ANCFIS-ELM: A Machine Learning Algorithm based on Complex Fuzzy Sets

Omolbanin Yazdanbakhsh
Dept. of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, Canada
yazdanba@ualberta.ca

Scott Dick
Dept. of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, Canada
dick@ece.ualberta.ca

Abstract— **The Adaptive Neuro-Complex Fuzzy Inferential System was the first neuro-fuzzy system employing complex fuzzy sets and rule interference. It was shown to be both accurate and parsimonious in time series forecasting. The main disadvantage of this system is its slow learning algorithm. One possible approach to speeding up this neuro-fuzzy system is to apply concepts from the Extreme Learning Machine family of architectures; specifically, we will randomly select the parameters of a "pool" of complex fuzzy sets, and then train the neural network by incrementally updating the parameters of a linear output function. We evaluate this new architecture on four software reliability growth datasets (a particular instance of time series forecasting).**

Keywords—Complex fuzzy sets; machine learning; neuro-fuzzy architecture; time series forecasting

I. Introduction

Machine learning algorithms based on Complex Fuzzy Sets and Logic (CFS&L) have shown promising results in time series prediction. CFS&L was first introduced by Ramot [1, 2] as an extension of type-1 fuzzy sets. The Adaptive Neuro-Complex Fuzzy Inference System (ANCFIS) is the first machine learning algorithm based on CFS&L to implement rule interference (the property of complex fuzzy rules interfering constructively or destructively with each other) [3]. It has been shown to perform well in univariate and multivariate time series forecasting [4, 5]. ANCFIS is based on Jang's ANFIS architecture [6], and likewise uses hybrid learning. Certain parameters are updated in both the forward and backward passes; the current design of the backward pass makes ANCFIS's learning speed slow and not suitable for stream mining.

The original Extreme Learning Machine (ELM) is a feed-forward neural network with a single hidden layer. Hidden layer weights and biases are assigned random values, and then output weights are determined via linear least squares [7]. As the hidden layer parameters in ELM do not need to be tuned, the training algorithm can be substantially faster than conventional architectures. Online Sequential ELM (OS-ELM) is a variation of ELM for online processes that employs the Recursive Least Squares (RLS) algorithm to update output weights [8].

Time series prediction is one of the most important tasks in various fields from humanity to engineering. One of the usages of time series analysis is in designing software reliability growth models, which use historic failures of a software system in order to forecast its reliability in the future. Reliability of software systems is critical in today's world since software systems have been combined with every aspect of our lives; we use them from checking our cellphone for new to accessing government information. However, software is more prone to failure than any other engineering product [9]. The U.S. economy may lose over $78 billion per year because of software failures [10]. Software Reliability Growth Models (SRGMs) can inform an outcomes-based approach for managing software quality improvement activities.

In this paper, we propose a new machine learning architecture which is an integration of ANCFIS and OS-ELM, which we refer to as "ANCFIS-ELM." We evaluate this system on the SRGM problem. Four software reliability interfailure time datasets are investigated here; the results of ANCFIS-ELM are compared against the well-known Radial Basis Function Network (RBFN), and a statistical model -- Autoregressive Fractionally Integrated Moving Average (ARFIMA).

The remainder of the paper is organized as follows: In section 2, we provide background on CFS&L, ANCFIS and ELM. Our new architecture is presented in section 3, and our experimental methodology and results are presented in section 4. We close with a summary and discussion of future work in the section 5.

II. Literature review

We review complex fuzzy sets and logic, ANCFIS and ELM. We also review essential background on RBFN, ARFIMA and time series forecasting.

A. Complex Fuzzy Sets

Ramot et al. proposed the complex fuzzy membership as a complex value drawn from a unit disk as follows [1]:

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

where r_s is the magnitude in polar form and is constrained to be ≤ 1, and w_s is the phase. Ramot et al. then proposed the first complex fuzzy logic based on IF-THEN rules and used algebraic product as fuzzy implication. He also defined the property of rule interference, and suggersted using a weighted complex sum of rule firing strengths to implement it [2]. Dick studied complex fuzzy logic by considering phase and magnitude simultaneously and showed algebraic product can be a candidate conjugate operator in this complex fuzzy logic [11]. He also proposed using sinusoidal membership function in order to model "approximately periodic" phenomena. Tamir et al. proposed "pure" complex fuzzy sets whose membership grades are drawn from the unit square of the complex plane [12]. Complex fuzzy logic for "pure" complex fuzzy sets was studied in [13-15]. Pythagorean membership degrees were studied in [16] and complex Atanassov's intuitionistic fuzzy sets were studied in [17]. Various operations for complex fuzzy sets were proposed in [18].

B. *Adaptive Neuro-Complex Fuzzy Inference System*

ANCFIS is a 6-layer network proposed in [3] as the first machine learning algorithm based on complex fuzzy systems and urle interference. The system is based on the well-known ANFIS system [6] with major differences in their architectures. The first difference is using sinusoidal functions as membership functions in ANCFIS:

$$r(\theta) = d\sin(a(\theta = x) + b) + c \tag{2}$$

where r and θ are the magnitude and phase of the membership grade, respectively. The parameters $\{a, b, c, d\}$ manipulate the shape of a sine wave; they change the frequency of the wave, cause a phase shift, shift the wave vertically and change its amplitude, respectively. The following conditions must be satisfied for the parameters to keep the membership grade in the unit disk [3]:

$$0 \leqslant d + c \leqslant 1, \qquad 1 \geqslant c \geqslant d \geqslant 0 \tag{3}$$

The sinusoidal membership function used in ANCFIS architecture makes it a candidate for time series prediction [4, 5, 19] based on the well-known ability of a Fourier series to approximate any periodic function; many time series do exhibit repeated behaviors, and this is what ANCFIS excels at modeling.

The second difference is in the input representation for ANCFIS. ANFIS and other systems take a sequence of previous measurements, and concatenate them into a feature vector (the "lagged" representation). Each observation is thus treated as a measurement from an orthogonal dimension. In ANCFIS, a window of previous observations is considered as one input to the systems. This preserves the time order and sampling rate between the observations, which will be modeled by membership phase. This leads directly to a third difference, the use of the complex convolution as a fuzzification operator, and the subsequent use of complex-valued signals throughout much of the ANCFIS network. In [19] we considered the possibility of subsampling input windows. We found that the method of delay embeddings (see section II.E of the current paper) defines a subsampling strategy that was particularly effective. This technique is usually employed to create lagged inputs, but we found that treating the resulting "delay vectors" as ANCFIS input windows instead worked extremely well.

The fourth difference is the addition of a new layer before the fourth layer in the original ANFIS architecture. This layer is for applying the concept of "rule interference" proposed in [2]; the idea is that the firing strengths of complex fuzzy rules may interfere constructively or destructively with each other based on the phase of the membership grade. In ANCFIS, the dot product operation is used in this layer.

The six-layer ANCFIS architecture is as follows [3]:

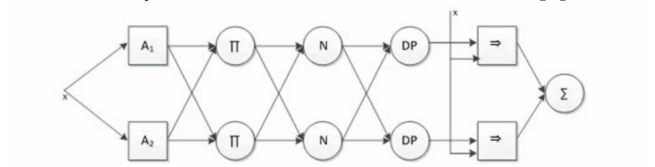

Fig. 1. ANCFIS architecture for univariate time series problems with two membership functions

- Layer 1: The membership grade of each input vector is calculated by applying complex convolution between an input vector and samples of the sinusoidal membership function, defined in (2), taken over one period and matched to the samples in the input vector. The results are normalized to the unit disk by the Elliot function.
- Layer 2: The firing strength of each rule is computed using the algebraic product.
- Layer 3: The firing strength of each rule is normalized.
- Layer 4: Rule interference is implemented by taking the dot product between the current rule's firing strength and the complex sum of all other rule's firing strengths.
- Layer 5: A linear consequent function is applied as in ANFIS, with parameters determined by a Kalman filter.
- Layer 6: All incoming signals are summed to obtain the final network output.

ANCFIS employs a hybrid learning system similar to ANFIS where in the forward pass, consequent parameters are determined by a Kalman filter and in the backward pass, antecedent parameters are obtained by back-propagation using gradient descent until the first layer. As there is no closed-form solution for the derivative of network error with respect to the first layer parameters, a derivative free optimization algorithm is used – the Variable Neighbourhood Chaotic Simulated Annealing (VNCSA) algorithm, developed in [3].

Other complex fuzzy machine-learning systems proposed in the literature include an online version of ANCFIS [20] and the family of Complex Neuro-Fuzzy System (CNFS) architectures [21-30] which uses complex Gaussian membership functions.

C. *Extreme Learning Machines*

For N pair of samples $(\mathrm{x}_i, \mathrm{t}_i)$, a single hidden-layer feed-forward neural network with $\tilde{N}$ hidden layers is defined as [7]

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i . \mathrm{x}_j + b_i\right) = \mathrm{t}_\mathrm{j}, \quad j = 1, \dots, N \tag{4}$$

where $\mathrm{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathcal{R}^n$ is the input vector, $\mathrm{t}_i = [t_{i1}, t_{i2}, \dots, t_{in}]^T \in \mathcal{R}^m$ is the outputs of the network. w_i is a vector of weights for the *i*-th hidden neuron, b_i is the bias of the *i*-th neuron, β_i is the weight from the *i*-th neuron to the *j*-th output neuron, and g is the activation function.

To describe the learning process in ELM, consider (4) as:

$$H\beta = T \tag{5}$$

where

$$H(w_1, \dots, w_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathrm{x}_1, \dots, \mathrm{x}_{\tilde{N}}) = \tag{6}$$

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

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \tag{6}$$

$$T = \begin{bmatrix} \mathrm{t}_1^T \\ \vdots \\ \mathrm{t}_N^T \end{bmatrix}_{N \times m} \tag{7}$$

In the first step of ELM, random values are assigned to the input weights (w_i) and biases (b_i) from a continuous probability distribution; H is thus calculated. Then, the output weights (β) are estimated as [7]:

$$\hat{\beta} = H^{\dagger} T \tag{8}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of H. This network can approximate any function as long as the activation function, g, is infinitely differentiable. Thus, logistic, exponential, sine and cosine functions are all candidate activation functions.

For incremental learning, Online Sequential ELM (OS-ELM) was proposed based on the recursive least square (RLS) algorithm [8]. In this network, output weights (β) are estimated as:

$$\hat{\beta} = (H^T H)^{-1} H^T T \tag{9}$$

The OS-ELM algorithm has two steps [8]:

- *Boosting Phase*: In this step, a small initial training set is considered $\{(\mathrm{x}_i, \mathrm{t}_i) | \mathrm{x}_i \in \mathcal{R}^n, \mathrm{t}_i \in \mathcal{R}^m, \ i = 1, \dots, \tilde{N}\}$, and random input weights and biases are assigned. Then $H_0 = [h_1, \dots, h_{\tilde{N}}]^T$ where $h_i = [g(w_1.\mathrm{x}_i + b_1), \dots, g(w_{\tilde{N}}.\mathrm{x}_i + b_{\tilde{N}})]^T$ is calculated. Then, initial output weights are estimated as: $\beta^{(0)} = M_0 H_0^T T_0$ where $M_0 = (H_0^T H_0)^{-1}$ and $T_0 = [t_1, \dots, t_{\tilde{N}}]^T$.
- *Sequential learning phase*: In this step, for any coming sample $(\mathrm{x}_i, \mathrm{t}_i)$, $h_{k+1} = [g(w_1.\mathrm{x}_i + b_1), \dots, g(w_{\tilde{N}}.\mathrm{x}_i + b_{\tilde{N}})]^T$ is calculated and the output weights are obtained based on the RLS algorithm as:

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}} \tag{10}$$

$$\beta^{(k+1)} = \beta^{(k)} + M_{k+1} h_{k+1} (t_i^T - h_{k+1}^T \beta^{(k)}) \tag{11}$$

Other ELM architecture have been proposed including fully complex ELM [31], incremental ELM [32] and pruning ELM [33]. A review of ELM architectures can be found in [34].

D. RBFN and ARFIMA

RBFN is a two layer neural network consisting of a hidden layer with nonlinear transformation and a linear output layer. The network follows a mapping from inputs to output according to [35]:

$$f_r(x) = \lambda_0 + \sum_{i=1}^{n_r} \lambda_i \varphi(\| x - c_i \|) \tag{12}$$

where $x \in R^n$ is the input vector, $\|.\|$ denotes the Euclidean norm, λ_i are the weights $0 \leqslant i \leqslant n_r$ (usually determined via least-squares estimation) $c_i \in R^n$ are the RBF centers (usually determined via clustering), $1 \leqslant i \leqslant n_r$, $n_r$is the number of centers, and $\varphi(.)$ is a kernel function, often a Gaussian [35]:

$$\varphi(x) = exp\left(\frac{-\| x - c_i \|^2}{\beta^2}\right) \tag{13}$$

where β is the spread parameter.

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

$$\left(1 - \sum_{i=1}^{p} \alpha_i L^i\right).(1 - L)^d X_t = \left(1 + \sum_{j=1}^{q} \beta_i L^i\right).\varepsilon_t \tag{14}$$

where p is the order of the auto-regression part of the model, α_i is the auto-regression coefficient of the i-th lag, L is the lag operator, d is the order of differencing, X_t is the t-th element of a time series, q is the order of the moving-average portion of the model, β_i is the moving-average coefficient of the i-th lag, and ε_t is the t-th error term; errors are assumed to be independent, identically distributed random variables drawn from a zero-mean normal distribution [36]. In the ARIMA model, p, d, and q are integers, but in ARFIMA model, d is allowed to take non-integer values [37]. This makes ARFIMA models particularly useful for self-similar time series, as the parameter d is directly related to the well-known Hurst parameter, which quantifies self-similarity.

E. Delay Embedding of a Time Series

Delay embedding is a commonly used approach for converting a time series into a state space that is at least equivalent to the original state space of the underlying system. Based on Taken's theorem [38], if the delay and dimension parameters selected are sufficient, the reconstructed state space is related to the original one by a smooth, invertible mapping. Thus, we can use the system's trajectory through the reconstructed state space to predict its future evolution.

A delay vector of a univariate time series is given by [39]:

$$S_n = (s_{n-(m-1)\tau}, s_{n-(m-2)\tau}, \dots, s_n) \tag{15}$$

where S_n is the delay vector with dimension m, and τ specifies the delay between successive observations; both of the parameters are determined heuristically. τ can be estimated by the first zero of the autocorrelation function, or the first minimum of the time-delayed mutual information function [38], and m is often estimated by the false nearest neighbors technique [40].

III. ANCFIS-ELM

In this section, we will develop a new network architecture based on ANCFIS and OS-ELM, which we will refer to as ANCFIS-ELM. Our objective in this design is to improve both the learning speed of ANCFIS (i.e. the execution time of the ANCFIS learning algorithm), while also maintaining or even improving its accuracy and parsimony.

Our approach is to replace the most time-consuming part of the ANCFIS learning algorithm – the optimization of the layer-

1 parameters by VNCSA – with the random-selection approach used in ELM. Thus, we will set the parameters of each CFS randomly in layer 1. This also eliminates the need for gradient descent, or indeed a backwards pass at all; the CFS parameters are the only ones adjusted in the ANCFIS backwards pass.

With that decision made, it is appropriate to stop and consider whether the other layers of ANCFIS are also needed. Let us first consider the layers accepting complex-valued inputs, which are layers 1-4 in ANCFIS. Layer 2 nodes take the algebraic product of their inputs, implementing a complex fuzzy conjunction to compute the firing strength of a complex fuzzy inference rule. The third layer normalizes this firing strength. The fourth layer then executes the operation of rule interference, via the dot product. Each of these steps is essential to the inference process defined by ANCFIS, and the models induced from a dataset would be profoundly changed if any of these layers were eliminated.

We next consider the remaining layers. In layer 5, a linear function of the input vector is computed in each node. These are the consequents of each rule; a sum of these consequents, weighted by the firing strengths of each rule (after the rule interference process) is taken in Layer 6. These operations seem less significant; while the consequent parameters grant more degrees of freedom to the network, a linear function is quite a limited model. The research into TSK fuzzy systems (which ANFIS mimics [6]) shows that a linear function is mathematically unnecessary; a TSK fuzzy systems whose consequents are merely constants remains a universal approximator [41]. Thus, we judge it would be possible to eliminate layer 5, and replace is with a weighted connection between each layer-4 node and the final summation node. These connection weights implement the constant-valued consequent function, and also match the ELM paradigm of optimizing only the weights on the output neurons of the network.

With these design decisions made, we can define the ANCFIS-ELM architecture. It is a five-layer network, with layers 1 to 4 identical to ANCFIS; however, the antecedent parameters $\{a, b, c, d\}$, are randomly drawn from a uniform distribution. Note, however, that the conditions in Eqs. (2) and (3) must be satisfied, and so each draw will be repeated until they are. Layer 4 connects directly to the output layer, which sums its inputs. These connections are weighted by the consequent parameters β_i.

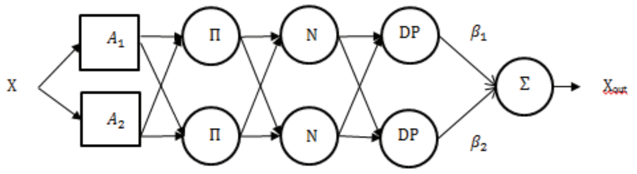

Fig. 2. ANCFIS-ELM network for a univariate time series with two membership functions.

The weights (β_i), are updated as in the sequential learning phase in OS-ELM. We do not perform the boosting phase as in the OS-ELM; instead initial output weights are determined as [35]:

$$\beta^{(0)} = 0 \quad (16)$$

$$M_0 = \lambda^{-1} I \quad \lambda \text{ is a small positive constant} \quad (17)$$

This network is thus trained by a fast, incremental algorithm, and is thus appropriate for data stream mining.

IV. Experimental Evaluation

In order to evaluate the ANCFIS-ELM algorithm, we have selected the software reliability growth modeling (SRGM) problem, which we treat as an instance of time series forecasting. Time series are known to be a subset of stream data, and SRGM is a particularly high-value example.

A. Datasets

The four time series we are working on in this paper are ODC1, ODC4, Android and Mozilla. The ODC1 and ODC4 data sets have 1207 and 2008 data points, respectively. These data sets were collected by IBM Corporation during their Orthogonal Defect Classification project [42]. These data sets are not originally interfailure times, but based on a recommendation in [42], we convert them to interfailure times by assuming that the failures occur randomly during day.

The Android data set has 20168 data points extracted from the bug reports and changes of the 2012 Mining Software Repositories Challenge dataset. To obtain the interfailure times, the date/time stamp of each bug report is processed and the time between the bugs are calculated with accuracy of one second [43].

The Mozilla data set has 86077 data points derived from the Bugzilla defect-tracking database used by the Mozilla project. The interfailure times are again obtained by processing the date/time stamps of "confirmed" bugs in the dataset between 1999 and 2003 [43].

A. Methodology

We follow a chronologically-ordered single-split experimental design. Each of our four datasets is split so that the earlier 2/3rds of the data forms a training set, and the final 1/3 is the test set. Then, input vectors are derived from the training and testing data set by using the delay embedding approach described in our Literature Review via the Tisean software package [39].

The results are compared against RBFN and ARFIMA. RBFN uses the same input vectors as ANCFIS-ELM and is implemented in the WEKA data-mining package [44]. For ARFIMA, we use the original training and testing set; the ARFIMA implementation is found in the "forecast" and "fracdiff" packages in R [45, 46].

The performance comparison is done through Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \quad (18)$$

To evaluate the statistical significance of our results, two tests are used; first, we use the paired Wilcoxon signed-rank test to compare the results of ANCFIS-ELM with RBFN and ARFIMA pairwise on each data set. Then, we calculate the effect size of those differences with Cohen's *d*.

The Wilcoxon is a non-parametric test for a difference between two distributions. The absolute error $|x_i - y_i|$ and the sign of the error $sgn(x_i, y_i)$ $i = 1, \dots, n$ of the predictions x_i and y_i are calculated for all *n* samples in each of the datasets, where we compare the predictions of ANCFIS-ELM against either RBFN or ARFIMA. Samples are ranked based on their absolute error, then the test statistic is obtained as:

$$W = \sum_i sgn(x_i - y_i). R_i \tag{19}$$

where R_i is the rank associated with $|x_i - y_i|$. The null hypothesis is that the ranks are symmetrically distributed around 0. Meaning that there is no difference in the prediction errors of the algorithms being compared. The distribution of this test statistic is known but complex; for a large number of samples, it can be approximated by the Normal distribution [47].

The effect size of each data set is computed by Cohen's *d* which measures the strength of difference between two sampled populations as:

$$d = \frac{\bar{x} - \bar{y}}{s} \tag{20}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of population *x* and *y*, and the pooled sample standard deviation *s* is calculated by

$$s = \sqrt{\frac{(n-1)s_x^2 + (n-1)s_y^2}{2n-2}} \tag{21}$$

where *n* is the sample size of two sets having equal size, s_x^2 is the variance of *x* and s_y^2 is the variance of *y*. Based on Cohen's suggestion in [48], an effect size less than 0.2 indicates no real relationship, $0.2 \leq d < 0.5$ indicates a weak effect, $0.5 \leq d < 0.8$ is a moderate effect, and $0.8 \leq d$ indicates a large effect.

C. Experimental Results

Table I presents the embedding delay and dimensions obtained for ODC1, ODC4, Android and Mozilla:

TABLE I. EMBEDDING DELAY AND DIMENSION FOR THE DATA SETS

	Dimension	Delay
ODC1	3	3
ODC4	9	2
Android	18	2
Mozilla	16	6

Table II presents prediction errors in terms of MAE for the three algorithms. Tables III and IV show the Wilcoxon test and effect size results for ANCFIS-ELM vs. ARFIMA and ANCFIS-ELM vs. RBFN, respectively.

Our results in Tables II-IV show that, although the differences between ANCFIS-ELM and ARFIMA are always significant, and those between ANCFIS-ELM and RBFN are significant on all datasets except ODC4, the effect size test indicates no meaningful differences with either algorithm on any dataset. We believe that this means the Wilcoxon test results are due to large sample size effects, and the effect size tests reveal a statistical tie between our new method and the existing ones.

TABLE II. PREDICTION ERRORS OF RBFN, ARFIMA AND ANCFIS-ELM

	RBFN	ARFIMA	ANCFIS-ELM
ODC1	0.028	0.032	0.029
ODC4	0.033	0.031	0.032
Android	0.066	0.064	0.062
Mozilla	0.0267	0.0274	0.0268

TABLE III. STATISTICAL SIGNIFICANCE OF DIFFERENCES BETWEEN ANCFIS-ELM AND ARFIMA

	W-Statistic	*p*-Value	Effect Size
ODC1	3.3e+4	0.0146	0.016
ODC4	1.3e+5	2.2e-7	0.036
Android	1.02e+7	7.1e-10	0.029
Mozilla	1.9e+8	7.8e-12	0.015

TABLE IV. STATISTICAL SIGNIFICAN OF DIFFERENCES BETWEEN ANCFIS-ELM AND RBFN

	W-Statistic	*p*-Value	Effect Size
ODC1	4.4e+4	0.05	0.019
ODC4	1.1e+5	0.76	0.004
Android	8.2e+6	2.2e-16	0.046
Mozilla	2.2+8	2.2e-16	0.003

V. CONCLUSION

We have proposed a new machine learning algorithm based on complex fuzzy sets. The system is a hybrid of ANCFIS and ELM. This system is able to solve one of the main problems of ANCFIS which is its learning speed. Comparison of ANCFIS-ELM with the well-known learning algorithms, RBFN and ARFIMA, shows that it is competitive with them as their results do not show any statistically meaningful difference from ANCFIS-ELM. In future work, we are going to extend and apply ANCFIS-ELM to the problem of stream data mining.

VI. REFERENCES

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

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

[3] Z. Chen, *et al.*, "ANCFIS: A neurofuzzy architecture employing complex fuzzy sets," *Fuzzy Systems, IEEE Transactions on,* vol. 19, pp. 305-322, 2011.

[4] O. Yazdanbaksh, *et al.*, "Predicting solar power output using complex fuzzy logic," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 2013, pp. 1243-1248.

[5] O. Yazdanbakhsh and S. Dick, "Multi-variate timeseries forecasting using complex fuzzy logic," in *Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing*

(WConSC), 2015 Annual Conference of the North American, 2015, pp. 1-6.

[6] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 23, pp. 665-685, 1993.

[7] G.-B. Huang*, et al.*, "Extreme learning machine: theory and applications," *Neurocomputing,* vol. 70, pp. 489-501, 2006.

[8] G.-B. Huang*, et al.*, "On-Line Sequential Extreme Learning Machine," *Computational Intelligence,* vol. 2005, pp. 232-237, 2005.

[9] C. Jones, *Software Assessments, Benchmarks, and Best Practices*. New York, NY, USA: Addison-Wesley, 2000.

[10] M. Levinson. (2001, October 15) Let's stop wasting $78 billion per year. *CIO Magazine*.

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

[12] D. E. Tamir*, et al.*, "A new interpretation of complex membership grade," *International Journal of Intelligent Systems,* vol. 26, pp. 285-312, 2011.

[13] D. E. Tamir and A. Kandel, "Axiomatic theory of complex fuzzy logic and complex fuzzy classes," *International Journal of Computers, Communications & Control VI (3),* 2011.

[14] D. E. Tamir*, et al.*, "Discrete complex fuzzy logic," in *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, 2012, pp. 1-6.

[15] D. E. Tamir*, et al.*, "The Theory and Applications of Generalized Complex Fuzzy Propositional Logic," in *Soft Computing: State of the Art Theory and Novel Applications*, ed: Springer, 2013, pp. 177-192.

[16] R. Yager, "Pythagorean membership grades in multi-criteria decision making," *Fuzzy Systems, IEEE Transactions* vol. 22, pp. 958-965, 2014.

[17] A. R. Salleh, "Complex intuitionistic fuzzy sets," in *AIP Conference Proceedings*, 2012, p. 464.

[18] G. Zhang*, et al.*, "Operation properties and δ-equalities of complex fuzzy sets," *International Journal of Approximate Reasoning,* vol. 50, pp. 1227-1249, 2009.

[19] O. Yazdanbakhsh and S. Dick, "Time-series forecasting via complex fuzzy logic," in *Frontiers of Higher Order Fuzzy Sets*, ed: Springer, 2015, pp. 147-165.

[20] S. Aghakhani and S. Dick, "An on-line learning algorithm for complex fuzzy logic," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1-7.

[21] C. Li, "Adaptive image restoration by a novel neuro-fuzzy approach using complex fuzzy sets," *International Journal of Intelligent Information and Database Systems,* vol. 7, pp. 479-495, 2013.

[22] C. Li and F. Chan, "Complex-Fuzzy Adaptive Image Restoration–An Artificial-Bee-Colony-Based Learning Approach," in *Intelligent Information and Database Systems*, ed: Springer, 2011, pp. 90-99.

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

[24] C. Li and T. Chiang, "Complex Neuro-Fuzzy ARIMA Forecasting—A New Approach Using Complex Fuzzy Sets," 2011.

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

[26] C. Li and T.-W. Chiang, "Function approximation with complex neuro-fuzzy system using complex fuzzy sets–a new approach," *New generation computing,* vol. 29, pp. 261-276, 2011.

[27] C. Li and T.-W. Chiang, "Complex Fuzzy Computing to Time Series Prediction A Multi-Swarm PSO Learning Approach," in *Intelligent Information and Database Systems*, ed: Springer, 2011, pp. 242-251.

[28] C. Li and T.-W. Chiang, "Intelligent financial time series forecasting: A complex neuro-fuzzy approach with multi-swarm intelligence," *International Journal of Applied Mathematics and Computer Science,* vol. 22, pp. 787-800, 2012.

[29] C. Li*, et al.*, "A novel self-organizing complex neuro-fuzzy approach to the problem of time series forecasting," *Neurocomputing,* 2012.

[30] C. Li*, et al.*, "Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling," *Neurocomputing,* vol. 94, pp. 121-139, 2012.

[31] G.-B. Huang*, et al.*, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing,* vol. 71, pp. 576-583, 2008.

[32] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing,* vol. 71, pp. 3460-3468, 2008.

[33] H.-J. Rong*, et al.*, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing,* vol. 72, pp. 359-366, 2008.

[34] G.-B. Huang*, et al.*, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics,* vol. 2, pp. 107-122, 2011.

[35] S. S. Haykin, *Neural networks and learning machines* vol. 3: Pearson Education Upper Saddle River, 2009.

[36] T. C. Mills, *Time Series Techniques for Economists*. Boston, MA, USA: Cambridge University Press, 1990.

[37] C. W. Granger and R. Joyeux, "An introduction to long‐memory time series models and fractional differencing," *Journal of time series analysis,* vol. 1, pp. 15-29, 1980.

[38] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*. New York, NY, USA: Cambridge University Press, 1997.

[39] R. Hegger*, et al.*, "Practical implementation of nonlinear time series methods: The TISEAN package," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 9, pp. 413-435, 1999.

[40] M. B. Kennel*, et al.*, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Physical Review A,* vol. 45, p. 3403, 1992.

[41] J. L. Castro, "Fuzzy logic controllers are universal approximators," *IEEE T. Systems, Man and Cybernetics,* vol. 25, pp. 629-635, 1995.

[42] M. R. Lyu, *Handbook of Software Reliabiity Engineering*. New York, NY: McGraw-Hill, 1996.

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

[44] M. Hall*, et al.*, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter,* vol. 11, pp. 10-18, 2009.

[45] R. J. Hyndman and Y. Khandakar, "Automatic time series for forecasting: the forecast package for R," Monash University, Department of Econometrics and Business Statistics2007.

[46] M. M. Maechler, "Package 'fracdiff'," 2009.

[47] S. Siegel, *Non-parametric statistics for the behavioral sciences*. New York, NY, USA: McGraw-Hill, 1956.

[48] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences, 2nd Ed.* Mahwah, NJ, USA: Lawrence Erlbaum Associates., 1988.