Machine Learning Engineering Nanodegree

Capstone Proposal

Sun 17 Febuary 2019

1. **Domain Background**

Algorithmic Trading

Algorithmic trading is automated (or semi-automated) pre-programmed trading instructions accounting input variables like time, price, volume, tweets, news, and company reports.
Algorithmic trading is developed so the traders do not need to constantly watch a stock price, making bad prediction or emotional decision. Traders can use the power of machines and algorithms to make better investment returns.

A basic price prediction models can be a linear regression. A more complex pattern recognition, or predictive model can also be developed by machine learning or deep learning. The common stock patterns are Mean Reversion, Momentum, and Breakout.

The most basic part of algorithmic trading is making a decision whether to buy or sell an asset. In any particular time, the trading system can access the current price level, and other related information, like volume of trade of the asset.

An example of algorithmic trading academic research paper:
http://qfrg.wne.uw.edu.pl/?page_id=645
https://www.wne.uw.edu.pl/files/2615/2405/6484/WNE_WP268.pdf

Notice this is a self motivated project on applying machine learning in trading. The academic research paper listed above is not the specification of this project, but a reference for ideas and information only.

2. **Problem Statement**

Mission: Create a Binary Classifier Prediction Model, to predict raise or fall in price of Ethereum 5 minute in the future.

Using only historical data in 2018 of two of the most popular crypto coin Bitcoin and Ethereum, build a machine learning model to predict the future price of Ethereum, 5 minutes in the future. Can machine learning give the trader an edge on decision making which perform better than random?

This project is a problem of Binary Classification.
If price predicted raise, the target label is Buy.
If price predicted fall, the target label is Sell.
The problem is NOT to predict an exact future price of Ethereum in 5 minute, but only predicting price either raise or fall in 5 minute, much simpler.

The output of the problem is the trained model predicting a raise and a fall in price of Ethereum 5 minutes in the future, with an accuracy greater than 0.5 (50%)

Notice
There is no porfolio handling in this project, simply create a model to make prediction better than random (benchmark).
Common algorithmic trading tools like Zipline, Alphalens or Pyfolio is not used in this project.

3. **Datasets and Inputs**

Using the freely available historical data of Bitcoin and Ethereum, in year 2018.
Since all cryptocurrency exchanges trade price are slightly different, use only one exchange historical data. I have decided to use exchange Gemini, this is a popular exchange in Europe, and it is the only exchange data I found with free minute data .csv files.

I have chosen to use the minute data for the problem, as minute have more data points than hourly or daily, and Deep Learning needs a lot of data points.

All of the input data are in .csv format, all included in the proposal zip file.

These .csv files can also be download for free at:
http://www.cryptodatadownload.com

Direct links to download:
http://www.cryptodatadownload.com/cdd/gemini_BTCUSD_2018_1min.csv
http://www.cryptodatadownload.com/cdd/gemini_ETHUSD_2018_1min.csv

Features
Close Price, Volume of Bitcoin
Close Price, Volume of Ethereum
(where DateTime is index in the dataset, not a feature)

Target
Predicting Raise or Fall in Ethereum Close Price, 5 minutes in the future
Binary values: Raise=Buy=1 or Fall=Sell=0
(Notice: The project is not predicting raise/fall in price of Bitcoin)

Correlation
Often similar type of stocks or crypto-currencies raise and fall together. e.g. when Japan made bitcoin a legal currency back in Nov 2017, the news likely to contributed the big raise in bitcoin value in late 2017, the altcoins also followed the raising trend. Bitcoin have the largest market share and it is often the trend setter. The correlation between Bitcoin and Ethereum will be used as inputs to predict the future price of Ethereum.

Data Limitation
The dataset include the historical price and volume for Bitcoin and Ethereum only, no other altcoin historical data will be use. It is common to use tweets or social media feeds to make trading decisions, but these datasets are very large and expensive to obtained, so they are not used in this project.

Datapoints
The dataset used in this project is year 2018 only, and the data samples are in minutes.
The number of data points in each .csv file will be:
365 days x 24 hours x 60 minutes = 525600 data points
(This is an approximation only, as there are likely missing gaps in data)

Data obtained already
I have already provided all the dataset i will be using for this project, the two .csv files in the zip file.
I have also provided download links in this document showing where i have obtained the dataset from.

4. **Solution Statement**

RNN LSTM
Trading historical price data is Time Sequence.
Recurrent Neural Network and Long Short Term Memory is ideal for solving Time Sequence problem.

Target Label
The inputs time, price, and volume data are features, there is no target label in the given data.
I will manually create a target label, binary possible values are ‘buy’ or ‘sell’, this target label is calculated by:

Future price (in 5 minutes) - Current price = Difference in Price
Positive Difference in Price = Price raised = Buy = 1
Negative Difference in Price = Price fall = Sell = 0

Feature Engineering
For each prediction, the model will not only use the features of the current data point, but it will also use a look back period (previous days/hours/minutes), like the last 10, 30, or 60 minutes etc of historical data. The length of history lookup period is a parameter to tweak in the project.
If the look back period is 100 days, then there are too many features generated and the model training will be slow.

If the look back period just 1 min, then it is probably not enough features for the model to learn the patterns.

Data split for training and validation
Data First 95% is used for training
Data Last 05% is used for validation
The split of data is by ordered datetime index, we first calculate what is the datetime of 95% of year 2018.
There is approx 365*24*60=525600 data points, if we dont count any missing gaps.
525600*0.95=499320
So the datetime at index 499320 is the 95% data point position where the split happens.
Data index 000001 to 499319 = training data
Data index 499320 to 525600 = validation data
(This is an approximation only, as there might be gaps in data)

Data Gaps Handling
There are two way considered to handle gaps in both the provided data and during processing the data, either drop any rows that have an empty value, or apply forward fill using the previous available value, which of these 2 approaches will be used, and where in code, will be decided during the code implemetation.

5. **Benchmark Model**

Pre-processing the data evenly so there are 50% buy and 50% of sell, this give a pure random guess prediction has a 50% chance of being right. This is the benchmark model.
Dropping any data points to balance the data will create "gaps" in the time-series. But there should still be enough data points to make the training possible, if not, then the model training would fail and should be detected in the validation accuracy. I am confident the selected features in dataset should be enough to train a model make predictions a bit better than random.

Model prediction validation accuracy (Benchmark) = 0.5

Logically, if the trained RNN model can predict correct target Buy/Sell by more than 50%, and many times, then the model is performing better than the benchmark (random), and the problem is solved.

The good trained RNN model can be calculated by the following simple equation:
Model prediction validation accuracy (Solved) > 0.5

6. **Evaluation Metrics**

The trained RNN model validation accuracy needs to be more than 0.5 (50%) to be successful.

Model prediction validation accuracy	Meaning
> 0.5 (>50%)	The RNN model is learning the patterns for a raise in future price, the model prediction is better than random. The problem is solved.
= 0.5 (=50%)	The RNN model is not learning any patterns from the input data, the model is stuck at random 50:50, the benchmark.
< 0.5 (<50%)	The RNN model is only memorizing input data, possible overfitting, The model has failed to learn.

To track the model prediction performance, I will use Tensorboard in development, and include graphs of validation accuracy in my Project Report.

7. **Project Design**

The workflow
- Read in the 2 .csv files into dataframes.
- Join all data into one main dataset, use column close as the price value, drop any inrelevant rows and columns.
- Data Handling - forward fill or drop any gaps in data, Date format yyyy-mm-dd hh:mm:ssss
- Assign a new column target label (Binary Classification) buy/sell value base on future price raised/falled.
- Normalizing numeric column values to between value 0.0 to 1.0
- Feature engineering, each data row lookback a number of sequence in the historical data, previous

minutes, these are additional features to be use to predict the future sequence price and target label.
- Seperate main dataset into Train data (95%) and Validation data (05%), do not shuffle, and must be ordered by Date.
- Create data for benchmark, by balancing data to 50% going up (Buy), 50% going down (Sell),
  by removing extra Buys or Sells if necessary.
- Create a deep learning RNN model using keras tensorflow LSTM
  Optimizer = Adam, Loss = Sparse Categorical Cross Entropy
- Train model, save logs to view in Tensorboard
- Hyperparameters tuning on number of layers in RNN model, layer size, batch size, dropouts etc.
- Analysis model training progress in Tensorboard, check validation accuracy.
  Once the validation accuracy is over 0.5, the problem is solved and stop the model training.
- Export Tensorboard graphs to use in the project report.
- Write project report and submit to Udacity.