Predicting Apple Stock Price with LSTMs

Edward Bickerton, rw19842@bristol.ac.uk
Department of Computer Science, University of Bristol

***Abstract*—This report was completed as coursework for the module Advanced Financial Technology (COMSM0090). I start of by explaining the functioning of a long short-term memory (LSTM) network (introduced by [1] in 1997), before demonstrating their effectiveness by forecasting the stock price of Apple. I conclude by exploring an alternative RNN architecture to LSTM.**

***Index Terms*—Long Short-Term Memory, Recurrent Neural Network, Machine Learning**

I. Long Short-Term Memory

LSTM networks are a variant of recurrent neural networks (RNNs), which, unlike traditional feed-forward neural networks, contain cycles allowing them to exhibit temporal behaviour. Thus, making RNNs very effective at processing sequential data which is commonplace in finance.

A. Neural Network Layers

Let $\boldsymbol{\alpha} = (\alpha_1,\ \alpha_2,\ \alpha_3)$, $\boldsymbol{\beta} = (\beta_1,\ \beta_2,\ \beta_3)$ be the input and output respectively of our neural network layer, shown in fig. 1 (note: their dimension need not be equal). The weights in the network are denoted as such; w_{ij} is the weight associated with the connection from α_j to β_i, the connections from the last input node (with value 1) are bias terms and are denoted as b_i. Each β_i is calculated according to:

$$\beta_i = \phi\left(\sum_{j=1}^{3} w_{ij}x_j + b_i\right), \quad W = \begin{pmatrix} w_{11} & w_{12} & w_{13} & b_1 \\ w_{21} & w_{22} & w_{23} & b_2 \\ w_{31} & w_{32} & w_{33} & b_3 \end{pmatrix} \tag{1}$$

where ϕ is an activation function, such as those shown in fig. 1, the purpose of which is to introduce nonlinearity into the model.

By letting W be the matrix of parameters (eq. 1 right) eq. 1 left simplifies to $\boldsymbol{\beta} = \phi\left(W[\boldsymbol{\alpha}, 1]\right)$, where $[\boldsymbol{\alpha}, 1]$ is the concatenation of $\boldsymbol{\alpha}$ and 1, $(\alpha_1,\ \alpha_2,\ \alpha_3,\ 1)$ and ϕ is applied point wise to the vector resulting from matrix multiplication. The weights and biases (parameters) can be preselected but are more frequently learned via backpropagation (or in the case of a RNN, backpropagation through time), a supervised learning technique.

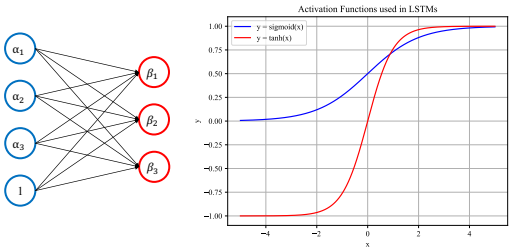

Fig. 1: Left: a single neural network layer. Each connection goes from an input node (left) to an output node (right) and has an associated weight. Right: The activation functions used in LSTMs.

B. LSTM Cell

Fig. 2 depicts the inner workings of a single cell of a LSTM network, (note: the neural network parameters of the cell are reused in each cell, so there is only one cell feeding its output back into itself at each time step).

The added complexity of LSTM over a standard RNN solves the problem of long-term dependencies [2] (resulting from the problem of vanishing gradient in backpropagation, affecting neural networks in general). This was done with the cell state, C_t, which allows for a better flow of information from the past to the present.

The cell state is updated via the forget and input gates, depicted by red and blue arrows respectively in fig. 2. The forget gate outputs $f_t = \sigma\left(W_f[h_{t-1},\ x_t]\right) \in (0,\ 1)^d \subset \mathbb{R}^d$ where d is the dimension of the cell state. The entries of f_t represent how much of their corresponding values in the cell state will be forgotten (0 being completely forget and 1 remember completely).

There are two stages to the input gate, $i_t = \sigma\left(W_i[h_{t-1},\ x_t]\right) \in (0,\ 1)^d \subset \mathbb{R}^d$ which determines what and to what degree should be added to the cell state, and $C'_t = \tanh\left(W_{C'}[h_{t-1},\ x_t]\right) \in (-1,\ 1)^d \subset \mathbb{R}^d$ which represents candidate values

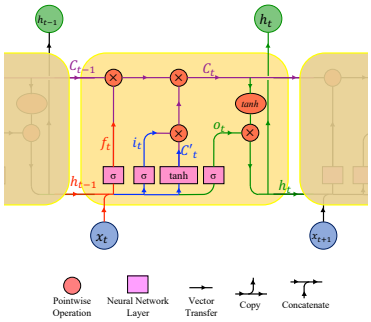

Fig. 2: A single cell in a LSTM network; x_t is the input at time t, while h_t is the output at time t.

which could be added to the cell state.

Finally, the cell state is updated (purple arrows) via, $C_t = f_t \times C_{t-1} + i_t \times C'_t$ where addition and multiplication are performed pointwise.

All that remains is to combine h_{t-1}, x_t and C_t to output h_t. This is done via the output gate (green arrows), which gives $h_t = o_t \times \tanh(C_t)$ where $o_t = \sigma(W_o[h_{t-1}, x_t]) \in (0, 1)^d \subset \mathbb{R}^d$ is used to decide what parts of the cell state to output, and tanh is applied pointwise to C_t to push its values between 0 and 1.

II. Forecasting Stock Price

The LSTM model used in fig. 3 used a cell state with 32 dimensions and achieved a root mean squared error of \$3.62.

Fig. 3: Daily open stock price of Apple.Inc from the 3rd of January to the 30th of December 2022, along with predictions made by a LSTM. Data from before the 26th of October (80% of total data) was used as training data.

Fig. 4 depicts the effect of the dimension of the cell state on the performance of the LSTM model on unseen test data (Apple.Inc stock price data from 26/10/22 to 30/12/22), we can see that d has little effect on the average RMSE (all lie between \$4.75 and \$5.50), however the IQR grows at the extreme values of d. The poor stability of the models with larger d could be due to having too many parameters to train.

On average models with $d = 256$ took 8 seconds to train whereas the models with $d = 32$ took just 1 second, which can be explained by the latter having an eighth as many parameters to train.

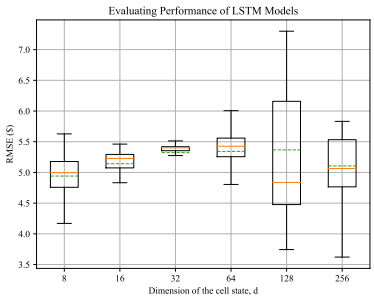

Fig. 4: Root mean squared error on test data of 100 LSTM models for each value of d. Dashed green line shows mean, orange line shows median.

III. LSTM Alternative

Introduced in [3], clockwork recurrent neural networks (CW-RNN) partition neurones into modules and assigns each one a clock period, T_i. At time-step t, only the modules whose clock period is a divisor of t are executed and recurrent connections from module j to i exist only if the period $T_i < T_j$.

Since not all modules are executed at every time step CW-RNN train and evaluate faster and have fewer parameters, because slower modules are not connected to faster ones. In [3] a CW-RNN was shown to outperform a LSTM of equal number of parameters at both sequence generation and spoken word classification.

References

[1] Hochreiter, S. and Schmidhuber, J. (1997) "Long short-term memory,"
[2] Bengio, Y., Frasconi, P. and Simard, P. (1994) "The problem of learning long-term dependencies in recurrent networks,"
[3] Koutník, J. et al. (2014) "A Clockwork RNN."