Noisy-Exponential Activation Function (NEAF)

Enrique Boswell Nueve IV
Department of Computer Science, Northern Illinois University
Email: enriquenueve9@gmail.com

***Abstract*— Due to the volatile nature of financial markets, creating accurate forecasts for financial assets is currently a rather difficult undertaking. To counter this difficulty, financial firms are employing Recurrent Neural Networks (RNN). Before they can forecast with any accuracy, RNNs have to go through a training process in which they analyze the data and then construct a function expressing said data. During the training process, RNNs can easily become corrupted due to the issues of exploding and vanishing gradients both of which are forms of neuron saturation. Neuron saturation forces the network to have to be retrained. In this paper, the author propose an activation function called the Noisy-Exponential Activation Function (NEAF) to counter this problem. Exploding and vanishing gradients are due to overflow or underflow which prevents the network from being able to update through backpropagation. Through NEAF, noise will be added based on the relative distance from the outputs of a linear function and hard sigmoid. Unlike previous noisy generating activation functions, NEAF introduces a unique attribute in which the noise added to the system is dampened over an exponential function. This prevents exploding and vanishing gradients but also continues to allow the activation function to have non-linear attributes. NEAF has been created for the intended purpose of Financial Forecasting. NEAF has a strong balance between functionality, in regard to reducing forecast error, but also is computationally cost-efficient, allowing fast real-time forecast. Extensive experiments with NEAF over historical stock data from the S&P 100 on a Recurrent Attention Unit, consistently showed improved performance through the use of NEAF.**

I. INTRODUCTION

In regard to the act of investing, the investor puts money into an investment with the intention that the return over time will be larger than the same money in the present. The intention for investing is rather clear, yet the method in which to make an economic profit through investing appears to be shrouded in mystery. The common consensus of financial analysts to moderately forecast the future lies in interpreting the past information of an asset. Although using this method is certainly not perfect, to any real degree of accuracy, forecasting based off of past trends is currently the accepted method. A common model or variants of Auto-Regressive Integrated Moving Average (ARIMA) is used for extended financial forecasting. Although ARIMA is still used for extended forecasting, the preferred method of the times is using a Recurrent Neural Network (RNN).

Recurrent Neural Networks are a type of Deep Network which includes the past state in order to give context to present information. This improves the prediction of the forecast, which is why financial analyst, use this method. This paper presents a new activation function called the Noisy-Exponential Activation Function (NEAF). NEAF improves the performance of RNNs' which prevents the problem of neuron saturation discussed in the following section.

II. PROBLEM DESCRIPTION

The core issue with RNNs is the trade off that exists between the desired result of accurate forecasting and the difficulty of initially training the RNNs. If the RNN learns the data, the forecast is highly accurate; however, this is not always the case. During training sessions, failure commonly occurs due to neuron saturation forcing the trial to be restarted. Neuron saturation, is the case in which overflow or underflow occurs during backpropigation of the error calculation, which prevents the RNN from further learning.

The aforementioned issue is why this experiment was proposed. It tested a new technique designed to prevent neuron saturation. This technique is called the Noisy-Exponential Activation Function (NEAF). NEAF was tested on a Deep Recurrent Attention Unit (RAU), a state of the art RNN, against an RAU without NEAF. These two networks were tested with the same data sets. The data sets consisted of historical data of the assets listed on the S&P 100 as of March 2019. The variables used in the data set were Open, High, Low, Close, and Mid-day average. The input variables were the previous day's Open, High, Low, Close, and Mid-day average while the target variable was the next day's Mid-day average. The two properties that were measured on both models over a five, ten, and fifteen day forecast were average Mean-Square Error and average Percent Error.

III. NETWORK

For this experiment, a Deep Recurrent Attention Unit with a Step-Down network was used to test NEAF. The Recurrent Attention Unit (RAU) is a variant of the Gated Recurrent Unit (GRU) which has an Attention Gate. Unlike GRU, "RAU can not only qualify which information is preserved over time, but also suppress redundant information through the use of the Attention Gate."[2] Overall, the RAU consists of four gates: Update, Reset, Candidate, and Attention. In general, gates consist of two parts. The first part is a weight layer, or a series of weight layers, that takes in new information. The second part is a different weight layer or layers that takes in the past state. Each of these gates serve a unique purpose. The first of which is the Update Gate.

The Update Gate is used to determine which information of the previous state is brought into the current cell state. This is expressed in the following formula such that its inputs are the previous hidden state h_{t-1} and the present input x_t. The σ

symbol stands for the sigmoid activation function, while U^z stands for recurrent weight and Wz stands for the new input weight.

$$z_t = \sigma\big(W^z x_t + U^z h_{t-1}\big) \tag{1}$$

The next gate is the Reset Gate which is denoted by r_t. This gate serves the purpose of deciding how much information to retain in the hidden state, which acts as memory system. The gate outputs a range of values between 0 and 1. The values between 0 and 1 represent the percentage of information deleted from the memory. Therefore, the output of the Reset Gate represents the information left after the deletion in memory. The Reset Gate is expressed by the following formula.

$$r_t = \sigma\big(W^r x_t + U^r h_{t-1}\big) \tag{2}$$

Following the Reset Gate is the Candidate Gate, which serves the purpose of creating the new hidden state value. The hidden state serves the purpose of transferring relative information between states. The hidden state is expressed by the notation $\tilde{h}_t$ and the following formula.

$$\tilde{h_t} = \tanh\big(W^c x_t + U^c h_{t-1}\big) \tag{3}$$

The last gate, the Attention Gate, is the unique aspect of an RAU. It serves the purpose of selecting which information is important in the current data and ignoring redundant information. The Attention gate is denoted by $\hat{h}_t$ and the following formula.

$$\hat{h_t} = \tanh\big(W^a \gamma_t\big) \tag{4}$$

The Attention Gate is built on an attention mechanism which consists of two parts: the scoring function and the probabilistic calculation performed through softmax. The general scoring function was used and is shown by the following equation.

$$\alpha_t = h_{t-1}^T W^{\alpha_t} x_t \tag{5}$$

The other part of the attention mechanism, the probabilistic calculation, is shown by the equation below.

$$\gamma_t = \frac{\exp\big(\alpha_t\big)}{\sum_{t=1}^{T} \exp\big(\alpha_t\big)} \tag{6}$$

After passing the values through the gates, the outputs are connected through the following formula to express the new state value.

$$h_t = \big(1 - z_t\big) \odot h_{t-1} + \big(\frac{z_t}{2}\big) \odot \big(\tilde{h_t} + \hat{h_t}\big) \tag{7}$$

Due to the differences in the dimensions of RAU's output of five dimensions (Open, High, Low, Close, and Mid-day average) compared to the target data dimension of one (next day's Mid-day average), a Step-Down network was added at the output of the RAU to convert the output to the same dimension size as target. The Step Down network is expressed by the following equation.

$$y = \sigma\big(W^T h_t\big) \tag{8}$$

Finally, the error function used for this network was the Pseudo-Huber loss function, shown by equation 9.

$$L_\delta(y, \hat{y}) = \delta^2\Big(\sqrt{1 + (\frac{y - \hat{y}}{\delta})^2} - 1\Big) \tag{9}$$

IV. SOLUTION

As stated, the main issue that occurs when training Deep Recurrent Networks is the issue of neuron saturation. In an attempt to solve this problem, the author decided to investigate modifications to one of the most influential aspects of Deep Networks, the activation function.

The activation function maps values to a desired space and adds non-linearity into the values flowing through the network. Deep Networks consist of weight matrices with activation functions between them. Without activation functions between the weight layers, the network would then resemble a very complex linear regression model. However, one of the key advantages of Deep Networks is that they can model nonlinear equations due to the activation functions with their non-linear features being present between the layers.

Often, to address the issue of neuron saturation, the activation function is modified since it controls the domain of the values flowing through the network. A unique approach in the paper "Noisy activation functions" proposed the idea of adding generated Gaussian noise to values in the activation function relative to their distance between a linear function and a clipped activation function [1]. This method did show improvement on the network's performance in terms of calculated lowered error, however, due to the method of the activation function behaved as a piece-wise linear function, which prevents the flow of non-linear values to be introduced into the network. That is why in this paper, an original idea of how the noise is to be added to the activation function is being proposed.

This method is called the Noisy-Exponential Activation Function (NEAF). NEAF injects noise into the network like previous noise generating activation functions; however, the noise is dampened over an exponential function. This prevents over-fitting but also continues to allow the activation function to have non-linear attributes. For this experiment, any gate that used sigmoid was replaced with NEAF. Specifically, the Update and Reset Gate, had NEAF implemented. NEAF is calculated as follows.

A. Linear Function

To begin, the values outputted from a weight layer are passed through a linear function as expressed by equation (10).

$$\phi(x) = .25x + .5 \tag{10}$$

B. Exponential-Noise Generator

Upon passing the values through the linear function, they are passed through equation (11).

$$\chi(\phi(x)) = \begin{cases} 1 - .01|\xi|\epsilon^{x - \sigma(x)} & x > .975 \\ x & o/w \\ .01|\xi|\epsilon^{\sigma(x) - x} & x < .025 \end{cases} \tag{11}$$

Equation (11) will find the distance between the linear values and the distance from hard sigmoid (12). The calculated distance between the values will act as a scaling value in regard to how much noise is added to each point. These values are then passed through an exponential function. By passing the values through the exponential function, non-linear values are still able to be introduced to the network.

$$\sigma(x) = \begin{cases} 1 & x > .975 \\ .25 & .025 \leq x \leq .975 \\ 0 & x < .025 \end{cases} \tag{12}$$

The noise, ξ , is generated randomly from a Gaussian distribution with the domain (-1,1). The noise is then put through an absolute value function to ensure the noise reduces, neuron saturation, rather than increases. The calculation of NEAF can be summarized by Algorithm 1. Although for this test, NEAF was applied to sigmoid, NEAF can be generalized to work on other closed-range activation functions with linear attributes. For example, NEAF can be generalized to work on hyperbolic tangent. This application of NEAF on hyperbolic tangent is demonstrated by the Algorithm 2 below. Htanh, as listed in the algorithm below, stands for Hard Hyperbolic Tangent. Htanh is calculated by Htanh(x) = max(-1,min(1,x)).

Algorithm 1 NEAF: Sigmoid

$y \leftarrow .25x + .5$ ▷ Eq(10)

$z \leftarrow \begin{cases} 1 - .01|\xi|\epsilon^{y-\sigma(y)} & y > .975 \\ y & o/w \\ .01|\xi|\epsilon^{\sigma(y)-y} & y < .025 \end{cases}$ ▷ Eq(11)

return z ▷ σ is Eq(12)

Algorithm 2 NEAF: Hyperbolic Tangent

$y \leftarrow x$

$z \leftarrow \begin{cases} 1 - .01|\xi|\epsilon^{y-Htanh(y)} & y > .975 \\ y & o/w \\ -1 + .01|\xi|\epsilon^{Htanh(y)-y} & y < -.975 \end{cases}$

return z

C. Visual of NEAF

To further explain NEAF, a visual (Fig 1) is provided. The graph shows that values close to zero or one, which are the max range of the sigmoid function, are pushed away due to the Gaussian noise which is introduced through NEAF. Additionally values which are farther away from the origin have have been displaced farther by the noise due to the relatively larger distance from the linear function value and the hard sigmoid value. It can also be observed that the noisy values have a skewed distribution due to the noise being passed through an exponential function. This visual illustrates that NEAF prevents overflow and underflow error by preventing values from reaching their limits of the respective activation function, but also maintains non-linearity as seen through the skewed shaped sample values.

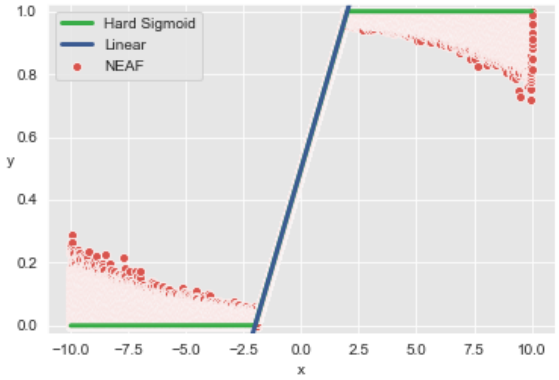

Fig. 1. NEAF sample points.

V. Experiment

For this experiment, NEAF was tested on a Deep Recurrent Attention Unit to forecast financial data. Historical stock data was retrieved through the use of the Alpha Vantage API for the assets listed on the S&P 100 as of March 2019. Of the assets listed on the S&P 100 as of March 2019, data was retrieved as far back as 20 years ago. For the assets listed on the S&P 100 as of March 2019 which had not been listed on the S&P 100 for the last 20 years, data was retrieved as far back as their initial date of being listed onto the S&P 100. Being that the collected data was stock data, the data only accounted for days in which the markets were open. This excludes days such as weekends and national holidays.

The data retrieved from the Alpha Vantage API came in the form of time-series data with the daily values for open, close, high, and low. For this experiment, the variables open, close, high, low, and mid-day average, which was derived from a day's high and low value, were used as input variables. As a target variable, the next day's mid-day average was used. For the sake of a baseline, testing occurred on an a Deep Recurrent Attention Unit without NEAF and a Deep Recurrent Attention Unit with NEAF.

Testing consisted of performing a five, ten, and fifteen day forecast. For training and testing the network, all data for the respective asset that was collected was used for training the network. The only data that was not used was the last five, ten, or fifteen days listed in the data, which was used as a test data set for the respective test. The two metrics of the network that were measured in this experiment were Percent Error and Mean-Square Error. The results of this experiment are discussed in the following section.

VI. Results

Testing of NEAF was performed on an Deep Recurrent Attention Unit to create a five, ten, and fifteen day forecast over financial assets. As a baseline, a Deep Recurrent Attention Unit without NEAF was also used to create a five, ten, and fifteen day forecast over financial assets. The two metrics

that were tested in this experiment were Percent Error and Mean-Square Error. The assets tested consisted of the listed stocks on the S&P 100 as of March 2019. The results varied between the five, ten, and fifteen day forecast. Across all three forecast, the error was lowered by NEAF to differing degrees.

Of the three different test the fifteen day forecast saw the largest difference in average percent error between the Deep RAU with NEAF and the Deep RAU without NEAF. The average percent error for the fifteen day forecast without NEAF was **%14.1904** while the average percent error with NEAF was only **%9.1922**. NEAF provided a **%4.9982** percent error improvement compared to the Deep RAU without NEAF. The improved average error percent with NEAF decreased as the length of the forecast decreased. For the ten day forecast, NEAF provided an improved percent error of **%2.0767**. Finally, with the least amount of improved percent error, the five day forecast with NEAF provided an improved percent error of **%.6917**.

For the fifteen day forecast the average MSE without NEAF was **0.0107** while with NEAF was **0.0178**. This outcome of increased MSE with lessened error could be explained because NEAF acted as a regressor due to the noise injection. This could have prevented over-fitting, allowing for an improved error rate but a larger MSE [3]. This holds true for each forecast except the ten day forecast, where the MSE is lower for the model with NEAF compared to the model without NEAF. This anomaly causes further testing to be required to explain the differences between the MSE of the two models for the different forecast lengths. The results of the experiment can be summarized by the table below.

Results		
	Sigmoid	NEAF
5 Day Error	6.6986	6.0069
5 Day MSE	0.0031	0.0036
10 Day Error	11.2689	9.1922
10 Day MSE	0.0107	0.0049
15 Day Error	14.1904	9.1922
15 Day MSE	0.0107	0.0178

VII. CONCLUSIONS

The results show that NEAF was able to improve the network's forecast by injecting noise into the activation function over an exponential function which is inferred by the improved error rate. These results promote that methods such as NEAF that improve recurrent networks' reliability and forecasting ability may one day allow recurrent networks to be easily trained and have more accurate predictions. However, more testing is required.

ACKNOWLEDGEMENTS

The author would like to give special thanks to Dr. Alhoori from the Northern Illinois University Computer Science department for advising during this research project. The author would also like to give special thanks to Doug Zubka and Dr. Friebus-Flaman for their support during this project. Finally, the author would like to give special thanks to Elizabeth Galetto for providing the numerous cups of coffee needed to perform this research.

REFERENCES

[1] Gulcehre, Caglar, et al. "Noisy activation functions." International conference on machine learning. 2016.

[2] Guoqiang Zhong, Guohua Yue, and Xiao Ling. "Recurrent Attention Unit." arXiv preprint arXiv:1810.12754 (2018).

[3] Zur, Richard M., et al. "Noise injection for training artificial neural networks: A comparison with weight decay and early stopping." Medical physics 36.10 (2009): 4810-4818.