2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Time Series Prediction Based on Randomly Weighted Neural Networks

Xizhao Wang
College of Computer Science and Software Engineering, Shenzhen University,
Shenzhen, China
xzwang@szu.edu.cn

Qin Wang
College of Computer Science and Software Engineering, Shenzhen University,
Shenzhen, China
wangqin2021@email.szu.edu.cn

Qiang Liu
College of Computer Science and Software Engineering, Shenzhen University,
Shenzhen, China
liuqiang2022@email.szu.edu.cn

***Abstract*—One of the most frequently used models for time series prediction is the Long Short Term Memory (LSTM). LSTM can leverage the past patterns to efficiently forecast the future observations but it is often criticized as very computationally expensive due to the iterative training. In this paper, to reduce the computational workload and improve the prediction performance of time series, we propose a novel auto-regression framework based on Random Vector Functional Link (RVFL). The new framework offers a lighter network structure with higher training efficiency compared to LSTM-based approaches. It is a new attempt to utilize randomized learning algorithms for time series prediction, providing valuable insights for developing faster and more efficient models in the future.**

***Index Terms*—time series prediction, auto-regression, long short-term memory, randomly weighted neural networks**

I. Introduction

Time series data consists of a sequence of observations ordered in time, and time series prediction aims to forecast future observations based on past patterns [1]. Auto-regression (AR) is a commonly used approach for time series prediction. However, due to the limited approximation ability of polynomials, AR struggles to handle complex nonlinear time series. With the advent of deep learning, LSTM-based methods have emerged as promising approaches for time series prediction [2], [3]. Nonetheless, the training of LSTM models can be computationally inefficient due to the large number of parameters in the gate cells. Additionally, as the number of network layers increases, LSTM may pose challenges in terms of hardware requirements and energy consumption.

To overcome these challenges, we propose a novel moving auto-regression framework called MAR-RVFL, based on Random Vector Functional Link (RVFL). RVFL is a type of randomly weighted neural network, where the weights between the input layer and hidden layer are randomly generated, and the weights between the hidden layer and output layer are calculated using matrix inverse operations [4]. RVFL's simple structure and non-iterative training mechanism enable reliable generalization and fast training speeds [5]. By combining RVFL with a specially designed sliding window mechanism, MAR-RVFL effectively learns the underlying patterns in time series data. Compared to LSTM, MAR-RVFL demonstrates higher training efficiency and robustness, providing valuable insights for more powerful time series prediction models.

II. Preliminarie

A. Auto-regression (AR)

AR is a method used for predicting future observations in a time series by analyzing the relationships between past observations. It assumes that the $(k+1)$th observation can be expressed as a linear combination of the previous k observations, as shown in Equation (1).

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{c} + \boldsymbol{\varphi}_1 \boldsymbol{x}_1 + \boldsymbol{\varphi}_2 \boldsymbol{x}_2 + \dots \boldsymbol{\varphi}_k \boldsymbol{x}_k + \boldsymbol{\varepsilon}_{k+1} \tag{1}$$

where $\hat{\boldsymbol{x}}_{k+1}$ is the evaluated observation at the $k+1$ th moment, $\boldsymbol{\varphi}_i$ is the influence factor of the i th moment, $\boldsymbol{x}_i$ is the i th observation, and $\boldsymbol{c}$ is a noise term. The objective is to estimate $\boldsymbol{\varphi}_i$ using methods such as least squares or gradient descent.

B. Random Vector Functional Link (RVFL)

RVFL is typical single hidden layer neural network. In RVFL, the weights between input layer and hidden layer are randomly assigned, the weights between input layer and output layer, as well as weights between hidden layer and the output layer (represented by $\boldsymbol{\beta}$) are calculated by solving the optimization problem in Eq.(2).

$$Minimize : \|\boldsymbol{H\beta} - \boldsymbol{Y}\| \tag{2}$$

The solution to the above problem is defined as Eq.(3)

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{Y} \tag{3}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\boldsymbol{H}$ [6].

III. Proposed Method

A. Moving Auto-regression RVFL(MAR-RVFL)

Assuming here we intend to predict future k moments' observations according to the past k observations in a time series. The original training data is a one-dimension sequence with a length of T.

$$\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3 \dots \boldsymbol{s}_t \dots \boldsymbol{s}_T\} \tag{4}$$

where $\boldsymbol{s}_t$ is the observation at moment t. According to the defination of the task, the sequence can be transformed into features matrix $\boldsymbol{X} \in R^{z \times k}$ and labels matrix $\boldsymbol{Y} \in R^{z \times k}$ ($\boldsymbol{z} = (\boldsymbol{T} - \boldsymbol{k})/\boldsymbol{k}$ is the amount of instances). Then we have:

$$\boldsymbol{x}_i = \left[\boldsymbol{s}_{(i-1)k+1}, \boldsymbol{s}_{(i-1)k+2}, \dots \boldsymbol{s}_{ik}\right], \boldsymbol{y}_i = \left[\boldsymbol{s}_{ik+1}, \boldsymbol{s}_{ik+2}, \dots \boldsymbol{s}_{ik+k}\right] \tag{5}$$

Based on the approach presented in [7], we first initialize the weights and biases between the input layer and the hidden layer ($\boldsymbol{W}$ and $\boldsymbol{b}$) with random values. The weights between the input layer and the output layer are ignored.Then we construct a initial window with a size of u:

$$\boldsymbol{U}_X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \ldots \boldsymbol{x}_u], \boldsymbol{U}_Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \ldots \boldsymbol{y}_u] \tag{6}$$

The output of hidden layer is $\boldsymbol{H} = \boldsymbol{g}(\boldsymbol{U}_X \cdot \boldsymbol{W} + \boldsymbol{b})$, where $\boldsymbol{g}$ is the activation function. According [7], the weights betweent hidden layer and output layer $\boldsymbol{\beta}$ can be calculated as Eq.(7).

$$\boldsymbol{\beta} = \boldsymbol{H}^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{\boldsymbol{C}} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\right)^{-1} \boldsymbol{U}_Y \tag{7}$$

We believe the latter observations in the time series show more relations with the observations in the future. Consequently, we use the latter observations in the window to further optimize the random weights $\boldsymbol{W}$ and $\boldsymbol{b}$ through Back Propagation (BP) algorithm.

$$\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{\alpha} \cdot \frac{\boldsymbol{\delta J}(\boldsymbol{W}, \boldsymbol{\beta}, \boldsymbol{U} \cdot \boldsymbol{M})}{\boldsymbol{\delta W}} \tag{8}$$

where M is a mask vector in which the former m elements are valued 0 and the latter $u - m$ elements are valued 1, J denotes the training loss, and $\boldsymbol{\alpha}$ is the learning rate.

The process of MAR-RVFL involves sliding the window $\boldsymbol{U}_X$ and $\boldsymbol{U}_Y$ over the input and output instances with a step size of $\boldsymbol{L}$. After each slide, the weights $\boldsymbol{\beta}$ are updated using Equation (10), which aims to minimize the training error under the new window while preserving the patterns learned from previous iterations. Assuming the slided new window is:

$$\boldsymbol{U}_X^* = [\boldsymbol{x}_{1+L}, \boldsymbol{x}_{2+L}, \ldots \boldsymbol{x}_{u+L}], \boldsymbol{U}_Y^* = [\boldsymbol{y}_{1+L}, \boldsymbol{y}_{2+L}, \ldots \boldsymbol{y}_{u+L}] \tag{9}$$

To adjust $\boldsymbol{\beta}$ for the new window and retain patterns in former iterations, we optimize $\boldsymbol{\beta}$ based on the following new object.

$$Minimize : \left\|\boldsymbol{H} \cdot \boldsymbol{\beta}^* - \boldsymbol{U}_Y^*\right\| and \left\|\boldsymbol{\beta}^* - \boldsymbol{\beta}\right\| \tag{10}$$

where $\boldsymbol{\beta}^*$ denotes the optimized $\boldsymbol{\beta}$. The first term represents the training error under new window, which is minimized in MAR-RVFL for fitting new window's observations. The second term $\|\beta^* - \beta\|$ is the L2-norm for updating $\boldsymbol{\beta}^*$, this term is minimized for inheriting former patterns. Based on lagrange method of multipliers, $\boldsymbol{\beta}^*$ is calculated as Eq.(11).

$$\boldsymbol{\beta}^* = \boldsymbol{H}^{\mathrm{T}} \cdot \left(\frac{\boldsymbol{I}}{\boldsymbol{C}} + \boldsymbol{H} \cdot \boldsymbol{H}^{\mathrm{T}}\right)^{-1} \cdot (\boldsymbol{U}_Y^* - \boldsymbol{H} \cdot \boldsymbol{\beta}) + \boldsymbol{\beta} \tag{11}$$

Then we again intend to optimize random weights based on latter observations in the new window. The MAR-RVFL method continues to slide the window and update the weights until it has iterated over the desired number of times. By doing so, it gradually incorporates information from different parts of the time series, capturing temporal dependencies and improving the prediction accuracy.

IV. Experiment

The proposed MAR-RVFL method is compared with LSTM on three real-word databases (DB1: daily temperature, DB2: sinewave, DB3: milk production). To ensure fairness, we use one LSTM layer as hidden layer in LSTM, and set the number of hidden nodes in both LSTM and MAR-RVFL as 2,000. Fig.1 shows the comparison for all three databases. It can be observed that MAR-RVFL outperforms LSTM in DB1 and DB2, showing better prediction ability. In DB3, LSTM performs slightly better than MAR-RVFL.

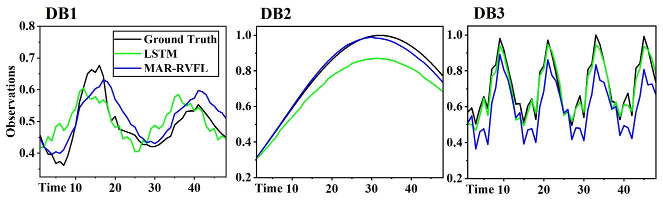

Fig. 1. Predicted and actual observations.

Fig.2 shows the MSE comparison between LSTM and MAR-RVFL for the three databases. It can be observed that MAR-RVFL achieves lower MSE values than LSTM in DB1 and DB2, indicating superior prediction performance. Assuming we have z instances after the pretreatment

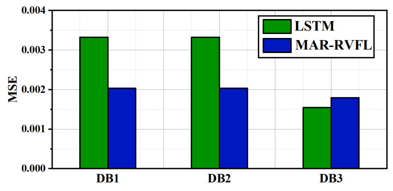

Fig. 2. Testing MSE of LSTM and MAR-RVFL on different databases.

Table.I shows the time cost during training. It can be seen that MAR-RVFL has a significantly shorter time compared to LSTM, making it computationally more efficient.

TABLE I
Time cost during training in LSTM and MAR-RVFL

Model	LSTM	MAR-RVFL
Time Cost (s)	192.7	41.9

V. Conclusion

In conclusion, the MAR-RVFL proposed in this paper offers a promising approach for time series prediction. By integrating RVFL into auto-regression, MAR-RVFL demonstrates competitive prediction ability compared to the widely used LSTM. Additionally, it exhibits faster training times, making it an efficient option for time series analysis.

References

[1] N. T. Koh, A. Sharma, J. Xiao, X. Peng, and W. L. Woo, "Solar irradiance forecast using long short-term memory: A comparative analysis of different activation functions," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2022, pp. 1096–1101.
[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[3] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, pp. 5929–5955, 2020.
[4] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.
[5] Q. Wang, J. Liu, W. Guo, and X. Wang, "Evolving stochastic configure network: A more compact model with interpretability," *Information Sciences*, vol. 639, p. 119006, 2023.
[6] D. Serre, "Matrices: Theory & applications additional exercises," *L'Ecole Normale Supérieure de Lyon*, 2001.
[7] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE transactions on cybernetics*, vol. 47, no. 10, pp. 3466–3479, 2017.