Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction

Hengjian Jia
Colyton Grammar School
E-mail: henryjia18@gmail.com

1 Acknowledgements

I would like to thank Alfie Howard for providing me with the necessary code to preprocess data. I would also like to thank François Chollet for creating the Keras framework which was used to create all the necessary code for this paper.

2 Abstract

We explore the effectiveness of using Long Short Term Memory Networks (LSTM) for predicting stock prices. We will construct, test and compare a range of architectures of LSTMs trained via backpropagation through time (BPTT).

3 Introduction

Stock prices are a form of time series data. There have been many existing business and economics based methods for predicting stock prices. These methods can be classed as fundamental and technical analysis. Technical analysis is based on the observing patterns in stock prices based on psychological effects (fear and greed) changing supply and demand. Fundamental analysis is based on observing current news and events such as the corporate profits and economic situation. However, all of these models ultimately rely on human judgement of the situation to make predictions and are not learned [1].

There are also other learning algorithms which have been used to takle the problem of predicting stock prices. These include using deep multilayer perceptrons [2] and convolutional neural networks [3]. However, these methods have limited capability for temporal memory which can be provided through a fixed sized sliding window for predicting future stock prices as a function of historical prices.

On the other hand, recurrent neural networks have a cycle feeds activations from the previous time step back in as an input and influences the activations of the current time step. Therefore the activations create an internal state. This in theory can store temporal information for a dynamic indefinite number of time steps in contrast to the fixed number of time steps of feed forward networks. LSTMs are a specific type of recurrent neural network which overcomes some of the problems of recurrent networks.[5].

4 Long Short Term Memory

LSTM hidden layers are made up of special cells with sigmoidal input, output and forget gates. This allows the network to learn when to forget, take input and output. The LSTM cell has an internal state which is updated based on the previous activations of the layer and inputs through connections to the previous layer and self connections[4].

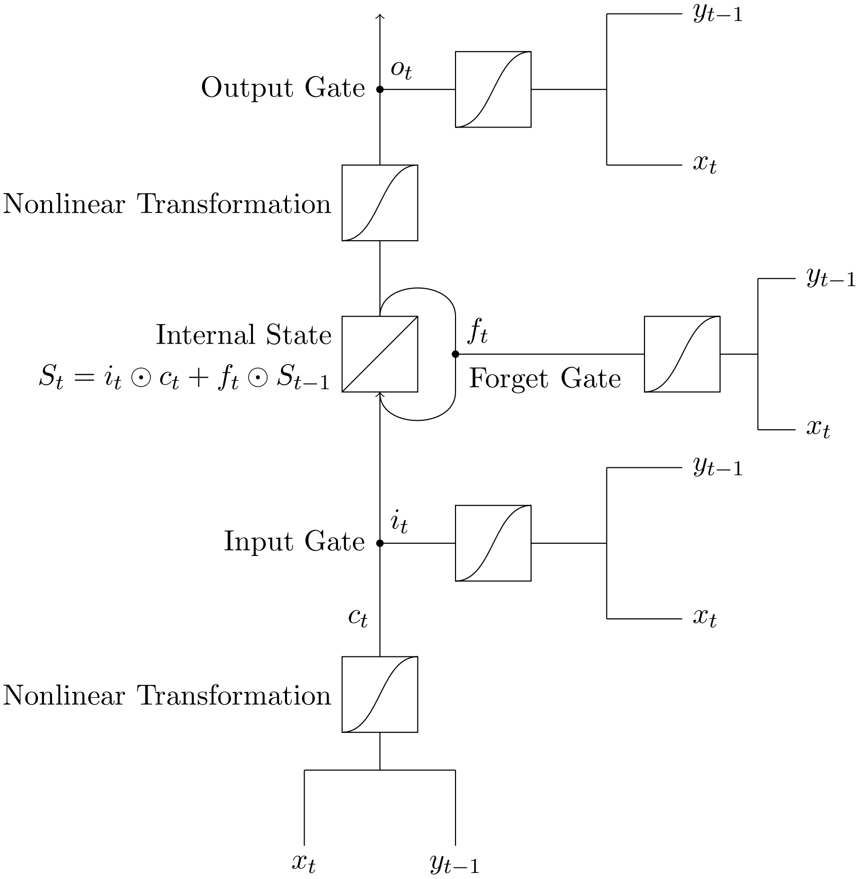

Figure 1: Diagram of the structure of a single LSTM cell

A layer of LSTM cells takes a sequence of vectors as input $x = (x_1, x_2, x_3, \ldots, x_T)$ and outputs a sequence of vectors $y = (y_1, y_2, y_3, \ldots, y_T)$. The output vectors are calculated by iterating through the following equations from $t = 1$ to T:

$$c_t = g(W_{cx}x_t + W_{cy}y_{t-1} + b_c)$$
$$i_t = \sigma(W_{ix}x_t + W_{iy}y_{t-1} + b_i)$$
$$f_t = \sigma(W_{fx}x_t + W_{fy}y_{t-1} + b_f)$$
$$o_t = \sigma(W_{ox}x_t + W_{oy}y_{t-1} + b_o)$$
$$S_t = i_t \odot c_t + f_t \odot S_{t-1}$$

$$y_t = o_t \odot \phi(S_t)$$

We define $\sigma(x)$ as a hard sigmoid function which can output 0 and 1. This means that the gates can fully close or open.

$$\sigma(x) = \begin{cases} 0 & x \leq -2.5 \\ 0.2x + 0.5 & -2.5 \leq x \leq 2.5 \\ 1 & 2.5 \geq x \end{cases}$$

And

$$\phi(x) = g(x) = tanh(x)$$

5 Network Topology

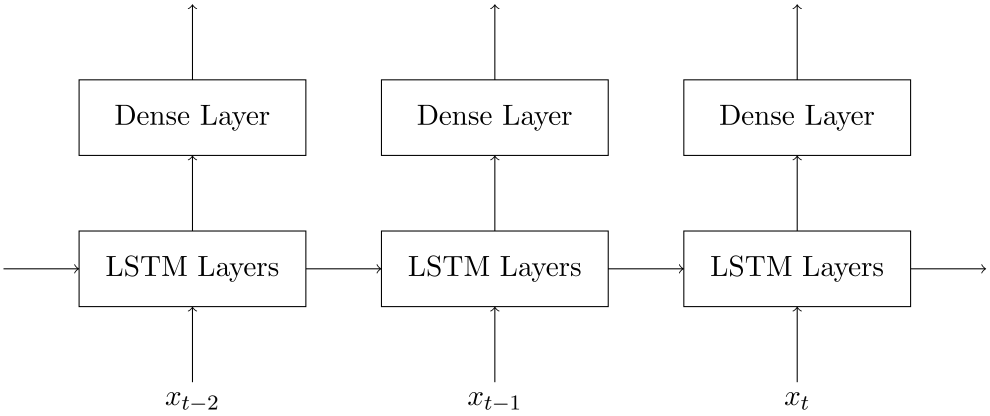

Figure 2: Network architecture

We will first use LSTM Layers as a set of learned feature generators. This will then allow us to stack a dense network on top of the LSTM layers to accumulate the outputs of the LSTM layers into the predictions of the network. As the LSTM layers are already nonlinear, only one dense layer is needed to accumulate their output. However, we varied the number of LSTM layers and the number of cells within them to experiment and see which archticture would perform the best.

6 Weight Initialisations

We initialised our feed forward weights by sampling from the uniform distribution [6]

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right]$$

We initialised our recurrent weights by first sampling from Gaussian distribution with 0 mean and unit variance. Then we performed singular vale decomposition

$$W = USV^T$$

Which gives us 2 random orthonormal matrices U and V, so we set

$$W := U$$

This gives us recurrent weight matrices with unit maximal eigenvalue. As a result, the internal state does not explode as multiplying by the weight matrices repeatedly does not increase the Euclidean norm of the internal state vector.

Finally, we initialised the forget bias units to 1 and the other biases to 0.

7 Hyperparameters, Feature Selection and Targets

In our experiments, we set the learning rate to 0.01 and trained our network for 200 epochs.

We used Google's daily stock prices from January 1st 2005 to December 31st 2014 to create our training set and from January 1st 2015 to June 21st 2015 to form our test set. Th data was optained from Yahoo finance. We only used the open, high, low, close and volume as features for our inputs. This is because any other technical indicators used in technical analysis are simply calculated from those. Therefore our network should learn any necessary patterns that are represented by the indicators.

We also normalised our data to aid training by converting it into returns through a percentage change calculation.

$$\hat{x_t} = \frac{x_t}{x_{t-1}} - 1$$

This scaled the values to make them all small but also transforms the time series to become bounded. This is due to the fact that stocks move very little from day to day. This allows our network to learn a pattern easier.

8 Training Methodology

Although LSTMs have mostly solved the vanishing/exploding gradient problem, they are still inadequate for sequences of such great lengths. Therefore we trained our network using the algorithm described as follows.

Algorithm 1 LSTM Pretraining Algorithm

```
$i \leftarrow 0$
$m \leftarrow$ sequence length
for $2^i < m$ do
    Reset network's internal state
    Truncate sequence using a sliding window of length $2^i$
    Train network on truncated sequence
    $i \leftarrow i + 1$
end for
Train sequence on whole dataset
Reset network's internal state
Test network on test sequence
```

However, we did not need the sequence length for training to be as long as the data goes back in time. So we truncated to a length of 256 using sliding windows as this was a little more than the number of trading days in one year. It is unlikely that data any further back than one year will affect future data.

We trained one epoch on each truncated sequence and then 100 epochs on the 256 days sequence. The batch size was kept constant at 20 and the Adam optimiser was used with 0.001 learning rate and default paramters [7].

9 Experiments

		Hidden Layer Size			
		50	100	250	500
Hidden Layers	1	0.0154	0.0236	0.0139	0.0135
	2	0.0152	0.0166	0.0141	0.0152
	3	0.0141	0.0134	0.0105	0.0130

Table 1: RMSE Test Error Of Specified Networks

10 Discussion

As shown in Table 1, the returns data is generally resistant to overfitting. The network generally performs better when made deeper and wider with a few exceptions even when the number of parameters in the network are far greater than then number of unique training examples.

Although to know in absolute terms whether the network performed, we compare the RMSE of the networks to the RMS of the returns itself. If the RMS of the returns itself is greater, then it would imply that the network is performing better than an algorithm that simply predicts no change and vice versa. The RMS of the test data is 0.0265 which shows that the networks are all performing nearly or more than twice as successfully as the benchmark.

Therefore from this we can conclude that LSTM networks trained with accelerated first order methods are an effective method of predicting stock returns. We also conclude that returns data and LSTM networks are generally resistant to overfitting and larger networks perform better.

References

[1] Barbara Rockfeller (2004). Technical Analysis for Dummies

[2] Wanjawa Barack Wamkaya, Muchemi Lawrence, (2014) ANN Model to Predict Stock Prices at Stock Exchange Markets

[3] Ashwin Siripurapu, Convolutional Networks for Stock Trading

[4] Sepp Hochreiter, Jurgen Schmidhuber (1997). Learning to Forget: Continual Prediction with LSTM

[5] Felix Gers (2001). Long Short-Term Memory in Recurrent Neural Networks

[6] Xavier Glorot, Yoshua Bengio (2010). Understanding the difficulty of training deep feedforward neural networks

[7] Diederik Kingma, Jimmy Ba (2014). Adam: A Method for Stochastic Optimization