Machine Learning Engineer Nanodegree

Capstone Proposal

Arnab Karmakar
Januray 30th, 2020

Proposal

Domain Background

Stock market is one of the most competitive financial markets. Traders need to compute the financial workloads with low latency and high throughput. In the past, people were using the traditional store and process method to calculate the heavy financial workloads efficiently. However to achieve low latency and high throughput, data-centers were forced to be physically located close to the data sources, instead of other more economically beneficial locations. This is the main reason, the data-streaming model was developed and it can process large amount of data more efficiently. It was shown in studies that using data streaming we can solve the options pricing and risk assessment problems using traditional methods, for example Japanese candlesticks, Monte-Carlo models, Binomial models, with low latency and high throughput. However instead of using those traditional methods, we approached the problems using machine learning techniques. We tried to revolutionize the way people address data processing problems in stock market by predicting the behaviour of the stocks. In fact, if we can predict how the stock will behave in the short-term future we can queue up our transactions earlier and be faster than everyone else. In theory, this allows us to maximize our profit without having the need to be physically located close to the data sources.

We examined two models.

Model 1 - We used a complete random model using a random number genertor. If the generated number is greater than `.7` (inclusive), we buy a stock and if it is less than `.3` (inclusive) we sell. We do nothing if the generated number is between `.3` (exclusive) and `.7` (exclusive). We close our position just before the exchange closes for the day.

Model 2 - In this model, we use multiple machine learning algorithms to get the final decision. For example, we will use classification techniques to classify stocks into different buckets (For us, it is three). Then we train three neural networks for these three buckets and take position in the market based on confidence levels of predicted values from these networks. Each of those models are applied on real stock market data and checked whether it could return profit.

Problem Statement

For the concept of this thesis we tried to predict the price of the stock in the short term future and decide whether it is better to buy, sell our stocks or do nothing(no trading). There is no strict definition of short term future. It can be any interval from nanoseconds until a few days. We decided that we will use 1-day interval as our prediction time. As the stock price depends on the time, time interval is a parameter that had to be decided. We think that 1-day can be a good representation of short term future. Also using a constant time interval simplifies the problem significantly. The main objective is to maximize the profit by trying to increase the capital. We tried to come up with an optimal trading strategy to maximize the potential profit. The main

idea is to model a stock trading into 1-day intervals and using historical information of the stock, we tried to predict if we should avoid trading or take a position which would either end above or below our entry price in the stock. Also we tried to train and test our model on historical stock data collected during the period of November 2010 to June 2019.

Datasets and Inputs

There are two types of data we have to download from external sources, the list of stocks we want to analyze and then prices for each stock in the given time period for analysis. DataHub provides API to download list of tickers(instruments). Secondly, yahoo finance provides free stock price data. We are going to use pandas_datareader package to download Yahoo finance data. The dataset will provide us Open, High, Low, Close, Adj. Close and Volume data.

Once we have data downloaded from the external sources, we are going to calculate different technical indicator values using the library talib. We will then choose our window for price change calculation. If we consider N as our window, we will first calculate changes for past 1, 2, 3, ..., N-1 days changes. Following table elaborates it further.

Downloaded stock price data

Date	Open	High	Low	Close	Adj. Close	Volume
Dec 13, 2019	361.05	365.21	354.64	358.39	358.39	6,570,900
Dec 16, 2019	362.55	383.61	362.5	381.5	381.5	18,174,200
Dec 17, 2019	378.99	385.5	375.9	378.99	378.99	8,496,800
Dec 18, 2019	380.63	395.22	380.58	393.15	393.15	14,121,000
Dec 19, 2019	397.32	406.85	396.5	404.04	404.04	18,107,100

With N = 2, we generate change data

Date	Open	High	Low	Close	Adj. Close	Volume	N - 1	N - 2
Dec 13, 2019	361.05	365.21	354.64	358.39	358.39	6,570,900		
Dec 16, 2019	362.55	383.61	362.5	381.5	381.5	18,174,200	23.11	
Dec 17, 2019	378.99	385.5	375.9	378.99	378.99	8,496,800	-2.51	20.6
Dec 18, 2019	380.63	395.22	380.58	393.15	393.15	14,121,000	14.16	11.65
Dec 19, 2019	397.32	406.85	396.5	404.04	404.04	18,107,100	10.89	25.05
Dec 20, 2019	410.29	413	400.19	405.59	405.59	14,752,700	1.55	12.44

With technical indicators calculated (for example Simple Moving Average, SMA)

Date	Open	High	Low	Close	Adj. Close	Volume	N - 1	N - 2	SMA (9)

Date	Open	High	Low	Close	Adj. Close	Volume	N - 1	N - 2	SMA (9)
Dec 13, 2019	361.05	365.21	354.64	358.39	358.39	6,570,900			
Dec 16, 2019	362.55	383.61	362.5	381.5	381.5	18,174,200	23.11		
Dec 17, 2019	378.99	385.5	375.9	378.99	378.99	8,496,800	-2.51	20.6	
Dec 18, 2019	380.63	395.22	380.58	393.15	393.15	14,121,000	14.16	11.65	
Dec 19, 2019	397.32	406.85	396.5	404.04	404.04	18,107,100	10.89	25.05	
Dec 20, 2019	410.29	413	400.19	405.59	405.59	14,752,700	1.55	12.44	
Dec 23, 2019	411.78	422.01	410	419.22	419.22	13,319,600	13.63	15.18	
Dec 24, 2019	418.36	425.47	412.69	425.25	425.25	8,054,700	6.03	19.66	
Dec 26, 2019	427.91	433.48	426.35	430.94	430.94	10,633,900	5.69	11.72	399.67
Dec 27, 2019	435	435.31	426.11	430.38	430.38	9,945,700	-0.56	5.13	407.67
Dec 30, 2019	428.79	429	409.26	414.7	414.7	12,586,400	-15.68	-16.24	411.36

Our label is going to be the next day change

Date	Open	High	Low	Close	Adj. Close	Volume	N - 1	N - 2	SMA (9)	Label
Dec 13, 2019	361.05	365.21	354.64	358.39	358.39	6,570,900				
Dec 16, 2019	362.55	383.61	362.5	381.5	381.5	18,174,200	23.11			-2.51
Dec 17, 2019	378.99	385.5	375.9	378.99	378.99	8,496,800	-2.51	20.6		14.16

Date	Open	High	Low	Close	Adj. Close	Volume	N - 1	N - 2	SMA (9)	Label
Dec 18, 2019	380.63	395.22	380.58	393.15	393.15	14,121,000	14.16	11.65		10.89
Dec 19, 2019	397.32	406.85	396.5	404.04	404.04	18,107,100	10.89	25.05		1.55
Dec 20, 2019	410.29	413	400.19	405.59	405.59	14,752,700	1.55	12.44		13.63
Dec 23, 2019	411.78	422.01	410	419.22	419.22	13,319,600	13.63	15.18		6.03
Dec 24, 2019	418.36	425.47	412.69	425.25	425.25	8,054,700	6.03	19.66		5.69
Dec 26, 2019	427.91	433.48	426.35	430.94	430.94	10,633,900	5.69	11.72	399.67	-0.56
Dec 27, 2019	435	435.31	426.11	430.38	430.38	9,945,700	-0.56	5.13	407.67	-16.24
Dec 30, 2019	428.79	429	409.26	414.7	414.7	12,586,400	-15.68	-16.24	411.36	

We are going to normalize our data using sklearn's MinMaxScaler.

Solution Statement

The number of stocks are more than 3000. We are going to discard stocks whose volume is less than `1,000,0000`. We reach our end result in two steps.

Step 1 - We will remove the label column and use all the normalized features and pass through different classification models. For example, we could use k-means algorithm to divide the data into three buckets. Later, we would analyze the clusters to understand how well the data have been categorized. We will have to do some parameter tuning to reach to our desired value.

Step 2 - Once data has been categoried, we are going to run three recurrent neural network for the three clusters. We are going to use following features -

```
- open
- high
- low
- close
- adj. close
- volumne
- Triangular Moving Average
- SAR
- MACD
- RSI
- STOCH
- AD
- ATR
- N  - x (changes each day compared one of the previous days going back upto N
days)
```

During prediction, data will be passed to all the three networks. The final decision would be the decision based on the RNN network with highest confidence level.

Benchmark Model

Trading is a probability game. Our first model is using a random number generator to generate a number between `0` and `1` and taking position based on the value. The goal is to prove that a systemic approach (in this case using machine learning models) yields higher returns as compared to taking position randomly.

If the value of random number generator is less `.3` (inclusive), we will short the stock and close the position before market closes for the day. Similarly, if the generated number is greater than `.7` (inclusive), we will buy the the stock. Any value between `.3` and `.7` will be excluded and no position in market would be taken. The success of this approach is going to be based on how many positive were closed with profit. This data is going to be our benchmark to compare with second model.

Evaluation Metrics

The evaluation metrics for both the models are simple. We calculate how many positions were closed profitably and divide the number with total positions taken.

```
Accuracy = Profitable positions / Total number of positions
```

The highest accuracy model is the winning model.

As of Jan 2020, most of the renowned brokers removed their round trip trading cost for stocks. Hence, we can safely ignore trading cost associated in our strategy.

Project Design

After downloading stock list, we are going to download prices for each stocks and calcualte features as described above. The stocks with less than 1,000,000 volume will be discarded. Using our *featuresets* - open,

high, low, close, adj close, volume, n - 1, n - 2, n - 3, n - 4, n - 5, n - 6, n - 7, n - 8, n - 9, tma, sar, MACD, RSI, STOCH, AD, ATR and *label* column, we are going to classify our records into three categories. We are then going to look into what trend each of these cluster/classes/category follow. For example, if cluster 1 is more bullish, cluster 2 is more bearish, then we would leave cluster 3 for no trade condition. Based on the clusters, we will divide the data set into three groups.

```
Group 1 data = Cluster 1 = Bullish (in this example)
Group 2 data = Cluster 2 = Bearish (in this example)
Group 3 data = Cluster 3 = No trading (in this example)
```

We are now going to divide each group into train and test data. This data will be used to train and validate three Recurrant Neural Network (RNN). The final decision whether to take a position or not will be based on following rules.

```
RNN 1 (Based on Cluster 1, Bullish in our example) = Predicted change, Confidence
RNN 2 (Based on Cluster 2, Bearesh in our example) = Predicted change, Condfidence
RNN 3 (Based on Cluster 3, No trading in our example) = Predicted change,
Confidence
```

The prediction result of the model with highest confidence will be followed. For example, if RNN 2 has the highest confidence, then we would short stock. The position will be closed before the close of the market on that day.

At the end, accuracy is calculated using

```
Accuracy = Profitable positions / Total number of positions
```

Since each stock behave differently, we are going to use separate classifiers and RNNs for each stocks. There are lot of parameters that we have to play around to reach the optimal value. We are going to experiment with different values of `x` in `n - x` features that we dicussed above. This is an optimization problem where we have to figure out what optimal value of x produces the best result without overfitting our model. We are also going to evaluate if using a different machine learning algorithm instead of RNN would perform better in our case.

References

1 Stock Market prediction using Artificial Neural Networks, Rafael Konstantinou

2 Predicting Stocks with Machine Learning, Magnus Olden