International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-2, December 2013

Effect of Normalization Techniques on Univariate Time Series Forecasting using Evolutionary Higher Order Neural Network

Sibarama Panigrahi, H. S. Behera

Abstract*— Over the last few decades, application of higher order neural networks (HONNs) to time series forecasting have shown some promise compared to statistical approaches and traditional neural network (NN) models. However, due to several factors, to date, a consistent HONN performance over different studies has not been achieved. One such factor is preprocessing of time series before it is fed into HONN models. Normalization is one of the important pre-processing strategies which have a significant impact on forecast accuracy. Despite its great importance, there has been no general consensus on how to normalize the time series data for HONN models. This paper investigates how to better normalize the univariate time series for HONN models especially, the Pi-Sigma network (PSN). For this five different normalization technique (Min-Max, Decimal Scaling, Median, Vector and Z-Score) are used to normalize four univariate time series and corresponding forecast accuracy are measured using an evolutionary Pi-Sigma network. Results show that forecast accuracy using HONN models depends on the normalization technique being used. It is also noted that with PSNs, decimal scaling and vector normalization techniques provide statistically meritorious results compared to other normalization techniques.

Index Terms*— Normalization, Higher Order Neural Networks, Pi-Sigma Network, Differential Evolution, Time Series Forecasting.

I. INTRODUCTION

Univariate time series forecasting (TSF) is the process of predicting the future outcomes based solely on past observations of a particular time series. Univariate TSF assists strategic decision making under uncertainty and has been successfully applied in the areas such as economics, finance, management, engineering etc. Over the past few decades, TSF were performed predominantly using various artificial neural networks (ANN) models because of its various unique characteristics. However, compared to traditional neural networks (NNs), higher order neural networks (HONNs) possess certain advantageous characteristics including: 1) stronger approximation capability; 2) faster convergence property; 3) greater storage capacity; and 4) higher fault tolerance capability. Due to these characteristics of HONN models, many studies have shown better performance than traditional NNs and other statistical models.

Despite of such better performance, HONNs have not been established as a valid and reliable tool to forecast a variety of time series. Various factors contribute to this inconsistent performance. One of the major factors is data pre-processing.

Data pre-processing is the process of analyzing and transforming the input and output variables to highlight the important relationships by reducing noise. It flattens the distribution of variables and assists neural networks in learning relevant patterns. In TSF, data pre-processing generally includes identification and treatment of various patterns (outliers, trend and seasonal components) and finally normalization of time series. Data normalization has a significant impact on the performance of any model because the sole purpose of data normalization is to guarantee the quality of the data before it is fed to any model. In literature, few studies were done in evaluating the sensitivity of the model's performance to different normalization techniques. Wan [1] used three different normalization techniques to evaluate the sensitivity of the proposed modeling scheme towards automatic time series forecasting and found that the modeling scheme is insensitive to Min-Max ([0,1]); Min-Max ([-1,1]); and mean and standard deviation normalization techniques. However, Mustaffa et al. [2] evaluated the sensitivity of min-max, decimal scaling and Z-Score normalization techniques in predicting future dengue outbreak using LS-SVM and neural network model (NNM) and suggested that both the models achieve better accuracy using decimal point normalization. Similarly, Eftekhary et al. [3] conducted a study on ranking five normalization techniques based on improved accuracy of support vector machine (SVM) using data envelopment analysis (DEA) method and concluded that non-monotonic normalization method performs better. Jayalakshmi et al. [4] suggested that: data classification using neural networks was dependent on the normalization methods and min-max normalization provides better classification accuracy than other methods. Nayak et al. [5] used five normalization techniques for predicting stock index of Bombay stock exchange using neuro-genetic models and suggested that these models are sensitive to different normalization techniques. In addition also suggested that not a single normalization technique is full proof for all models rather different normalization techniques should be tried for better results. Literature revealed that normalization techniques have a significant impact on the performance of a model and the normalization technique should be chosen based on the problem and model in hand.

Manuscript published on 30 December 2013.
* Correspondence Author (s)
Sibarama Panigrahi*, Computer Science and Engineering, MIRC Lab, Majhighariani Institute of Technology and Science, Rayagada, India.
H. S. Behera, Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla (Odisha), India.

Despite the great importance of normalization technique and better performance of HONN models in forecasting across various disciplines, there has been no study to evaluate the sensitivity of normalization techniques for forecasting accuracy using HONNs. Motivated by this need, this paper attempts to evaluate the effect of various normalization techniques on univariate TSF using HONN models.

The rest of the paper is organized as follows. Section 2 briefly describes different normalization techniques, Pi-Sigma neural network. Section 3 explains the methodology used for univariate TSF using evolutionary Pi-Sigma neural network. Extensive Simulation results are presented in Section 4. Finally, conclusions are drawn in Section 5.

II. PRELIMINARIES

A. Normalization

Normalization is a pre-processing strategy that transforms the data points into a small pre-specified range generally 1 to -1 or 0. In this study five popular normalization techniques (such as: decimal scaling, median, min-max, vector and z-grade) are considered. All the techniques are discussed briefly using the following notation.

Consider a time series T= T_1, T_2, T_3, …… T_k and the normalized series N= N_1, N_2, N_3, … N_k.

Decimal Scaling

In this method the decimal point of every data point move 'P' number of places towards the left, where 'P' is the number of digits of the maximum absolute value of the dataset.

Mathematically

$$N_i = \frac{T_i}{10^P}$$

where i = 1,2,…k and P = length(max(| T |))

Median

In this method all data points are normalized by the median of the original series. This method is useful when ratio between two hybridized samples need to be computed or when performing distributions.

Mathematically

$$N_i = \frac{T_i}{\text{median}(T)}$$

Min-Max Normalization

This method performs a linear transformation of data values based on the maximum (Max_T) and minimum (Min_T) value of the original series (dataset). This method maps the data values from a range [Min_T, Max_T] to a range [Min_N, Max_N].

Mathematically

$$N_i = Min_N + \frac{T_i - Min_T}{Max_T - Min_T} \times (Max_N - Min_N)$$

Min-Max normalized data values preserve the relationship of the original data values. However, using this technique in time series forecasting, a problem may occur if any of the out-of-sample (to be predicted) data points is out of the range [Min_T, Max_T]. To overcome this problem, few authors [6, 7] used modified min-max normalization by considering a practical increment and decrement on maximum and minimum limits of known time series respectively. However, by how much factor the practical increment and/or decrement will be considered is again a big question and erroneous consideration of factor value may cause significant information loss leading to loss of quality in learning technique [8, 9].

Vector Normalization

In this method the time series is considered as a single vector and normalization is carried out by dividing each data value by the root sum squared value of the original series.

Mathematically

$$N_i = \frac{T_i}{\sqrt{\sum_{j=1}^{k} T_j^2}}$$

Z-Score

The data values are normalized using the mean (μ_T) and standard deviation (σ_T) of the original data values (series). This method is also called Zero-Mean normalization because after this normalization the mean of normalized series becomes zero.

Mathematically

$$N_i = \frac{T_i - \mu_T}{\sigma_T}$$

This method is applicable to any stationary series even if the minimum and maximum value of out of sample data values is unknown. However, this method may be sensitive to small value of σ_T and does not perform well with non-stationary time series (as mean and standard deviation vary over time).

B. Pi-Sigma Network

The Pi–Sigma Network was introduced by Shin and Ghosh [10] and shown greater promise in solving several difficult tasks such as zeroing polynomials [11] and polynomial factorization [12]. PSN (e.g. As shown in Figure 1) is a special type of feed forward neural network which has a single hidden layer of summing units with linear transfer function and an output layer of product units with non-linear transfer function. Thus, the output of PSN is obtained by making product of the sum of the inputs and passing it through a nonlinear function. The weights between input and hidden layer are adapted during the learning process whereas the weights between hidden and output layer are fixed to one. Therefore, with one layer of trainable weights the training time of PSN reduces drastically [10, 13]. Consider a PSN with NOIN (number of input neurons) input neurons, NOHN (number of hidden neurons) hidden neurons and NOON (number of output neurons) output neurons. For such a PSN the number of trainable weights is (NOIN+1) × NOHN considering each hidden neuron is associated with NOIN inputs and a bias unit.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The output of the PSN is computed by making product of the output of NOHN hidden units and passing it to a nonlinear function, which is defined as follows:

$$Y = \sigma\left(\prod_{j=1}^{NOHN} h_j\right)$$

$$h_j = \sum_{i=1}^{NOIN} (w_{ij} x_i + w_{j0})$$

Where h_j is the output of the j^{th} hidden unit, σ a nonlinear transfer function, w_{ij} is the weight connecting i^{th} input neuron with the j^{th} hidden neuron and w_{j0} is the bias associated with j^{th} hidden neuron.

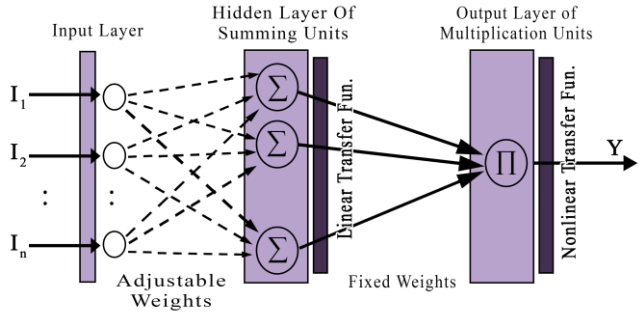

Fig.1. Architecture of a Typical Pi-Sigma Network

III. METHODOLOGY

The main goal of this paper is to find the best data normalization technique to forecast univariate time series using an evolutionary higher order neural network (HONN). For this the Pi-Sigma network (PSN) has been chosen as a computational model because, PSNs are not only computationally much more efficient than other HONN models but also manages to incorporate the capability of first orders HONN indirectly. However, in order to make a fair comparison among various normalization techniques a robust training algorithm for PSN is required. Shin et al. [13] used the back propagation algorithm for PSN training. But, the gradient based training methods often suffer from several shortcomings, including: 1) easily trapping to local minima; 2) slow convergence properties; 3) training performance is sensitive to initial values of its parameters. For this reason, few methods have been proposed to train PSN using evolutionary algorithms like differential evolution (DE) [14], particle swarm optimization (PSO) [14] and chemical reaction optimization (CRO) [15].

Pseudo Code of Methodology

1. Normalize the time series using a normalization technique.
2. Transform the normalized time series into patterns using sliding window method (Depends on the number of input neurons and forecasting horizon)
3. Divide the normalized patterns into three segments Train (70%), Validation (15%) and Test (15%) patterns.
4. Train the PSN using DE-ANNT+ method considering the train and validation patterns.
   a. Randomly Initialize the population with each chromosome representing a weight-set of PSN having length ([NOIN + 1] × NOHN i.e. NOIN×NOHN weights connecting between NOIN inputs and NOHN hidden neurons and NOHN bias units for NOHN hidden neurons), and each gene representing a weight of PSN.
   b. Calculate the fitness (mean square on train set) of each chromosome.
   c. Generate multiple scale factors (from uniform distribution between 0-2) to produce multiple mutant vectors after Mutation.(Note: here DE/rand/1/bin mutation strategy is followed.)
   d. Select the best mutant vector.
   e. Apply binary crossover between the target vector and best mutant vector to generate the trial vector.(Note: Unlike DE-ANNT+, the crossover probability Cr= Gaussianrand (0.6, 0.1), where Gaussianrand is a function which generates random numbers from Gaussian distribution with mean=0.6 and standard deviation=0.1, It is regenerated if the random number falls out of the range[0,1])
   f. Perform Selection between trial vector and target vector
   g. Termination Criteria: Use the fittest chromosome of the population to perform single-step-ahead (SSA) or multiple-step-ahead (MSA) forecasting on validation set. If the forecast accuracy on validation set of present generation best chromosome performs worse than that of previous generation then terminate and go to step-5 otherwise go to step-4.c.
5. Obtain the output of PSN on the test set using the optimal weight set obtained from step-4 for the time series.
6. De-normalize the output of PSN to obtain the actual forecasts.

Measure the forecast accuracy on Train, Validation and Test patterns.

In this paper we have considered DE with multiple trial vectors for ANN training (DE-ANNT+) algorithm [16] (with a little modification) to train the PSN because of its efficiency. Slowki proposed DE-ANNT+ [16] for ANN (multilayer perceptron model) training. In the DE - ANNT+ algorithm, in each generation and for each target vector multiple mutant vectors are generated. The best mutant vector is selected for crossover with target vector to produce the trial vector. The selection between the target and trial vector is carried out in order to obtain the vector for the next generation. It was applied to classify the parity-p problem more efficiently than that of DE-ANNT, extended back propagation (EBP), EA-ANNT and results are comparable to that of Levenberg-Marquardt (LM) algorithm. It was also found that DE-ANNT+ takes less memory than LM algorithm. Interested readers may go through [16] to have a detail description regarding DE-ANNT+ method.

For every time series the above methodology is applied to perform SSA forecasting (forecasting horizon=1) and multiple five-step-ahead forecasting (forecasting horizon=5) using different normalization technique. The SSA is relatively easy and widely discussed in the literature. For MSA forecasting two approaches found in the literature, such as: direct and recursive.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

In direct MSA approach k-step-ahead forecasting is obtained directly without obtaining the intermediate 'k-1' forecasts whereas in recursive approach the k-step-ahead forecasting is obtained by recursively performing 'k' SSA forecasting recursively. In this paper direct approach is used for MSA forecasting because several studies [1, 17] have shown better performance of a direct MSA approach than the recursive MSA approach. The experimental setup and simulation results are shown in the next section.

IV. EXPERIMENTAL SETUP AND RESULTS

All simulations in this paper were implemented on a system with Intel ® core (TM) 2Duo E7500 CPU, 2.93 GHz with 2GB RAM and implemented using Matlab R2009a. Note that all PSNs considered in this paper have linear transfer function at the hidden layer and tan-hyperbolic activation function at the output layer. The number of input neurons and hidden neurons of PSN for a time series is obtained by plotting autocorrelation graph whereas the consideration of number hidden neurons of PSN for a time series is based on minimizing the degree of freedom of PSN, thus simpler networks that are capable of solving each problem can be chosen. All PSNs are trained using DE-ANNT+ [16] with population size 50, five numbers of trial vectors and initial value of each chromosome (representing a PSN weight-set) is initialized to uniform distributed random values drawn from a range [-1, 1]. All the normalization techniques considered in this paper transforms the time series into a range [-1, 1].

A. Time Series

For experimental analysis four univariate time series have been considered from the well-known Hyndman's time series data library exported from datamarket.com. These time series are: Wisconsin employment time series measured from January 1961 till October 1975, Monthly interest rates of Government Bond Yield 2-year securities Reserve Bank of Australia measured from January 1969 till September 1994, monthly milk production per cow in pounds measured from January 1962 till December 1975 and mean summer temperature in degree centigrade from 1781 till 1988.

B. Performance Measure

Literature revealed the use of four major types of measures to evaluate forecast accuracy such as: Scale-dependent, Percentage error, Relative error, and Scale-free. Hyndman and Koehler [18] conducted a comparative study on different measures of forecast accuracy. . Percentage errors have the advantage of being scale independent, so they are often used to compare forecast performance between different data series. Thus, the NN3 competition organizers have chosen the symmetric mean absolute percentage error (SMAPE) for model evaluation. Hence, for evaluating the forecast accuracy using various normalization techniques, SMAPE measure has been used which is defined as follows.

$$\text{SMAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i| - |F_i|}{(|Y_i| + |F_i|)/2} \times 100$$

Where Y_i and F_i are true and forecasted values respectively at i^{th} time point, N is the number of forecasting points.

C. Results and Discussion

In order to evaluate the effectiveness of normalization techniques on univariate time series forecasting using evolutionary PSN, ten independent simulations were carried out using each of the five normalization techniques for each of the four time series and corresponding 1-step-ahead and 5-step-ahead forecast accuracy was measured. To evaluate the sensitivity of the methodology to these five normalization techniques, we performed post-hoc analysis using duncan's multiple range test on the SMAPEs obtained from simulation. Table 1, Table 2, Table 3, Table 4 and Table 5 show the experimental results on four time series and their average using five different normalization techniques.

Table 1. SMAPE (%) on Wisconsin employment time series with PSN (6-3-1)

Normalization Technique	1-Step-Ahead Forecast			5-Step Ahead Forecast		
	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.
Decimal Scaling	**1.24 ± 0.006^{a}**	1.15 ± 0.053^{a}	1.47 ± 0.144^{b}	2.25 ± 0.012^{c}	**3.01 ± 0.086^{a}**	**5.40 ± 0.213^{a}**
Median	2.71 ± 0.024^{e}	6.85 ± 0.029^{d}	13.08 ± 0.011^{e}	2.99 ± 0.024^{d}	6.91 ± 0.035^{d}	13.17± 0.010^{b}
Min-Max	1.49 ± 0.012^{c}	2.63 ± 0.059^{b}	6.68 ± 0.094^{c}	**2.18 ± 0.025^{b}**	3.45 ± 0.103^{b}	7.50 ± 0.631^{a}
Vector	1.26 ± 0.022^{b}	**1.09 ± 0.108^{a}**	**1.33 ± 0.228^{a}**	2.25 ± 0.011^{c}	3.04 ± 0.060^{a}	5.83 ± 0.347^{a}
Z-Score	1.91 ± 0.051^{d}	4.05 ± 0.043^{c}	10.84 ± 0.044^{d}	2.09 ± 0.017^{a}	4.16 ± 0.058^{c}	10.90 ± 0.002^{b}

*Means within a column the same letter(s) are not statistically significant (p=0.05) according to Duncan's Multiple Range Test (SPSS V.16.0.1)

Table 2. SMAPE (%) on monthly interest rates government bond yield 2-year securities reserve bank of Australia time series with PSN (12-6-1)

Normalization Technique	1-Step-Ahead Forecast			5-Step Ahead Forecast		
	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.
Decimal Scaling	**3.03 ± 0.292^{a}**	3.73 ± 0.228^{ab}	**6.12 ± 0.537^{a}**	8.08 ± 0.388^{c}	8.77 ± 0.432^{a}	16.85 ± 0.721^{a}
Median	7.22 ± 0.119^{d}	7.69 ± 0.073^{c}	9.27 ± 0.389^{b}	9.93 ± 0.093^{d}	9.13 ± 0.189^{ab}	19.12 ± 0.754^{b}
Min-Max	3.48 ± 0.163^{b}	3.92 ± 0.198^{b}	6.77 ± 0.521^{a}	**7.11 ± 0.220^{a}**	9.27 ± 0.380^{bc}	17.76 ± 1.728^{a}
Vector	3.08 ± 0.375^{a}	**3.64 ± 0.266^{a}**	6.38 ± 0.676^{a}	8.04 ± 0.259^{c}	**8.75 ± 0.672^{a}**	**16.62 ± 0.751^{a}**
Z-Score	6.74 ± 0.169^{c}	7.69 ± 0.283^{c}	10.96 ± 1.643^{c}	7.50 ± 0.161^{b}	9.63 ± 0.716^{c}	17.32 ± 2.143^{a}

*Means within a column the same letter(s) are not statistically significant (p=0.05) according to Duncan's Multiple Range Test (SPSS V.16.0.1)

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Table 3. SMAPE (%) on monthly milk production per cow in pounds time series with PSN (6-3-1)

Normalization Technique	1-Step-Ahead Forecast			5-Step Ahead Forecast		
	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.
Decimal Scaling	3.19 ± 0.247[b]	3.97 ± 0.525[b]	4.61 ± 0.568[b]	2.78 ± 0.244[b]	**3.31 ± 0.131[a]**	3.67 ± 0.404[a]
Median	4.19 ± 0.114[e]	9.19 ± 0.584[c]	10.18 ± 0.651[d]	4.03 ± 0.179[c]	8.80 ± 0.689[b]	10.43 ± 0.808[b]
Min-Max	**2.04 ± 0.122[a]**	**3.11 ± 0.147[a]**	**3.65 ± 0.157[a]**	**2.21 ± 0.082[a]**	3.42 ± 0.327[a]	**3.35 ± 0.557[a]**
Vector	3.61 ± 0.320[c]	3.45 ± 0.383[ab]	3.74 ± 0.411[a]	3.08 ± 0.556[b]	3.67 ± 0.542[a]	3.40 ± 0.342[a]
Z-Score	3.85 ± 0.269[d]	8.70 ± 0.947[c]	9.26 ± 0.902[c]	4.12 ± 0.405[c]	8.34 ± 1.350[b]	9.75 ± 1.790[b]

*Means within a column the same letter(s) are not statistically significant (p=0.05) according to Duncan's Multiple Range Test (SPSS V.16.0.1)

Table 4. SMAPE (%) on mean summer temperature time series with PSN (12-6-1)

Normalization Technique	1-Step-Ahead Forecast			5-Step Ahead Forecast		
	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.
Decimal Scaling	3.71 ± 0.017[c]	**4.38 ± 0.054[a]**	**3.48 ± 0.037[a]**	3.74 ± 0.052[c]	4.32 ± 0.068[a]	**3.63 ± 0.102[a]**
Median	4.27 ± 0.031[e]	5.88 ± 1.966[b]	4.18 ± 0.094[b]	4.37 ± 0.555[e]	6.27 ± 2.053[b]	4.40 ± 0.002[b]
Min-Max	3.10 ± 0.085[b]	5.45 ± 0.523[b]	4.06 ± 0.284[b]	3.25 ± 0.068[b]	5.45 ± 0.280[b]	3.89 ± 0.281[a]
Vector	3.89 ± 0.009[d]	4.49 ± 0.065[a]	3.64 ± 0.034[a]	4.02 ± 0.011[d]	**4.32 ± 0.031[a]**	3.64 ± 0.018[a]
Z-Score	**2.25 ± 0.110[a]**	4.50 ± 0.428[a]	4.19 ± 0.510[b]	**2.19 ± 0.105[a]**	4.54 ± 0.309[a]	4.44 ± 0.539[b]

*Means within a column the same letter(s) are not statistically significant (p=0.05) according to Duncan's Multiple Range Test (SPSS V.16.0.1)

Table 5. Average SMAPE (%) on all time series.

Normalization Technique	1-Step-Ahead Forecast			5-Step Ahead Forecast		
	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.	Train Mean ± St.D.	Validation Mean ± St.D.	Test Mean ± St.D.
Decimal Scaling	2.79 ± 0.96[a]	3.31 ± 1.31[a]	3.92 ± 1.76[a]	4.21 ± 2.33[a]	4.95 ± 2.31[a]	7.39 ± 5.60[a]
Median	4.60 ± 1.66[c]	7.40 ± 1.57[c]	9.18 ± 3.27[c]	5.33 ± 2.75[b]	7.77 ± 1.61[c]	11.77 ± 5.39[b]
Min-Max	**2.53 ± 0.81[a]**	3.77 ± 1.12[a]	5.29 ± 1.49[b]	**3.69 ± 2.05[a]**	5.40 ± 2.43[a]	8.13 ± 5.95[a]
Vector	2.96 ± 1.06[a]	**3.17 ± 1.30[a]**	**3.77 ± 1.85[a]**	4.35 ± 2.27[ab]	**4.85 ± 2.35[a]**	**7.37 ± 5.51[a]**
Z-Score	3.68 ± 1.94[b]	6.23 ± 2.09[b]	8.82 ± 2.94[c]	3.97 ± 2.23[a]	6.67 ± 2.51[b]	10.60 ± 4.84[b]

*Means within a column the same letter(s) are not statistically significant (p=0.05) according to Duncan's Multiple Range Test (SPSS V.16.0.1)

It can be observed from the above tables (Table 1 to Table 4) that the Median and Z-Score normalization techniques were outperformed by the other three normalization techniques in all the four time series considered. Though Min-Max normalization technique performs better than decimal scaling and vector normalization on milk time series, for all other time series gives worse SMAPE. In order to obtain the best normalization technique for univariate TSF using evolutionary PSN, the average SMAPEs of all four time series for each normalization technique is obtained and shown in Table 5.

One can observe from Table 5 that decimal scaling and vector normalization are the best practical normalization methods and provide statistical meritorious results for univariate time series forecasting using evolutionary higher order neural network (HONN) compared to other normalization techniques. Although Vector normalization have shown better forecast accuracy (in both 1-step-ahead and 5-step-ahead), it is not statistical significance to that of decimal scaling.

V. CONCLUSION

This paper evaluates the impact of normalization techniques on univariate TSF using evolutionary PSNs. For this, five popular normalization techniques (Min-Max, Decimal Scaling, Median, Vector and Z-Score) and four univariate time series are considered. The experimental results revealed that normalization techniques have a significant impact on both single and multiple step-ahead forecasting. The decimal scaling and vector normalization techniques gave statistically better forecast accuracy compared to median, min-max and z-score normalization techniques. Although vector normalization on an average provides the best forecast accuracy, it is statistically insignificant to decimal scaling. Therefore, one can use decimal scaling or vector normalization method to forecast any univariate time series using evolutionary PSN. This finding is also applicable to ridge polynomial higher order neural networks (RPNNs) since RPNNs are a generalization of PSNs.

REFERENCES

[1] Y. Shin, and J. Ghosh, "The pi–sigma network: An efficient higher-order neural network for pattern classification and function approximation," In: Neural Networks, IJCNN-91-Seattle International Joint Conference on Neural Networks; pp. 8-14, 1991.

[2] D. S. Huang, H. H. S. Ip, K. C. K. Law, and Z. Chi, "Zeroing polynomials using modified constrained neural network approach," IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 721-732, 2005.

[3] S. Perantonis, N. Ampazis, S. Varoufakis, and G. Antoniou, "Constrained learning in neural networks: Application to stable factorization of 2-d polynomials," Neural Processing Letter, vol.7, no. 1, pp. 5-14, 1998.

[4] C. Voutriaridis, Y. S. Boutalis, and G. Mertzios, "Ridge polynomial networks in pattern recognition," EC-VIP-MC 2003, 4th EURASIP Conference focused on Video / Image Processing and Multimedia Communications, vol. 2, pp. 519-524, 2003.

[5] P. Liatsis, and A. J. Hussain, "Nonlinear one-dimensional DPCM image prediction using polynomial neural networks," In Proc. SPIE: Applications of Artifcial Neural Networks in Image Processing IV San Jose, California, pp. 58-68, 1999.

[6] A. Hussain, A. Knowles, P. Lisboa, W. El-Deredy, and D. Al-Jumeily, "Polynomial pipelined neural network and its application to financial time series prediction," Lecture Notes in Artifcial Intelligence 4304, pp. 597-606, 2006.

[7] Y. Shin, and J. Ghosh, "Ridge Polynomial Networks," IEEE Transactions on Neural Networks, vol. 6, no. 3, pp. 610-622, 1995.

[8] Y. L. Karnavas, and D.P. Papadopoulos, "Excitation control of a synchronous machine using polynomial neural networks," Journal of Electrical Engineering, vol. 55, no. 7, pp. 169-179, 2004.

[9] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Hardware-friendly Higher-Order Neural Network Training using Distributed Evolutionary Algorithms," Applied Soft Computing, vol. 10, no 2, pp. 398-408, 2010.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

[10] K. K. Sahu, S. Panigrahi, and H. S. Behera, "A Novel Chemical Reaction Optimization Algorithm for Higher Order Neural Network Training", Journal of Theoretical and Applied Information Technology, vol. 53, no. 3, pp. 402-409.
[11] R. Storn, and K. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.
[12] D. Goldberg, "Genetic Algorithms in Search," Optimization and Machine Learning Reading, MA:Addison-Wesley, 1989.
[13] J. Kennedy, R. C. Eberhart, and Y. Shi, "Swarm intelligence," San Francisco CA:Morgan Kaufmann, 2001.
[14] K. Socha, and M. Doringo, "Ant colony optimization for continuous domains," Europian Journal of Operation Research, vol. 185, no. 3, pp. 1155-1173, 2008.
[15] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm-A novel tool for complex optimization problems," in IPROMS Oxford, U.K.: Elsevier,2006.
[16] H. G. Beyer, and H. P. Schwefel, "Evolutionary Strategies: A Comprehensive introduction", Natural Computing, vol. 1, no. 1, pp. 3-52, 2002.
[17] K. H. Han, and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 580-593, 2002.
[18] A. Y. S. Lam, and V. O. K. Li, "Chemical-Reaction-inspired metaheuristic for optimization," IEEE Transaction on Evolutionary Computation, vol. 14, no.3, pp. 381-399, 2010.
[19] A.Y.S. Lam, "Real-Coded Chemical Reaction Optimization," IEEE Transaction on Evolutionary Computation, vol. 16, no. 3, pp. 339-353, 2012.
[20] A. Y. S. Lam, and V. O. K. Li, "Chemical Reaction Optimization: a tutorial," Memetic Computing, vol. 4, no. 1, pp. 3-17, 2012.
[21] J. Ghosh, and Y. Shin, "Efficient higher-order neural networks for classification and function approximation," in: International Journal on Neural Systems, vol. 3, pp. 323–350, 1992.
[22] Y. Shin, and J. Ghosh, "Realization of Boolean functions using binary pi-sigma networks," in: Proceedings of Conference on Artificial Neural Networks in Engineering, 1991.
[23] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighbourhood based mutation operator," IEEE Transaction on Evolutionary Computation, vol. 13, no.3, pp. 526-553, 2009.
[24] S. Rahnamayan, H. R. Tizhoosh, and M. A. Salama, "Opposition based differential evolution", IEEE Transaction on Evolutionary Computation, vol. 12, no.1, pp. 64-79, 2008.
[25] S. Das, and P. N. Suganthanam, "Differential Evolution: A Survey of the state-of-the-Art," IEEE Transaction on Evolutionary Computation, vol. 15, no.1, pp. 4-31, 2011.
[26] K. Price, R. Storn, and J. Lampinen, "Differential Evolution-A Practical Approach to Global Optimization," Berlin, Germany: Springer, 2005.
[27] K. V. Price, "An introduction to differential evolution," in: New Ideas in Optimization, U.K.: McGraw-Hill, pp. 79–108, 1999.
[28] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An Adaptive Differential Evolution Algorithm with Novel Mutation and Cross Over Strategy for Global Numerical Optimization," IEEE Transaction on System, Man, and Cybernetics-PART B: Cybernetics, Vol. 42, No.2, pp.482-500, 2012.
[29] J.J.Q. Yu, A.Y.S. Lam, and V.O.K. Li, "Evolutionary Artificial Neural Network based on chemical reaction optimization", IEEE Congress on Evolutionary Computation, pp. 2083–2090, 2011.