International Journal of Approximate Reasoning 105 (2019) 417–430

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

FANCFIS: Fast adaptive neuro-complex fuzzy inference system

Omolbanin Yazdanbakhsh, Scott Dick *

Dept. of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada

A R T I C L E I N F O

Article history:
Received 30 March 2018
Received in revised form 9 August 2018
Accepted 29 October 2018
Available online 1 November 2018

Keywords:
Complex fuzzy systems
Machine learning
Neuro-fuzzy systems
Time series forecasting
Data stream mining

A B S T R A C T

Large-scale multivariate time series forecasting is an increasingly important problem, with sensor networks pouring out sampled data at unprecedented rates, trillions of dollar's worth of stock trades made every data, and zettabytes of traffic transmitted over the Internet every year. While they 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 Adaptive Neuro-Complex-Fuzzy Inferential System 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 complex fuzzy rules. 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.

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 [13,41]. 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 [30]. 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 [26]. Volatility, in the meantime, refers not only to a distributional shift in the data

* Corresponding author at: Dept. of Electrical & Computer Engineering, University of Alberta, 9211-116 Street NW, Edmonton, Alberta, T6G 1H9, Canada.
E-mail addresses: yazdanba@ualberta.ca (O. Yazdanbakhsh), sdick@ualberta.ca (S. Dick).

https://doi.org/10.1016/j.ijar.2018.10.018
0888-613X/© 2018 Elsevier Inc. All rights reserved.

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 behavior of the spammer and in the perspective of the receiver regarding a spam category [14]. The volume, velocity and volatility of stream data imply that forecasting models must be incrementally learned for a limited historical time window. 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 [5]. Again, an incremental algorithm generally lends itself to this adjustment. 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, wherein each data point is timestamped. Statistical models (e.g. FARIMA [11], GARCH [15], etc.) and machine-learning have all been employed in forecasting, and found to be effective on a wide variety of datasets. As one example, machine learning algorithms based on CFS&L have been shown to be very accurate and parsimonious forecasting models [1,7,65,67,68]. The Adaptive Neuro-Complex Fuzzy Inference System (ANCFIS) was the first machine learning algorithm based on Ramot’s CFS&L [44,45] and Dick’s findings in [8]; the system is a derivative of Jang’s Adaptive Neuro-Fuzzy Inferential System [25], modified to employ complex fuzzy sets in its rule antecedents and rule interference in its inferential process. Results in [7] 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, ANCFIS employs a batch learning algorithm combining gradient descent and chaotic simulated annealing, which makes the algorithm relatively slow. It is thus ineffective as a stream-forecasting algorithm. 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 paper, we develop fast incremental algorithms for time series forecasting based on ANCFIS [7]. 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 (i.e. having the largest absolute value of their coefficients), and a CFS matching each one of these is added to the model. Then, the consequent parameters of the FANCFIS model are updated incrementally via recursive least squares. As a contrast, we also modify the RANCFIS randomized-learning variant of ANCFIS [62] for incremental learning (we denote this new variant as RANCFIS-Stream). In our experiments, we find that FANCFIS, while being slower than RANCFIS-Stream, is generally more accurate.

Our research in this paper make three contributions to the fuzzy and neuro-fuzzy systems literature. Firstly, we have designed fast and accurate stream-mining algorithms based on the ANCFIS architecture, demonstrating that complex fuzzy logic can usefully be applied to this class of problems. Secondly, we present a method for inductively determining the membership function parameters of a sinusoidal CFS directly form the Fourier transform of a time series, which has never previously been explored. Finally, we investigate the hypothesis that, model structures being equal, a fully adaptive model will be more accurate than one using randomized learning – albeit slower. This illuminates a trade-off that is not often empirically explored in the randomized learning literature.

The remainder of the paper is organized as follows: in Section 2, we provide background on data stream mining, CFS&L, ANCFIS, RANCFIS, delay embedding techniques and the discrete Fourier transform. Our methodology is described in Section 3, and our experimental results in Section 4. We close with a summary and discussion of future work in section 5.

2. Literature review

We review, complex fuzzy sets and logic, ANCFIS and RANCFIS structure, and delay embeddings for a time series.

2.1. Complex fuzzy sets

Ramot et al. proposed the complex fuzzy membership grade as [45]:

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

where r_s and w_s are the amplitude and phase of the membership grade, respectively. The membership is restricted to the unit disk in the complex plane ($r_s(x) \in [0, 1]$), while w_s can take any real value. The complex fuzzy membership reduces to type-1 fuzzy sets when $w_s(x) = 0$. Each element of the universe of discourse is thus associated with a vector-valued membership, as in Fig. 1.

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. The unit disk D is projected along the universe of discourse, U, producing a cylinder within which the complex fuzzy grades are bounded. 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 Fig. 1, the intersection of the orthogonal unit disk (outlined in red) and a membership function gives the corresponding grade.

Ramot et al. proposed the first complex fuzzy inferential system in [44]. Antecedent and consequent predicates consisted of complex fuzzy sets, and the complex fuzzy conjunction from [45] was used to combine the antecedents. The algebraic product was suggested as the implication operator, and a “vector aggregation” operator was proposed for combining rule consequents [44]:

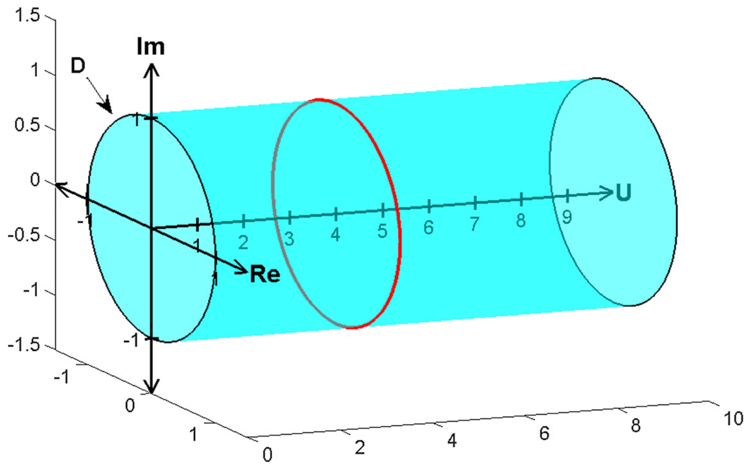

Fig. 1. Complex fuzzy membership grades. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$v : \{a \mid a \in \boldsymbol{C}, |a| \le 1\}^n \to \{b \mid b \in \boldsymbol{C}, |b| \le 1\} \tag{2}$$

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

where $w_i \in \{a \mid a \in \boldsymbol{C}, |a| \le 1\}$ for all i, and $\sum_{i=1}^{n} |w_i| = 1$. This aggregation allows constructive or destructive interference, based on the phase of the rule consequents.

Ma et al. introduced the product-sum aggregation operation for complex fuzzy sets [38], which is an extension of the aggregation in Eq. (3). Dick [8] showed that the membership grades proposed by Ramot et al. [44,45] are rotationally invariant (phase is only a relative quantity); as a result, algebraic product cannot be its conjunction operator. He further showed that, when amplitude and phase are coupled (i.e. membership is a vector), algebraic product is one possible conjunction operator. Sinusoidal membership functions were also proposed as candidate complex fuzzy membership functions; they have the advantage of being able to capture periodic behavior [8]. Recently, Dick et al. have examined the relationship between Pythagorean fuzzy sets [58] and complex fuzzy sets. This led to the identification of two new DeMorgan triples (union, intersection, negation) for complex fuzzy sets, and an analysis of their properties. Various complex fuzzy operations have been introduced [45,57]. Zhang et al. introduced distance between two complex fuzzy sets and δ-equalities of complex fuzzy sets [70].

Tamir et al. proposed “pure” complex fuzzy sets with membership grades drawn from a unit square $[0, 1] \times [0, 1]$ [50]:

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

where $\mu_r(V)$ and $\mu_i(z)$ are the real and imaginary part of the pure complex fuzzy membership grade, $\mu(V, z)$. Union and intersection operators have been proposed for the pure complex fuzzy set [50]. Pure complex fuzzy logics were developed in [51–54].

Yager et al. introduced Pythagorean membership grade as [58,59]:

$$\begin{aligned} &A = \{\langle x, A_Y, A_N\rangle \mid x \in U\} \\ &A_Y = r(x)\cos\big(\theta(x)\big) \\ &A_N = r(x)\sin\big(\theta(x)\big) \end{aligned} \tag{5}$$

where $A_Y : U \to [0, 1]$ and $A_N : U \to [0, 1]$ are the membership grade and non-membership grade of object x in the fuzzy set A, respectively, provided that $(A_Y)^2 + (A_N)^2 \le 1$ (with membership and non-membership having the same meanings as in intuitionistic fuzzy sets). This membership degree can be considered as complex fuzzy grade provided that $r \in [0, 1]$ and $\theta \in [0, \frac{\pi}{2}]$. Yager also introduced conjunction and aggregation operation for Pythagorean grade [59]. Salleh et al. proposed Complex Atanssov's Intuitionistic Fuzzy Sets (CAIFS) [46]; its membership grade contains a membership ($\mu_A(x) : U \to [0, 1]$) and non-membership ($\gamma_A(x) : U \to [0, 1]$) grade, provided that $0 \le \mu_A(x) + \gamma_A(x) \le 1$. Salleh et al. in [46] defined complement, union and intersection for CAIFS based on [45,70], and [3] extended the intuitionistic possibility and necessity measures defined by [4] for CAIFS.

Interpretability – a key advantage of fuzzy systems – arises due to the use of linguistic variables, the association of natural-language words with fuzzy sets having congruent semantics. However, the two-dimensional nature of CFS membership functions has so far made finding such congruences extremely difficult, and this is perhaps the most important

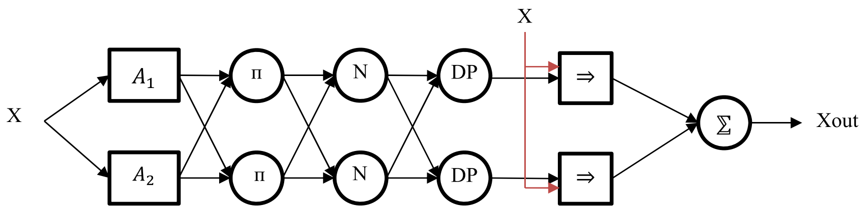

Fig. 2. ANCFIS architecture for univariate time series problems with two membership functions.

open problem in CFS. Currently, Alkouri [2] defines complex linguistic variables as a 6-tuple associating different terms – representing uncertainty and periodicity – with the magnitude and phase of the CFS, respectively. Six linguistic hedges (as introduced by Zadeh [69]) are also defined: *very A, very-very A, indeed A, a little A, slightly A, more or less A and extremely A*. Tamir et al. suggest that propositions of the form "x...A...B," with A the main linguistic term and B a linguistic modifier could be represented by a complex fuzzy class [51]. Finally, Dick et al. extended the Pythagorean fuzzy sets to the entire unit disc, defining the negative real and negative imaginary axes as anti-membership and anti-non-membership, respectively. They suggest that this could be a unified model for negation and antonym in fuzzy logic, as it captured the key characteristic that "the negation of the antonym is the antonym of the negation" from linguistics [10]. A detailed review of complex fuzzy sets and logic may be found in [64].

2.2. ANCFIS

The Adaptive Neuro-Complex Fuzzy Inference System (ANCFIS) is a six-layer architecture created as the first machine learning algorithm based on CFS&L [7]. Its structure is based on the well-known ANFIS (Adaptive Neuro-Fuzzy Inference System) [24] with five major modifications, and focused on time series prediction as its major application area. First, ANCFIS uses sinusoids as its membership functions based on Dick's suggestion in [8] as opposed to the bell-shaped and triangular membership functions in ANFIS. Second, observations from the input time series are windowed for input to ANCFIS; this design maintains the temporal relation between data points and reduces the number of inputs. In ANFIS, each lagged observation is treated as a separate input to the system. Third, input fuzzification in ANCFIS is accomplished through a complex-valued convolution in the first layer. Fourth, there is one more layer in ANCFIS located before the output layer; this layer implements the rule interference property suggested in [44], using the dot product. The last difference is in their learning algorithm. Similar to ANFIS, ANCFIS uses hybrid learning to train its parameters, with a least-squares algorithm employed in the forward pass to estimate the linear consequent parameters. However, in the backward pass, ANCFIS employs a combination of gradient descent and a derivative free optimization algorithm to update the sinusoidal membership function parameters.

The sinusoidal membership function used in ANCFIS is defined as [7]:

$$r(\theta) = d\sin\big(a(\theta = x) + b\big) + c \tag{6}$$

where $r(\theta)$ is membership amplitude, θ is membership phase, and $x \in X$ is an object of the universal set X. $\{a, b, c, d\}$ are the membership function parameters; a changes frequency, b is responsible for phase shift, c shifts the sine wave vertically and d changes the amplitude. The following conditions must be satisfied to ensure the grade remains in the unit disk.

$$0 \leqslant d + c \leqslant 1, \qquad 1 \geqslant c \geqslant d \geqslant 0 \tag{7}$$

ANCFIS has a six-layer architecture as follows [7] (Fig. 2):

- *Layer 1:* Input vectors are fuzzified in this layer by computing the complex convolution between the input vector and samples of the sinusoidal membership function (Eq. (6)) taken over one period; the number of the samples is equal to the length of the input vector.

$$conv = \sum_{k=1}^{2n-1} \sum_{j=\max(1,k+1-n)}^{\min(k,n)} f(j)g(k+1-j) \tag{8}$$

where $f(.)$ is a data point in the n-element input vector with length of n, and $g(.)$ is the sampled membership function. The grades are then normalized to the unit disk by the Elliot function:

$$O_{1,ji} = \frac{conv}{1+|conv|} \quad i = 1, 2, \ldots, m, \;\; j = 1, 2, .., n \tag{9}$$

where m is the number of membership functions, and $O_{1,ji}$ is the output of the i-th node in Layer 1 for the j-th input vector.

- *Layer 2:* The algebraic product is used as conjunction operator in this layer to obtain the firing strength of each rule.

$$O_{2,i} = \prod_{\substack{j=1 \\ k_j \in K_j}}^{n} O_{1,jk_j} \quad i = 1, 2, \ldots, m^n \tag{10}$$

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

- *Layer 3:* Firing strengths are normalized in this layer.

$$O_{3,i} = \hat{w}_i = \frac{O_{2,i}}{\sum_{j=1}^{|O_2|} |O_{2,j}|}, \quad i = 1, 2, \ldots, |O_2| \tag{11}$$

where $|O_2|$ is the number of rules.

- *Layer 4:* Rule interference is implemented by taking dot product of the current node's output and the complex sum of all other nodes' output in the layer 3.

$$O_{4,i} = w_i^{DP} = \hat{w}_i \cdot \sum_{j=1}^{|O_3|} \hat{w}_j \tag{12}$$

where $|O_3|$ is the number of nodes in layer 3.

- *Layer 5:* Consequent parameters are calculated by a Kalman filer.

$$O_{5,i} = w_i^s = w_i^{DP} \left[\sum_{k=1}^{j} \sum_{l=1}^{n} p_{i,kl} x_{kl} + r_i \right] \tag{13}$$

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 l-th data point of the k-th input vector, and $p_{i,kl}$, r_i are the consequent parameters of the linear function.

- *Layer 6:* All incoming signals to this layer are summed to compute the final output of ANCFIS

$$O_{6,j} = \sum_{i=1+(j-1)*N}^{j*N} w_i^s \tag{14}$$

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

Outputs from layer 1 through layer 3 in ANCFIS are complex-valued. After the dot product in layer 4, all the outputs of the remaining layers are real-valued. In the backward pass, from layer 6 to layer 1, gradient descent is applied to propagate the network error to each neuron; however, as there is no closed-form solution for the derivative of the network error with respect to the membership function parameters, a derivative free optimization algorithm (VNCSA) is implemented [7] to perform this last step.

The Complex Neuro-Fuzzy System (CNFS) is a group of related machine learning algorithms based on CFS&L, also derived from ANFIS but using a form of complex Gaussian membership functions [31–36]. The CNFS architectures were also designed for time series prediction; using the "dual-output property," CNFS can produce forecasts for a bi-variate time series using only one output node [32,34–36].

2.3. RANCFIS

The Randomized Adaptive Neuro-Complex Fuzzy Inference System (RANCFIS) is a six-layered feed-forward network inspired by ANCFIS [7]. However, there are two major differences between them. Firstly, RANCFIS employs randomized learning in its architecture. The membership function parameters, $\{a, b, c, d\}$, are selected randomly from a uniform distribution and held constant; this eliminates the backwards learning pass completely. Moreover, output weights, β_i, are considered in the connection between layer 5 and layer 6, and trained using least squares estimation. Thus, the inputs to the layer 6 are [62]:

$$I_{6,i} = w_i^{DP} \left[\sum_{k=1}^{j} \sum_{l=1}^{n} p_{i,kl} x_{kl} + r_i \right] . \beta_i = w_i^{DP} \left[\sum_{k=1}^{j} \sum_{l=1}^{n} \gamma_{i,kl} x_{kl} + \eta_i \right] \tag{15}$$

where $\gamma_{i,kl} = p_{i,kl} . \beta_i$ and $\eta_i = r_i . \beta_i$. As with ANCFIS, layer 6 sums its inputs to produce the final network output.

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

where j is the number of outputs and N is the number of rules. To obtain the parameters $\gamma_{i,kl}$ and η_i in Eq. (15), Eq. (16) is rewritten as:

$$\mathrm{T} = \mathrm{H} \cdot \beta \tag{17}$$

where

$$\mathrm{T}^T = \begin{bmatrix} O_{6,1} & \cdots & O_{6,j} \end{bmatrix} \tag{18}$$

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

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

β is estimated (as also in the earlier ANCFIS-ELM architecture [60]) using the RLS algorithm:

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}} \tag{19}$$

$$\beta^{(k+1)} = \beta^{(k)} + M_{k+1} h_{k+1} \left(t_i^T - h_{k+1}^T \beta^{(k)}\right) \tag{20}$$

with the initial weights obtained as [18]:

$$\beta^{(0)} = 0 \tag{21}$$

$$M_0 = \lambda^{-1} I, \quad \lambda \text{ a small positive constant.} \tag{22}$$

2.4. Delay embedding

Forecasting based on machine learning algorithms is based on the assumption that time series are deterministic; this means that the state-space trajectory of the system underlying the time series is unique, and the trajectory to time t completely determines the future evolution of the system. However, time series are simply sequences of observations and do not directly provide any information regarding either the system or the state space [28]. Thus, there is a need to reconstruct the state space in order to exploit this determinism – or at least, construct an equivalent state space. The most common approach for doing so is Takens' delay embedding technique.

A delay vector is an ordered sequence of prior values of the time series (we will refer to these as *lags*) relative to the current observation, and the process of generating delay vectors relative to each point in the time series is a delay embedding. The number of lags (which we will denote m) is the dimensionality of the embedding. Lags do not need to be successive observations, but the number of observations between each lag should be constant; this is the delay parameter (sometimes also called the *lag temperature*), which we will denote by τ. According to Takens' embedding theorem [49], if the embedding dimensionality is large enough (at least $2D + 1$, where D is the dimensionality of the actual system state space), then the delay vectors can be interpreted as points in an equivalent (isomorphic) state space to the original one. Thus, the trajectory sampled by the delay vectors (arranged in time order) again completely determines the future evolution of the system, and thus the delay vectors can be used to forecast the time series. Most commonly, the next k values of the time series are appended to a delay vector as the dependent variables to be predicted by a machine learning algorithm (this is referred to as the k-steps-ahead forecasting problem).

Delay vectors in a univariate time series have the following form [20]:

$$S_n = (s_{n-(m-1)\tau}, s_{n-(m-2)\tau}, \dots, s_n) \tag{23}$$

where τ and m are defined above. The main difficulty in performing a delay embedding is to determine the values of τ and m for a given time series; Takens' theorem provides no useful guidance on this question. Instead, heuristics are normally used [28]. The delay (τ) is often determined by the first minimum of the time-delayed mutual information function or the first zero of the autocorrelation function. The dimension (m) can be estimated by the False Nearest Neighborhood (FNN) algorithm [29] or the approach proposed by Cao in [6]. The FNN heuristic is based on the idea that an insufficient dimensionality results in a projection of the true trajectory to a subspace, meaning that some apparent nearest neighbors are actually far distant in the true manifold. Detecting these FNN examples is based on observing an exceptionally large difference in the one-step-ahead evolution of that point versus its neighborhood. One then seeks a dimensionality for which the number of FNNs drops to a very small fraction of the dataset. Cao's approach is similar in that the one-step-ahead prediction error of a nearest-neighbor classifier is determined; however, the criterion here is to select the dimensionality that minimizes the total prediction error over the whole dataset. This approach requires no visual inspection of FNN plots, and in our experience [61] the resulting forecasts are just as accurate as those where we used FNN to determine the dimensionality. We thus use Cao's approach in the current work, employing a KD-tree-based algorithm for finding nearest neighbors.

3. Methodology

3.1. FANCFIS

The Fast Adaptive Neuro-Complex fuzzy Inference System (FANCFIS) is a six-layered feed forward neural network. Much like RANCFIS, FANCFIS was designed to overcome the slow learning speed of ANCFIS, while remaining a highly accurate time series forecasting algorithm. Unlike RANCFIS, however, FANCFIS accomplishes this goal without the use of random learning. Instead, the sinusoidal membership functions are induced directly from the time series by taking an FFT of a portion of it. Since the membership functions are thus tuned specifically to the dataset, we would thus generally expect that a FANCFIS network will be somewhat smaller than a RANCFIS network that achieves the same accuracy on the time series.

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 embedding are determined. Then, the initial output weights of the FANCFIS network are calculated. In the incremental learning step, the output weights are updated via recursive least squares as each new observation arrives.

In the initialization step, the time series is partitioned, with the chronologically first segment used to induce the CFS membership function parameters through a Fourier transform. The discrete Fourier transform of a time series with N data points $A(k)$ $k = 1, 2, \dots, N$, is calculated as [48]:

$$F_A(n) = \sum_{k=0}^{N-1} \frac{A(k)}{N} e^{-i2\pi nk/N} \tag{24}$$

where $F_A(n)$ is complex-valued; the real and imaginary components are the amplitude of a sine and a cosine wave, respectively, at frequency n. The inverse DFT reconstructs a time series as [48]:

$$\begin{aligned} A(k) &= \sum_{n=0}^{N-1} F_A(n) e^{\frac{i2\pi nk}{N}} \\ &= \sum_{n=0}^{N-1} F(n)_{real} \cos\left(\frac{2\pi nk}{N}\right) - \sum_{n=0}^{N-1} F(n)_{imag} \sin\left(\frac{2\pi nk}{N}\right) \end{aligned} \tag{25}$$

We begin by using the Tisean software to select a subsequence of the initialization partition with a minimal endpoint mismatch. The Fourier transform of this subsequence is computed (using the Fast Fourier Transform algorithm), and the N_{mf} frequencies with the highest power identified, where N_{mf} is the number of CFS to be used. Membership functions are determined as:

$$\begin{aligned} & r(\theta) = F(n_i)_{real} \cos(\theta = x) - F(n_i)_{imag} \sin(\theta) + DC \quad i = 1, 2, \dots, N_{mf} \\ & \theta = \frac{2\pi n_i k}{N} \quad k = 1, 2, \dots, N \end{aligned} \tag{26}$$

where r and θ are the amplitude and phase of the complex membership grade, respectively, $x \in X$ is an element of the universe of discourse X, n_i is the frequency with the i-th highest power $F(n_i)_{real}$ and $F(n_i)_{imag}$ are the real and imaginary parts of the Fourier transform in the frequency of n_i, N is length of the subsequence, and DC is the zero frequency component, $F_A(0) = \sum_{k=0}^{N-1} \frac{A(k)}{N}$. We use the fast Fourier transform implementation in [12]. For a univariate time series, this results in rules of the form:

$$\text{If } X = [x_1, x_2, \dots, x_n] \text{ is } A\cos(\theta = x) - B\sin(\theta) + C \text{ then } z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b \tag{27}$$

This rule has no easy linguistic interpretation that we are aware of, demonstrating the interpretability problem discussed in Section 2.1. Furthermore, as the initialization partition may only be a small fragment of the time series, it seems inadvisable to compute the delay embedding parameters using it. We thus compute delays and dimensions over the entire training

Table 1
FANCFIS algorithm.

```
FANCFIS algorithm

A. Initialization Step:
    1. Normalize the training set as:

       $$\frac{x_i - \max(X)}{\max(X) - \min(X)} \qquad (28)$$

       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, Eq. (29) repeats for each variate.
    2. Obtain delay and dimension for delay vectors
        a. Use the mutual information heuristic to estimate $\tau$
        b. Use the k-NN approach to estimate the dimension $m$ as proposed in [6]. The nearest neighbor search is carried out by a KD-tree algorithm.
    3. Form the delay vectors for multivariate time-series and 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 Eq. (26)
    5. For N epochs repeat:
        a. Calculate output weights (Eqs. (17)–(22))
B. Incremental Learning Step: for each new data point,
    6. 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 (Eqs. (17)–(22))
```

set, using (respectively) the time-delayed mutual information heuristic and Cao's method from [6], implemented using the k-nearest-neighbor search from [40].

After obtaining the both membership function parameters (Eq. (26)) and the delay vectors, we can calculate the initial output weights of FANCFIS. In the first layer of FANCFIS, complex membership grades of the input delay vectors are calculated by applying the complex convolution (Eq. (8)) between the delay vectors and each membership function (Eq. (26)), followed by the Elliot function (Equation (9)). Layer 2 through layer 6 are then identical to RANCFIS with output weights estimated by the RLS algorithm using Eqs. (17)–(22).

In the incremental learning step, each delay vector FANCFIS receives is used to update output weights via RLS. We note that FANCFIS' architecture is suitable for data stream forecasting. We have also re-implemented RANCFIS as a stream miner, in order to directly compare the two algorithms on similar tasks. As with FANCFIS, RANCFIS Stream computes its initial output weights from the initialization partition (but the membership function parameters are chosen randomly). The output weights are updated via RLS in the incremental learning step, as with FANCFIS.

3.2. Complexity analysis

We will study the time complexity (order of growth) for FANCFIS and RANCFIS Stream in this section. We consider six different inputs: the number of variates in the time series (*NumVar*), the number of membership functions (*NumMF*), the number of outputs (*NumOut*), number of training data used for the initial learning step (*NumTr*), the maximum dimensionality allowed in Cao's method [6] (*DMax*), and finally the number of observations encountered during the incremental learning step (*NumObs*). For simplicity of presentation in this analysis, we assume that *NumMF* is same for all the variates in the time series; the extension for a non-uniform number of membership functions is trivial. We further assume that the length of the initialization partition is chosen to be much greater than the internal time-scales of the system (i.e. the length of one orbit of the state-space attractor) [19,28]. Finally, the number of delay vectors *NumDV* is given by $NumTr - (m-1) * D$, with m the actual dimensionality of the delay reconstruction and D the delay parameter (a constant determined by examining heuristics). This is bounded below by $NumTr - (DMax - 1) * D$, and bounded above by *NumTr*. For simplicity of presentation, we include *NumDV* as an expression in our analysis.

In each step of Table 1, 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 less than or equal to *DMax*. The running time of the K-nearest-neighbor search algorithm (*KnnSearch*) when implemented via KD-trees is estimated as $O(Knear * \log(NumDV))$ where *Knear* is the number of neighbors to find 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 step 2.
4. $numVar * Fourier$, where the running time of the FFT is estimated as $O(NumTr * \log(NumTr))$.

Table 2
RANCFIS algorithm.

```
RANCFIS algorithm

A. Initialization Step:
   1. Normalize the training set per Eq. (29).
   2. Obtain delay and dimension for delay vectors
      a. Use mutual information to estimate $\tau$
      b. The approach from [6] to estimate $m$
   3. Form the delay vectors for multivariate time-series and 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 (Eqs. (17)–(22))
```

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 step, we thus have the following number of operations:

$$\begin{aligned} & NumTr * numVar + D * numVar + DMax^{numVar} * (NumDV * numVar * DMax) * KnnSearch \\ & \quad + NumDV * numVar * numDim + numVar * Fourier + epoch * \big(numVar \\ & \quad * numMF * DMax^2 + numVar * numMF + numMF^{numVar} * numVar \\ & \quad + numOut * \big(numMF^{numVar} * DMax^{numVar} + numMF^{numVar}\big)^2\big) \end{aligned}$$

Broadly speaking, for large time series this expression will be dominated by the number of training samples. *DMax* is constrained by practical considerations; very few phenomena to date have been observed to exhibit *high*-dimensional deterministic dynamics. Equilibrium points and limit cycles are obviously low-dimensional constructs, and the strange attractors observed in chaotic systems also usually have low fractal dimensions [28]. The delay length D must, by its nature be very small compared to the length of the initialization partition (specifically, the final delay vectors (a multiple of D) should be substantially smaller than one full orbit of the attractor, whatever its geometry). *NumVar* and *NumOut* are constants, and *NumMF* is a user-defined parameter. Thus, the running time of the initialization step should be $O(NumTr * \log(NumTr))$.

For incremental learning, we have the following number of operations for each step, which repeat for each new data point:

6. $NumObs * numVar$
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 incremental learning step are:

$$\begin{aligned} & NumObs * \big(NumObs * numVar + numVar * numDim + numVar * numMF * DMax^2 + numVar \\ & \quad * numMF + numMF^{numVar} * numVar \\ & \quad + numOut * \big(numMF^{numVar} * DMax^{numVar} + numMF^{numVar}\big)^2\big) \end{aligned}$$

Again, for large time series the number of observations will dominate the other inputs; thus the running time for incremental learning is estimated as $O(NumObs^2)$.

From Table 2, all the steps are same as in FANCFIS except step four (determining membership functions); we treat the complexity of a single random draw as $O(1)$. Thus, the complexity in the initial step is dominated by the determination of dimensions via Cao's method, estimated as $O(Knear * \log(NumDV))$, and for the incremental learning, the running time is estimated as $O(NumObs^2)$.

Table 3
Univariate time series.

Univariate time series	Num observation	Train/test
Solar Power [65]	20,000	2/3 – 1/3
Santa Fe Laser A [65]	1000	9/10 – 1/10
Stellar [65]	600	4/5 – 1/5
Mackey–Glass [25] [65]	1000	1/2 – 1/2
Sunspot [65]	280	~3/4 – ~1/4
Wave [65]	320	4/5 – 1/5
Mozilla [66]	86,077	2/3 – 1/3
Android [66]	20,168	2/3 – 1/3
ODC1 [9,37]	1207	2/3 – 1/3
ODC4 [9,37]	2008	2/3 – 1/3

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 in the testing set. The initialization partition is in turn the chronologically earliest part of the training set. We explore the size of this initialization set, repeating our experiments with 10%, 20% and 50% of the training set dedicated to it. We furthermore explore the number of CFS per variate, varying it from one to five. This matches our goal of exploring the performance difference between randomized and fully inductive learning, as we are comparing the two approaches on networks of the same size.

Our results in RANCFIS and FANCFIS are compared using the root mean square error (RMSE) and mean absolute error (MAE) statistics:

$$\begin{aligned} \mathrm{RMSE} &= \sqrt{\frac{1}{K}\sum_{i=1}^{K} MSE_i} \\ MSE_i &= \frac{\sum_{j=1}^{n}(y_j-\hat{y}_i)^2}{n} \end{aligned} \tag{29}$$

$$\begin{aligned} \mathrm{MAE} &= \frac{1}{K}\sum_{i=1}^{K} MAE_i \\ MAE_i &= \frac{\sum_{j=1}^{n}|y_j-\hat{y}_i|}{n} \end{aligned} \tag{30}$$

where K is the number of variates in the time series; 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, respectively, and y_j and $\hat{y}_i$ shows actual and desired values, respectively. We employ MAE for the software reliability growth time series, for consistency with other papers analyzing them [60] [62]; for all other time series, RMSE is used as the performance measure.

To determine if there is a significant difference between the results of RANCFIS and FANCFIS, the Friedman test is applied. The Friedman statistic, S, is calculated as [21]:

$$S = \frac{12n}{k(k+1)}\sum_{j=1}^{k}\left(\overline{R_j}-\frac{k+1}{2}\right)^2 \tag{31}$$

where n is the number of time series and k is the number of different approaches applied on them, and $\overline{R_j}$ is the average rank of the j-th method applied on the n various time series. The null hypothesis H_0 is compared against the alternative H_1:

$$\begin{aligned} &H_0 : [\tau_1 = \cdots = \tau_k] \\ &H_1 : [\tau_1, \ldots, \tau_k \text{ not all equal}] \end{aligned} \tag{32}$$

where τ_i is the i-th method effect. For either n or $k > 5$, if H_0 is true, S is approximately chi-square distributed with $k-1$ degrees of freedom; thus, 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 [21,56]. $\chi^2_{k-1,\alpha}$ is obtained from the *NSM3* package in the R environment [47].

3.4. Data sets

We have selected 14 time series for our experiments; ten univariate times series, including four software reliability growth data sets, and four multivariate time series (Tables 3 and 4).

Table 4
Multivariate time series.

Multivariate time series	Num observation	Num variate	Train/test
Motel [23] [63]	186	2	2/3 – 1/3
Precipitation [39] [63]	420	3	6/7 – 1/7
Flour [23] [63]	100	3	9/10 – 1/10
NASDAQ [42] [63]	1000	2	1/2 – 1/2

Table 5
Number of membership functions, delay vector's parameters and portion of data saved.

	FANCFIS				RANCFIS Stream			
	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 6
Comparing FANCFIS and RANCFIS Stream. The Friedman statistic shows a significant difference.

	FANCFIS	RANCFIS-Stream
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 (MAE)	0.0273	0.0274
Android (MAE)	0.0639	0.0624
ODC1 (MAE)	0.0299	0.0304
ODC4 (MAE)	0.0325	0.0322
Motel	0.1928	0.2433
Flour	0.1447	0.1697
NASDAQ	23.957	24.027
Precipitation	0.20206	0.20206

4. Results

We now present our experimental results based on the discussions in Section 3.1. We have implemented the FANCFIS and RANCFIS-Stream algorithms discussed above as stand-alone application programs, written in C. The full source code for both FANCFIS and RANCFIS-Stream is available at https://github.com/SonbolYb/Stream-ANCFIS-FeedForward.

We begin by comparing FANCFIS and RANCFIS Stream against each other, in terms of both accuracy and running time (in the initialization step; as above, the two are identical during incremental learning). We then compare FANCFIS against our existing batch-learning approaches, RANCFIS [62] and multivariate ANCFIS [61]. This second investigation should reveal if the stream-learning design of FANCFIS has substantially reduced its accuracy.

We begin with the hyper-parameter settings that we found most effective for FANCFIS and RANCFIS Stream. Table 5 presents the number of membership functions, embedding delays and dimensions, and the size of the initialization set for both algorithms on each of our datasets. Table 6 presents our comparison of the accuracy of the two methods.

In order to determine if the differences between FANCFIS and RANCFIS above are significant, we calculate the Friedman statistic, S, in Eq. (31) 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 is above the critical value of 3.84, and so the null hypothesis is rejected and there is a significant difference between FANCFIS and RANCFIS Stream; plainly, FANCFIS is somewhat more accurate.

To compare the execution time of FANCFIS and RANCFIS Stream, Table 7 presents the time taken for the initial learning step (seconds) with the initialization set fixed at 10% of the training set, and only two CFS per variate (this should be the

Table 7
Time Taken (seconds) for initializing FANCFIS and RANCFIS Stream.

	FANCFIS	RANCFIS-Stream
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 8
Performance of FANCFIS against other approaches. No significant difference between FANCFIS and CFL-based batch learning algorithms are observed based on the Friedman test.

	FANCFIS	RANCFIS [62]	ANCFIS [61,65]	Other methods
Solar Power	4.8948	1.949	3.106	–
Santa Fe Laser A	0.0557	0.0523	0.114	0.037 [55]
Mackey–Glass	0.0136	0.021	0.015	0.0047 TSK-NFIS [17]
Stellar	0.01560	0.0120	0.014	0.071 Neural Net [17]
Sunspot	0.0988	0.112	0.103	0.036 TSK-NFIS [17]
Wave	0.0622	0.121	0.009	0.08 [22]
Mozilla (MAE)	0.0273	–	–	0.027 RBFN [66]
Android (MAE)	0.0639	–	–	0.066 RBFN [66]
ODC1 (MAE)	0.0299	–	–	0.028 RBFN [66]
ODC4 (MAE)	0.0325	–	–	0.33 RBFN [66]
Motel	0.1928	0.041	0.155	–
Flour	0.1447	0.198	0.196	0.46 DAN2 [16]
NASDAQ	23.957	23.49	23.50	66.22 CNFS-ARIMA [36]
Precipitation	0.20206	0.206	0.198	0.608 VTG Scheme [27]

most advantageous set of parameters for FANCFIS, so these results should be considered a lower bound on the difference between the two algorithms). We use the clock() function in C to obtain the CPU time consumed by the program.

What we can conclude form Tables 6 & 7 is that FANCFIS is somewhat more accurate than RANCFIS Stream with a similar number of membership function nodes, but at the cost of requiring a longer computation time. We do strongly suspect that, given a sufficiently large number of randomly-chosen CFS, RANCFIS Stream would close that accuracy gap; however, as pointed out in the Introduction, this tends to become unwieldy in large-scale learning.

We next consider how FANCFIS compares against classical batch-learning approaches on the same datasets. The experiments for RANCFIS [62] and ANCFIS [61,65] are all the classic chronologically-ordered single-split design; the algorithms are trained on the full training set, and tested on the test set. As per Section 3.4, the training and test sets for all datasets match across all three algorithms (although we do not have prior results on the SRGM datasets, and so omit those). Table 8 presents the performance of all three algorithms. Employing the Friedman test again, we find that the test statistic $S = 0.2$, which is far below the critical value for $\chi_{2,0.05}$; thus there is no significant difference between FANCFIS and the two CFL-based batch-learning approaches.

In Table 9 we investigate the question of model compactness for FANCFIS, comparing it with the RANCFIS-Stream algorithm from Table 6, and the models from the general literature identified in Table 8. As FANCFIS and RANCFIS-Stream are neuro-fuzzy implementations of a complex fuzzy inferential system, our complexity measure for them is the number of (complex) fuzzy rules, as these are the first-class units of model functionality. As for the other architectures, we adopt a complexity measure for each which is both appropriate to that architecture and congruent to our measures for FANCFIS and RANCFIS-Stream. For example, the models from [22] and [66] are radial basis function networks; the number of radial basis function neurons in the hidden layer is thus a sensible measure of complexity. Likewise, [16], [17], [27] are variations on the multilayer perceptron (with one hidden layer) for the datasets so marked, and thus the number of hidden neurons is again reasonable. [17] also employs a modified ANFIS network for two of the other datasets, and so the number of fuzzy rules is again a sensible complexity measure. [36] reports on one of the variants of the CNFS architecture, the other major approach to inductive learning of a complex fuzzy system in the literature; the number of CFL rules is thus directly comparable to our architecture (note that [36] uses fuzzy clustering to reduce its rulebase, while ANCFIS simply forms a complete rulebase over all variates).

Table 9
Model Complexities for the Results in Tables 6 and 8.

	FANCFIS complexity	RANCFIS-Stream complexity	Model complexity in the literature
Solar Power	3 CFL Rules	1 CFL Rule	N/A
Santa Fe Laser A	4 CFL Rules	1 CFL Rule	500 Consequent Neurons [55]
Mackey–Glass	2 CFL Rules	1 CFL Rule	4 Fuzzy Rules [17]
Stellar	3 CFL Rules	1 CFL Rule	3 Hidden Neurons [17]
Sunspot	5 CFL Rules	1 CFL Rule	18 Fuzzy Rules [17]
Wave	5 CFL Rules	2 CFL Rules	47 RBF Neurons [22]
Mozilla	4 CFL Rules	1 CFL Rule	$>$ 100 RBF Neurons [66]
Android	5 CFL Rules	3 CFL Rules	$>$ 100 RBF Neurons [66]
ODC1	5 CFL Rules	3 CFL Rules	$>$ 50 RBF Neurons [66]
ODC4	1 CFL Rule	3 CFL Rules	$>$ 50 RBF Neurons [66]
Motel	4 CFL Rules	4 CFL Rules	N/A
Flour	1 CFL Rule	27 CFL Rules	8 Hidden Neurons [16]
NASDAQ	9 CFL Rules	1 CFL Rule	5 CFL Rules [36]
Precipitation	1 CFL Rule	1 CFL Rule	5 Hidden Neurons [27]

5. Summary and future work

We have proposed the Fast Adaptive Neuro-Complex Fuzzy Inference System, a novel data-stream forecasting algorithm based on complex fuzzy sets and logic. FANCFIS maintains the advantages of compactness and accuracy from our earlier ANCFIS architecture, but can be trained far more rapidly. As a part of this learning algorithm, we have also shown how the parameters of a sinusoidal CFS can be directly determined from a Fast Fourier Transform of a time series. We have compared FANCFIS against 1) RANCFIS Stream, a randomized learning variant of FANCFIS; 2) our batch-learning variant of RANCFIS; and 3) the original ANCFIS architecture (also a batch-learning architecture). Our main findings are that FANCFIS is somewhat more accurate, but somewhat slower than RANCFIS Stream; and that FANCFIS is as accurate as the batch-learning versions of RANCFIS and ANCFIS.

Our future work in this area will focus on large-scale data stream mining. Our experiments with FANCFIS and RANCFIS Stream have together begun to illuminate the design space for stream-mining algorithms based on complex fuzzy sets and logic; we expect that there is a great deal more to explore. In addition, the demand for explanation mechanisms in practical machine-learning deployments (i.e. XAI) underlines the importance of interpretability for complex fuzzy sets. One possible direction is to merge Tamir's Complex Fuzzy Classes with the ideas of *fuzzy signatures* from [43].

Acknowledgement

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant no. RGPIN 262151.

References

[1] S. Aghakhani, S. Dick, An on-line learning algorithm for complex fuzzy logic, in: Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, 2010, pp. 1–7.
[2] A.U.M. Alkouri, A.R. Salleh, Linguistic variables, hedges and several distances on complex fuzzy sets, J. Intell. Fuzzy Syst.: Appl. Eng. Technol. 26 (2014) 2527–2535.
[3] A.U.M. Alkouri, 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.
[4] K.T. Atanassov, Intuitionistic fuzzy sets, Fuzzy Sets Syst. 20 (1986) 87–96.
[5] D. Brzeziński, Mining Data Streams with Concept Drift, Master's thesis, Poznan University of Technology, 2010.
[6] L. Cao, A. Mees, K. Judd, Dynamics from multivariate time series, Phys. D: Nonlinear Phenom. 121 (1998) 75–88.
[7] Z. Chen, S. Aghakhani, J. Man, S. Dick, ANCFIS: a neurofuzzy architecture employing complex fuzzy sets, IEEE Trans. Fuzzy Syst. 19 (2011) 305–322.
[8] S. Dick, Toward complex fuzzy logic, IEEE Trans. Fuzzy Syst. 13 (2005) 405–414.
[9] S. Dick, C. Bethel, A. Kandel, Software reliability modeling: the case for deterministic behavior, IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum. 37 (2007) 106–119.
[10] S. Dick, R.R. Yager, O. Yazdanbakhsh, On Pythagorean and complex fuzzy set operations, IEEE Trans. Fuzzy Syst. 24 (2016) 1009–1021.
[11] S. Dick, O. Yazdanbaksh, X. Tang, T. Huynh, J. Miller, An empirical investigation of Web session workloads: can self-similarity be explained by deterministic chaos?, Inf. Process. Manag. 50 (2014) 41–53.
[12] M. Frigo, 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.
[13] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Data stream mining, in: Data Mining and Knowledge Discovery Handbook, Springer, 2009, pp. 759–787.
[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (2014) 44.
[15] R.C. Garcia, J. Contreras, M. Van Akkeren, J.B.C. Garcia, A GARCH forecasting model to predict day-ahead electricity prices, IEEE Trans. Power Syst. 20 (2005) 867–874.
[16] M. Ghiassi, S. Nangoy, A dynamic artificial neural network model for forecasting nonlinear processes, Comput. Ind. Eng. 57 (2009) 287–297.
[17] D. Graves, W. Pedrycz, Fuzzy prediction architecture using recurrent neural networks, Neurocomputing 72 (2009) 1668–1678.
[18] S.S. Haykin, Neural Networks and Learning Machines, vol. 3, Pearson Education, Upper Saddle River, 2009.
[19] R. Hegger, H. Kantz, T. Schreiber, Practical implementation of nonlinear time series methods: the TISEAN package, Chaos: Interdiscip. J. Nonlinear Sci. 9 (1999) 413–435.

[20] R. Hegger, H. Kantz, T. Schreiber, Practical implementation of nonlinear time series methods: the TISEAN package, arXiv:chao-dyn/9810005, 1998.
[21] M. Hollander, D.A. Wolfe, E. Chicken, Nonparametric Statistical Methods, John Wiley & Sons, 2013.
[22] X. Hong, C.J. Harris, Experimental design and model construction algorithms for radial basis function networks, Int. J. Syst. Sci. 34 (2003) 733–745.
[23] R.J. Hyndman, M. Akram, Time series data library, 2006.
[24] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Englewood Cliffs, NJ, USA, 1997.
[25] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685.
[26] A. Jha, M. Dave, S. Madan, A review on the study and analysis of big data using data mining techniques, Int. J. Latest Trends Eng. Technol. 6 (2016).
[27] T. Jo, VTG schemes for using back propagation for multivariate time series prediction, Appl. Soft Comput. 13 (2013) 2692–2702.
[28] H. Kantz, T. Schreiber, Nonlinear Time Series Analysis, Cambridge University Press, Cambridge/New York, 1997.
[29] M.B. Kennel, R. Brown, H.D. Abarbanel, Determining embedding dimension for phase-space reconstruction using a geometrical construction, Phys. Rev. A 45 (1992) 3403.
[30] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, Open challenges for data stream mining research, ACM SIGKDD Explor. Newsl. 16 (2014) 1–10.
[31] C. Li, Adaptive image restoration by a novel neuro-fuzzy approach using complex fuzzy sets, Int. J. Intell. Inf. Database Syst. 7 (2013) 479–495.
[32] C. Li, T.-W. Chiang, Complex fuzzy computing to time series prediction a multi-swarm PSO learning approach, in: Intelligent Information and Database Systems, Springer, 2011, pp. 242–251.
[33] C. Li, T.-W. Chiang, Function approximation with complex neuro-fuzzy system using complex fuzzy sets – a new approach, New Gener. Comput. 29 (2011) 261–276.
[34] C. Li, T.-W. Chiang, Intelligent financial time series forecasting: a complex neuro-fuzzy approach with multi-swarm intelligence, Int. J. Appl. Math. Comput. Sci. 22 (2012) 787–800.
[35] C. Li, T.-W. Chiang, L.-C. Yeh, A novel self-organizing complex neuro-fuzzy approach to the problem of time series forecasting, Neurocomputing (2012).
[36] C. Li, T. Chiang, Complex neuro-fuzzy ARIMA forecasting—a new approach using complex fuzzy sets, IEEE Trans. Fuzzy Syst. 21 (2012) 567–584.
[37] M.R. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill, New York, NY, 1996.
[38] J. Ma, G. Zhang, J. Lu, A method for multiple periodic factor prediction problems using complex fuzzy sets, IEEE Trans. Fuzzy Syst. 20 (2012) 32–45.
[39] T. Masters, Neural, Novel and Hybrid Algorithms for Time Series Prediction, John Wiley & Sons, Inc., 1995.
[40] M. Muja, D. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 2227–2240.
[41] S. Muthukrishnan, Data Streams: Algorithms and Applications, Now Publishers Inc., 2005.
[42] NASDAQ Composite Index, Yahoo Finance. Available http://finance.yahoo.com/q?s=^IXIC, 2014.
[43] C. Pozna, N. Minculete, R.-E. Precup, L.T. Kóczy, A. Ballagi, Signatures: definitions, operators and applications to fuzzy modelling, Fuzzy Sets Syst. 201 (2012) 86–104.
[44] D. Ramot, M. Friedman, G. Langholz, A. Kandel, Complex fuzzy logic, IEEE Trans. Fuzzy Syst. 11 (2003) 450–461.
[45] D. Ramot, R. Milo, M. Friedman, A. Kandel, Complex fuzzy sets, IEEE Trans. Fuzzy Syst. 10 (2002) 171–186.
[46] A.R. Salleh, Complex intuitionistic fuzzy sets, in: AIP Conference Proceedings, 2012, p. 464.
[47] G. Schneider, E. Chicken, R. Becvarik, NSM3: Functions and Datasets to Accompany Hollander, Wolfe, and Chicken – Nonparametric Statistical Methods, Third Edition, 1.3 ed. 2015.
[48] R.B. Stull, An Introduction to Boundary Layer Meteorology, vol. 13, Springer Science & Business Media, 2012.
[49] F. Takens, Detecting strange attractors in turbulence, in: Dynamical Systems and Turbulence, Warwick 1980, 1981, pp. 366–381.
[50] D.E. Tamir, L. Jin, A. Kandel, A new interpretation of complex membership grade, Int. J. Intell. Syst. 26 (2011) 285–312.
[51] D.E. Tamir, A. Kandel, Axiomatic theory of complex fuzzy logic and complex fuzzy classes, Int. J. Comput. Commun. Control 6 (3) (2011) 562–576, https://doi.org/10.15837/ijccc.2011.3.2135.
[52] D.E. Tamir, M. Last, A. Kandel, The theory and applications of generalized complex fuzzy propositional logic, in: Soft Computing: State of the Art Theory and Novel Applications, Springer, 2013, pp. 177–192.
[53] D.E. Tamir, N.D. Rishe, A. Kandel, Complex fuzzy sets and complex fuzzy logic an overview of theory and applications, in: Fifty Years of Fuzzy Logic and Its Applications, Springer, 2015, pp. 661–681.
[54] D.E. Tamir, H.N. Teodorescu, M. Last, A. Kandel, Discrete complex fuzzy logic, in: Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American, 2012, pp. 1–6.
[55] J.A.B. Tomé, J.P. Carvalho, One step ahead prediction using fuzzy Boolean neural networks, in: EUSFLAT Conf., 2005, pp. 500–505.
[56] D. Wackerly, W. Mendenhall, R. Scheaffer, Mathematical Statistics with Applications, Cengage Learning, 2007.
[57] L. Wan, W.K. Ng, X.H. Dang, P.S. Yu, K. Zhang, Density-based clustering of data streams at multiple resolutions, ACM Trans. Knowl. Discov. Data 3 (2009) 14.
[58] R. Yager, Pythagorean membership grades in multi-criteria decision making, IEEE Trans. Fuzzy Syst. 22 (2014) 958–965.
[59] R.R. Yager, A.M. Abbasov, Pythagorean membership grades, complex numbers, and decision making, Int. J. Intell. Syst. (2013).
[60] O. Yazdanbakhsh, S. Dick, ANCFIS-ELM: a machine learning algorithm based on complex fuzzy sets, in: World Congress on Computational Intelligence, Vancouver, Canada, 2016.
[61] O. Yazdanbakhsh, S. Dick, Forecasting of multivariate time series via complex fuzzy logic, IEEE Trans. Syst. Man Cybern. Syst. 47 (2017) 2160–2171.
[62] O. Yazdanbakhsh, S. Dick, Induction of complex fuzzy infernetial system via randomized learning, Int. J. Intell. Syst. (2018), submitted for publication.
[63] O. Yazdanbakhsh, 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.
[64] O. Yazdanbakhsh, S. Dick, A systematic review of complex fuzzy sets and logic, Fuzzy Sets Syst. (2018), in press.
[65] O. Yazdanbakhsh, S. Dick, Time-series forecasting via complex fuzzy logic, in: A. Sadeghian, H. Tahayori (Eds.), Frontiers of Higher Order Fuzzy Sets, Springer, Heidelberg, Germany, 2015.
[66] O. Yazdanbakhsh, S. Dick, I. Reay, E. Mace, On deterministic chaos in software reliability growth models, Appl. Soft Comput. 49 (2016) 1256–1269.
[67] O. Yazdanbaksh, A. Krahn, S. Dick, Predicting solar power output using complex fuzzy logic, in: IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint, 2013, pp. 1243–1248.
[68] M. Yeganejou, S. Dick, Inductive learning of classifiers via complex fuzzy sets and logic, in: Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on, 2017, pp. 1–6.
[69] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—I, Inf. Sci. 8 (1975) 199–249.
[70] G. Zhang, T.S. Dillon, K.-Y. Cai, J. Ma, J. Lu, Operation properties and δ-equalities of complex fuzzy sets, Int. J. Approx. Reason. 50 (2009) 1227–1249.