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. We imagine a cooperative utopia for man and machine to harmoniously unite in intuitive and logical unison for the continual improvement and sustainability of the economic machine by which we are all inter-dependently associated.

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 Classification Model 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 risk return characteristics are in parity across the set of tactical parameters as minimizing reduces value-at-risk in aggregate for a trading program. Having a sensible target price aids in tactical trading decisions such as risk-return calculations and exit point determination.

2 Related Work

Several papers are relevant to our work including the R2N2 paper [2] which covered x,y...

3 Methods

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 Scaling

Classification Model

We scale the both the price and volume series to adjust the range of values to be gaussian with zero mean and unit variance, without changing the distribution. This will prevent a feature with high variance that is orders of magnitude higher than other features from dominating and making the estimator unable to learn from other features correctly.

Linear Model

Ryan

3.1.2 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 regularize both the classification and linear models using a Kalman filter with the pykalman library.

Kalman Filter

The Kalman filter process has 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 states estimated covariance.

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

3.1.3 Preprocessing

Classification Model

Eric TODO (windowing and balancing the data) [3]

Linear Model

Ryan TODO

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:

Classification Model

Price and Volume were used in the classification model for simplicity to ensure model sensitivity to the most widely distributed data features in the financial asset space.

"Everything should be made as simple as possible, but not simpler. " - Albert Einstein

Linear Model

Ryan TODO (ta library) [1].

3.3 Activation, Loss

Classification Model

Eric TODO (windowing and balancing the data) [3]

Linear Model

Ryan TODO (ta library) [1].

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

The hyperparameter decisions that were most effective at tuning the model were smaller number of hidden units per layer, dropout, scaling and normalizing the data in the pre-processing step, learning rate, and batch normalization.

3.4.1 Network Architecture

Classification Model

A 3 layer RNN model for Classification Model with no kernel or activity regularization was effective to a desirable level of accuracy.

Linear Model

The linear model (Ryan)...

3.4.2 Number of Epochs

Classification Model

A 50-75 epoch run was most effective for the classification model as validation loss increased beyond with a slight decline in accuracy, and X-X for the linear model.

Linear Model

The linear model (Ryan)...

3.4.3 Batch Size

Classification Model

Due to the 'small data' nature of our inquiry, a lower batch size of 64 for the binary model sufficed.

Linear Model

A batch size of 32 was adequate for the linear model.

3.4.4 Learning Rate

Classification Model

.001

Linear Model

.001

3.4.5 Dropout

Classification Model

A dropout ratio of 0.4 improved the volatility of the classification model significantly from baseline (sans any regularization).

Linear Model

The linear model (Ryan)...

3.4.6 Loss Functions

Classification Model

Eric TODO 'adam'

Linear Model

The linear model (Ryan)...

3.4.7 Activation Functions

Classification Model

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

Linear Model

3.4.8 Hidden Units

Classification Model

Given the 'small data' nature of our inquiry, a lower number of units was most effective for increasing accuracy, reducing loss, and decreasing overall volatility of metrics.

Linear Model

3.5 Tactical Parameters

Our research showed a 120 period window with a forecast period of 3 gave the greatest accuracy with the lowest loss and the least volatility of all other tested combinations, taking into account risk-parity considerations (4 periods showed very similar results, with slightly more stability in accuracy across epochs, but 3 gives less value-at-risk in practical application in finance).

3.6 Results

3.6.1 Classification Model

A 120 period window with a forecast period of 3 was the optimal random swing trade for bitcoin since 10/11/2015. While this may change in the future, we are certain that additional data and feature engineering, novel approaches, and additional human insights will enhance the tactics disclosed herein.

3.6.2 Linear Prediction

Ryan

4 Conclusions

Deep Recurrent Neural Networks are

5 Future Work

6 Code

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

References

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

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

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