Time Series Forecasting with Deep Learning Models

Eric Steen, Orion Darley, Ryan Silva – Stanford University

May 2019

Abstract

Deep Learning models are increasingly used for time series forecasting and are showing promise as an industry-leading methodology for such analysis. In particular, Deep Recurrent Neural Networks(RNN) with Long Short Term Memory(LSTM) are the most widely accepted Deep Learning approach for time series. In this paper we approach the problem of predicting Bitcoin(BTC) prices with RNN and evaluate their efficacy compared to widely accepted traditional time series approaches, in particular the (S)AR(I)MA(X) models.

1 Introduction

The purpose of this research is to benchmark the effectiveness of deep learning neural networks on price forecasting of bitcoin. Long-Short Term Memory (LSTM) and Recurrent Neural Networks (RNN) are explored due to their cutting-edge effectiveness. We evaluate against (S)AR(I)MA(X) models.

2 Related Work

Several papers are relevant to our work including x, y, z..

3 Methods

Our team experimented with several LSTM models and are still deciding a model strategy. The following steps are taken to measure the predicted values distance from ground truth:

1. Import bitcoin pricing from 2012-current, to include open price, close price, high and low prices, volume transacted, and percentage change over a daily interval.

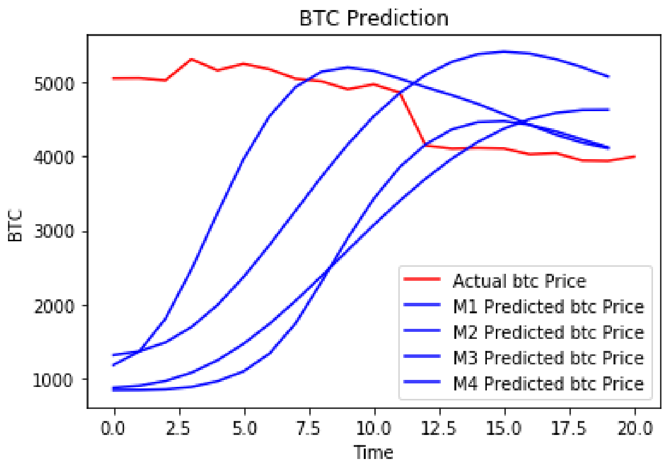
2. Plotted the distributions of each and determined the threshold for the training data. Most of the data is heavily skewed as to where past price changes and autocorrelation do not necessarily represent the current trend nor the near-future trend of Bitcoin prices.

3. Scaled the data and created the training and test structure.

4. Built four models using various parameters (various activation functions, hidden nodes, dropout rates, batch sizes, etc.)

5. Benchmarked the predicted values for four models versus the actuals.

The results are fairly similar, however a more focused strategy for deep learning in time series needs to be developed (see results below).



3.1 Dataset

Our dataset consists of the following:

1. Bitcoin prices on a daily basis since 2012

2. Bitcoin prices on an hourly basis since july 2017

3.1.1 Regularization

We regard the old wall street maxim 'The trend is your friend' to be a good starting point for the evaluation of deep learning on financial time series, so to reduce noise that might prevent the machine learning algorithm from learning the trend, we will explore regularizing the data with wavelet transformation using the PyWavelets library.

The process to regularize is as follows:

1. The data is transformed using a Wavelet transform.
2. Coefficients more than 1 standard deviation away from the mean of coefficients are removed.
3. Inverse transform the new coefficients to get the denoised data.

Here is the integral form of the wavelet transformation used by the *PyWavelets* library:

$$\left[W_\psi f\right](a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi\left(\frac{x-b}{a}\right)} f(x)dx$$

3.2 Feature Engineering

We explore a variety of data preprocessing and augmentation techniques gathered from the literature, and implement them as part of our EDA (Exploratory Data Analysis). Our EDA includes the following:

1. We draw inspiration from the R2N2 [3] algorithm, which uses the VAR and VARMAX algorithms to pre-process the time series data and produce residuals, which are used as input to an LSTM. We show that the VAR model performs quite well in one day forecasts. We can use these techniques to augment our data, providing both the raw features, as well as VAR and/or VARMAX residuals.
2. Found some success by pre-processing the data using filters, namely the Kalman and Wavelet filters [1].
3. [2] gives multiple technical indicators for data augmentation including momentum, trend, volatility, and volume. These can be calculated with the python 'TA' library . These features can be used to further augment the LSTM input.
4. [4] proposes the technique of segmenting the time series into a number of consecutive 'windows' , and a label which is the next window of data. Preprocessing the data into this format and saving to disk can save a lot of computation time each run. There are also many hyperparameters here, such as the size of the window, how many windows to include in a training example, if and how much he windows overlap, etc.

We plan to use the root mean squared error (RMSE) between predicted and true financial time series as the cost function and evaluation metric for our models, as implemented in the EDA notebook. We also plan to build out a data pipeline using simple data stores (redis or sqlite).

3.3 Hyperparameter Tuning

3.3.1 Network Architecture

3.3.2 Number of Epochs

3.3.3 Batch Size

3.3.4 Learning Rate

3.3.5 Dropout

3.3.6 Loss Functions

3.3.7 Hidden Units

3.3.8 Window Length

3.3.9 Forecast Length

3.4 Results

After 600 hours of testing we found that the hyperparameters and architecture that showed the most promising results were x, y, z...

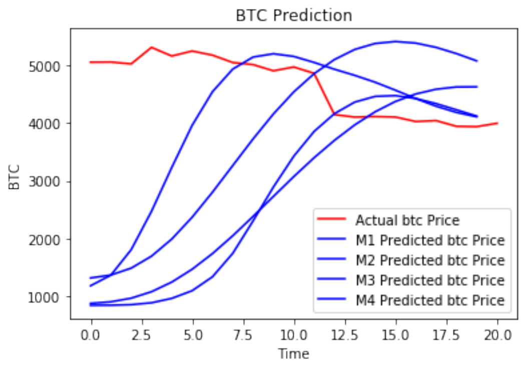


Figure 1: Our Results

4 Conclusions

5 Future Work

6 Code

Our code is at https://github.com/ericsteen/crypto_data The code of particular interest is in `btc_lstm_final_v2.ipynb` and `data_experiments.ipynb`. Data gathering scripts are located in /lib as well.

References

[1] Zhang Jiang. Exploration of predicting power of arima..

[2] Kaur. Quantitative trading strategies using deep learning, pairs trading.

[3] Shaw Persson, Slottje. Hybrid autoregressive recurrent neural network architecture for algorithmic trading of cryptocurrencies.

[4] Lilian Wang. Predict stock market prices using rnn model with multilayer lstm cells + optional multi-stock embeddings.