NYU TANDON SCHOOL OF ENGINEERING

SUMMER 2022

MIDTERM DRAFT

BOA Reinforcement Learning Capstone

Nicole Zhao (xz2616)
Teresa Duan (rd3111)
Tianjing Cao (tc3385)
Wenqing Gao (wg2125)
Zhuotian Tang (zt2017)

Supervised By
Dr. Cristian Homescu , Dr. Francisco Rubio

1 Introduction

Quantitative strategy is a significant topic in modern Finance. Among distinct topics in quantitative finance, asset allocation, which is the process of finding optimal weights for component assets in a portfolio, is one of the most important areas . By tilting toward exposure to different kinds of assets, investors could obtain great diversified portfolio. For decades, the research on asset allocation problems to guide investment actions is always a blooming direction.

This paper mainly focuses on combining Deep Q-Network (DQN) on global indices' asset allocation. We made a few changes to the previous work introduced, and the experimental outcomes will further prove the profitability and generalization of our models.

1.1 Related Work

Reinforcement Learning (RL) is a blooming field famous for addressing a wide scope of complex decision-making tasks.

As illustrated by Das [1], a widely used dynamic programming known in RL literature is solving the "planning problem", which works backward from all values of mean-variance pairs to achieve the best outcomes. The approach is model-free and may be applied to problems with path-dependency and extremely large state spaces, which is often seen in the finance world. To deal with the commonly high dimensional data in wealth management tasks, Dixon introduced G-Learner as an RL algorithm that operates with explicitly defined one-step rewards, does not assume a data generation process, and is suitable for noisy data[2].

Researchers also paid efforts applying these techniques in Goal-Based Investing (GBI). According to [3], Sanjiv developed a dynamic programming methodology with investor preferences to maximize investor outcomes over multiple, potentially competing goals, even when financial resources are limited. Two years later, he reported an improved algorithm to construct an optimal portfolio trading strategy that maximizes the probability of attaining an investor's specified target wealth at the end of a designated time horizon, given a set of efficient or even inefficient portfolios [4].

On top of that, within a Deep Reinforcement Learning (DRL) framework, the RL agents construct and learn their knowledge through deep learning of neural networks. As pointed out by Arulkumaran [5], DRL can learn from raw sensors directly as input, which makes it an efficient solution to the traditional portfolio management problem. Katongo and Bhattacharyya in [6], seeking to increase the portfolio's risk-adjusted returns by capturing market trends and anomalies and reallocating the component assets, employed DRL Algorithms (PPO, A2C, and DDPG) as well as Neural Network Autoencoders. Similarly, Yang et al. in [7] reveals a DRL scheme that automatically integrates the three actor-critic algorithms. Focusing on Convolutional Neural Network, Benhamou et al., in [8] used three individual networks as inputs, including portfolio returns, asset features, and the previous portfolio allocation.

2 Asset Allocation Problems

Our team aim at using the RL techniques to solve asset allocation problem.

Typically, an asset allocation problem can be described as a mean-variance optimization (MVO) problem.

$$w^* = \arg\max_{w \in C} \{w'\mathbb{E}[r] - \frac{\lambda}{2} w' var[r] w\} \tag{1}$$

where
$w-$ the portfolio weights / holdings
$\mathbb{E}[r]-$ the vector of expected returns
$var[r]-$ the covariance matrix of returns
$\lambda-$ the risk aversion
$\mathcal{C}-$ the set of constraints (linear equality/inequality constraints, etc.)

This economic theory has been evolving for several decades, and has an enormous application on related areas. During the time MVO shows its flaws. First, the efforts takes to estimate the covariance matrix grow dramatically with the increase of assets. Second, the optimized results are highly sensitive to the expected returns and the covariance matrix, so that a small deviation of estimate could lead to a large bias of the allocation. Furthermore, the MVO is constructed on the assumption of market equilibrium, which can not be reached in reality. Several techniques were introduced to decrease these influence, the Black Litterman model is a good example. On the other hand, with the blooming of machine learning, more studies focus on solving asset allocation via reinforcement learning.

Reinforcement learning allows us to solve these dynamic optimization problems in an almost model-free way, relaxing the assumptions often needed for classical approaches. However, how to formulate the expected utility maximization problems using modern machine learning techniques remains a huge challenge for us. We also notice that the goal of asset allocation is to obtain the weights of the assets that maximize the rewards in a given state of the market considering risk and transaction costs. So the first step should be solving a prediction problem for the vector of expected returns and covariance matrix, which is exactly the framework of Deep Q-Network (DQN).

3 Reinforcement Learning Theory

Different from ML, RL develops from the idea of "interacting with the environment", which means the RL training agent gains experience and improves performance by constantly interacting with an environment. More specifically, the basic framework of RL problem consists of agent (the learner and decision maker)and environment that is defined by the Markov Decision Process (MDP) which is denoted by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. $\mathcal{S}$ is the set of all possible states (state space), $\mathcal{A}$ is collection of all available actions (action space), and $\mathcal{R}$ decides the reward received after each transformation between states.The goal of agent is to implement the optimal policy π^* which directs to take actions and gets the maximum total amount of reward (expected return) in the whole trajectory starting from any state.

Another important concept is value function. Formally, value function v under policy π is defined by:

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} d^k R_{t+k+1}]$$

This state-value function tells the value of state s. Similarly, the action-value function that tells the value of taking action a at state s is given by:

$$q_\pi(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} d^k R_{t+k+1} | S_t = s]$$

A fundamental property of value function is the recursive relationship it satisfies.The value under policy π at state s can be expressed by summation of the value of all possible successor states weighted by the transformation probability:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + dv_\pi(s')]$$

It is also called the Bellman equation for v_π. The value function is a powerful tool to measure the performance of policies, and a better policy is defined to be the one that has no less value at any state than the other one. From this well defined order we can find the optimal policy which takes the maximum value at every state. Thus the evaluation of a policy can be naturally transformed to the estimation of its value function v_π, and performance of actions can be directly illustrated as the difference of value in next state and value in present state. In the training process, the goal-seeking agent trades off be- tween exploration (taking actions it has not known the reward) and exploitation (taking actions with the greatest expected reward) to decide the next action, and gradually drives the value function at each state towards the maximum. The value function under optimal policy in the form of Bellman equation can be expressed by:

$$v_{\pi^*}(s) = \max_{a \in A(s)} \sum_{s',r} p(s', r|s, a)[r + dv_{\pi^*}(s')]$$

Equivalently, for the action-value function under optimal policy it is:

$$q_{\pi^*}(s, a) = \sum_{s',r} p(s', r|s, a)[r + dv \max_{a' \in A(s')} q_{\pi^*}(s', a')]$$

In the most of cases, people do not have the complete knowledge of the environment, as the state transformation probability $p(\cdot)$ is always unknown. The different approaches to approximate the solution to optimal Bellman equation forms the basis of a number of RL algorithms.

3.1 Q-Learning Algorithm

Q-learning is a model-free RL algorithm and it can handle problems with stochastic transitions and rewards without requiring adaptations. For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy. "Q" refers to the function that the algorithm computes – the expected rewards for an action taken in a given state.

After Δt steps into the future the agent will decide some next step. The weight for this step is calculated as $\gamma^{\Delta t}$, where γ (the discount factor) is a number between 0 and 1 $(0 \leq \gamma \leq 1)$and has the effect of valuing rewards received earlier higher than those received later (reflecting the value of a "good start"). γ may also be interpreted as the probability to succeed (or survive) at every step Δt.

The algorithm, therefore, has a function that calculates the quality of a state–action combination:

$Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Before learning begins, Q is initialized to a possibly arbitrary fixed value (chosen by the programmer). Then, at each time t the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action), and Q is updated. The core of the algorithm is a Bellman equation as a simple value iteration update, using the weighted average of the old value and the new information:

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \bigg)}^{\text{temporal difference}}$$

where r_t is the reward received when moving from the state s_t to the state s_{t+1}, and α is the learning rate $(0 < \alpha \leq 1)$.

Note that $Q^{new}(s_t, a_t)$is the sum of three factors. The first one is $(1-\alpha)Q(s_t, a_t)$, the current value weighted by the learning rate. Values of the learning rate near to 1 make the changes in Q more rapid. The second one is $\alpha\, r_t$, the reward $r_t = r(s_t, a_t)$ to obtain if action a_t is taken when in state s_t (weighted by learning rate). The third one is $\alpha\gamma \max_a Q(s_{t+1}, a)$, the maximum reward that can be obtained from state s_{t+1}(weighted by learning rate and discount factor)

An episode of the algorithm ends when state s_{t+1} is a final or terminal state. However, Q-learning can also learn in non-episodic tasks (as a result of the property of convergent infinite series). If the discount factor is lower than 1, the action values are finite even if the problem can contain infinite loops. For all final states s_f, $Q(s_f, a)$ is never updated, but is set to the reward value r observed for state s_f. In most cases, $Q(s_f, a)$ can be taken to equal zero.

3.2 Q-Learning Implementation

From the pseudo code Algorithm 1, we can see the only difference is the update rule for value function. In the process of Q-learning, generated actions in a trajectory does not follow the action selected in Q's updating. This implies that the learning data is derived from the policy that is "off" the policy being estimated, and this kind of algorithms represented by Q-learning is termed "off-policy". Q-learning algorithms directly learns the optimal value function thus it usually converges in faster speed, and the result of Q-learning may be affected by the large variance of training data and could be trapped in the local optimal control.

Algorithm 1 Q-Learning Algorithm

```
Require: State space $\mathcal{S}$, action space $\mathcal{A}$,discount rate $d$ and learning rate $\alpha$
  Initialize $Q(\mathcal{S}, \mathcal{A})$ with zeros
  repeat forever (for each episode):
      Initialize $S$
      Generate action $A$ under $\epsilon-$soft policy from present $V(A)$
      repeat forever (for each episode):
          Take action $A$, observe $R, S'$
          $Q(S,A) \leftarrow Q(S,A) + \alpha[R + d \max_a Q(S',a) - Q(S,A)]$
          $S \leftarrow S'$
          Choose $A$ from $S$ using $\epsilon$-soft based on the present $Q$
      until S is terminal
```

4 Design of Experiment

Inspired by the huge progress made from previous work, we decide to implement an application of multi-indices asset allocation based on Deep Q-learning Network. Our overall framework is described as follows:

- **Step1**: Model the asset allocation problem into the reinforcement learning framework. Determine settings of necessary details such as the choice of reward function, time period, etc.
- **Step2**: Retrieve historical daily-basis price of multiple important global indices using package *yfinance*. Clean the data by standard data-processing methods.
- **Step3**: Implement a series of commonly used neutral networks, including CNN, RNN, LSTM, etc. Predict indices' future price at each time spot.
- **Step4**: Train a q-learning RL agent on historical price from 2010-2018 and backtest the trading agent's performance on 2019-2022.
- **Step5**: Analyze NN model's predictions under different network structures. Using metrices such as volatility, Sharpe Ratio (SR), and maximum drawdown to compare the back-testing performance between strategy from mean-variance optimization and deep-Q-network.

4.1 XCS For Rules Generation

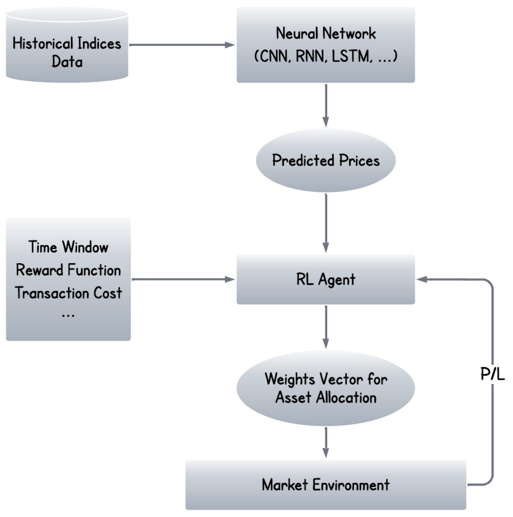

Figure 1: Flow Chart of Trading System

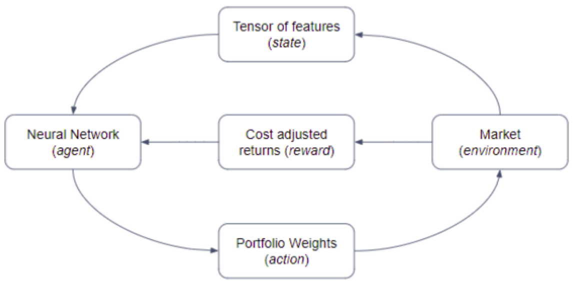
Tensor of features (state)
Neural Network (agent)
Cost adjusted returns (reward)
Market (environment)
Portfolio Weights (action)

Figure 2: MVO in a RL framework

4.2 Back-tested Trading System

The whole train-test trading process is briefly shown by a flowchart in Figure 1 . In order to obtain well balance between portfolio return and risk control, we will calculate its volatility, Sharpe ratios etc. as our risk metrics. We will compare the result with our benchmark, the equal weighted buy-and-hold strategy.

5 Description

A market index is a collection of investments, such as stocks, that are grouped to track the performance of a particular segment of the financial market. Stock market indices around the world are powerful indicators for global and country-specific economies. In the United States the S&P 500, Dow Jones Industrial Average, and NASDAQ Composite are the three most broadly followed indexes by both the media and investors. Additionally, there are approximately 5000 others that make up the U.S. equity market. Indices can be constructed in a wide variety of ways but they are commonly identified generally by capitalization and sector segregation (Investopedia). Investment managers use indices as benchmarks for performance reporting. Meanwhile, investors of all types use indexes as performance proxies and allocation guides. Indexes also form the basis for passive index investing often done primarily through exchange-traded funds that track indices specifically.

The S&P 500, which is short for Standard Poor's 500, was introduced in 1957 as a stock market index to track the value of 500 corporations that have their stocks listed on the New York Stock Exchange (NYSE) and the NASDAQ. Its index symbol is ˆGSPS. The collection of stocks that make up the S&P 500 is designed to represent approximately 80% of the total value of the U.S. stock market. The S&P 500 Index is a market-weighted index (also referred to as capitalization-weighted). Therefore, every stock in the index is represented in proportion to its total market capitalization. The value of the S&P 500 and various stocks within the index is closely watched by market participants since their performance represents a gauge of the health of the U.S. economy (Investopedia). That is why we choose S&P 500 as the first index to test our strategy.

Dow Jones Industrial Average (DJIA) is one of the oldest, most well-known, and most frequently used indices in the world. It includes the stocks of 30 of the largest and most influential companies in the United States. Its index symbol is ˆDJI. The DJIA is a price-weighted index and represents about a quarter of the value of the entire U.S. stock market. A change in the Dow represents changes in investors' expectations of the earnings and risks of the large companies included in the index. In general, the Dow is known for its listing of the U.S. markets best blue-chip companies with regularly consistent dividends (Investopedia).

The NASDAQ Composite Index is known for being a heavily tech weighted index, which includes software, biotech, semiconductors, and more sub-sectors across the tech market. The NASDAQ Composite Index is a market-capitalization-weighted index of all the stocks traded on the NASDAQ stock exchange and it also includes some non-US based companies. Its index symbol is ˆIXIC. Unlike the S&P 500 and the Dow Jones Industrial Average, NASDAQ also includes many speculative companies with small market capitalization. As a result, its movement generally indicates the performance of the technology industry as well as investor's attitudes toward more speculative stocks (Investopedia).

6 Timeline

Now we completed literature review and identified the most promising methods and references. We also did some research on corresponding packages which implement these methods and obtained relevant data sets. Next, we plan to use 1-2 weeks to design codes and identify python packages, and 3-4 weeks to implement coding framework and conduct testing and debugging. After completing the base code part, we are going to spend 1-2 weeks to compare methods and analyze results. In the end, we plan to finish our project report and presentation in one week.

References

[1] S. R. Dasa and S. Varmaa, “Dynamic goals-based wealth management using reinforcement learning,” *Journal Of Investment Management*, vol. 18, no. 2, pp. 1–20, 2020.

[2] M. Dixon and I. Halperin, “G-learner and girl: Goal based wealth management with reinforcement learning,” *arXiv preprint arXiv:2002.10990*, 2020.

[3] S. R. Das, D. Ostrov, A. Radhakrishnan, and D. Srivastav, “Dynamic portfolio allocation in goals-based wealth management,” *Computational Management Science*, vol. 17, no. 4, pp. 613–640, 2020.

[4] ——, “Dynamic optimization for multi-goals wealth management,” *Journal of Banking & Finance*, vol. 140, p. 106192, 2022.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[6] M. Katongo and R. Bhattacharyya, “The use of deep reinforcement learning in tactical asset allocation,” *Available at SSRN 3812609*, 2021.

[7] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, “Deep reinforcement learning for automated stock trading: An ensemble strategy,” in *Proceedings of the First ACM International Conference on AI in Finance*, 2020, pp. 1–8.

[8] E. Benhamou, D. Saltiel, J. J. Ohana, J. Atif, and R. Laraki, “Deep reinforcement learning (drl) for portfolio allocation,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 527–531.