Crypto Currency Market Prediction using LSTM Neural Networks

(Crypto Trade Bot)

Sam Badger
Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85719
smalbadger@email.arizona.edu

Nicholas Everhart
Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85719
neverhart@email.arizona.edu

Abstract

The boom in cryptocurrencies over the past year has made many people very wealthy; but investing in cryptocurrencies comes with a significant amount of risk. With only a few well-timed trades, one could greatly improve their financial situation. Unfortunately, the reverse is also true. With such volatility, it can be difficult to predict when the good times to buy and sell are. That's where algorithmic trading comes into play. Our goal is to design a cryptocurrency-trading bot (CTB) that utilizes Long Short-Term Memory (LSTM) Neural Networks to strategically place trades. Some possible desired outcomes of these explorations are: (a) beat the market, (b) decrease uncertainty in trade decisions, (c) decrease or eliminate risk by utilizing "safety nets" of sorts.

1 Introduction

The initial goal of the Crypto Trade Bot was to make a trading bot, based off a simple state machine, to automate buying and selling in a cryptocurrency market. The scope of the project quickly changed when it was determined that a state machine was too susceptible to market volatility and ended in multiple failed attempts. After many attempts to get the state machine to work and show a profit, it was decided that machine learning would be necessary to have the Crypto Trade Bot be worthwhile and profitable. The ultimate goal of the Crypto Trade Bot is to generate passive income, when the bot is turned on, as well as give insight into market patterns.

Information regarding trades is collected in real time from the GDAX (a subsidiary of Coinbase) websocket and stored into a document database (MongoDB). To be able to apply machine learning techniques to this problem, there is a nontrivial amount of data preprocessing that has to be done. Transforming our data from hundreds of thousands or even millions of raw trades to a more useful form for learning proved to be quite challenging and is far from perfect.

Based on research and previous studies, neural networks proved to be the way to go. Upon further review, it was determined that the Long Short-Term Memory (LSTM) neural network would have the greatest success assuming infinite data and computing power is available. A major (and valid) concern is that an LSTM network would not be able to keep up with the incoming data which needs to be processed in real-time. To cope with this, we'll focus on making fewer trades only when we are very confident in the decision rather than placing rapid trades in hopes to turning small profits. This is also a more practical approach since each trade will cost 0.3% of the funds being bought or sold. By accepting that there will always be a nontrivial delay between the time that we get the data and the time we receive a prediction from our model, we can gather data within a specified time frame, give it more meaning, and finally predict whether or not it would be a good time to buy or sell.

2 Methods and approach

Like any problem where machine learning is used, there are specified steps that need to be followed; the first being data retrieval. Once a large amount of data is collected and a steady stream of incoming data is established, the next step is data preprocessing where the raw data is converted into something that the model can use. However, to know what preprocessing techniques need to be used, a model has to be picked out. The model chosen is a multivariate LSTM network which has very specific requirements for its incoming data. Neural networks, like Long Short-Term Memory (LSTM) recurrent neural networks are able to almost seamlessly model problems with multiple input variables and time dependence [1]. Once all of the prior details are laid out, a labeling scheme needs to be decided upon so the model can appropriately update during the learning process.

With the general outline of the machine learning aspect of this problem in place, there is a need to start thinking about how to use the predictions from the model to accurately place trades. Essentially, the model needs to be able to predict if there is about to be a large increase or decrease in the price of a cryptocurrency. Using this prediction, the trading algorithm will simply be a series of conditions that will only place trades at the most optimal times. This is discussed further in section 2.6 Forecasting.

To understand how an LSTM works, let's think of a neural network like a human brain. Humans don't start their thinking from scratch every second. You understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again [5]. Traditional neural networks can't handle this type of computation and this means that a recurrent neural network is a must. LSTM's are a type of RNN that are capable of learning long-term dependencies. All recurrent neural networks have the form of a chain of repeating modules of neural network.

Below is a visual representation of how a regular RNN looks like with a tanh layer.

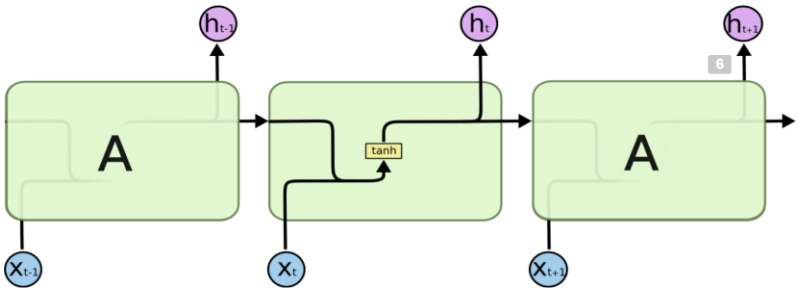

Figure 1: Repeating module in a standard RNN containing a single layer [5]

Like a regular RNN, an LSTM network have a similar chain like loop structure. However, the repeating module has a very different structure. Instead of the single neural network layer, the LSTM repeating module has four that interact in a very specific way.

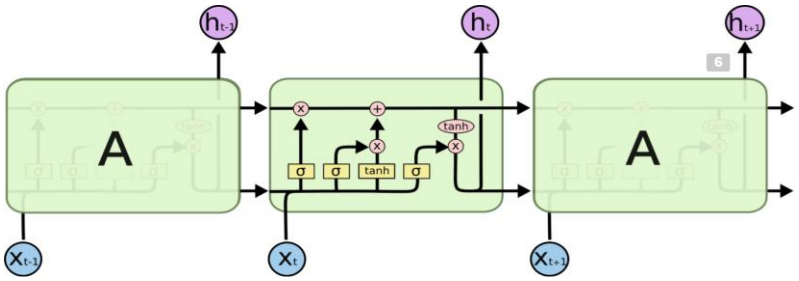

Figure 2: Repeating module in an LSTM RNN containing four interacting layers [5]

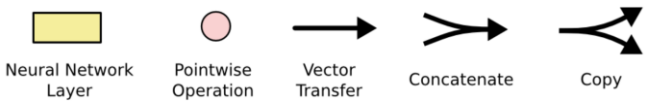

Figure 3: Details of the RNN's modules

For a deeper understanding of how LSTM's work, please reference Christopher Olah's cited web page.

2.1 Data Retrieval

As mentioned earlier, before performing any type of machine learning, a significant amount of data must be readily accessible. This statement remains true for offline learning but for online learning, a steady and reliable stream of *incoming* data is also required. Although the scope of this paper doesn't quite extend to online learning, it was still accounted for in the data retrieval phase of this problem.

Raw trading data is being pulled straight from the GDAX websocket using the Application Program Interface (API) available for python programs and stored in a MongoDB collection. MongoDB is used because of its flexibility with allowing document layout to change between entries [6]. Documents from MongoDB can also be fetched very easily in the form of python dictionaries. Only trades that have been successfully processed are recorded rather than all orders that have been placed. Each trade consists of the product (i.e. Bitcoin, BitcoinCash, Litecoin, or Ethereum), the price and time at which the trade was successfully completed, the side of the transaction (buy/sell), and a number that indicates the sequence of the trades.

To establish a reliable stream of incoming data, a python script was created that persistently starts the data retrieval program. This is important because the data retrieval program loses connection with the websocket fairly consistently especially when local network traffic is high. Due to this, there is a high risk of having short bouts of missing data. The more serious problem is that the operating system may lose connection with the internet and not reconnect automatically. To account for this, a more robust python script could have been created that executes a command to reconnect to the internet. Even with this feature, there is still a risk of missing data and this will have to be addressed in the preprocessing stage.

2.2 Giving the data more meaning

Although an LSTM network may be able to find some patterns in the price, given a sequence of trades as raw as they are, it would be easier to predict if the static was somehow removed. Static can be a serious problem because within a fraction of a second, a cryptocurrency can trade within about a 1% range (at its most volatile state). To smooth our data without losing meaning, data points are grouped into three-minute windows and a few more features are extrapolated from the individual trades that make up the window. For each window, the price is averaged, and the standard deviation is recorded and added as a feature. For each window, the numbers of buys and sells are also calculated and stored as features. The time associated with each window is the time of the first trade that the window comprises.

The window method was first tried with non-overlapping windows, but overlapping windows were found to eliminate static more effectively. This also allows data to be processed faster for online training since a small amount of previous data will also be used to create a new window. The alternative would be to wait for a whole new window to be collected which would cause a delay that could seriously hinder the Crypto Trade Bot's ability to make accurate trades.

2.3 Labeling Data

Once the raw data has been transformed into windows, it needs to be labeled. This is essential for any machine learning problem – without it, the model would not be able to update its weights appropriately. For this problem, there are many different labeling schemes that would work, of which only two were considered.

The first consideration has each window utilize a set of discrete trinary flags (a 1, 0, or -1) that would be turned on: (1) if the price were to increase by a specified percentage in a specified time-frame, (0) if there was not a significant change in price, or (-1) if there was a significant drop in price. Each flag would be associated with a specified time frame so that the model could capture both long and short-term behavior at any given point. This method comes with a set of complications; the biggest being that it's difficult to capture precise behavior with such a simple model. What if there were a sudden drop and increase within the same time-frame? How would the model know how much to trade?

With these complications, comes the need for a labeling scheme that allows the bot to decide when to trade, what side to trade on, and how much to actually trade. To accomplish this, a formula is needed that allows each window to be scored based on each of the following windows within a given time frame. This formula should give more weights to windows that come sooner rather than later and should have the ability to go positive or negative representing a price increase or drop. The following formula was used to score each window:

$$\frac{1}{N}\sum_{i=1}^{N}[\log((p_i+1)/(a_i+1))]^2$$

where $j \geq i$, s_i = the score at window i, p_j = price at window j, p_i = price at window i. The summation max limit is of all windows within a specified time frame rather than a specified number of windows.

At any given window, if the average price were to increase significantly soon after, the window would be assigned a high score telling the bot to buy. A negative score with a large magnitude would signify that the bot should sell. A score that is around zero would mean that no prediction could be

made. The last possibility is a big deal because if the average price were to increase and decrease rapidly enough, the positive and negative scores would cancel each other out telling the bought to make no action and avoid false positives.

2.4 Preprocessing

After the data has been labeled and scored, it still needs to be preprocessed further for the LSTM to accept it. There are many steps that can be taken to preprocess data, but the biggest requirement with LSTM networks is that the data has to be in chronological order and the time field must be removed (i.e. all features, including time, must converted to sequential numbers). Since the data is already in chronological order, nothing needs to be sorted. Time can simply be taken out of the input data. It may be a good idea to fill in any missing values with "NaN" (Not a Number), but this was ignored since there are simply large gaps where there is no data at all in our stored data (caused by computer issues and not retrieving data from GDAX). There may be a better way to handle this such as uniformly distributed time steps regardless of whether trades happen in a window, but this would come with its own consequences since trades are so sporadic by nature. By only starting the window at a trade, it ensures that every window has at least one trade in it.

The other major requirement is that the data has to be in a three-dimensional form with the dimensions being the following:

1. Samples - the outer dimension is a list of samples.
2. Time steps (windows) - a short sequence of time steps between 200 and 500 long.
3. Features - the number of features in a feature vector.

It was a conscious decision to not scale the input data since this problem has a strong likelihood of exceeding previous maximums during online learning, in which case all the data would need to be rescaled and the LSTM would need to be retrained.

2.5 Training

To train the LSTM model, we split the data into a 95% training set and a 5% test set. This will help when the Crypto Trade Bot is brought online as most of our training data will be from previous trades and only a small percentage will be from new incoming trades.

To determine the error between the training data and the network predictions, we used the Mean Squared Logarithmic Error (MSLE). MSLE was used because it does not penalize large differences in the predicted and the actual values when both predicted and true values were large numbers. MSLE measures the ratio between the actual and predicted values as follows:

$$s_i = \sum_{j=i} (\frac{p_i - p_j}{2}) e^{i-j}$$

where p_i = predicted value and a_i = actual value. The "Adam" optimizer was used and is a method for Stochastic Optimization. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. It is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients [3].

Parameters of the LSTM network were adjusted in order to provide the best results based on many training and testing runs. It was determined that 200 epochs trained the network enough without allowing too much overfitting to occur. 50 nodes were used in the LSTM network. This number was based on related work from Jason Brownlee (referenced in citations [1]) that used LSTM networks to predict weather patterns. Further testing would be necessary to make this conclusion. The batch size was chosen with regard to a few aspects. Through testing, it was seen that with too small of a batch size, the algorithm took much longer to complete the training. On the opposite end, too large a batch size led to bad predictions. Through many tests, it was found that a batch size about 500 out of 50,000 windows was the optimal range.

2.6 Forecasting

Once the LSTM model has been trained on the first 95% of the data, the features of the remaining 5% are put into the model to obtain the predicted scores. Plotting these predicted scores against the actual scores of the testing data should reveal a fairly linear relationship with an ideal slope of 1.

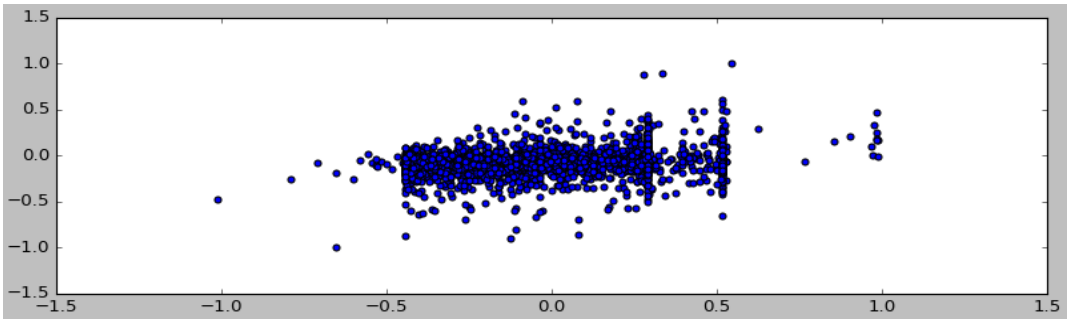

figure 4: the x-axis is the predicted score; the y-axis is the actual score.

Ideally, there would be a strong linear correlation with a slope of 1 which would show that we can accurately predict the score. However, it's obvious that this correlation does not exist

Although the desired relationship does not appear, upon plotting the predicted scores with the testing price, it is obvious that the predicted score tends to spike at local maxima on the price chart. Although the magnitude of these spikes is not reliable, the placement of them is fairly consistent and has potential to serve as a flag to sell. The bot could then wait for the price to decrease to a local minimum to buy again. With a clever buying scheme, it's possible to create an algorithm that could turn very large profits. Of course, the trading algorithm needs to be aware that the magnitude of the spikes can vary greatly from run to run.

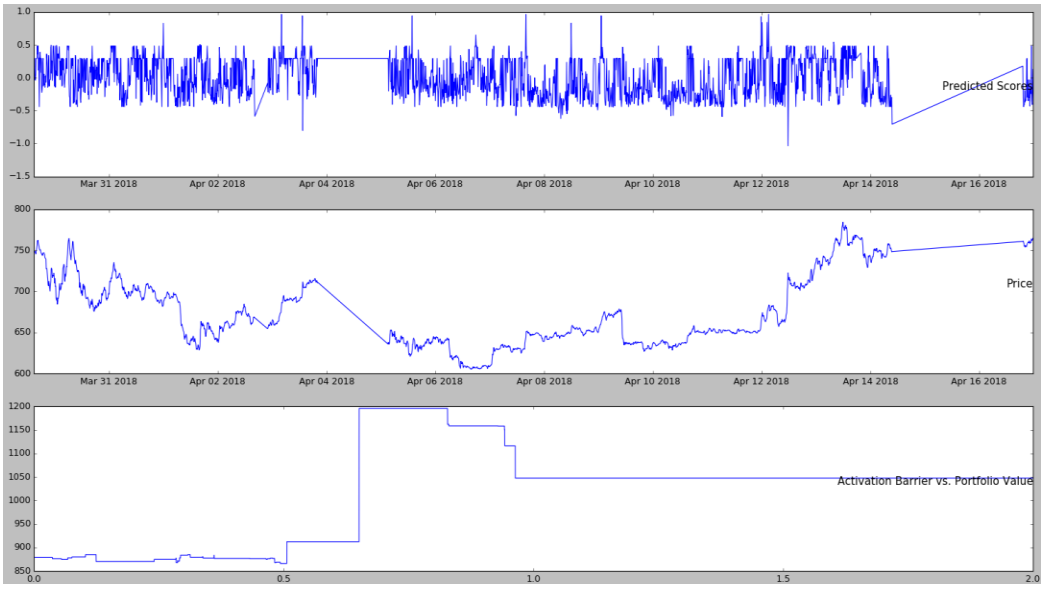

figure 5: (upper) LSTM score predictions w.r.t. time, (middle) price of the cryptocurrency w.r.t. time, (lower) portfolio value w.r.t. activation barrier.

The activation barrier decides which magnitude spikes will trigger trades. When an activation barrier is selected, all spikes with a magnitude greater than or equal to it will trigger a sell if and only if the last trade was a buy. Figure 5 shows just how profitable the bot could be with accepting the right

magnitude of spikes. Setting an activation barrier between 0.65 and 0.8 would result gains of 4 times what it would be without trading at all. If the activation barrier is set at .93 or above, only an initial buy is made without selling.

3 Related Work

3.1 Different method for a like problem

Sebastian Heinz, CEO of STATWORX, and his team members designed a similar program to predict the value of the S&P 500 using data from the Google Finance API. His team used multilayer perceptron's (MLPs, a type of neural network) to predict the S&P 500 market values [2]. Based on a graphical output, he shows that their MLP program had a linear output for the scaled values between the true S&P price vs. the observed S&P predictions. It is not directly expressed in the article, but the comments between Sixue Qin and Sebastian on January 15, 2018 lead one to believe that this model is meant to be a tutorial and not a forecasting model.

3.2 Same method LSTM, different problem

Jason Brownlee, Ph.D. in machine learning, uses an LSTM neural network to predict air pollution. Much like the Crypto Trade Bot, Jason shows how the model takes collected data and prepares it for use by the forecasting model. He then goes on to show how he fits the LSTM to his problem. The forecasting model follows many of the same steps to label, prepare and train the data as the Crypto Trade Bot. Jason concludes his article with a rendering of how the test data had a lower error rate than his training data when compared to his original data. His final error using the model achieves a respectable RMSE of 26.496, which is lower than an RMSE of 30 found with a persistence model [1].

3.3 Similar method with a similar problem

Raoul Malm, a scientist at the University of Mainz, Germany, uses and LSTM to predict the NY Stock Exchange. His work demonstrates the future price prediction for different stocks using recurrent neural networks in TensorFlow. Recurrent neural networks with basic, LSTM or GRU cells were implemented [4]. His work, like the Crypto Trade Bot, follows the same pattern for manipulating the data. The data is analyzed, manipulated, modeled and validated, and finally predicted. Since Raoul's Kaggle site simply list his code with a few graphs depicting his data, his end results were determined from his graphical comparisons. When specifically looking at his future stock prices graph, we see that his test prediction is very close in line with his test target. This leads to the conclusion that his predictions followed very closely to the actual market prices and would be a great model to test with our specific problem.

4 Results and conclusion

With a testing RMSE value of 0.927, there isn't much to brag about since the predictions had a range of 2 (-1 to 1). Although the model does not seem to accurately predict the desired scores, the predictions still hold value and could be used to create a very profitable trading algorithm. Before launching the live Crypto Trade Bot, more research should be done in the field of fine tuning LSTM networks to get reproducible and consistent predictions. With a model that can predict local maxima in price, an immediate sell could be initiated, and another LSTM network may be used to predict a price limit at which the currency should be repurchased. By placing limit orders, an investor can avoid being charged, making the actual cost of running the CTB much lower.

Acknowledgments

We would like to thank Professor Gregory Ditzler for providing us with guidance and direction with this project.

References

[1] Brownlee, Jason. “Multivariate Time Series Forecasting with LSTMs in Keras.” Machine Learning Mastery, 23 Oct. 2017, machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/.

[2] Heinz, Sebastian. “A Simple Deep Learning Model for Stock Price Prediction Using TensorFlow.” A Medium Corporation, Augmenting Humanity, 9 Nov. 2017, medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877.

[3] Kingma, Diederik, and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” [1402.1128] Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition, 23 July 2015, arxiv.org/abs/1412.6980v8.

[4] Malm, Raoul. “NY Stock Price Prediction RNN LSTM GRU | Kaggle.” Countries of the World | Kaggle, May 2017, www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru.

[5] Olah, Christopher. “Understanding LSTM Networks.” Understanding LSTM Networks -- Colah's Blog, 27 Aug. 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.

[6] “What Is MongoDB?” MongoDB, 2018, www.mongodb.com/what-is-mongodb.