaysia, 2012, pp. 464–470.

[64] A. A. P. Santos, F. J. Nogales, and E. Ruiz, "Comparing univariate and multivariate models to forecast portfolio value-at-risk," *J. Financ. Economet.*, vol. 11, no. 2, pp. 400–441, 2013.

[65] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 24–38, May 2009.

[66] G. Schneider, E. Chicken, and R. Becvarik, *NSM3: Functions and Datasets to Accompany Hollander, Wolfe, and Chicken—Nonparametric Statistical Methods*, 3rd ed., accessed on Sep. 1, 2016. [Online]. Available: https://rdrr.io/cran/NSM3/

[67] A. Sfetsos and A. H. Coonick, "Univariate and multivariate forecasting of hourly solar radiation with artificial intelligence techniques," *Solar Energy*, vol. 68, no. 2, pp. 169–178, 2000.

[68] R. Shoorangiz and M. H. Marhaban, "Complex neuro-fuzzy system for function approximation," *Int. J. Appl. Electron. Phys. Robot.*, vol. 1, no. 2, pp. 5–9, 2013.

[69] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[70] L.-Y. Su, "Prediction of multivariate chaotic time series with local polynomial fitting," *Comput. Math. Appl.*, vol. 59, no. 2, pp. 737–744, 2010.

[71] H. Sun, S. Wang, and Q. Jiang, "FCM-based model selection algorithms for determining the number of clusters," *Pattern Recognit.*, vol. 37, no. 10, pp. 2027–2037, 2004.

[72] Z.-L. Sun, T.-M. Choi, K.-F. Au, and Y. Yu, "Sales forecasting using extreme learning machine with applications in fashion retailing," *Decis. Support Syst.*, vol. 46, no. 1, pp. 411–419, 2008.

[73] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, WARWICK 1980*. Berlin, Germany: Springer, 1981, pp. 366–381.

[74] D. E. Tamir, L. Jin, and A. Kandel, "A new interpretation of complex membership grade," *Int. J. Intell. Syst.*, vol. 26, no. 4, pp. 285–312, 2011.

[75] D. E. Tamir and A. Kandel, "Axiomatic theory of complex fuzzy logic and complex fuzzy classes," *Int. J. Comput. Commun. Control*, vol. 6, no. 3, pp. 508–522, 2011.

[76] D. E. Tamir, M. Last, and A. Kandel, "The theory and applications of generalized complex fuzzy propositional logic," in *Soft Computing: State of the Art Theory and Novel Applications*. Berlin, Germany: Springer, 2013, pp. 177–192.

[77] D. E. Tamir, N. D. Rishe, M. Last, and A. Kandel, "Soft computing based epidemical crisis prediction," in *Intelligent Methods for Cyber Warfare*. Cham, Switzerland: Springer, 2015, pp. 43–67.

[78] D. E. Tamir, H. N. Teodorescu, M. Last, and A. Kandel, "Discrete complex fuzzy logic," in *Proc. Annu. Meeting North Amer. Fuzzy Inf. Process. Soc. (NAFIPS)*, Berkeley, CA, USA, 2012, pp. 1–6.

[79] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[80] P. Thirunavukarasu, R. Suresh, and P. Thamilmani, "Complex neuro fuzzy system using complex fuzzy sets and update the parameters by PSO-GA and RLSE method," *Int. J. Eng. Innov. Technol.*, vol. 3, no. 1, pp. 117–122, 2013.

[81] U. Thissen, R. Van Brakel, A. P. De Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometr. Intell. Lab. Syst.*, vol. 69, nos. 1–2, pp. 35–49, 2003.

[82] D. Wackerly, W. Mendenhall, and R. Scheaffer, *Mathematical Statistics With Applications*. Belmont, CA, USA: Cengage Learn., 2007.

[83] W.-C. Wang, K.-W. Chau, C.-T. Cheng, and L. Qiu, "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series," *J. Hydrol.*, vol. 374, nos. 3–4, pp. 294–306, 2009.

[84] R. R. Yager, "Pythagorean membership grades in multicriteria decision making," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 958–965, Aug. 2014.

[85] R. R. Yager and A. M. Abbasov, "Pythagorean membership grades, complex numbers, and decision making," *Int. J. Intell. Syst.*, vol. 28, no. 5, pp. 436–452, 2013.

[86] O. Yazdanbakhsh and S. Dick, "Multi-variate timeseries forecasting using complex fuzzy logic," presented at the NAFIPS-WConSC, Redmond, WA, USA, 2015, pp. 1–6.

[87] O. Yazdanbakhsh and S. Dick, "Time-Series forecasting via complex fuzzy logic," in *Frontiers of Higher Order Fuzzy Sets*. New York, NY, USA: Springer, 2015, pp. 147–165.

[88] O. Yazdanbaksh, A. Krahn, and S. Dick, "Predicting solar power output using complex fuzzy logic," in *Proc. IFSA World Congr. NAFIPS Annu. Meeting Joint (IFSA/NAFIPS)*, Edmonton, AB, Canada, 2013, pp. 1243–1248.

[89] G. Zhang, T. S. Dillon, K.-Y. Cai, J. Ma, and J. Lu, "Delta-equalities of complex fuzzy relations," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Perth, WA, Australia, 2010, pp. 1218–1224.

[90] G. Zhang, T. S. Dillon, K.-Y. Cai, J. Ma, and J. Lu, "Operation properties and δ-equalities of complex fuzzy sets," *Int. J. Approx. Reason.*, vol. 50, no. 8, pp. 1227–1249, 2009.

[91] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[92] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003.

Omolbanin Yazdanbakhsh (M'13) received the B.Sc. and M.Sc. degrees from Islamic Azad University, Najafabad Branch, Esfahan, Iran, in 2007 and 2010, respectively, both in electrical engineering. She is currently pursuing the Ph.D. degree in computer engineering with the University of Alberta (U of A), Edmonton, AB, Canada.

Her current research interests include machine learning and data mining focusing on time series forecasting.

Ms. Yazdanbakhsh was a recipient of the Best Student Paper Award at NAFIPS 2015, the Queen Elizabeth II Graduate Scholarship in 2015, and the U of A Doctoral Recruitment Scholarship in 2012.

Scott Dick (M'02) received the B.Sc. and M.Sc. degrees in computer science and the Ph.D. degree in computer science and engineering from the University of South Florida, Tampa, FL, USA in 1997, 1999, and 2002, respectively.

He was an Assistant Professor of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, from 2002 to 2008, where he has been an Associate Professor, since 2008. His current research interests include computational intelligence, machine learning, and data mining.

Dr. Dick is the Chair of the IEEE CIS Fuzzy Systems Technical Committee Task Force on Complex Fuzzy Sets and Logic.