Stock Price Prediction Using RNN

Moaz Mohamed
The University of Adelaide
Adelaide SA 5005
a1779177@student.adelaide.edu.au

Abstract

the purpose of this project is to study the ability of machine learning algorithm at predicting future stock market prices, the challenge is an effective model that can capture long term dependencies in memory like elements and use that feature to make predictions. RNN and LSTM have been utilize to tackle such problem. With having LSTM having a better performance than RNN with RMSE value of 7.63 , 10.63 *respectively*

1. Introduction

Stock market prediction is the act of trying to determine the future value of company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield a significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable.

In this era, where machine learning and deep learning is at full swing, An attempt will be made to use machine learning as a predictor to tame the inherent unpredictability of the stock market.

2. Background

there are certain constraints and limitation that prevent the usage of typical neural network architectures in time series data-sets. In vanilla neural network (NN) and convolution neural network (CNN). both architectures cant capture long term dependencies among the data. in the example in eq 1 the answers for both sequences are 13,16 the machine in this case the machine learning algorithm should be able to output both answers using/looking backwards/previous values hence the output is dependent on the previous values. Which inspired researchers to come up with new architectures that can capture the sequential information present in the data or the dependency of previous values while making prediction. While NN and CNN are suitable for such type of tasks that is such tasks require some sort of memory function that can hold information that NN and CNN lacks unlike recurrent neural networks (RNN) [1].

$$\begin{array}{l}\{10, 11, 12, ?\}\\ \{4, 8, 12, ?\}\end{array} \tag{1}$$

RNN networks depicted in figure 1 are similar to a typical neural network the difference centralize around the feedback loop in the weights that are being feed into the same weights and that feedback connection is the representation of the memory element that makes RNN the edge where having a memory like element is necessary.

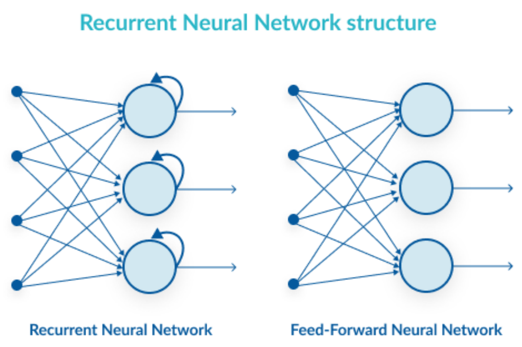
Recurrent Neural Network structure
Recurrent Neural Network
Feed-Forward Neural Network

Figure 1. RNN Basic Structure

3. Methodology

3.1. RNN

The main motivation for RNN structure is to have a memory element that can share data across layers. Working recursively in eq 3 governs the flow of information from input feature vectors X to output vector O where the output vector O is a function of the h_t and matrix transformation V similarly, eq 2 governs the flow of the information horizontally and transforming the information from the first hidden layer to the last hidden layer. Due to the flow of information in those two orthogonal directions governed by eq 3 and 2 what ever information captured in the first layer will be passed in to the next layer thus RNN is capable of capturing long term dependencies effect.

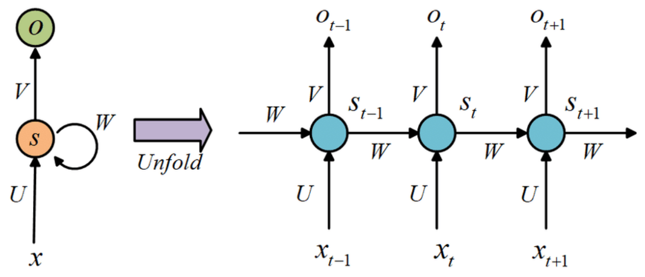

Figure 2. RNN

$$h_t = \tanh\left(ux_t + wh_{t-1}\right) \tag{2}$$

$$\hat{O}_t = \mathrm{Soft\,max}\left(Vh_t\right) \tag{3}$$

3.1.1 RNN Drawbacks

RNN architecture inherent drawback is exploding and vanishing gradients. Performing backpropagation in time (BPTT) of RNN loss functions the term $\prod_{m=k+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}$ in both Eq 5 and 4 can become extremely small and close to zero (vanishing) or extremely big (exploding) which in tern affects the gradients calculated when performing BPIT [2] [3]. And the mechanism is as follows.

$$\frac{\partial L}{\partial W} = \sum_{i=1}^{T}\sum_{k=1}^{i}\frac{\partial L_i}{\partial y_i}\frac{\partial y_i}{\partial h_i}\prod_{m=k+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}\frac{\partial^+ h_k}{\partial W} \tag{4}$$

$$\frac{\partial L}{\partial U} = \sum_{i=1}^{T}\sum_{k=1}^{i}\frac{\partial L_i}{\partial y_i}\frac{\partial y_i}{\partial h_i}\prod_{m=K+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}\frac{\partial^+ h_k}{\partial V} \tag{5}$$

Eq 2 have been rewritten as follows.

$$\begin{aligned}\phi_j &= Ux_j + wh_{j-1}\\ h_j &= \tanh\left(\phi_j\right)\end{aligned} \tag{6}$$

what we are interested in is the term $\prod_{m=k+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}$ so lets expand and take the Jth term. Which is the partial derivative of

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial h_j}{\partial \phi_j}\cdot\frac{\partial \phi_j}{\partial h_{j-1}} \tag{7}$$

the first element is Eq 7 is α, when taking the derivative of tanh the maximum value is 1

$$\frac{\partial h_j}{\partial \phi_i} = \mathrm{diag}\left(\tanh'\left(\varnothing_j\right)\right) \tag{8}$$

the second element is Eq 7 is β and equal to

$$\frac{\partial \phi_j}{\partial h_{j-1}} = w \tag{9}$$

rewriting Eq 7 and taking the magnitude

$$\left\|\frac{\partial h_j}{\partial h_{j-1}}\right\| = \left\|\mathrm{diag}\left(\tanh'\left(\phi_j\right)W\right)\right\| \tag{10}$$

so the term $\prod_{m=k+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}$ is bounded by $(\alpha\beta)^{(i-k)}$ and if the term $(\alpha\beta)$ is bigger than 1 the gradients will explode and if its less than 1 the gradient will vanish hence the term $\prod_{m=k+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}$ directly affects both Eq 5 and 4 which intern is affects the BPIT mechanism.

$$\left\|\frac{\partial h_i}{\partial h_k}\right\| = \left\|\prod_{m=1+1}^{i}\frac{\partial h_m}{\partial h_{m-1}}\right\| \leqslant \prod_{m=k+1}^{i}\alpha\beta \tag{11}$$

Fortunately, when encountering this such problem it can be mitigated using gradient clipping to have a cut off value for the gradients or use gradient norm scaling changing the derivatives of the loss function to have a given vector norm when the L2 vector norm (sum of the squared values) of the gradient vector exceeds a threshold value. Another alternative is using LSTM network.

3.2. Long Term Short Memory (LSTM)

The motivation for inventing LSTM network was to counter the drawbacks for RNN networks, hence the vanishing and exploding problem Which limited RNN to handling short term sequences. as the sequences got longer RNN performance got worse and LSTM was answer to that problem [4].

LSTM cell consist of number of gates that are able to modulate the input. the gates are (input, forget and output)

Forget Gate is responsible for deciding which information gets removed from the cell state C_{t-1} as described in figure 3 and 12. if f_t is 1 the information will be included into the cell state and not of its 0.

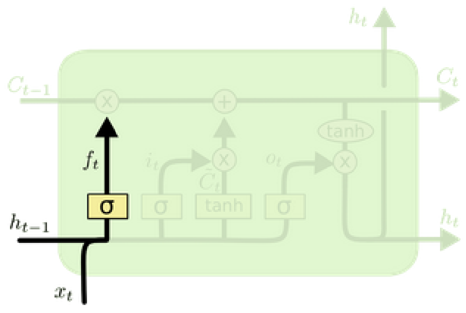

Figure 3. Forget Gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{12}$$

Input gate is responsible for deciding what new information should be stored in the cell state. that is achieved in two steps. the tanh layer holds the new information in holding state and if the information is passed to the cell state is decided by the sigmoid layer as described in figure 4 and Eq 13,14

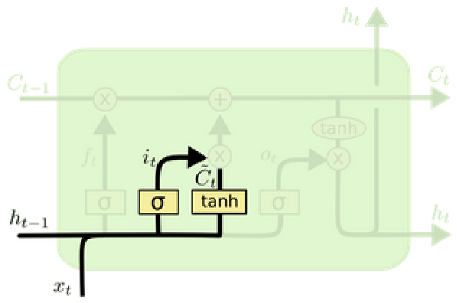

Figure 4. Input Gate

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{13}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{14}$$

Output gate is responsible for modulating the C_t state through the sigma layer with using tanh to push the values to be between negative and positive 1 as described in figure 5 and Eq 15

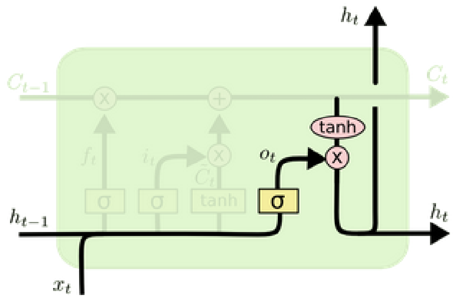

Figure 5. Output Gate

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{15}$$

4. Experimental Analysis

In the provided notebook both RNN and LSTM share the same network structure. it consist of 4 layers in total with all sharing the same number of neurons with drop out layer after each layer. all drop out layers share the same drop value except the last layer.

The data-set has been standardized and split to create a validation set of last 200 days of the training set. both networks will have an input of 120 days time-steps. both models are uni-variate models that will predict the opening price of the stock the next day. Both models have been trained with a max epoch of 2000 and a batch size of 512 while using Adam optimizer with its default settings. Early stopping method was used with a patience of 200 epochs to stop the training of the model tended to over-fit.

RNN tends to over fit when attempting to raising the drop out values to combat this problem either the model loss will fluctuate and becomes unstable or over fit when using a modest drop out value. LSTM also exhibit similar trend when using a high drop out value even with using a relatively low number of neurons as seen in figure 6 and 8. Through hyper-parameter tuning through units values of $[20, 30, 40, 50, 60, 80, 120, 150]$, drop out values of $[0.1, 0.2, 0.25, 0.3]$ and drop out for the last layer of $[0.1, 0.2, 0.3, 0.4, 0.5]$ the best performing parameters for both networks have been found as seen in table 1 and 2 with a RMSE value of 7.63 for LSTM network and 10.63 for RNN network with their prediction plotted in Figures 7 and 9

Algorithm	Units	Drop Out 1	Drop Out 2
LSTM-Overfitting	50	0.25	0.5
LSTM-Overfitting	150	0.25	0.2
LSTM-Best	80	0.1	0.5

Table 1. LSTM Performance

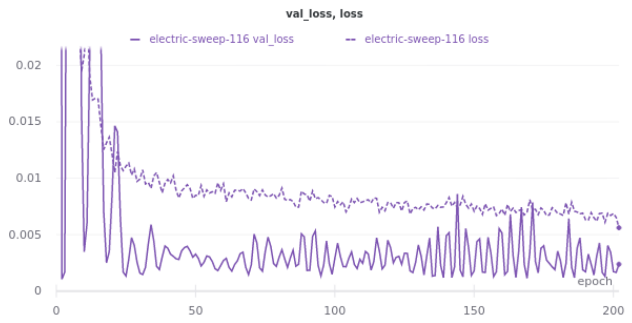

Figure 6. LSTM Overfitting

Algorithm	Units	Drop Out 1	Drop Out 2
RNN-Overfitting	160	0.2	0.25
RNN-Overfitting	80	0.2	0.25
RNN-Best	80	0.25	0.4

Table 2. RNN Performance

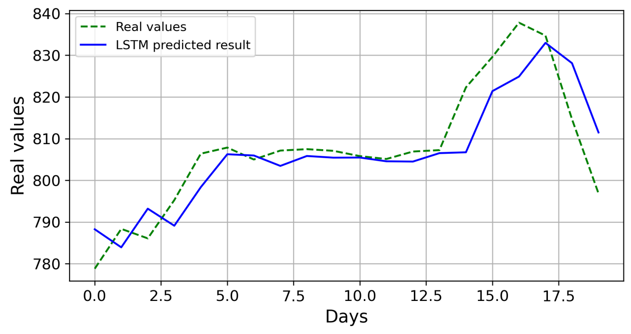

Figure 7. LSTM Predictions

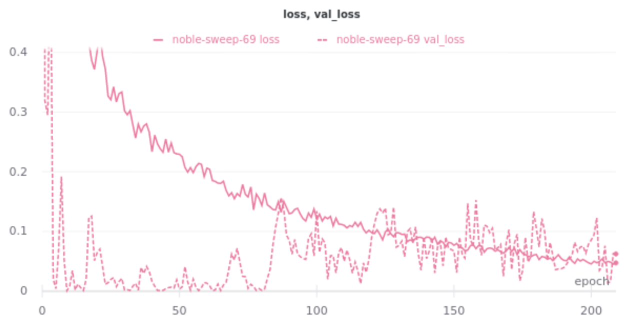

Figure 8. RNN Overfitting

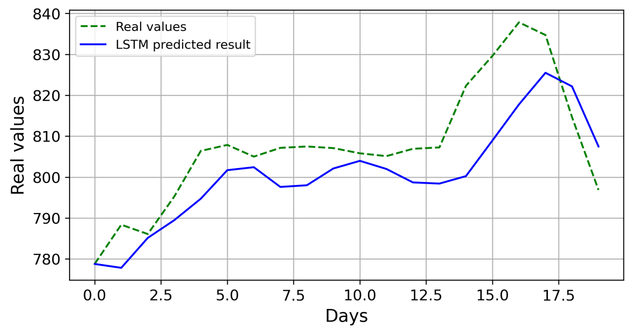

Figure 9. RNN Predictions

5. Conclusion

LSTM did preform better than RNN network. both networks were unstable with high drop out values which limited the ability to control over-fitting, it should be interesting comparing the performance of drop out with L1 and/or L2 normalization and see if that might counter the problem of unstable performance. In general LSTM does have Higher capabilities to capture long term dependencies setting high and low time-steps for both network and seeing how each network preform should be interesting. also in this experiment only uni-variate in theory having more data by using a multivariate model by combing closing and opening prices to predict future opening price should help the model capture more useful information to aid the model during training and prediction. even more interesting is having a multi-modal model that can combine news sentiment with other stock prices to make a more accurate predictions.

6. Code

all the code for this assignment is provided in this GitHub repo `https://github.com/RoastedKernel/uni_Machine_Learning/tree/master/Deep_Learning_Fundamentals/assign_3`

the Hyper-parameters results for the RNN network are provided here using weights and biases IDE.

Link `https://wandb.ai/roastedkernel/RNN_4Layers/table?workspace=user-roastedkernel`

LSTM are in here `https://wandb.ai/roastedkernel/LSTM_4Layers/table?workspace=user-roastedkernel`

References

[1] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar 2020.

[2] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. B. Schön, “Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness,” 2020.

[3] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” 2013.

[4] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.