Time Series Forecasting with Deep Learning Models

Ryan Silva, Eric Steen, Orion Darley – Stanford University

May 2019

Abstract

Deep Learning models are increasingly used for a variety of time series forecasting applications and show significant promise as the industry-leading methodology for the foreseeable future, as previously used methods can be integrated into them easily. In this research, Deep Recurrent Neural Networks(RNN) with Long Short Term Memory(LSTM) are explored to handle the problem of uncertainty in Bitcoin(BTC) prices. We evaluate their efficacy in relation to more generally accepted time series approaches, in particular the (S)AR(I)MA(X) models.

1 Introduction

We apply deep learning to a duple of tactical parameters: lookback period and forecast length, hereby denoted as $\langle wlen, flen \rangle$ for window length and forecast length respectively. We execute a two-pronged strategy.:

1. We use a discrete binary classification RNN model to determine the optimal hold period $flen$ for a position held over a given $wlen$ from a semi-random sequence of $\langle wlen, flen \rangle$ pairs.

2. We use a continuous linear RNN model to predict the price following each set of chosen tactical parameters $\langle wlen, flen \rangle$.

Minimization of hold period is highly desirable when parity exists across the set of tactical parameters $\langle wlen, flen \rangle$, so as to reduce value-at-risk in aggregate. Having a target price clearly aids in tactical trading decisions such as exit point determination.

2 Related Work

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

3 Methods

Ryan, Orion, Eric TODO

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 in order to reduce noise that might prevent the machine learning algorithm from learning the trend, we explore regularizing the data using a Kalman filter with the pykalman library.

The Kalman filter process is divided into two steps:

- The prediction step, which uses a previously estimated state and the linear model to predict the value of the next state as well as the state estimate covariance:

  $\hat{x}_{k|k-1} = Ax_{k|k-1} + Bu_k$
  $P_{k|k-1} = AP_{k-1|k-1}A^T + Q$

- The update step, which uses the current measurement of the output together with the statistical properties of the model, to correct the state estimate. The values calculated is the innovation covariance, the Kalman gain resulting in the updated state estimate and state estimate covariance:

  $\mathrm{S}_k = CP_{k|k-1}C^T + R$
  $K_k = P_{k|k-1}C^T S_k^{-1}$
  $\hat{x}_{k|k} = Ax_{k|k-1} + K_k(Z_k - C\hat{x}_{k|k-1}$
  $P_{k|k-1} = (I - K_kC)P_{k|k-1}$

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. Ryan TODO R2N2 [3]

2. Ryan/Eric TODO Kalman(ryan) and Wavelet(eric) filters [1].

3. Ryan TODO (ta library) [2].

4. Eric TODO (windowing the data) [4]

3.3 Activation, Loss

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).

3.4 Hyperparameter Tuning

3.4.1 Network Architecture

3.4.2 Number of Epochs

3.4.3 Batch Size

3.4.4 Learning Rate

3.4.5 Dropout

3.4.6 Loss Functions

Eric TODO 'adam'

3.4.7 Activation Functions

Eric TODO '$\text{sparse}_c ategorical_c rossentropy'$

3.4.8 Hidden Units

3.4.9 Window Length

3.4.10 Forecast Length

3.5 Results

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

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.