School of Informatics

Informatics Project Proposal
Diversified Portfolio Stock Selection Agents

B174744
April 2021

Abstract

Automatic stock trading systems should trade diversified portfolios in order to minimize risk. Researches have implemented multi asset reinforcement learning based systems that outperform markets. But such systems usually trade only the one stock that is predicted to perform the best in the future. We propose a project plan to built a new system, based on two agents. The signals of the two agents are passed through a mathematical framework to trade diversified decisions.

Date: Tuesday 13th April, 2021

Tutor: S. Guo
Supervisor: Prof. V. Restocchi

1 Motivation

Researchers have built many models that give signals for trading assets. Consider for example the work of Ding *et al.* [1, 2], who built a model using sentiment analysis, that predicts if a stock will move up or down. Imagine this model gives a buy signal for a stock. Now imagine that you have twenty more such models that give different buy and sell signals for different stocks. What are you supposed to do?

A human selecting which model to trust is sub-optimal and time consuming. Instead, we should have an automatic mechanism that takes action by taking into consideration the signals by the models and the market conditions. Reinforcement learning agents are appropriate for this job. A reinforcement learning agent, observes the conditions (state) of an environments and selects an action. In our case, the agent would observe the market conditions and the trading signals of the models (state) and chose which stocks to buy or sell (action). After choosing an action the agent receives a reward and moves on to the next trading day, i.e. in the next state. The sequence of state, action, reward and next state is defined as a Markov Decision Process (MDP) [3]. Through appropriate algorithms the agent would be able to maximize the reward in the MDP.

Finding the appropriate signals to trade is black art and time consuming, so we will not consider that. Instead we will only build the system that trades multiple assets. Keep in mind that the system we want to built is supposed to be our final decision maker before investing our money. Hence, it should not invest only in the one or two assets it predicts to do the best in the future. Rather, the system should adopt a more diversified approach and invest in multiple stocks.

1.1 Problem Statement

An inspiring trader can choose between two solutions for an automatic multi asset trading system. However, both have significant limitations. First choice is the classical mathematical programs used in financial optimization such as the famous min variance portfolio [4]. Such methods do not scale well, it is not clear how to add trading signals from other trading systems and are not very practical for small investor.

The second choice is to use Reinforcement Learning based multi asset trading system [5, 6, 7, 8]. However, RL agents tend to pick the one or two stocks that are believed to perform the best in the future. If the agent believes stock XYZ will be the best performer in the future, then the agent has no incentive to select any other stock. This is highly problematical as investing all the capital in one or two stocks is risky. Researchers are forcing diversification by the punishing non-diversified agents through the reward function [8] or the loss function of the neural networks [7]. Such approaches though are artificial and more importantly do not minimize risk [1] in trading. The latter is highly important, because it has been reported that less volatile stocks have higher returns [9].

One may also attempt to change the reward function of the RL agent. Instead of rewarding the agent based on returns, we reward it by some risk adjusted reward function [10]. Althought the agent will select less risky stocks, there is no guarantee the agent will select multiple stocks. The agent will select only the one or two stocks that are predicted to have the best risk adjusted reward in the future.

We want to have a trading system that (1) trades diversified stocks, (2) minimizes risk, (3)

[1] Reduces risk but not minimize.

maximizes returns, (4) is easily scalable and practical and (5) can use signals from other trading systems. To the best of our knowledge, current work in literature does not satisfy all five criteria.

1.2 Research Hypothesis and Objectives

We propose to use two reinforcement learning agents. The first agent will decide the percentage of capital to be used for short selling[2], cash and long buys[3]. The second agent will output which stocks are "good" or "bad". The outputs of the trading agents will be fed through a mathematical framework similar to Markowitz's [4], to give the final portfolio of the system.

We make the following assumptions. (1) Our decisions do not affect the market. (2) We can sell or buy stocks at any time. (3) Investor needs to have money aside to short sell.[4] (4) No cost for short selling. In practice, when we short sell we have to pay a small fee every day. We will assume this fee is zero.

1.3 Timeliness and Novelty

The project proposes a new framework to trade assets.

The project can be thought in three parts. (1) Building a prototype. This would include the trading environment, the mathematical framework and the two agents. It would be expected that this step would take 4 weeks. Given that I have experience with the dataset, it would be faster than normal to built the trading environment. Also, we will use standard RL algorithms for the agents. The challenge will be with the mathematical framework and merging the components together. Each of the two though, would be doable within a week each. The final two parts are, (2) Tuning the system and (3) writing the report. We expect that (2) would take three weeks. Moreover, while waiting for the experiments to run, I will be able to work on (3).

1.4 Significance and Beneficiaries

The proposed system has an immediate application in stock trading. In order to optimize portfolios, some companies use linear or quadratic programs. Such programs optimize some value given some conditions. It is very hard to impose custom conditions on such programs, like taxes. Such conditions introduce if/then statements over the region we optimize. This makes the region highly non-convex, hence hard to optimize and often requiring companies to buy custom solutions. The proposed system would allow to pass any condition through the MDP with few lines of code.

Example 1: Taxes. Adding custom tax brackets in a linear program is clearly very hard. In our framework, we would just change the reward function. Example 2: Illegal buys. Some laws prevent large companies to buy stocks that they sold the last 30 days. It is also clear that it is hard to impose such conditions on a mathematical program. In the proposed framework, we would ban that action for the agent. Any restriction can be modelled through the MDP.

Some other benefits of the proposed framework include the following. First, the system may outperform the basic program used. This is attributed to two factor; (1) the system takes more data than just closing prices, (2) the intelligent agents may be able to find tricks in the MDP.

[2]Short selling is taking a negative position on a stock.

[3]Buying a stock is also referred as long position.

[4]For example if one short sells 20 dollars worth of a stock, that person must have 20 dollars in the account.

A second advantage is speed. Gradient descent used in neural nets is likely more efficient than algorithms used to solved mathematical programs over non-convex regions.

More importantly, as alluded in the introduction, the framework would be ideal for hands off trading. A trader can pass all the signals of trading algorithms through the inputs of the neural networks and get a diversified portfolio based on the ideas of modern portfolio theory and artificial intelligence.

2 Background and Related Work

2.1 Value vs Policy RL

Reinforcement learning is split into value and policy based. Value based RL, like Q-learning [11], predicts the future reward for an action. The agent then picks the action that gives the most reward. Policy based learns the best action directly without calculating the expected reward. Both value [7, 12, 6] and policy [13, 10, 5, 8] based approaches have been cited in literature for stock trading. In the next paragraph we explain why we will use policy based.

Markets are known to be volatile and hard to predict [14]. It is therefore argued that predicting the exact reward is not possible, hence policy based RL is a better approach [10, 13]. Imagine a stock moving down. It may be hard to predict the exact reward, however it may be significant easier to learn that the agent should clear the position.

It is worth noting that some Q-learning trading researchers are clipping the reward of their agents. For example [12], rewarded the agents with $+1$ for profitable trades and -1 for non-profitable or not allowed trades. Such Q-learning agents do not predict the returns, rather the probability of a profitable trade.

2.2 Related Work

Patel used two agents in one trading systems to find ideal limit orders[5] for Bitcoin prices [12]. He used one agent to signal if Bitcoin is a good buy and another agent to decide the price of the limit order. Both agents were trained using Q-learning. The author gave different reward to each agent. The agent that decided if Bitcoin is a good buy got a reward of -1 for a losing trade and 1 for a positive trade. The other agent got a reward based on the optimal limit order. We are going to use those ideas.

Lee *et al.* used multiple Q-Learners to built a diversified portfolio system [7]. Each Q-learner gave $-1, 0$ or 1 for each ticker. Then the authors collected those signals and passed them through a function that took averages to built a diversified portfolio. There was an extra term in the loss function of the Q-learners that was based on the correlation of the outputs of the agents.

Other multi asset trading strategies include the following. Park *et al.* used a Q-learning to train an agent which trades three tickers [6]. The agent took either short, neutral or long position in each ticker. The output of the Q-network was and 8 dimensional vector for each possible action. One non Q-learning is strategy is that of Yang *et al.* [5]. The authors trained three different reinforcement learning algorithms that learn the policy directly. Then they trade at quarter t based on which of the three algorithms performed the best at quarter $t-1$.

[5]Request to buy a stock. Stock is bought only if someone is willing to buy at the price of the limit order.

2.3 REINFORCE

REINFORCE, or policy gradient as is known today, is a policy based RL algorithm. We briefly explain REINFORCE here. Let $\pi(\cdot \mid \theta)$, be the policy given by parameters θ and $\gamma \in [0, 1]$ be the discount factor constant. We generate an episode using π. Say we get the following state, action, reward sequence;

$$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T_1}, R_T$$

. We first find the total loss. Initialize $L = 0$ and $G = 0$ and loop backward for $t = T \ldots 1$;

$$G \leftarrow R_{t+1} + \gamma G$$
$$L \leftarrow L - G \log \pi(A_t | S_t, \theta)$$

The total loss per step is then $L \leftarrow L/T$. We then update θ using a step of gradient descent towards L.

2.4 Markwotiz Portfolio

We discussed that the decisions of our agents will pass through a mathematical framework, similar to Markwotiz [4]. If x is some portfolio of stocks, we will need some measurement $\mathfrak{R}(x)$ of how risky this portfolio is. The idea is that we then select the portfolio x that minimizes the risk $\mathfrak{R}(x)$ given that the portfolio satisfies some conditions.

Typical conditions is that the portfolio is expected to return some threshold. We will use different conditions to adapt to our problem. We can find the x that minimizes $\mathfrak{R}(x)$ by solving a linear or quadratic program.

A typical risk measure used is the covariance matrix of past returns [4]. If we trade n stocks, then let Σ be the $n \times n$ covariance matrix of past returns. If x is some portofolio, then $\mathfrak{R}(x) = x^T \Sigma x$.

3 Programme and Methodology

We want to built a reinforcement learning trading agent that trades diversified portfolios. The novelty of this work is that we use the following two agents. The first agent is called the capital allocation agent and returns a three dimensional vector (c_1, c_2, c_3), where $c_1 + c_2 + c_3 = 1$ and $c_1, c_2, c_3 \in [0, 1]$. c_1 is the ratio of capital allocated to short selling, c_2 is the ratio of capital allocation to cash and c_3 to long positions. The second agent is the stock selection agent, which for each stock outputs $-1, 0$ or 1, corresponding to short sell, hold and buy signal for the corresponding stock.

If we trade n stocks then denote the portfolio x^t at time t by $x^t = (x_1^t, x_2^t, ..., x_n^t)$. We will make the assumption that $\sum |x_i| = 1$. This is assumption (3) in section 1.2. Let $\mathfrak{R}(x)$ denote some risk measure of x, then at time $t + 1$ we select the portfolio that minimizes the following program,

$$
\begin{aligned}
&min \;\; \mathfrak{R}(x^{t+1}) \;\; s.t. \\
&x_i^t < x_i^{t+1} \;\; if \;\; a_i = 1 \;\; \forall i \\
&x_i^t = x_i^{t+1} \;\; if \;\; a_i = 0 \;\; \forall i \\
&x_i^t > x_i^{t+1} \;\; if \;\; a_i = -1 \;\; \forall i \\
&\sum_i max(x_i^{t+1}, 0) = c_3 \\
&\sum_i max(-x_i^{t+1}, 0) = c_1
\end{aligned}
$$

, where a_i is the output of the stock selection agent for asset i. The first conditional equation ensure that if the stock selection agent likes a stock, we buy the stock at time $t+1$. Similarly for conditional equations two and three. The last two ensure that the capital are allocated according to the capital allocation agent. The optimization program can be solved using the CVXOPT library in Python. The risk measure $\mathfrak{R}(x)$, will be either linear or quadratic. CVXOPT contains solvers for both linear and quadratic programs. The constrains in this program are linear. The CVXOPT documentation states clearly that the library is able to solve linear constrains.

The risk measures considered will be the covariance matrix [15] and conditional value at risk [16] if time permits. The last measure has different values for short selling and long buys. One easy way is to not allow short selling, but we could do some engineering to make it work. At each time step a stock will be classified as a potential short sell or a long buy. Let s^- be the set of stocks classified as potential short sells and s^+ the ones as long buys. Define $z_i^+ = 0$ if stock i is in s^-, otherwise $z_i^+ = x_i$. Similarly define z_i^-.Then if $\mathfrak{R}^+$ is the risk for buys and $\mathfrak{R}^-$ is the risk for shorts, we define,

$$\mathfrak{R}(x) = \mathfrak{R}^+(z^+) + \mathfrak{R}^-(z^-)$$

The risk measures outlined will be estimated using past data. We chose the above risk measures ,because the covariance matrix is the most known risk measure and CVaR is considered the state of art of risk measures because it satisfies the mathematical properties of coherence.

Now the program defined may be infeasible depending on the outputs of the agents. However we can do the following to ensure this never happens. We first run the stock selection agent. We introduce two boolean flags. The first boolean flag is on if the stock selection agent has output -1 for at least one stock. Similarly for the second boolean flag. If we feed those boolean flags through the capital allocation agent along with its own previous predictions, the agent will be able to learn when the program is infeasible.

The capital allocation agent (CA) will be controlled by a CNN. We avoid RNN, due to computational constrains and dense networks because they do not capture the sequential structure of data. The inputs of the CNN, or the state space of the CA agent, will be a $d \times n$ matrix, which will be the closing prices of n stocks over the past d time periods. Potentially, we could add more data by adding channels. For example, we make a channel with the high prices as well. The CNN will be eventually flattened out and concatenated with the Boolean flags discussed. The agent is rewarded by the profit it achieves.

The stock selection agent will also be controlled by a CNN. Define $\mathfrak{c}$ to be the closing price at some time divided by the closing price 20 days ago. Similarly $\mathfrak{h}$ is the high price divided by the

closing price 20 days ago. Similarly for open $\mathfrak{o}$ and low $\mathfrak{l}$. We do this transformation in order for the CNN to generalised between different price ranges for stocks. The input of the CNN agent will be a $d \times 7$ matrix. For last d time periods, the inputs of the CNN include $\mathfrak{c}$,$\mathfrak{o}$,$\mathfrak{h}$,$\mathfrak{l}$, Volume, SPY price, Gold price, Bitcoin price. Here is where the trading signals of other models could be inserted.

The stock selection agent will be the same for all stocks. At each episode, the SS agent will be optimized with respect to one stock only, but take decision for all stocks. The agent will be rewarded with $+1$ for positive trade and -1 for negative trade or not allowed trade. This is done for two reasons. First there due to time constrains and second if the agent does not select the outputs independently for all stocks, it may be able to coordinate the actions and bypass the restrictions imposed by the mathematical program. We use REINFORCE to optimize both agents.

3.1 Risk Assessment

1. The agents could fail to give diversified results. There have been reported two techniques to encourage diversified stock selection. Aboussalah *et al.* [8] penalized the agent through the reward function. Another approach is through the loss function [7]. If agents do not give diversified portfolios, both approaches will be tried. This has *low* chance of happening due to the fact that SS agent takes independent decisions. The impact would be *medium* to high depending on the degree of severity.

2. REINFORCE has high variance. This has a *medium* chance of happening with a *medium* impact on the project. If we face such an issue we will use the PPO algorithm [17]. This algorithm clips the policy based on the previous policy and is smoother, so it could solve the problem.

3. The CA agent reward depends on the actions of the SS agent. There is a *medium* chance of the CA agent being unstable, which would have a *medium* impact on the project. If it gets bad, we can merge both agents into a single one.

3.2 Ethics

There is no ethics consideration in terms of researching the topic as it does not involve humans. However, the work could motivate someone (directly or indirectly) to trade in the stock market. People usually lose money in the stock market. Some people even treat the stock market as a casino and get addicted. For these reasons, we may put in the end of the work a warning about the potential dangers.

4 Evaluation

The data-set used will be stock data in 1 min intervals. The data contains forty thousands tickers from 6 February 2020. At the time the project starts, the data will end at 1st of June 2021. Due to computational constrains we will use data only from the stocks in the Dow Jones index. The author has been collecting the data-set since the start date using Google Finance API and storing it. The Google Finance API is reliable and has no cost.

The data is already cleaned for the tickers in Nasdaq and New York Stock exchange. Specifically the data has been adjusted for stock splits and merged together. All of the Dow Jones stocks

belong in those two exchanges. Minor adjustments need to be made in the GOLD, BTC and SPY prices.

If time permits, we will use also the daily closing prices of the Dow Jones stocks since 2010 in order to be able to compare results with literature. This data can be downloaded using any financial API.

Instead of train,validation and test data we use in sample and out of sample data as done in [5, 13]. Data is split into two periods; in sample and out of sample data. In sample data is used for training and validation. We start by the first x data points. $[0, x)$. The agent is trained on those timesteps and takes a decision for the next m timesteps. We then use a rolling window, meaning at the next step, we train on $[m, x + m)$ to take a decision for the next m timesteps and so on. The insample data is used to find the hyper-parameters along with x and m. During out of sample data we test the results. We fix the hyper-parameters but retrain continuously as before.

We evaluate the final trading agent on three standard financial metrics.(1) Annualized returns, (2) Sharpe ratio [15]. If agent takes n trades with returns r_i, then the Sharpe Ratio is

$$\frac{\frac{1}{n}\sum r_i - r^f}{var(r_i)}$$

where r^f is the risk free return. Risk free investment depends on the magnitude of money the inventor has. In any case, nowadays this value is close to 0 for any investor. For simplicity, we assume $r^f = 0$.

(3) Max drag-down. This is the biggest drop in the portfolio. We can eyeball this value. If the system trades diversified portfolios and has comparable annualized returns to the buy and hold strategy we would have a good system. We would expect the system to have a better values on metrics (2) and (3) since the system is designed to minimize risk and those metrics measure risk.

We will experiment with adding white noise to the dataset in every episode played by the agent to reduce overfitting. This will be a minor part of the project.

5 Expected Outcomes

The system selects good stocks to trade through recent advances in artificial intelligence and minimize risk through modern portfolio theory. An interesting comparison will be the Sharpe ratio given by our system against the Sharpe ratio given by the standard min variance portfolio used. If our system has better Sharpe ratio, then it would give evidence that the min variance portfolio does not minimize future risk ideally. Moreover, the research would provide evidence that the mathematical programs like Markowitz's [4] can be improved through artificial intelligence.

6 Research Plan, Milestones and Deliverables

We present the Gannt Chart in figure 1. We expect the (1) trading evniroment, (2) mathematical framework, (3) reinfoce and cnn and (4) merging all together would take one week each. After that we would need two weeks to tune the hyper-parameters. We also give one and half week

as flexible week. If the project goes well, we can consider some extensions discussed, otherwise we can use the extra time for what needs more time. After, July we can start writing the dissertation on top of tuning the hyper-parameters.

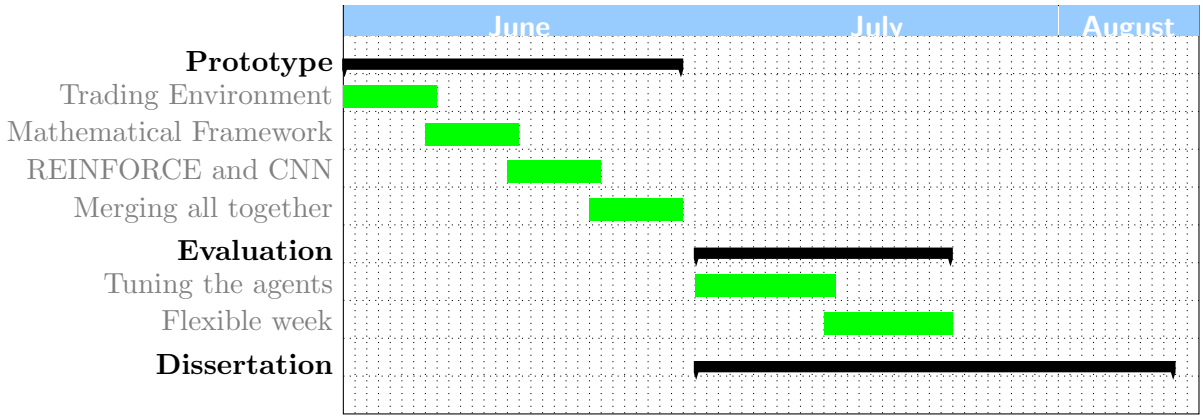

Figure 1: Gantt Chart of the activities defined for this project.

Milestone	Week	Description
M_1	2	Trading Environment and mathematical framework
M_2	4	Learning algorithms and prototype completed
M_3	7	Evaluation completed
M_4	10	Submission of dissertation

Table 1: Milestones defined in this project.

Deliverable	Week	Description
D_1	4	?????
D_2	7	?????
D_3	10	Dissertation

Table 2: List of deliverables defined in this project.

References

[1] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, 2014.

[2] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[3] Wendell H Fleming and Halil Mete Soner. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

[4] Harry Markowitz. Portfolio selection, 1959.

[5] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. *Available at SSRN*, 2020.

[6] Hyungjun Park, Min Kyu Sim, and Dong Gu Choi. An intelligent financial portfolio trading strategy using deep q-learning. *Expert Systems with Applications*, 158:113573, 2020.

[7] Jinho Lee, Raehyun Kim, Seok-Won Yi, and Jaewoo Kang. Maps: Multi-agent reinforcement learning-based portfolio management system. *arXiv preprint arXiv:2007.05402*, 2020.

[8] Amine Mohamed Aboussalah and Chi-Guhn Lee. Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. *Expert Systems with Applications*, 140:112891, 2020.

[9] Malcolm Baker, Brendan Bradley, and Jeffrey Wurgler. Benchmarks as limits to arbitrage: Understanding the low-volatility anomaly. *Financial Analysts Journal*, 67(1):40–54, 2011.

[10] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] Yagna Patel. Optimizing market making using multi-agent reinforcement learning. *arXiv preprint arXiv:1812.10252*, 2018.

[13] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[14] Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[15] William F Sharpe. The sharpe ratio. *Journal of portfolio management*, 21(1):49–58, 1994.

[16] Carlo Acerbi and Dirk Tasche. On the coherence of expected shortfall. *Journal of Banking & Finance*, 26(7):1487–1503, 2002.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.