Examination of stock predictions using RNN & LSTM

Talor R Braly
College of Natural Sciences and Mathematics
University of Texas at Dallas
Plano, TX, USA
Trb180000@utdallas.edu

Gary Chen
School of Engineering and Computer Science
University of Texas at Dallas
Richardson, Texas
gxc097020@utdallas.edu

Nancy Dominguez
School of Engineering and Computer Science
University of Texas at Dallas
Richardson, Texas
dominguez.nancy@utdallas.edu

Jonathan Hulsey
College of Natural Sciences and Mathematics
University of Texas at Dallas
Grand Prarie, Texas
jdh150330@utdallas.edu

***Abstract*—As they are effective in a broad range of applications, LSTMs are extensively covered in a number of technical and scientific blogs and journals. This breadth of applications led us to focus our attention here, as its examination is likely to provide utility in any of a number of future fields. The goal of this paper is to present the essentials of RNN and its refinement, LSTM, in a clear and approachable format. We present a brief look at the background and origins of the modeling technique. Then, we discuss the theory underlying the technique, explaining how raw data is transformed into useful information and predictions. Next, we will examine the results obtained from the model and discuss the expectations these imply. Finally, we will present our conclusions, in particular the benefits gained by adding LSTM to the base RNN.**

Keywords—recurrent neural network, long short-term model, prediction, modeling

I. Introduction and Background

A. *Recurrent Neural Networks*

Recurrent Neural Networks (RNNs) are specialized Artificial Neural Networks (ANNs) which are particularly well suited for dealing with sequences, especially text and time sequences. RNNs were originally based upon the 1974 work of David Rumelhart, an American psychologist, specializing in the formal study of human cognition, and with essential work in the application of back-propagation, deep learning, and Artificial Neural Networks (ANNs) [1]. This foundation lead the way to Hopfield Networks, an inter-referential network of binary nodes, developed in 1982 by John Hopfield, an American Physicist who would become the Howard A. Prior Professor of Molecular Biology, Emeritus [2]. The first true RNN emerged in 1993, when a “very deep learning” task was solved by a neural history compression system; using over 1,000 subsequent layers in an RNN unfolded in time [3].

B. *Long Short-Term Memory*

Long Short-Term Memory (LSTM) is a refinement of RNNs which allow a model to include and extrapolate from dependencies between nodes which are more remote in the sequence than those available in traditional RNN models. LSTM was developed, as a further development of the earlier RNN framework, by the joint work of Sepp Hochreiter, a German Computer Scientist, specializing in Machine Learning, Deep Learning, and Bioinformatics, who now leads the Institute for Machine Learning at the Johannes Kepler University of Linz [4], and Jürgen Schmidhuber a Computer Scientist specializing in Artificial Intelligence, Deep Learning, and ANNs, who would later become a co-director of the Dalle Molle Institute for Artificial Intelligence Research in Manno [5], in 1997 [6].

II. Theoretical Framework

A. *Recurrent Neural Networks*

i. Purpose and Structure

RNNs primary mechanism is to iteratively update a hidden state *h_i*, a vector of arbitrary dimensionality, at any step *t_i*, when given *x_i*, the input vector, to predict *y_i*, with *x_i* and *y_i* also having arbitrary and independent dimensionality. Unlike classic NN, which have may have distinct weights for each vertex and biases for each layer, RNNs only have three weights:

W_{xh} The weight for each connection $x_t \to h_t$

W_{hh} The weight for each connection $h_{t-1} \to h_t$

W_{hy} The weight for each connection $h_t \to y_t$

and two biases:

b_h The bias for determining each h_t

b_y The bias for determining each y_t

The whole of the RNN may thus be represented as the weights in a *matrix* and the biases in a *vector*. The hidden states may be found using:

$$h_t = [some\ activation\ function](W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

and the output at any given step is:

$$y_t = W_{hy}h_t + b_y$$

ii. Training

With the structure established, we now consider the training process. Taking the following definitions for our training model:

$$y = \text{raw output from the RNN}$$

$$p = \text{final probabilities} = \text{softmax}(y)$$

$$c = \text{the true value of the target}$$

L = the cross-entropy loss = -ln(p_c)

$$W_{xh}, W_{hh}, W_{hy} = \text{weight matrices of the RNN}$$

$$b_h, b_y = \text{bias vectors of the RNN}$$

First, we calculate $\frac{\partial L}{\partial y}$

Since

$$L = -\ln(p_c) = -\ln\,(\text{softmax}(y_c))$$

Then

$$\frac{\partial L}{\partial y_i} = \begin{cases} p_i & \text{if } i \neq c \\ p_i - 1 & \text{if } i = c \end{cases}$$

Next, the gradients for W_{hy} and b_y

$$\frac{\partial L}{\partial W_{hy}} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial W_{hy}}$$

$$\frac{\partial y}{\partial W_{hy}} = h_n$$

Where h_n is the final hidden state

$$\frac{\partial L}{\partial W_{hy}} = \frac{\partial L}{\partial y} h_n$$

and

$$\frac{\partial y}{\partial W_b} = 1$$

$$\frac{\partial L}{\partial b_y} = \frac{\partial L}{\partial y}$$

Finally, we find the gradients for W_{xh}, W_{hh}, and b_h.

$$\frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial y} \sum_t \frac{\partial y}{\partial h_t} * \frac{\partial h_t}{\partial W_{xh}}$$

As changes to W_{xh} alter each h_t, it is necessary to backpropagate across each timestep, a process called Backpropagation Through Time (BPTT). W_{xh} is used to calculate each $x_t \rightarrow h_t$, so we must backpropagate through each of those links.

For each step t we must calculate $\frac{\partial h_t}{\partial W_{xh}}$

Using tanh as the activation function

$$h_t = tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

[1] Nature, Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. Volume 323, Issue 6088, pp. 533-536 (1986)

[2] “John Hopfield.” *Wikipedia*, Wikimedia Foundation, 23 Sept. 2019, en.wikipedia.org/wiki/John_Hopfield.

[3] Schmidhuber, Jürgen. Juergen Schmidhuber's Online Publications, Apr. 1996, people.idsia.ch/~juergen/onlinepub.html.

[4] “Sepp Hochreiter.” *Wikipedia*, Wikimedia Foundation, 12 Oct. 2019, en.wikipedia.org/wiki/Sepp_Hochreiter.

[5] “Jürgen Schmidhuber.” *Wikipedia*, Wikimedia Foundation, 13 Oct. 2019, en.wikipedia.org/wiki/J%C3%BCrgen_Schmidhuber.

[6] Neural Computation, Hochreiter, Sepp; Schmidhuber, Jürgen Volume 9, Issue 8, pp. 1735–1780 (1997-11-01)

[7] Zhou, Victor. “An Introduction to Recurrent Neural Networks for Beginners.” *Victor Zhou*, Victor Zhou, 24 July 2019, victorzhou.com/blog/intro-to-rnns/.

[8]