**Evaluation Comparison of Gated Recurrent Unit (GRU) Neural Networks and Multivariable Linear Regression**

Ha G. Huy

STAT3013.N12.CTTT – EN

University of Information Technology

[20521385@gm.uit.edu.vn](mailto:20521385@gm.uit.edu.vn)

***Abstract – Deep learning algorithms have been implemented in several application domains and as such new algorithms have been developed to improve the performance. This paper presents a comparative study of three deep learning algorithms, namely Multiple Linear Regression, Multiple Non-Linear Regression and Gated Recurrent Unit, on the accuracy aspect.***

I. INTRODUCTION

Recurrent Neural Network (RNN) models are the most effective sequence model and they have been applied in many applications with temporal or sequential data. For performance improvement and specific purposes, people have developed many variants of RNN algorithms including Gated Recurrent Neural Network (Gated RNN) with two main concepts, Long Short -Term Memory (LSTM) and Gated Recurrent Unit (GRU). Long Short-Term Memory (LSTM) Recurrent Neural Network and Gated Recurrent Unit (GRU) Recurrent Neural Network have been applied successfully in sequence application. For example, people have been used extensively in polyphonic music processing, namely speech signal processing [1], natural language processing [2], and sequence generation [3].

The main element leads to the success of them is the gating network signals that is used to control the current input values and old memory to update the current activation neural layer [4]. The set of weights in Gated RNN can be changed after each time step.

Multivariable Linear Regression(MLR) and Multivariable Non-linear Regression (MNLR) are the primitive algorithms presenting simple functions. Regression Models are mostly used for functional Data Analyst. By D. Nguyen et al in 2011, Linear Regression Model has been used for Author age prediction via text dataset. In 2008, KW. Lau et al, have implemented Non-Linear Regression Model for Local Prediction on complex time series.

In this paper, we focus on the GRU RNN model and comparatively evaluate the performance of Multivariable Linear Regression (MLR) model on a public dataset. Using the BTC dataset, we will predict the Bit-coin value after each day based on the “Close” column. The observational data will be put in a 2123 x 1 matrix. In next section, we will introduce to some related research about Model Comparation and application of RNN Models. The Models’ architectures and settings will be also showed in details and lead to the conclusion at the last part.

II. RELATED WORK

Since RNN is introduced in 1986, there was many research papers about this concept and its variant. Some researchers revealed that some complex model such as RNN are more trustworthy than some simple one.

M. Ravanelli at al. [1] and Nana liu [2] applied RNN Models to process sound and speech signals.

*A. Graves* [4] showed that the power of a RNN Model to predict complex sequences just by processing a data point at a time.

Recently, A. Kumar [5] showed how important of model complexity in model selection. The article also showed the disadvantage of simple models (such as Linear Regression).

Nan Liu at al. [6], the research concluded that deep learning models are good at identifying objects but they need to be improved more to understand the relationship of objects. This means we need to build a complex architecture model to apply to a complex dataset.

Concluded by K. Cho at al. [7] and G. Chrupala at al. [8], a RNN Model can automatically learn a grammatical structure in a sentence.

J. Chung at al. [9] revealed that a GRU RNN Model is comparable to LSMT Model. Showing that GRU Model is also one of the best model in Time Series.

**III. DATA SOURCE AND MODEL ARCHITECTURES**

*A. Data Source*

*This paper focuses on predict bitcoin price was recorded daily from 6/1/2019 to 6/1/2022. The data frame consisting of 1097 observations.*

Chart, line chart

Description automatically generated

Figure . Bitcoin price from Dataframe after MinMax Normalization

*B. Recurrent Neural Network*

Sequential data types are typically processed by RNN. The RNN models have a recurrent hidden state as in

where is an *m-dimension* input vector at the current time (*t*), is the *activation function*, such as logistic function, or the Rectified Linear Unit (ReLU) [2, 10]. are defined as sized parameters where is a *n* x *m* matrix, U is an *n* x *n* matrix, and *b* is a vector with *n* elements. These sized parameters in this case are treated as two weights and one bias.

Tsungnan Lin at al. [11] showed that the gradients may vanish or explode after a number of timesteps if use such as a simple RNN. In [2-4], the idea of using some variants of RNN (LSTM and GRU) to solve the problem. We will present these two models below in details for our purposes.

*C. Long Short-Term Memory (LSTM)*

S. Hochreiter at al. [12] showed the idea of making a path that can let the gradient flow for a long time. F.A. Gers at al. [13] revealed that a crucial improve of LSTM model is make the *weight parameters* can be adaptively change through each time step by making *gated self-loop* to control the *weights* and the time for each integration can be adapted dynamically. The time in each integration can be also changed by *fixed parameters (weights)* with a set of suitable input values since the output of the model is the time for each time step.

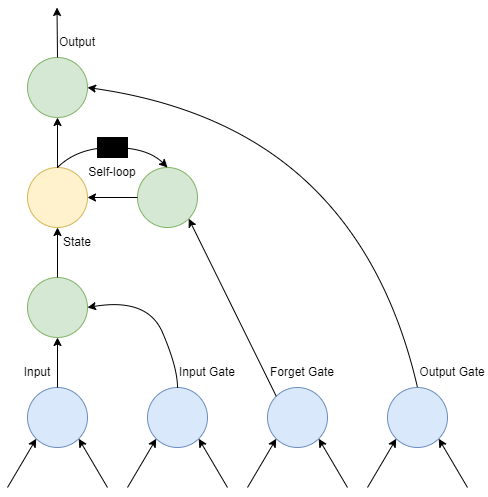


Figure . Diagram of a “cell” in LSTM model. Each cell connects with each other instead of regular hidden unit as in Graph Neural Network. Each input features are the result of a regular neural unit. The values can be stored in the State Unit of the cell. The State Unit has a linear self-loop with the weights are controlled by Forget Gate. All the Gate-Units can use non-linear sigmoid function, while the Input can use any non-linear function. The black square is the delay of each time step.

The abstract of a cell of LSTM model show in the *Figure 1.*

*LSTM RNN* use “LSTM cells” that have a self-loop. Each cell has the input and output values like a primitive recurrent network with more parameter and use *a system of gates* to control the information flow. As mentioned in the previous part, the *parameters (weigths)* of the self-loop are controlled by the *Forgot Gate* (time step t and cell i) to calculate the parameter within 0 to 1 using Sigmoid function.

where is the current input vector and is the current vector in hidden layer, contains the output values of the cell. is the *bias value, weight* of input and *recurrent weight* of forget gate. The LSTM internal state is updated as follows:

= +

where is calculated like the forget gate but with its own parameter

The output can be shut off by the output gate and can be calculated by:

*D. Gated Recurrent Unit (GRU) RNN*

Get the idea from LSTM architecture, Gated Recurrent Unit inherited some its necessary features. Answered in [7, 8], the main difference between LSTM and GRU is that GRU have a single gating unit to update the state unit and control the forgetting factor.

By J. Chung at al. [9], the result showed that GRU RNN are much more advance than LSTM in most cases. There is a variant of GRU RNN, e.g. the Minimal Gated Unit (MGU) RNN which only use one gate equation and give the compatible performance (in some cases) to the LSTM RNN.

The update equations:

where *u* is update gate and *r* is reset gate. Update gate value is followed by the equation:

and the reset gate value equation is:

In this paper, we will focus on GRU RNN only and compare with the very basic Linear Regression model.

*E. Multivariable Linear Regression (MLR)*

Maverick [14] showed that multivariable Linear Regression (MLR) models have been used in various area to predict or classify data . MLR is a very basic model that take a vector containing input values and vector containing the weight values of each feature. We can define the output value by the following formula:

where is the predicted value at the output.

In this case, is a parameter(weight) to multiply with feature and sum up all the value from feature to get the predicted value (). Each weight in a MLR model is show that how importance of the data. If the weight is a positive number, increasing the value of its feature will also increase the predicted value. The predicted value in output neural will also decrease if the feature’s value is increased while the weight is negative. This also concluded that if the weight is *zero,* the feature is mean nothing in our model.

In real-world situation, the MLR is sometimes more complicated with *intercept parameter (bias):*

where b is the bias value.

This will keep our model is still described as a straight line but does not need to go through the *origin point.*

Linear regression, basically, is a very basic algorithm and has a lot of disadvantages. But it will give us an overall view of *Machine learning algorithm.*

*F. Multivariable Non-Linear Regression (MNLR)*

Multivariable Non-Linear Regression is a version of regression analysis. The observational data are put in a non-linear model. We can define the output of the model by the following formula:

where y is the output value. The function is nonlinear and takes and as the vector of independent variables and bias values respectively.

IV. MODEL SETTINGS

A. GRU RNN

In this task, we built a RNN model prediction with GRU layer. Each layer has 20% dropout rate to prevent the result from overfitting and the activation function is set as *tanh* function. The model is performed by Python language using *Keras library* with *Tensorflow* as the backend library. The hyperparameter of this model is set as Learning rate: 0.0001, Hidden Unit: 64, Batch size: 256, Epoch: 100, Optimizer: Adam.

In the dataframe, we divined the dataset into three set. 70% for training dataset, 20% for validation dataset and 10% for testing dataset.

Training data set:

Chart, line chart

Description automatically generated

Figure . Training set (70%)

Validation data set:

Graphical user interface, chart

Description automatically generated

Figure . Validation set (20%)

Testing data set:

Graphical user interface, chart, line chart, scatter chart

Description automatically generated

Figure . Testing set (10%)

B. MLR Model

In this test, we run a very basic MLR Model using Sklearn library in Python language. The dataset is split to training set (80%) and testing set (20%).

C. MNLR Model

In this model, we used Random Forest Regression on the test. The dataset is split to training set (80%) and testing set (20%).

V. RESULT AND CONCLUSION

In this section, we will show the test results on three mentioned models and evaluate the result by MAPE Loss, RMSE Loss and APE Loss.

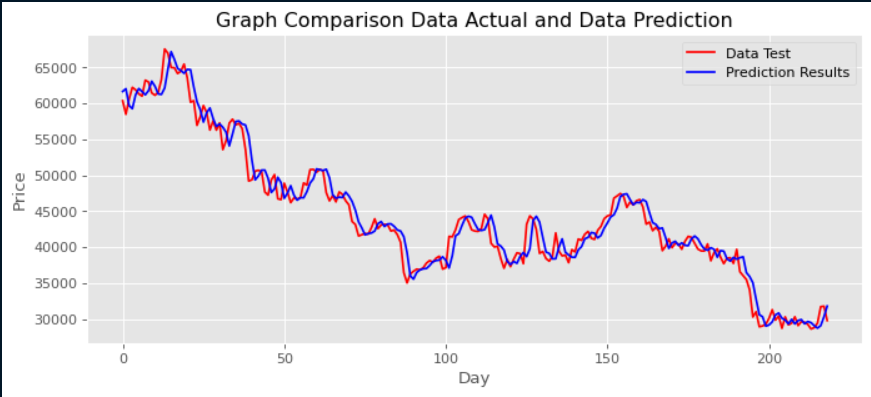


Figure . Compare the actual value and predicted value in GRU RNN Model

Chart, line chart, histogram

Description automatically generated

Figure . Compare the actual value and predicted value in Multivariable Linear Regression Model

Chart, line chart, histogram

Description automatically generated

Figure . Compare the actual value and predicted value in Multivariable Non-Linear Regression Model

|  |  |  |  |
| --- | --- | --- | --- |
|  | MAPE | RMSE | APE |
| GRU RNN | 23.6 | 1772 | 0.03 |
| MLR | 3.69 | 2131 | 0.03 |
| MNLR | 23.1 | 1885 | 0.02 |

Based on the result, we can conclude that MLR Model is the less reliable, since the function is just described in a straight line when it has the highest RMSE value. The GRU RNN has the highest accuracy based on the RMSE value. Otherwise, MLR has the smallest MAPE value. S. Kolassa’s [15] research showed that different accuracy measures are minimized by different functionals of the unknown future distribution. Answer in [16], the biased forecasts are much more awarded by MAPE value, explained for the unexpected MAPE value of MLR Models

REFERENCES

[1] M Ravanelli. Light Gated Recurrent Units for Speech Recognition. arXiv:1803.10225, 2019

[2] Nana Liu. Music Emotion Recognition Model Using Gated Recurrent Unit Networks and Multi-Feature Extraction. Hindawi || Article ID: 5732687, 2022

[3] A Graves. Generating Sequences With Recurrent Neural Networks. arXiv:1308.0850, 2013

[4] Ian Fellow. Deep Learning Book pg 407-410. 2015

[5] A Kumar. Model Complexity & Overfitting in Machine Learning. [Model Complexity & Overfitting in Machine Learning - Data Analytics (vitalflux.com)](https://vitalflux.com/model-complexity-overfitting-in-machine-learning/#What_is_model_complexity_why_its_important), 2022

[6] Nan Liu, Shuang Liu and Yilun Du. Learning to Compose Visual Relations. arXiv:2111.09297, 2021

[7] K Cho, B v MerrienBoer, D Bahdanau and Y Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259, 2014

[8] G Chrupala, A Kádar and A Alishahi. Learning language through pictures. arXiv:1506.03694, 2015

[9] J Chung, C. Gulcehre, K Cho and Y Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555, 2014

[10] Quoc V Le. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. arXiv:1504.00941, 2015

[11] Tsungnan Lin, Bill G. Horne, Peter Tino, C. Lee Giles. Learning long--term dependencies is not as difficult with NARX recurrent neural networks, 2001

[12] S Hochreiter and J Schmidhuber. Long short-term memory. Neural Computation 9(8):1735-80, 1997

[13] F A Gers, J Schmidhuber and F Cummins. Learning to forget continual prediction with LSTM. IEEE Xplore, 10.1049/cp:19991218, 2002

[14] Maverick 01. Predicting using Linear Regression in R. analyticsvidhya.com, 2020

[15] S. Kolassa. Why the “best” point forecast depends on the error or accuracy measure, 2020

[16] [accuracy - What are the shortcomings of the Mean Absolute Percentage Error (MAPE)? - Cross Validated (stackexchange.com)](https://stats.stackexchange.com/questions/299712/what-are-the-shortcomings-of-the-mean-absolute-percentage-error-mape)