Project Overview

Jashua Luna
August 17, 2021

Objective

The aim of this project is to develop and implement a neural network capable of trading bitcoin (or any other cryptocurrency). Ideally this network will be able to profit from these trades.

Basis for Network (problem definition)

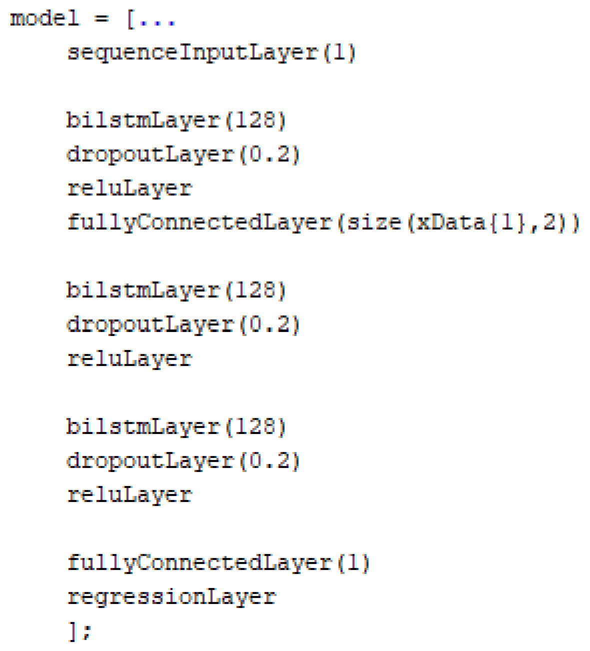
Fundamentally, there are only two actions a trading network can take; buy or sell the asset in question. Since the aim is to always profit, selling the asset is considered shorting and buying considered longing. The objective becomes to maximize profit on a given time interval by longing as many increases in price, and shorting as many decreases in price, as possible.

Considering the price of the asset in question as unchangeable by the network's actions, a model can learn to maximize profits by maximizing the length of the line made between selected peaks and valleys (on the price chart); trades are made at peaks and valleys. In an ideal scenario, where trades are executed the instant the network makes the request, and at the ideal price, and with no trading fees, a network could learn to make trades on a millisecond-to-millisecond basis, since maximum line length means maximum profits. However, this is not the case. Request-response lag, and trading fees mean a trade must be of a certain size for it to be profitable.

Proposed Solution

Since the network has two effective states, long and short, the asset's price can be reduced to a binary signal. One for long, zero for short. The proposed network is therefore a transfer function which converts the asset's price data to this binary signal, from which trades can be made.

To train the network, historical data must be converted into its binary representation; that also satisfies the minimum trade size constraint. For this, a recursion is performed where a new point is inserted between the point at time t, and its right-hand neighbor, until a stopping condition is met, or the end of data is reached. Stopping condition at the time of this report is that the trade to the right of the trade at time t can be no closer than t + 30 minutes (data was sampled at 1minute intervals). The new point is inserted at the maximum of the absolute difference between the secant, of the point at time t and its righthand neighbor, and the price data. Then, once the recursion is complete, the slopes between points obtained are calculated, and if the slope is negative, the binary signal at that point is zero, and if positive one.



```
model = [...
    sequenceInputLayer(1)

    bilstmLayer(128)
    dropoutLayer(0.2)
    reluLayer
    fullyConnectedLayer(size(xData{1},2))

    bilstmLayer(128)
    dropoutLayer(0.2)
    reluLayer

    bilstmLayer(128)
    dropoutLayer(0.2)
    reluLayer

    fullyConnectedLayer(1)
    regressionLayer
    ];
```

From this price – binary data, a network is trained. For good measure, the binary signal (yData) for the network is time shifted forward. This mitigates request-response lag. A basic lstm network model was developed in matlab and trained on price-binary data. The price data was converted to the difference between closing and opening price for that time interval, and maxmin scaled, to normalize data around zero. At the time of writing this report, positive results have been achieved with the model (above), and time shifts of zero and five minutes.

Next Steps

The next steps are to port the matlab solution to python (keras). Then develop a method of parsing price data, passing it through the trained network, and making trades (or not) accordingly. Then implement this on Azure. Write a formal report, and possibly make a jupyter notebook for the training/data-generation python code.