Machine Learning Engineer Nanodegree

Capstone Proposal

Jacob Anthony

(jacobscottanthony@gmail.com)

February 3rd, 2020

Proposal

Domain Background

The domain of this proposal is finance and more specifically algorithmic trading. Algorithmic trading has existed for many decades (Gupta, 2015) and has increased in popularity since the digitalization of exchanges since the late 1980's. This has coincided with a simultaneous reduction in computing cost due to Moore's law (Wikipedia, 2020), an increase in Artificial Intelligence research (Wikipedia, 2020) and an increase in value of global markets (Federal Reserve Bank of St. Louis, 2020). Financial markets are ripe for the application of Artificial Intelligence techniques given they are typically data-rich, are vast and varied, are difficult to model through quantitative means and offer a substantial financial reward to players able to successfully develop them.

My motivation for choosing this topic is my personal interest in the subject. I have tried implementing trading strategies before and have had this idea in the back of my mind since I started this program.

Problem Statement

For the lay investor, most (active) trading strategies are too onerous to implement as they require one to be constantly watching and reacting to the market. Thus, most individual investors adopt a buy and hold strategy that seeks to take advantage of longer term moves in the market. Training and deploying a suitable Reinforcement Learning Agent to act on one's behalf in the market is a potential solution for individual investors to increase their returns versus a Buy and Hold strategy.

Datasets and Inputs

This proposal will make use of public financial data in the form of daily (open, high, low, close, volume) or intraday (open, high, low, close, volume) pricing data for the financial instruments considered. Published research in this area shows models being successfully trained on both daily and intraday data. Sources will include:

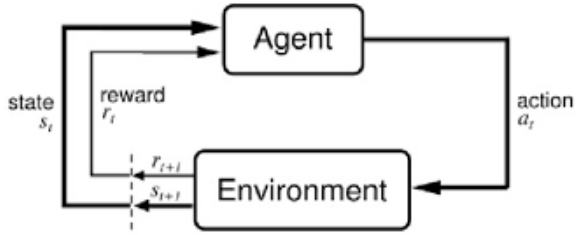
- Stock and Market Index's, daily resolution:
  - Alphavantage API
    - https://www.alphavantage.co/
- Stock and Market Index's, intraday resolution:
  - Purchased from:
    - https://pitrading.com/historical-stock-data.html
    - https://firstratedata.com/
    - https://www.backtestmarket.com/en/

Pricing and trading volume data are both time-series data. This data forms the basis of the environment that I will train the agent to act within. In addition to the time-series data, a number of leading and lagging market measures (also known as

Technical Indicators) will be added to the state which will help to describe the past, current and (possible) future state of the market at each time step. (Krolner, Vanstone, & Finnie, 2010) find the most commonly used indicators as:

- Simple moving average, SMA
  - https://www.investopedia.com/terms/s/sma.asp
- Exponential moving average, EMA
  - https://www.investopedia.com/terms/e/ema.asp
- Relative strength index, RSI
  - https://www.investopedia.com/terms/r/rsi.asp
- Rate of change, ROC
  - https://www.investopedia.com/terms/r/rateofchange.asp
- Moving average convergence / divergence, MACD
  - https://www.investopedia.com/terms/m/macd.asp
- Williams oscillator, WO
  - https://www.investopedia.com/terms/w/williamsr.asp
- Average true range, ATR
  - https://www.investopedia.com/terms/a/atr.asp

This set of technical indicators will be the starting point for the Environment. It is noted that there are a large number of additional possibilities that could be considered for the agent (TradingView, 2020) & (Wikipedia, 2020).



Following the diagram above the agent will receive a state from the Environment in the form of price, volume and an array of technical indicators. It will then decide on an action (Buy, Sell or Hold). The Environment will return a reward (or punishment) and the next state of the environment. The agent will then continue to take actions and receive rewards seeking to maximise its reward until either the episode ends or another stopping criteria is reached (For eg. Training Loss, Max Drawdown).

The maximum time horizon for training I am going to consider will be limited by the data available. From the sources above, it'll be easy to collect daily data for most US Stocks going back to the early 2000's. At a daily resolution this will be ~ 20y x 250d = 5000 rows per instrument. If I increase this to minute resolution this becomes ~ 5000d x 24h x 60m = 7.2M rows per instrument. I will only use stocks currently found in the S&P500 index such as Ecolab Inc (NYSE: ECL).

Solution Statement

Reinforcement Learning is an ideal solution for training a trading agent. The market can be described as a continuous Markov Decision Process (MDP). The Environment (market) can be specified as an array of real numbers at each timestep, *Actions* can be simplified to Buy, Sell, Hold and *Rewards* can be calculated at each timestep. The action space is continuous as trades can be executed at any price someone is willing to sell at and with any percentage of the current trading account balance.

A number of approaches to solving this problem have been considered. Dynamic Programming could be applied to a problem like this but it is computationally very expensive due to the large possible state space. Proximal Policy Optimization is also reported in the literature although it can be unstable in learning due to its Monte-Carlo sampling of the Environment.

Due to my previous experience in implementing the algorithm, its dominance in the literature and applicability to continuous action spaces I've chosen to train the agent using the Deep Deterministic Policy Gradient (DDPG) algorithm. This algorithm is also known as Actor/Critic.

Benchmark Model

I am proposing one benchmark model to compare the results achieved by this proposal.

- "Practical Deep Reinforcement Learning Approach for Stock Trading" (Xiong, Liu, Zhong, Yang, & Walid, 2018)
  - https://arxiv.org/pdf/1811.07522
  - This team trained a DDPG agent to successfully trade on stocks within the DJIA and were able to achieve a Sharpe Ratio of 1.77 with an annualized return of 22.24%.

In addition, I harbour some doubt that this model represents results that are representative of state-of-the-art models from the domain. This is due to the missing incentive for someone to publish an exception model for the world to profit from. This is further compounded by most financial companies likely considering any implementations of said models to be a combination of highly proprietary, strictly confidential and intellectual property.

Evaluation Metrics

There are a number of well understood metrics to evaluate the performance of a trading system. The key ones I will use for this proposal are:

- Buy and Hold of the S&P500, B&H
- Final Portfolio Value, FPV
- Sharpe Ratio, SR

- https://en.wikipedia.org/wiki/Sharpe_ratio
- Maximum Drawdown, MDD
  - https://www.investopedia.com/terms/d/drawdown.asp

If these measures need to be substituted (or supplemented) to help root out and adjust the agent's behaviour (positively or negatively), I will investigate the implementing Risk Adjusted Return, Time Discounted Return, Mean Positive Return and Value at Risk as additional ways to evaluate the agent's performance.

Project Design

Below is the approximate workflow that I am planning to follow in completing this project.

- **Data Acquisition**
  - Acquire the daily and intraday pricing and volume data for all the stocks in the S&P500 index.
- **Data Pre-Processing**
  - Pre-process the pricing and volume data into a continuous time series by removing the dates and replacing them with whole numbers.
  - Add all the technical indicators as mentioned above to the dataset.
  - Save the dataset in a format that is easy (and fast) for the Environment to load and process during training.
- **DDPG Architecture Setup**
  - Starting with the openai Spinning up implementations I will create the appropriate files in a Python Jupyter workspace.
  - I will need to write a custom environment class.
  - I will need to write a reward function.
  - Thoroughly test the implementations written here to ensure they make sense.
  - *N.B I will refer to the benchmark models and other papers referenced in the Bibliography, my Quadcopter Project DDPG project implementation and other online resources as necessary here. I am expecting this stage to be difficult.*
- **DDPG Training**
  - Initialize all hyper-parameters using reasonable estimates from the referenced papers in this proposal.
- **DDPG Hyper-Parameter Tuning**
  - I am planning to start with the manual grid-searching.
  - If this manual grid search is wasting too much time I will setup a more formal grid search of the variables of interest.
  - Stretch Goal: Experiment with Random Walk and Bayesian Optimization of the hyper-parameter tuning to see if this can further improve the grid search.
- **DDPG Agent Performance Evaluation vs. Benchmark Model**
  - I will chart and discuss the performance of the agent in training, validation and deployment.

- **Loop through Architecture, Training, Tuning and Performance Evaluation Steps**
  - Loop through steps as necessary or as additional tweaks are made to the Environment, Reward Function and/or Actor/Critic Network Architectures.
- **Write up the Report**
  - Using the Udacity Project Rubric, write up the work completed and submit.
- **Real-World Deployment…….;-)**

Stretch Goals:

- **Add Additional Measures to Environment**
  - Next day and next week LSTM Price Forecasting following an implementation by (Lim, Zohren, & Roberts, 2019).
  - Current Twitter Sentiment at each timestep following an implementation by (Nan, Perumal, & Zaiane, 2020).
- **Performance With & Without New Measures**
  - Chart and discuss the differences seen in training, validation and deployment.
- **Principal Component Analysis (PCA) for each Technical Indicator**
  - Conduct a PCA for each technical indicator in the data set with respect to price. Simplify the state array to the Technical Indicators that represent 80% of the PCA and see if this reduces noise in training without sacrificing model performance.

Bibliography

Federal Reserve Bank of St. Louis. (2020, February 2). *Wilshire 5000 Total Market Full Cap Index*. Retrieved from https://fred.stlouisfed.org/series/WILL5000INDFC

Gupta, A. (2015, June 2). *History of Algorithmic Trading, HFT and News Based Trading*. Retrieved from https://blog.quantinsti.com/history-algorithmic-trading-hft/

Krolner, B., Vanstone, B., & Finnie, G. (2010). Financial Time Series Forecasting with Machine Learning Techniques: A Survey. *ESANN 2010 Proceedings* (pp. 25-30). Bruges, Belgium: ESANN.

Lim, B., Zohren, S., & Roberts, S. (2019). Enhancing Time-Series Momentum Strategies Using Deep Neural Nets. *The Journal of Financial Data Sciences*, 19-38.

Liu, J., Chao, F., Lin, Y., & Lin, C. (2015). Stock Prices Prediction using Deep Learning Models.

Nan, A., Perumal, A., & Zaiane, O. (2020). Sentiment and Knowledge Based Algorithmic Trading with Deep Reinforcement Learning.

TradingView. (2020, February 2). *Category:Indicators and overlays*. Retrieved from https://www.tradingview.com/wiki/Category:Indicators_and_overlays

Wikipedia. (2020, February 2). *History of aritifical intelligence*. Retrieved from https://en.wikipedia.org/wiki/History_of_artificial_intelligence#Deep_learning,_big_data_and_artificial_general_intelligence:_2011%E2%80%93present

Wikipedia. (2020, February 2). *Moore's law*. Retrieved from https://en.wikipedia.org/wiki/Moore%27s_law

Wikipedia. (2020, February 2). *Technical Indicator*. Retrieved from https://en.wikipedia.org/wiki/Technical_indicator

Xiong, Z., Liu, X., Zhong, S., Yang, H., & Walid, A. (2018). Practical Deep Reinforcement Learning Approach for Stock Trading.