Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 03, No. 1, March : 2022

Synthesis of Dempster-Shafer Evidence and Trimmed-Winsorized Means for Use in Neural Network Time Series Prediction

P. BHARATH KUMAR
Assistant Professor
bharathkumar1218@gmail.com

E. ANITHA
Assistant professor
eanithaalts@gmail.com

H. PRASANTH KUMAR
Assistant professor
hprasanth183@gmail.com

Abstract: Artificial neural networks are suitable for all machine learning and design recognition tasks. A synthetic neural system, in contrast to conventional methods for time series assessment, requires guidance for the time series data and may be used to a wide range of problems. We experimented to find the best criteria for a forecasting system in order to be used. Under a well-known feature selection technique, the newly developed artificial neural networks offered improved predicting accuracy.

Keywords:

1. INTRODUCTION

Normally, the guidelines of the typical approach [2] are based on frequency range and the auto connection of time series. The utilization of artificial neural networks for time sequence analysis relies just about the data which were discovered. A *synthetic neural system is strong enough to signify any kind of time series, as multiple layer feed forward systems with an adequate amount of hidden models and one or more hidden layer are able of approximating function [8, 11]to any considerable. The forward and backward paths of the completely connected feed forward network may be applied by internal and external products of matrices and vectors in several outlines of APL code. For the application, we chose to make use of a completely joined, layered, feed forward artificial neural system with a single hidden layer and the back-propagation learning algorithm. The following section gives a brief review of the applicable definitions and calculations.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 03, No. 1, March : 2022

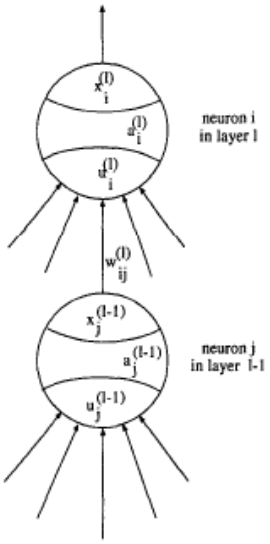

Figure 1: Exchange of activation values between neurons

1.1 A Very Short Introduction to Artificial Neural Networks

The fundamentals of the method for understanding in neural nets were set by [10] artificial neural nets contain many simple processing devices (called processing factors or neurons) grouped in levels. Each layer is recognized from the index l=O,..., M. The processing elements are interlocked as follows: Conversation between processing elements is permitted for processing elements of nearby levels. Neurons in a level cannot convey. Each neuron has a particular activation level a. Data is processed by the network from the exchange of service levels between neurons (see figure1):

The output value of the i-th neuron in sheet l is denoted by $\chi_i^{(l)}$,It is calculated with the formula

$$\chi_i^{(l)} = g(a_i^{(l)})$$

where $g(\cdot)$ is a monotone growing function. For our examples, we use the function $g(y) = \frac{1}{1+e^{-y}}$ (the "squashing function"). The commencement level $a_i^{(l)}$ of the neuron i in layer l is calculated by

$$a_i^{(l)} = f(u_i^{(l)})$$

Where $f(\cdot)$ is the activation function (in our case the characteristics function is used).

The net input $u_i^{(l)}$ of neuron i in layer l is projected as

$$u_i^{(l)} = (\sum_{j=1}^{n^{(l)}} \omega_{ij}^{(l)} \chi_j^{(l-1)}) - \theta_i^{(l)}$$

Where $\omega_{ij}^{(l)}$ is the weight of neuron j in layer $l-1$ connected to neuron i in layer l, $\chi_j^{(l-1)}$ the output of neuron j in layer $l-1$. $\theta_i^{(l)}$ is a bias charge that is subtracted from the sum of the weighted activations.

The computation of the network position starts at the input layer and ends at the output layer. The input vector I initializes the establishment levels of the neurons in the input layer:

$a_i^{(0)} = i^{th}$ element of I

For the input layer, $g(\cdot)$ is the identity function. The service degree of one coating is spread to another layer of the system. Then the dumbbells between the nerves are altered by the back-propagation learning rule. The artificial neural network discovers the enter / output mapping by a change of the dumbbells and reduces the distinction between the genuine and wanted output vector.

The simulator can be split in to two principal periods during network training: A arbitrarily chosen input/output pair is offered to the input tier of the network. The service is subsequently spread to the hidden levels and eventually to the output level of the system. Next action the real output vector is in contrast to the specified effect. The mistake values are disseminated back from your output level towards the hidden levels. The weights are altered so that there's less error for a recent demo of exactly the exact same design.

The weight change in layer l at time υ is calculated by

$$\Delta\omega_{ij}^{(l)}(\upsilon) = \eta\delta_i^{(l)}\chi_j^{(l-1)} + \alpha\Delta\omega_{ij}^{(l)}(\upsilon - 1)$$

where $\eta \in (0,1)$ is the learning rate and $\alpha \in (0,1)$ is the impetus. Both are kept steady during learning. $\delta_i^{(l)}$ is distinct

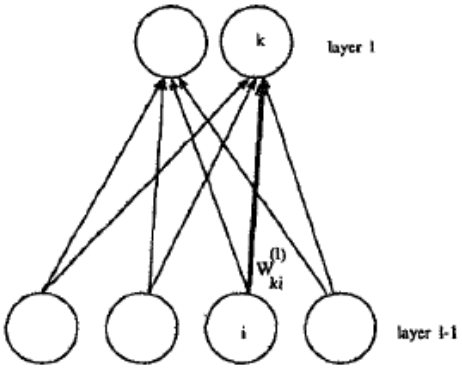

Figure 2: Weight adaptation among two neurons

1. for the output layer $(l = L)\delta_i^{(L)}$ as

$$\delta_i^{(L)} = (d_i - \chi_i^{(L)})g'(u_i^{(L)})$$

Where $g^{'}(u_i^{(L)})$ is the incline of the output function at $u_i^{(L)}$ .The gradient of the output function is always optimistic.

The formula can be elucidated as follows: When the output $\chi_k^{(l)}$ of the neuron i in layer l is too small, $\delta_k^{(l)}$ has a negative assessment. Hence the output of the neuron can be raised by increasing the net input $u_k^{(l)}$ by the following change of the burden values:

If $\chi_i^{(l-1)} > 0,$ then increase $\omega_{ki}^{(l)}$

If $\chi_i^{(l-1)} < 0,$ then decrease $\omega_{ki}^{(l)}$

The regulation applies vice versa for a neuron with an output charge that is too high (see figure 2).

2. for the output layer $(l < L)\delta_i^{(l)}$ is defined by:

$$\delta_i^{(l)} = g^{'}(u_i^{(l)})(\sum_{k=1}^{n^{(l+1)}} \delta_k^{(l+1)}\omega_{ki}^{0l+1})$$

Finally the weights of layer l are adjusted by

$$\omega_{ij}^{(l,new)} = \omega_{ij}^{(l)} + \Delta\omega_{ij}^{(l)}(\upsilon)$$

2. IMPLEMENTATION

The time series modeling and forecasting classification was implemented using core duo processor with 4gb ram of an auxiliary workstation. The system consists of two main components: A toolkit of APL functions that constrain the neuralnetwork and log parameters and results of the replication runs to APL component files.A graphical user interface that allows the user to navigate during the simulations and to evaluate the actual time series with the onegenerat;d by the neural network,

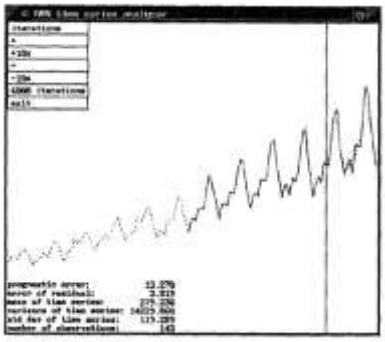

Figure 3: User interface of the forecasting system

Figure 3 offers a display dump of the interface: the damaged line displays the real time sequence; the strong line signifies the network's output. The outlook data is divided from the historic data - the network's training set - by the vertical bar in the best quarter of the chart, the menu in the top left part of number 3 enables the person to choose a perspective of the network's forecasting capacity at different countries through the entire learning stage.

By looking at the record files of the simulator runs, current and previous outcomes could be assessed and compared.

3. MODELING

3.1 The Training Sets

As test bed for our forecasting program we utilized two well-known time sequence from [2]: The monthly totals of international flight passengers (thousand of passengers)from 1949 to 1960 (see number 4), as well as the daily closing costs of share inventory (see figure 5).Stand one provides features to some of the two-time series : u g the mean, is the typical deviation, and n the amount of findings.

Time Series	σ	μ	n
Share Price	84.11	478.47	369
Grid power unit Price	119.55	280.30	144

Table 1: Properties of time series

The next section is disturbed with the question: How can a neural network learn a time series?

3.2 The Algorithm

The neural network sees the time series Xl,. . . . Xm in the form of many mappings of an input vector to an output value (see figure 6). This technique was presented by [4].A number of adjoining data points of the time series (the input window Xt_s, X,-S+,,.., X,) are planned to the interval [0,1] and utilized as service levels for the models of the enter level. The problem useful for that back-propagation learning algorithm is currently computed by evaluating the worth of the result unit together with the value of the time sequence at time t one. This mistake is spread back for the contacts between concealed and result level and to these between hidden and output level. One demonstration is finished, after all dumbbells have been updated so. Training a sensory network with all the back-propagation algorithm generally demands that representations of the enter set (called one epoch) are offered several times.

For your understanding of period series data, the representations were offered in a randomly fashion: As reported [4], picking a random place for every portrayal's input window ensures better system functionality and prevents local minima.

The following section is focused on the choice of the appropriate guidelines for the learning criteria and the choice of the ideal topology for the predicting network.

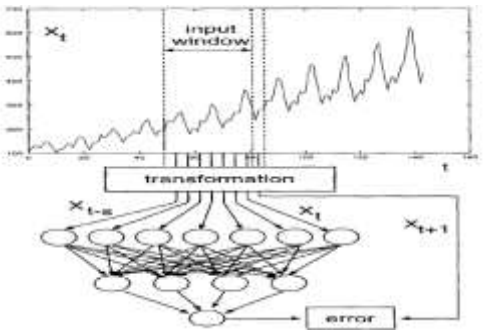

Figure 3: Learning a Time Series

3.3 Network Parameters

The following parameters of the artificial neural network were selected for a closer inspection:

- The learning rate η
  η ($0 < \eta < 1$) is a scaling factor that tells the knowledge algorithm how strong the weights of the associations should be familiar for a given error. A higher η can be used to speed up the knowledge process, but if η is too high, the algorithm will‘(step over” the most favorable weights. The learning rate η is regular across presentations.
- The momentum α The momentum restriction a ($0 < \alpha < 1$) is another number that affects the grade descent of the weights: To prevent each association from following every little modify in the solution space immediately, the impetus term is added that keeps the course of the previous step [7],thus avoiding the drop into local minima. The momentum term is steady across presentations.
- The number of input and the number of concealed units (the network topology).The number of input units conclude the number of periods the neural network “looks into the past” when predicting the future, the number of input units is corresponding to the size of the input window.
  Whereas it has been exposed that one hidden layer is adequate to approximate continuous function [8], the number of hidden units essential is “not known in general [7]”. Other move toward for time series analysis with artificial neural networks report working network topologies (number of neurons in the input-hidden output layer) of 8-8-1, 6-6-1 [CMMR9Z], and 5-5-1[13].

To inspect the distribution of these parameters, we behavior a number of experiments: In succeeding runs of the network, these parameters were methodically changed to explore their consequence on the network’s modeling and forecasting capabilities.

We used the following terms to calculate the modeling quality and forecasting quality of our system: For a time series $X_1, \ldots . X_n$

$$s_m = \sqrt{\frac{\sum_{i=1}^{n} (X_i - X_i)^2}{n}}$$

Where*Xi* is the approximation of the artificial neural network for phase*i* and *r* is the number of forecasting periods. The error s_m (equation 1) estimates the capability of the neural network to mimic the known information set, the error *Sf* (equation 2) judges the networks’s forecast potential for a forecast period of length *r*. In our experiments, we used $r = 20$.

4. COMPARISON BETWEEN DIVERGENT FEATURE SELECTION MODELING’S

We compared results gained under feature selection methods called Dempster–Shafer evidence theory and trimmed-Winsorized means. The execution of the procedure uses the applications explained by Jenkins and Box in Component V of their classic [2].The approach is known as an autoregressive integrated moving average method of order (p,d,q). It's described by the equation

$$a(z)\nabla^d X_t = b(z)U_t$$

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 03, No. 1, March : 2022

where X_t stands for in time ordered values of a time series, $t = 1, \ldots, n$ for n observations. U_t is a sequence of random values called "white noise" process. The backward difference operator V is defined as

$$\nabla X_t = X_t - X_{t-1} = (1 - z)X_t$$

We fitted these feature selection models for each time series and let it predict the next 20 observations of the time series. The last 20 observations were dropped from the time series and used to calculate the prediction error of the models.

The following feature selection models were calculated for the power unit price time series (after a logarithmic transformation):

$(1 - z)(1 - z^{12}) X_t = (1 - 0.24169,2 - 0.47962.z^{12}) U_t$ and for the Share time series:

$(1 - z) X_t = (1 - 0.10538z) U_t$

In the forecast errors for the man-made neural network (ANN), the artificial neural network utilizing the logarithmic and change (ANN record,) and the feature selection models opted are compared: The artificial neural network utilizing the logarithmic and changed time series out-performed for both time series, whereas the "simple" artificial neural network expected more precisely just for the Shares time series. This behavior could be described as follows: the bigger data selection of the flight passenger time series results in a reduction in precision for the untransformed in place set. Logarithmic and differencing transformations helped to remove the tendency and planned the time series data in to a smaller variety.

Time series	Dempster–Shafer evidence theory	Trimmed-Winsorized means
power unit prices	20.97	18.72
Share price	10.97	11.35

Table 2: Forecasting errors for ANN under Dempster–Shafer evidence theory and Trimmed-Winsorized means

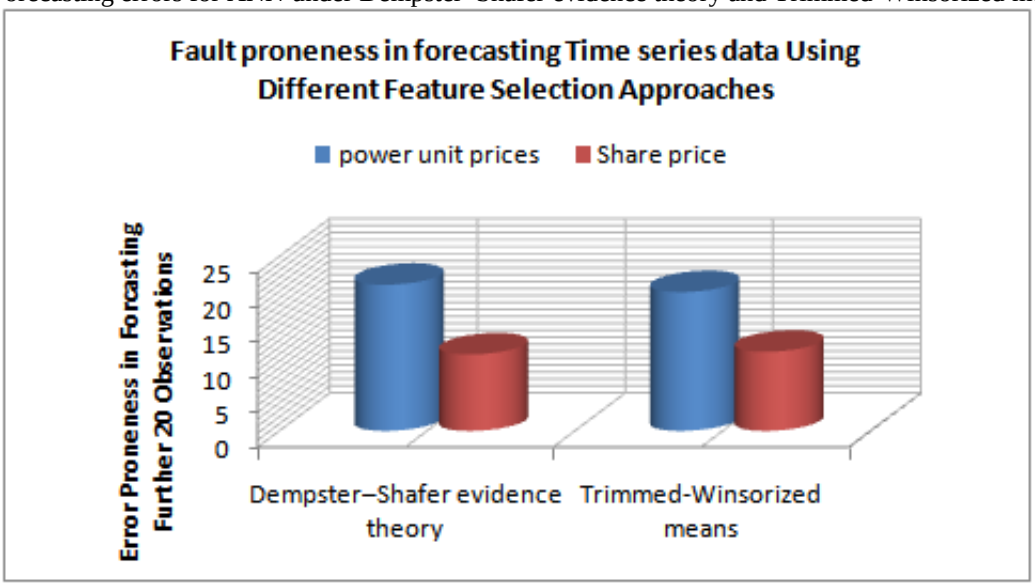

Figure 1: Fault proneness of ANN under Dempster–Shafer evidence theory and Trimmed-Winsorized means

ANN Code

These functions present a complete implementation of the definitions given in the introduction. The main function NET learns the well known XOR (exclusive or) mapping. For reasons of simplicity, the input-output mapping of the XOR function is coded into the function NET.

Conclusion and Further Work

We've offered a predicting program for univariate time sequence that utilizes artificial neural networks. These processing devices proved themselves to be feasible options to classic approaches. The machine may be utilized along with other approaches for time series evaluation or just as a standalone device.

More work may contain the assessment with additional time series analysis approaches, development of hybrid approaches that blend the power of standard methods with artificial neural networks and the use of our program to multivariate time series.

References

[1] M. Alfonseca. Advanced applications of APL: logic programming, neural networks and hypertext. IBM Systems Journal, 30(4):543—553, 1991.
[2] George E. P. Box and Gwilym M. Jenk¬ins. Time Series Analysis - forecasting and control. Series in Time Series Anal-ysis. Holden-Day, 500 Sansome Street, San Franciso, California, 1976.
[3] E. Chatfield. The Analysis of Time Series. Chapman and Hall, New York, fourth edi¬tion, 1991.
[4] Kanad Chakraborty, Kishan Mehrota, Chilukuri K. Mohan, and Sanjay Ranka. Forecasting the Behaviour of Multivariate Time Series Using Neural Networds. Neu¬ral Networks, 5:961-970, 1992.
[5] Dyadic Systems Limited, Riverside View, Basing Road, Old Basing, Basingstoke, Hampshire RG24 OAL, England. Dyalog Apl Users Guide, 1991.
[6] Richard M. Evans and Alvin J. Surkan. Relating Numbers of Processing Elements in a Sparse Distributed Memory Model to Learning Rate and Generalization. ACM APL Quote Quad, 21(4):166-173, 1991.
[7] John Hertz, Anders Krogh, and Richard G. Palmer. Introduction to the Theory od Neu¬ral Computation. Addison Wesley, Red¬wood City, California, 1991.
[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. Neural Networks, 2:359-366, 1989.
[9] Howard A. Peele. Teaching A Topic in Cy¬bernetics with APL: An Introduction to Neural Net Modelling. ACM APL Quote Quad, 12(l):235-239, 1981.
[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning repre¬sentations by back-propagating errors. Na¬ture, 323(9):533-536, October 1986.
[11] Hava Siegelmann and Eduardo D. Sontag. Neural Nets Are Universal Computing De¬vices. Technical Report SYSCON-91-08, Rutgers Center for Systems and Control, May 1991.
[12] Alexei N. Skurihin and Alvin J. Surkan. Identification of Parallelism in Neural Net¬works by Simulation with Language J. ACM APL Quote Quad, 24(l):230-237, 1993.
[13] Halbert White. Economic prediction us¬ing neural networks: the case of ibm daily stock returns. In Proceedings of the IEEE International Conference on Neural Net¬works, pages II—451—11—459, 1988.
[14] G. P. Zhang, B. E. Patuwo, and M. Y. Hu, “Forecasting with artificialneural networks: The state of the art,” Int. J. Forecast., vol. 14, pp.35–62, Mar. 1998.
[15] S. Crone and P. Grafeille, “An evaluation framework for publications onartificial neural networks in sales forecasting,” in Proc. Int. Conf. Artif.Intell., Las Vegas, NV, Jun. 2004, pp. 221–227.
[16] T. Hill, M. O’Connor, and W. Remus, “Neural network models for timesseries forecasting,” Manage. Sci., vol. 42, no. 7, pp. 1082–1092, Jul.1996.
[17] N. R. Swanson and H. White, “Forecasting economic time seriesusing flexible versus fixed specification and linear versus nonlineareconometric models,” Int. J. Forecast., vol. 13, no. 4, pp. 439–461, 1997.
[18] W. Zhang, Q. Cao, and M. J. Schniederjans, “Neural network earningper share forecasting models: A comparative analysis of alternativemethods,” Decision Sci., vol. 35, no. 2, pp. 205–237, 2004.
[19] S. Heravi, D. R. Osborn, and C. R. Birchenhall, “Linear versus neuralnetwork forecasts for European industrial production series,” Int. J.Forecast., vol. 20, no. 3, pp. 435–446, Jul.–Sep. 2004.
[20] L. J. Callen, C. C. Kwan, P. C. Yip, and Y. Yuan, “Neural networkforecasting of quarterly accounting earnings,” Int. J. Forecast., vol. 12,no. 4, pp. 475–482, 1996.
[21] M. Adya and F. Collopy, “How effective are neural networks at forecastingand prediction? A review and evaluation,” Int. J. Forecast., vol.17, nos. 5–6, pp. 481–495, Nov. 1998.
[22] M. Nelson, T. Hill, T. Renas, and M. O’Connor, “Time series forecastingusing NNs: Should the data be deseasonalized first?” J. Forecast., vol.18, no. 5, pp. 359–367, 1999.

[23] G. P. Zhang and D. M. Kline, “Quarterly time series forecasting withneural networks,” IEEE Trans. Neural Netw., vol. 18, no. 6, pp. 1800–1814, Jun. 2007.
[24] S. D. Balkin and J. K. Ord, “Automatic neural network modeling forunivariate time series,” Int. J. Forecast., vol. 16, no. 4, pp. 509–515,2000.