Improving normalization method of higher-order neural network in the forecasting of oil production

Joko Prasetyo[*], *Noor Akhmad* Setiawan, and *Teguh Bharata* Adji

Department of Electrical Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia

Abstract. One of the challenges in the oil industry is to predict well production in the absence of frequent flow measurement. Many researches have been done to develop production forecasting in the petroleum area. One of the machine learning approach utilizing higher-order neural network (HONN) have been introduced in the previous study. In this study, research focus on normalization impact to the HONN model, specifically for univariate time-series dataset. Normalization is key aspect in the pre-processing stage, moreover in neural network model.

Keywords—oil production forecast, time-series, higher-order neural network, normalization.

1 Introduction

Oil production forecasting is important for petroleum industry to deliver planning and investment strategy. Many researches have focused to deliver prediction model for the oil production with various approaches. Artificial neural network with imperialist competitive algorithm have been used to predict oil flow rate [1]. The utilization of nonlinear autoregressive neural network with exogenous input (NARX) also have been used to forecast oil production [2]. One variance neural network algorithm, called higher-order neural network (HONN), have been introduced in the previous study [3]. The study also introduced dataset of 5 (five) oil production well from Campay Basin Field, India. In the study, the research was focused on predicting cumulative production of 5 wells, instead of predicting individual well.

As key process to deliver robust machine learning model, pre-processing stage is very important step. One of the steps in the pre-processing stage is standardization or normalization. Almost all previous studies applied some kind of standardization before delivering learning step [4, 5, 6], especially for neural network with gradient-based estimator. In general, learning algorithms benefit from standardization of the data set. A noteworthy exception are decision tree-based algorithms that does not need scaling the data.

There are many standardization methods to be chosen including linear transformation and non-linear transformation [https://scikit-learn.org/stable/modules/preprocessing.html]. In this study, non-linear transformation called quantile transformation is proposed to improve prediction accuracy and the result acquired by various standardization method are evaluated in this paper.

2 Method

2.1 Dataset

In the previous study, oil production in Cambay basin (CB), Gujarat, India was introduced [3]. There are 5 well monthly production data from actual oil field for over five years (from 2004 to 2009). In this study, two datasets were chosen, which is CB1 and CB2 (first and second well). The production chart for CB1 can be seen in Figure 1, while for CB2 can be seen in Figure 2. Each dataset consists of 63 observations which determine oil production measurement for each well monthly.

As a benchmark, other dataset will be used in this study. This dataset is very well-known public time-series dataset, the Wolf's sunspot data [7]. This dataset contains annual number of spots occurrence present on the face of the sun. The dataset includes the number of observed yearly from 1700 to 2019, which is totally 321 observations (see Figure 3).

Those three datasets are categorized as univariate time-series data.

2.2 Pre-processing

The main focus on this study is in this stage. The standardization or normalization happened in the pre-processing step. It is a best practice to deliver such activity especially when dealing with neural network algorithm. There are three normalization method to be observed in this study. The most common one is the min-max method, which can be seen as follow:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

[*] Corresponding author: joko.nugroho.p@mail.ugm.ac.id

Where x_{norm} is the normalized value, x is the original value, x_{max} is the maximum raw value and x_{min} is the minimum raw value. This normalization also called as linear transformation [6][8]. The new value will be spread from 0 to 1. In the scikit-learn, this method called as MinMaxScaler. One variance of normalization is to extent the new value from -1 to 1 as in (2). This variance is to adapt with neural network architecture which using tanh activation function as its result to value between -1 to 1.

$$x_{new} = 2 \times x_{norm} - 1 \qquad (2)$$

However, both normalization methods are very sensitive to the outliers [8]. Another method that less sensitive to outlier is non-linear transformation. One of example of non-linear transformation is quantile transformation. This normalization method is successfully applied in biological study [9]

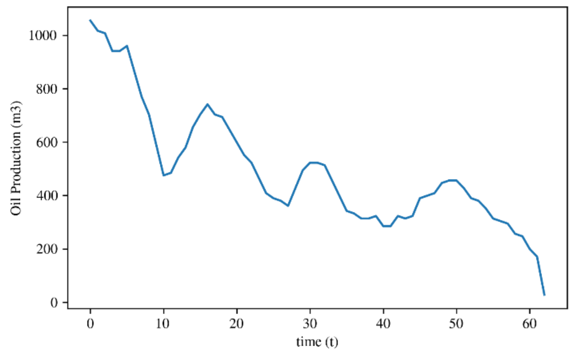

Fig. 1. Production profile of CB1 well.

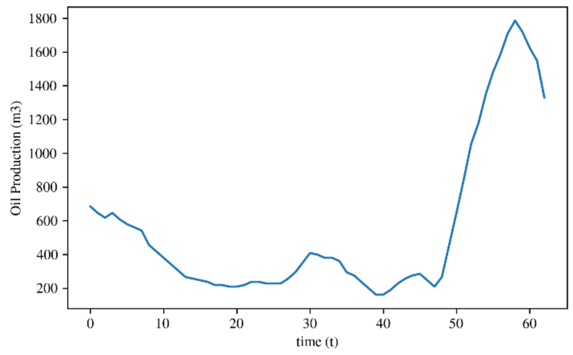

Fig. 2. Production profile of CB2 well.

Uniform quantile transformation (UQT) observed all distributions and then perform a non-parametric transformation to map the data to a uniform distribution. This transformation uses estimation of cumulative distribution function and then map the original values to a uniform distribution. The obtained values map to desired output distribution using associated quantile function. This method tends to spread out the most frequent values [10].

This transformation method can be described as follows:

1. With one time series data, sort all the values from low to high
2. Determine the cumulative distribution function for the series.
3. Determine the quantile (i.e. define 5 to retrieve 0.0, 0.25, 0.5, 0.75 and 1.0 quantile)
4. Calculate the normalization (using minmax) for each quantile with max value defined for each quantile.
5. The normalized value will be spread out from 0 to 1.

For this research, those three datasets will be normalized using three methods as follows:

- Norm1: normalization with range -1:1,
- Norm2: normalization with range 0:1,
- quantile transformation.

The result of each datasets can be seen in Figure 4, Figure 5 and Figure 6. The result of normalization with range of -1 to 1 seem similar with the original value since it is linear transformation. The normalization of 0 to 1 also similar with less steep. The quantile transformation result looks different with the other two, especially in CB2. The transformation makes the first value and the peak value less distinction than in original chart.

After being normalized, to transform time-series data to be used in supervised learning, the lag features have been applied. This approach is the classical way converting time series to supervised learning [11][12]. Time series data is shifted as feature and the next time (t+1) given as target. For this research, the initial setup will be two prior time steps to be used as feature (t-2 and t-1) and subsequent time step will be used as target (t+1). This approach has disadvantage since not all observation can be used, the first set should be dismissed since no feature or target is available. In this setup, first two data solely to be feature only. And then, trial will be continued up to 4 lag features (t-4, t-3, t-2, t-1 as features).

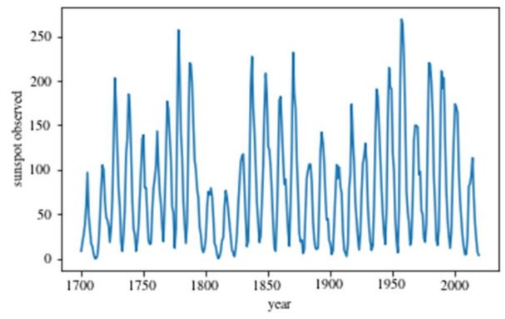

Fig. 3. Normalized value for CB1 dataset

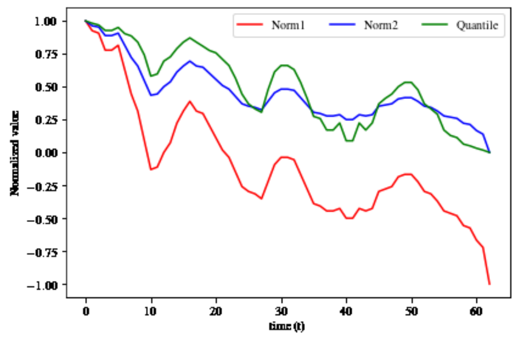

Fig. 4. Sunspot series (1700-2019)

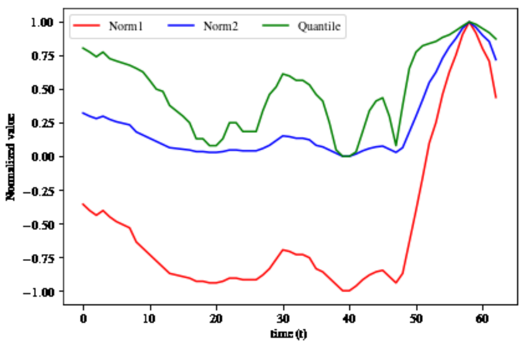

Fig. 5. Normalized value for CB2 dataset

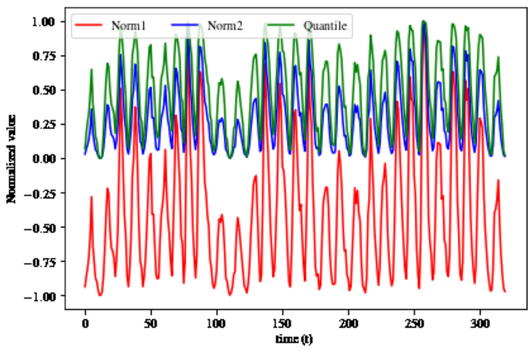

Fig. 6. Normalized value for Sunspot dataset.

2.3 Training methods

Higher-order neural network borrows features from conventional neural network (ANN) with the different synaptic operation. Major difference between HONN and ANN is how to calculate the sum-product of synaptic weight and input vector. As in (1), synaptic operation of conventional ANN is the sum-product of weight and input. While for HONN, the synaptic operation will be as (2). The difference is how the HONN apply polynomial calculation of input vector.

$$y = \varphi(\textstyle\sum_{i=0}^{n-1} w_i x_i) \tag{1}$$

$$y = \varphi(w_0 x_0 + \textstyle\sum_{i_1=1}^{n} w_{i_1} x_{i_1} + \sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n} w_{i_1 i_2} x_{i_1} x_{i_2} + \cdots + \sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n} \cdots \sum_{i_N=i_{N-1}}^{n} w_{i_1 i_2 \cdots i_N} x_{i_1} x_{i_2} \cdots x_{i_N}) \tag{2}$$

For this study, the architecture of HONN being proposed is third-order operations, called cubical synaptic order (CSO), with 2 (two) inputs and one hidden layer with different number of hidden neurons. The activation function being selected is tanh-tanh (hyperbolic tangent). The comparison of high-level architecture of neural network and the synaptic operation of CSO can be seen in Figure 7 and Figure 8. . Both use 2 input, 1 hidden layer with 3 neurons and one output layer. In the CSO side, the 3-rd degree polynomial applied into each input. The equation for each later is similar to (2). To optimize the accuracy, HONN model will be trained using back-propagation method to seek minimum loss during feed-forward.

The hyperparameter for HONN learning being chosen with learning rate of 0.01 and randomized initial weight. The learning rate is dynamically changed. If error is increasing, the learning will be multiplied by 0.7 and if error decreased then learning rate being multiplied by 1.05. The iteration for the training cycle is 600 and the momentum is 0.9. The activation function for both layers (hidden and output) is hyperbolic tangent (tanh). The hidden neuron will be set from 2 to 10.

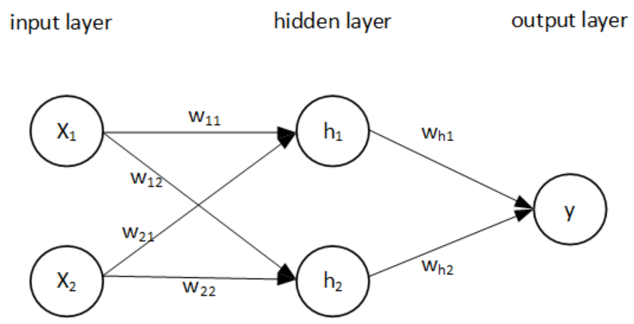

Fig. 7. Conventional neural network with 2 (two) inputs and 2 (two) hidden neurons

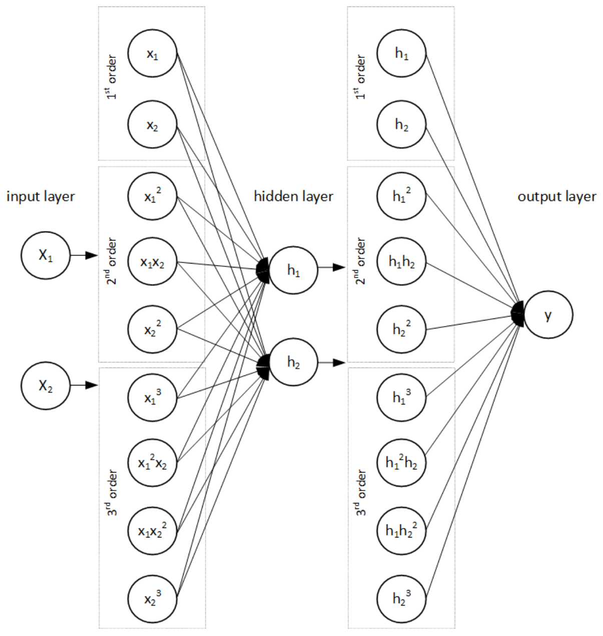

Fig. 8. Higher-order neural network with 2 input and 2 hidden neurons.

2.4 Evaluation

Various statistical methods are available to evaluate the performance of neural network model. In this study, performance evaluation will utilize the coefficient of determination or R squared. This evaluation method is statistical measure how closed the data pairs with the regression line, as stipulated in (3) where y_i is the actual value, $\hat{y}_i$ is the predicted value at given time. For the evaluation, transformed dataset was split into 80:20.

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{3}$$

3 IMPLEMENTATIONS AND RESULT

All implementation of the proposed approach has been deployed using Python 3.7 with a Intel Xeon system having Windows 10 OS. The additional libraries used to include NumPy, scikit-learn and matplotlib for the visualization. For the back-propagation, it has been deployed with mini batches approach of gradient descent.

To ensure repeatability, each combinations of neurons in the hidden layer have been tested with 10 trial. With R^2 score measured for every run, the mean score will be reported. In the proposed work, three datasets being used with three different normalization method, thus summary report consists of each dataset separately as in Table 1. In Table 1, the best result for each dataset have been highlighted. The proposed method using UQT outperform other normalization method in CB2 and sunspot datasets, which the R2 of quantile transformation is 0.69582 and 0.72431. While in CB1 dataset, UQT achieved the lowest score with R2 score of 0.43458. The comparison of actual and forecast value of all method is given in Figure 9. Figure 10 and Figure 11.

The lesser performance of UQT compared to Norm1 (normalization with range -1 to 1) may be acceptable, since the selected activation function in HONN algorithm is hyperbolic tangent. This activation function output is similar with Norm1 output, between -1 and 1. However, it is noted that UQT outperform Norm2 in all datasets, even both transformation output has similar range (0 to 1).

TABLE I. Best Result of Each Dataset (Mean Of 10 Run)

Dataset	Norm. Method	No. of lag features	No. of hidden neurons	R^2	RMSE[a]
CB1	**Norm1**	**4**	**1**	**0.62228**	**0.11095**
	Norm2	2	6	0.44515	0.06927
	Quantile	2	6	0.46050	0.07908
CB2	Norm1	2	9	0.55888	0.15015
	Norm2	3	10	0.50047	0.07838
	Quantile	**3**	**7**	**0.69582**	**0.02535**
Sunspot	Norm1	4	10	0.72021	0.14506
	Norm2	4	4	0.59050	0.08834
	Quantile	**4**	**8**	**0.72431**	**0.12695**

[a] In this study, RMSE cannot be used to compare between model due to difference scale

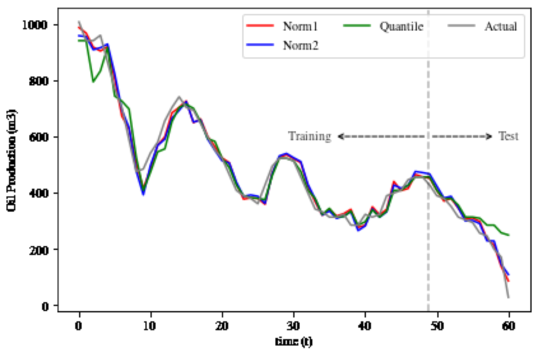

Fig. 9. Prediction for CB1 dataset

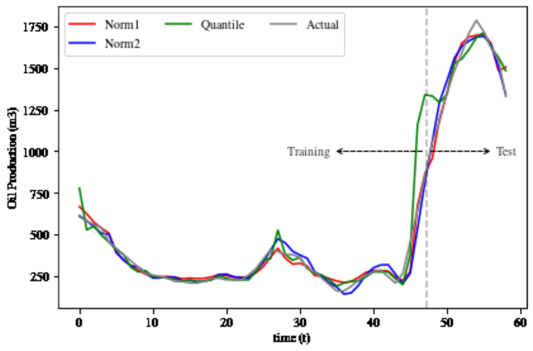

Fig. 10. Prediction for CB2 dataset

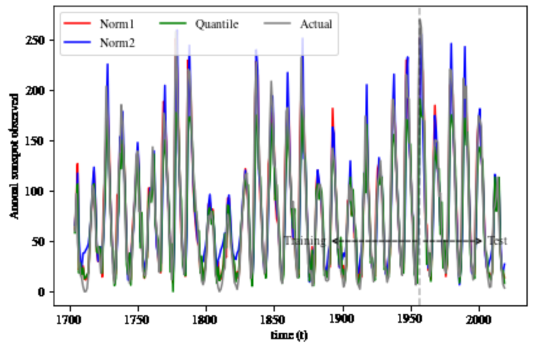

Fig. 11. Prediction for Sunspot dataset

4 CONCLUSION

In this paper, alternative of normalization method during pre-processing time-series data is proposed, in conjunction with HONN algorithm. Notably, in the case studies described in this paper, the UQT has promising method to improve prediction accuracy.

Future research may extent to evaluate UQT with various activation function which align with the range of normalized value (0 to 1). Sigmoidal function might be suitable for the UQT. It is also noted that another quantile transformation variance is more popular, which is normal quantile transformation (NQT). The comparison of UQT and NQT might be interesting to observe.

References

[1] M. A. Ahmadi, M. Ebadi, A. Shokrollahi, and S. M. Javad Majidi, "*Evolving artificial neural network and imperialist competitive algorithm for prediction oil flow rate of the reservoir*," Appl. Soft Comput. J., vol. **13**, no. 2, pp. 1085–1098, 2013, doi: 10.1016/j.asoc.2012.10.009.

[2] L. Sheremetov, A. Cosultchi, J. Martínez-Muñoz, A. Gonzalez-Sánchez, and M. A. Jiménez-Aquino, "*Data-driven forecasting of naturally fractured reservoirs based on nonlinear autoregressive neural networks with exogenous input*," J. Pet. Sci. Eng., vol. **123**, pp. 106–119, 2014, doi: 10.1016/j.petrol.2014.07.013.

[3] N. Chithra Chakra, K. Y. Song, M. M. Gupta, and D. N. Saraf, "*An innovative neural forecast of cumulative oil production from a petroleum reservoir employing higher-order neural networks*

(HONNs)," J. Pet. Sci. Eng., vol. **106**, pp. 18–33, 2013, doi: 10.1016/j.petrol.2013.03.004.

[4] J. Wang and J. Wang, "*Forecasting stochastic neural network based on financial empirical mode decomposition,*" Neural Networks, vol. **90**, pp. 8–20, 2017, doi: 10.1016/j.neunet.2017.03.004.

[5] [5] M. Fayaz, H. Shah, A. Aseere, W. Mashwani, and A. Shah, "*A Framework for Prediction of Household Energy Consumption Using Feed Forward Back Propagation Neural Network,*" Technologies, vol. **7**, no. 2, p. 30, 2019, doi: 10.3390/technologies7020030.

[6] T. Zhang and X. You, "*Improvement of the Training and Normalization Method of Artificial Neural Network in the Prediction of Indoor Environment*," Procedia Eng., vol. **121**, pp. 1245–1251, 2015, doi: 10.1016/j.proeng.2015.09.152.

[7] "SILSO | World Data Center for the production, preservation and dissemination of the international sunspot number." [Online]. Available: http://www.sidc.be/silso/home. [Accessed: 15-Apr-2020].

[8] "Compare the effect of different scalers on data with outliers — scikit-learn 0.22.2 documentation." [Online]. Available: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html. [Accessed: 15-Apr-2020].

[9] S. C. Hicks and R. A. Irizarry, "*When to use Quantile Normalization?*," bioRxiv, p. 012203, 2014, doi: 10.1101/012203.

[10] "sklearn.preprocessing.QuantileTransformer — scikit-learn 0.22.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html#sklearn.preprocessing.QuantileTransformer. [Accessed: 18-Apr-2020].

[11] "Basic Feature Engineering With Time Series Data in Python." [Online]. Available: https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/. [Accessed: 18-Apr-2020].

[12] "Forecasting time series: using lag features | Bartosz Mikulski." [Online]. Available: https://www.mikulskibartosz.name/forecasting-time-series-using-lag-features/. [Accessed: 18-Apr-2020].