Joe Puhalla

HW 6

Big Data

<https://arxiv.org/ftp/arxiv/papers/2002/2002.11523.pdf>

Github: <http://github.com/evgps/a3c_trading>.

**Using Reinforcement Learning in the Algorithmic Trading Problem**

E. S. Ponomareva, \*, I. V. Oseledetsa, b, and A. S. Cichockia

This paper proposes using a variety of ANN (artificial neural network) structures, including LSTM, to create a trading strategy. First, the stock exchange is interpreted into a game with Markov properties consisting of states, actions and rewards. The paper focuses on predicting futures prices. The program implements the actor-critic problem with the training of a neuron network every N= 200 steps. Every training epoch was conducted on the three-month data which contained 50,000 one-minute steps. The data is split into two sets training and testing with each period being six months long. Trades are only profitable if they generate a profit of more than 2.50 rubles due to the transaction cost of commission. The architecture of the neural network is then varied. The activation function used was the sigmoid function and the initial choice of total hidden layers was one. These six assumptions were used when testing the variety of architectures:

Assumption 1. Using a different reward function.

Assumption 2. Introducing a recurrent layer (LSTM).

Assumption 3. Adding a dropout layer.

Assumption 4. Increasing the number of neurons in the hidden layers.

Assumption 5. Using a more complicated architecture of the cost function.

Assumption 6. Combining the attributes for several minutes in a common vector.

The best architecture provided for a 66% annum return when accounting for commission costs (See paper for entire data results).

<https://arxiv.org/pdf/1912.08791.pdf>

Forecasting Significant Stock Price Changes Using Neural Networks

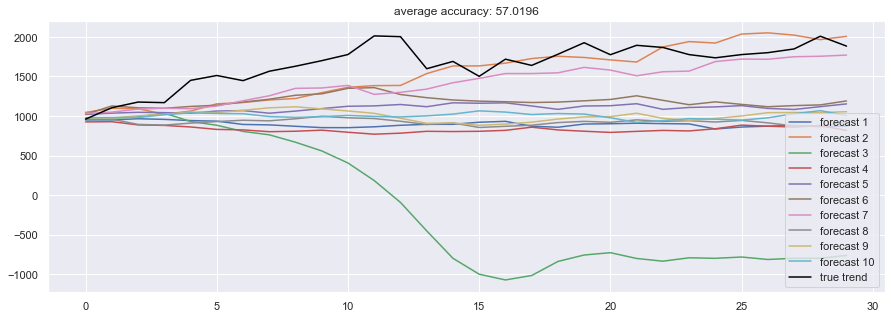
Firuz Kamalov

This paper proposes three different neural network architectures in order to predict significant changes in price data of four major US stocks. The three models used are MLP (multi-layer perceptron), CNN, and LSTM. The results obtained from these models are compared against a random forest and relative strength index methods. A normal MLP consists of an input layer, hidden layer and then an out put layer. The LSTM model is a type of RNN where the output is calculated based on the current input and previous hidden state, where the hidden state is calculated during the previous time step. Hence, the network remembers previous states when computing the current state. For this experiment, each model contains an input layer, two hidden layers, and an output layer. In addition, a dropout rate of 0.2 is applied to certain layers within the CNN and LSTM. The models use a threshold of 1.2 standard deviations of a price move as a defined major move. The LSTM model is shown to be able to accurately predict large price changes up to a certain threshold with each of the four stocks chosen. In general, the models are better at predicting more significant price changes than less significant price changes. It is mentioned that the MLP is significantly less computationally expensive than the LSTM model but produces slightly less accurate results. This idea must be taken into consideration when determining which model is appropriate; there is always an accuracy-complexity optimization that must be considered for each unique problem. Lastly, the research performed differs from a lot of published research in that these models are only used to predict that a significant price change occurs rather than predicting whether that change is positive or negative.

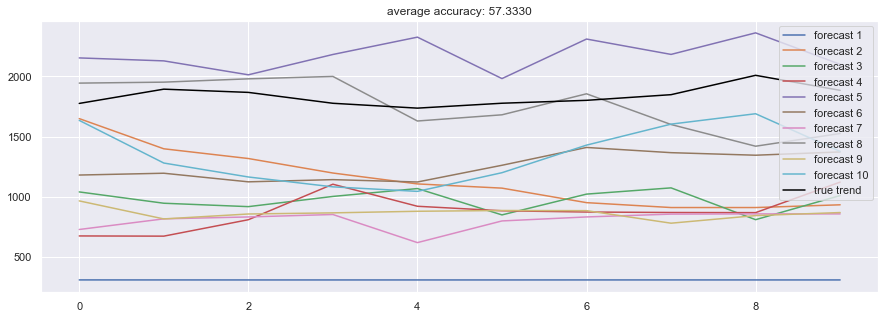
For the above referenced paper, I have not yet found the GitHub for the code used in this paper, but the paper provides a solid framework of ideas. The below Githubs could be used to help recreate this paper by altering the LSTM architecture.

For the below referenced GitHubs, there is not a published paper associated with either of the accounts, however, the depth of source code available is very significant and useful for all teams looking to have an initial model or strategy already coded. This code can then just be tweaked to fit the needs of the research for each team.

<https://github.com/huseinzol05/Stock-Prediction-Models>

Ran the LSTM code on AMZN stock using single layer (128 neurons) with 300 epochs and a dropout rate of 0.8. Received accuracy results and simulations as image below. Shows forecasted price of stock over time. Need to add floor of 0. Unsure why one forecast is so negative.

Ran LSTM but changed number of layers to 2, epochs to 100, and train sets to 10 instead of 30.



<https://github.com/firmai/financial-machine-learning>

Again another repository full of already completed model code, but lacks a paper associated with it.