Reinforcement Learning For Tactic Portfolio Allocation

Jay Xiao
University College London

April 3, 2023

1 Introduction

Portfolio management is one of the most important areas in finance, from large entities to every individual, a good portfolio helps the growth of wealth. It includes selecting a set of assets among equity, bonds and derivatives etc. by continuously redistributing a certain amount of wealth to achieve the strategic goals under minimal risk. One of the mainstream theories that solve this problem was summarized by Markowitz (1952), who concluded using the mean-variance optimization model for single-period portfolio selection can significantly reduce risk, this theory is regarded by the financial community as the starting point of modern portfolio theory. Another classical theory is capital growth theory developed by Hakansson and Ziemba (2011) which focuses on dynamic portfolio selection, the algorithms developed upon this theory can be classified into four categories, "Follow-the-Winner", "Follow-the-Loser", "Pattern-Matching", and "Meta-Learning" Li and Hoi (2012).

The question is whether a reinforcement learning agent can do better than those financial experts. Because human is basically irrational, Shiller (2015) argues that asset prices can become overvalued due to irrational exuberance, which can lead to bubbles and crashes, whereas reinforcement learning agent makes decisions based on the policy trained by the historical data, so they do not have emotional impulses on decision making. Good portfolio management can significantly reduce risk during a financial crisis. Machine learning has recently been widely applied to this problem, some of them use historical data to predict future returns and made adjustments based on the future prediction Sen et al. (2021), others like Gao et al. (2020) treated the portfolio management problems as finite Markov decision process, the author create discrete action and state spaces and use the deep-Q network reinforcement learning algorithm. Until recently, the continuous actor-critic deterministic policy gradient method is introduced Lillicrap et al. (2015), which allows the action space to be continuous by using an approximation method to the policy function.

In the paper I will apply continuous-action reinforcement learning algorithms to address the stocks portfolio optimization problem, the Markowitz mean-variance model will be our benchmark to compare the feasibility of the RL model. The FinRL library was used, it provides a good reinforcement learning environment. The contribution of this report includes: comparing the influence of different rewards functions on the reinforcement learning agent, successfully implementing commission fees and considering the risk-free asset.

2 Problem statement and data structure

2.1 Assumption

1. We skipped the stock selection, 5 assets will be randomly selected from Dow Jones Index list to test the performance of the Deep Reinforcement learning model.
2. Due to our active portfolio adjustments, the transaction cost serves as a constraint that the agent should consider in trading. The transaction fee Tx is 0.01%.
3. The risk-free rate here is the effective Federal Fund rate which can perform as the safe haven when the economy goes down.
4. Zero market impact: we assume that the agent's actions do not affect the market price
5. Zero slippage: there are enough volumes of stocks to trade.

2.2 problem statement and data structure

The stock data is downloaded from Yahoo Finance, and the effective federal fund rate is used. The total number of assets here is six. Assume there are n risky assets with a risk-free asset in the market, the return of each asset is r_t^i, and the allocation weight of investor in each asset is w_t^i respectively at time t. Given that the investor has initial capital $V_0 = \$1000000$, with initial weights $\mathbf{w}_0 = (0, 0, 0, 0, 1)$ all in cash, the goal is to actively reallocate the positions to optimise the investor's final capital.

$$\underset{[\mathbf{w}_1,\ldots,\mathbf{w}_T]}{max} \sum_{j=0}^{T} \sum_{i=0}^{n} r_t^i * w_t^i$$

No margin and short action are considered in this report, i.e. The weight $w_t^i >= 0 \; \forall i, t$. And the total dollar value holding of assets $\sum_{i=0}^{n} r_t^i * w_t^i$ is equal to V_t.
This problem can be addressed as a Markov decision process.
Definition 2.2.1 A Markov Decision Process is a tuple $< \mathbb{S}, \mathbb{A}, \mathbf{P}, \mathbf{R}, \gamma >$ where,

- $\mathbb{S}$ is a set of states,
- $\mathbb{A}$ is a set of actions,
- $\mathbf{P}$ is a state transition probability matrix, $\mathrm{P}_{ss\prime}^{a} = \mathbb{P}[S_{t+1} = s\prime | S_t = s, A_t = a]$
- $\mathbf{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is a reward function,
- $\gamma \in [0, 1]$ is a discount factor.

2.2.1 Action space

In portfolio allocation, an action $\mathbf{a}_t$ is the vector of the percentage of wealth distribution in each asset at time t, which forms a continuous action space.

$$\mathbf{a}_t := \mathbf{w}_t = (w_t^1, w_t^2, w_t^3, w_t^4, w_t^5, w_t^6)$$

In the algorithm we set each weight $\mathbf{w}^i \in [0,1]$, then use the softmax function to normalize the weights to make sure the action add up to 1:

$$\text{softmax}(\mathbf{a}_t)_i = \frac{\exp(a_t^i)}{\sum_{j=1}^{6} \exp(a_t^i)}, \quad \forall i \in [1,6] \tag{1}$$

2.2.2 STATE SPACE

The state space that the agent observes is today's price, which is given by a price matrix $\mathbf{p}_t^{close}$, here we simplify the allocation procedure see below explanations about the allocation process. We also added a technical indicator matrix $\mathbf{ti}_t$[1]:

$$\mathbf{ti}_t = [\mathbf{macd}_t, \mathbf{bollub}_t, \mathbf{bolllb}_t, \mathbf{rsi30}_t, \mathbf{cci30}_t, \mathbf{dx30}_t, \mathbf{ma30}_t, \mathbf{ma60}_t, \mathbf{cov}_t]$$

where the **cov** is given by:

$$\mathbf{cov} = \Big[[cov_{t,(1,1)} \cdots cov_{t,(1,6)}], \cdots, [cov_{t,(6,1)} \cdots cov_{t,(6,6)}]\Big]$$

Covariance is a good indicator of the correlation between the return of each asset, we will look back at one year's returns of assets and calculate their covariance,

$$cov_{T,(i,j)} = \frac{\sum_{t=T-252}^{T}(r_{t,i} - \hat{r}_i)(r_{t,j} - \hat{r}_j)}{251}$$

So at time t that the agent observe includes $\hat{\mathbf{s}}_t = (\mathbf{P}_t^{close}, \mathbf{s}_t, \mathbf{a}_{t-1})$, where s_t is of the dimension 14×6. (note that only $\mathbf{ti}_t$ will be feed into the neuron network, see section 3.1.2)

2.2.3 REWARD FUNCTION

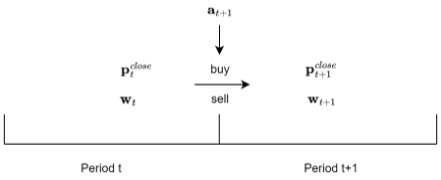

Figure 1: Allocation process

From figure 1, if the portfolio value at period t is V_t, after receiving agent action, the transaction cost on each asset is calculated by $tx\% = |\mathbf{w_t} - \mathbf{a}_{t+1}| \times 0.01\%$, it can be regarded as the shrinkage of the portfolio value. The portfolio return $return_{t+1}$ by doing action $\mathbf{a}_{t+1}$ will then be $(\frac{\mathbf{p}_{t+1}^{close}}{\mathbf{p}_t^{close}} - 1) - tx\%$, thus portfolio value at $t+1$ will be

$$V_{t+1} = (1 + return_{t+1}) \times V_t$$

[1] check https://pypi.org/project/stockstats/ for different names

The immediate reward function will be built in two different structures to test the convergence of the models:

$$r_t = return_t \qquad \text{or} \tag{2}$$

$$r_t = V_t \tag{3}$$

The following are some basic definitions in reinforcement learning,

Definition 2.2.2 A policy π of a reinforcement learning agent is the probability distribution over actions given state s_t:

$$\pi(a_t|s_t) = \mathbb{P}[A_t = a_t|S_t = s_t] \tag{4}$$

Definition 2.2.3 The immediate reward at t in a state s, r_t is the expected reward at next time i.e. $\mathbf{E}[r_{t+1}|S_t = s]$, the return G_t at time t then is given by the total discounted reward from t till the end of the episode, $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$

Definition 2.2.4 The state-value function $V_\pi(s)$ of an MDP is

$$V_\pi(s) = \mathbb{E}[G_t|S_t = s] \tag{5}$$

The action-value function $Q_\pi(s, a)$ is the expected return starting from state s

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{6}$$

by Bellman equation,

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(s_{t+1})] \tag{7}$$

3 Methodology

3.1 Advantage Actor-Critic reinforcement learning model

3.1.1 A2C algorithm

In this section, we describe the implementation of the Actor-Critic method with the Advantage Actor-Critic (A2C) algorithm for asset allocations. A reinforcement learning agent is an algorithmic agent that learns to interact with an environment following a certain policy π and forming a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, s_2...)$, the goal is to optimize the cumulative rewards. A2C is a model-free, on-policy reinforcement learning algorithm that uses both an actor network and a critic network to learn a policy and estimate the value function. The actor network is responsible for selecting actions based on the current policy, while the critic network estimates the value function of the state.

The major difference between AC and A2C is the A2C algorithm introduces the advantage function to solve the high variability problem of value-based methods, it is calculated as the difference between the state-action function $Q_\pi(s_t, a_t)$ of taking a particular action a_t at state s_t, and the state-value function $V_\pi(s_t)$. The advantage function is defined as:

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{8}$$

$$= \underbrace{r_{t+1} + V_\pi(s_{t+1}) - V_\pi(s_t)}_{\text{TD error}} \qquad \text{(by Bellman equation)} \tag{9}$$

The TD error is the unbiased estimation of the advantage function, the advantage function measures how much better or worse a particular action is compared to the average action in that state, and the A2C algorithm updates the actor and critic networks simultaneously using the advantage function. However, due to the state space and action space being large, it is almost impossible to obtain various value functions $V_\pi(s)$ and $\pi(a|s)$ precisely. It is necessary to find an approximate function. Specifically, neural networks can be used to approximate, i.e.

$$\pi(a_t|s_t) \approx \pi_\theta(a_t|s_t) \tag{10}$$

$$V_\pi(s_t) \approx V_{\pi_\theta}(s_t; \phi) \tag{11}$$

The actor network is updated using policy gradient descent to maximize the expected cumulative reward, which is defined as the sum of rewards discounted by a factor γ:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{12}$$

where π_θ is the policy defined by the actor network with parameters θ, and r_t is the reward received at time step t. The policy gradient is computed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)\right] \tag{13}$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{14}$$

where $\log \pi_\theta(a_t|s_t)$ is the logarithm of the probability of selecting action a_t at state s_t under the policy π_θ.

The critic network is updated using mean-squared error to minimize the difference between the estimated value function $V(s; \phi)$ and the actual rewards received:

$$L(\phi) = \mathbb{E}_{\pi_\theta}\left[\left(V(s_t; \phi) - (r_t + \gamma V(s_{t+1}; \phi))\right)^2\right] \tag{15}$$

$$\phi \leftarrow \phi + \beta \nabla_\phi L(\phi) \tag{16}$$

The A2C algorithm is trained using batches of experiences collected by the agent while interacting with the environment. Each batch is used to update the actor and critic networks, and the process is repeated until convergence. We used a discount factor of 0.99 and an entropy coefficient of 0.05 in our experiments to encourage exploration and prevent premature convergence.

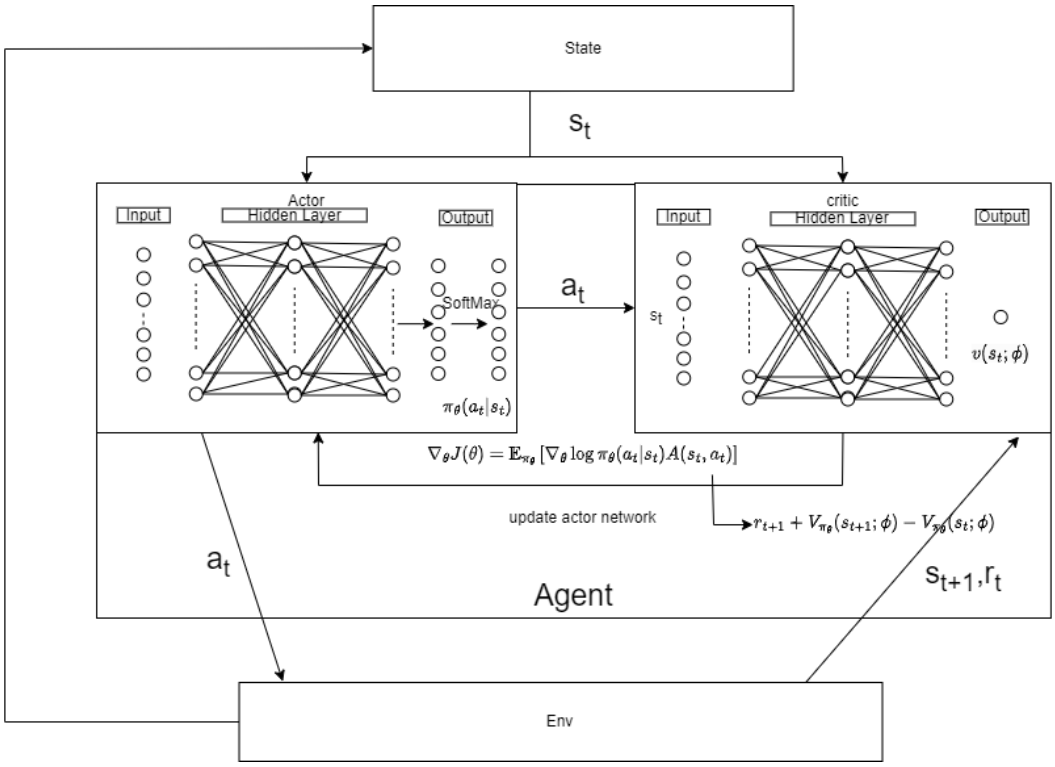

Figure 2: A full update cycle of single step A2C algorithm at state s_t

3.1.2 Network structure

We know from experience that a well-performing deep neural networking can greatly improve the performance of DRL applications Munikoti et al. (2022), therefore, the neural network design is an important part of DRL. Because the input data s_t is non-time dependent data, it is enough to just multi-layer perceptron, complex network I tried overfitted easily in the testing set which give us bad results. The neural network in both actor and critic networks are multi-layer perceptron, the input is the technical indicators with three hidden layers $64 \times 64 \times 16$. Two different immediate reward functions will be implemented in the algorithm to see the results

3.2 Baseline Model

The benchmarks we are going to use are as follows:

3.2.1 The uniform buy and hold

we equally divide the initial capital and invest it into all assets for the whole trading period.

3.2.2 Markowitz mean-variance optimization model

The classical methodology of portfolio management is the Markowitz mean-variance optimization model. The Markowitz model assumes that every investor is risk averse, that is, hoping to minimize

the risk of the portfolio under the condition of obtaining a certain return or obtaining the maximum benefit under certain conditions of risk. And the Markowitz portfolio theory shows that under certain conditions the investor's investment decision is only determined by the expected value and variance of the return of an asset. The risk of the asset can be regarded as the variance of the asset return, the risk of a portfolio is not only related to the risk of an individual asset but also related to the covariance between each asset.

Following from the problem statement, in the setting of the Markowitz model we take the whole trading period as a single time period, considering the maximum expected risk an investor can take is fixed σ_{max}, the mean-variance problem is given by:

$$\max_{\vec{\mathbf{w}}} \sum_{j=0}^{T} \sum_{i=0}^{n} r_i * w_i$$

$$s.t. \begin{cases} \sigma_{max} \geq \sum_{i=0}^{n} \sum_{j=0}^{n} \quad w_i w_j cov(r_i, r_j) = \sigma_p \\ \sum_{i=0}^{n} w_i = 1 \\ \sum_{i=0}^{n} w_i \mu_i = \mu_p \end{cases}$$

where μ_i is the expected return of asset i and μ_p is the expected return of the portfolio.

In practice, we will use the historical return to calculate the mean and variance of the underlying asset. And use them as the expected return and the risk of the underlying asset.

4 Result

The data will be split into a 70% training set and a 30% test set.

Figure 3: Training and Test periods

4.1 Training results

The results in figure 4 and 5 clearly show that using daily portfolio return as the reward function has better convergence property, whereas using portfolio value function did not show good convergence of value function.

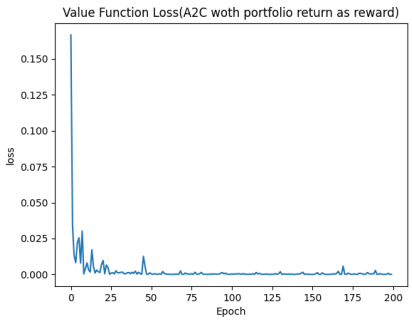

Figure 4: A2C model with rewards equals portfolio return

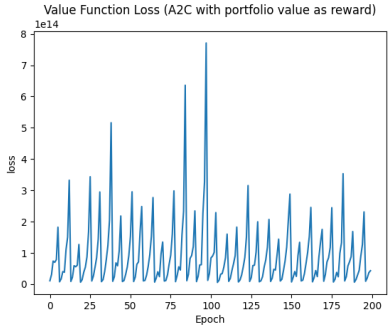

Figure 5: A2C model with rewards equals portfolio value

4.2 Performance metrics

The financial measurement we use includes the Sharpe Ratio, the Sortino Ratio, Maximum Drawdown and the Calmar Ratio, the formulas are:

$$\textbf{Annual Sharpe Ratio} = \frac{\mathbb{E}(r_t)}{\sigma_t} \times \sqrt{252} \tag{17}$$

$$\textbf{Annual Sortino Ratio} = \frac{\mathbb{E}(r_t)}{\sqrt{\frac{1}{N-1}\sum_{t=1}^{N} r_t^2}} \times \sqrt{252} \tag{18}$$

$$\textbf{Maximum Drawdown} = \frac{\textbf{MinThroughValue - MaxPeakValue}}{\textbf{MaxPeakValue}} \tag{19}$$

$$\textbf{Annual Calmar Ratio} = \frac{\mathbb{E}(r_t)}{\textbf{Maximum Drawdown}} \times 252 \tag{20}$$

where r_t is the daily return of the portfolio, σ_t is the standard deviation of the daily percentage portfolio return, we assume there are 252 trading days in a year.

In table 1, the Markowitz mean-variance model is to find the minimum variance, calculated based on the previous day's return to adjust asset weights. We can see that it has the lowest annual volatility, and the A2C model with the portfolio return as the immediate reward function outperforms all other models. However, the performance of another A2C model is similar to the Uniform buy-and-hold model. Both A2C models did not reduce the risk in trading. But with the addition of the cash in the portfolio, the annual volatilises are smaller than the market average volatility.

Table 1: The performance of each model

Model	Test set					
	Annual return	Annual Volatility	Sharpe Ratio	Sortino Ratio	Maximum Drawdown (%)	Calmar ratio
DJI index(market average)	0.111	0.186	0.659	0.912	**-37.1**	0.299
Uniform buy and hold	0.213	0.168	1.234	1.774	-28	0.763
Markowitz mean-vairance	0.180	**0.158**	1.129	1.618	-27.4	0.657
A2C(portfolio value)	0.217	0.170	1.239	1.783	-27.9	0.777
A2C(portfolio return)	**0.233**	0.176	**1.279**	**1.839**	-28.7	**0.809**

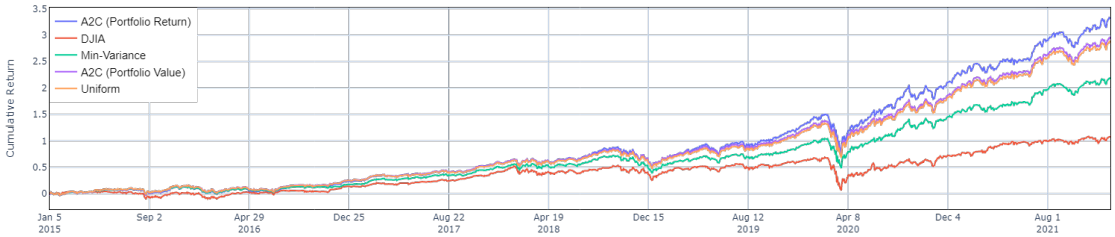

Figure 6: Trading cumulative returns

4.3 Stocks Backtesting

In the case of no leverage, the real portfolio asset performance should be between the stock with the best price performance and the stock with the worst price. We can see in figure 7 the performance of our best A2C model is quite satisfactory, it lies in between Apple and other stocks. Figure 8 shows that the weights of almost all assets are between 0.12 and 0.27, apple takes most of the capital, and there is not a lot of money invested in a single asset, which fully disperses the risk. The model has the ability to diversify the risk and identify the best stock.

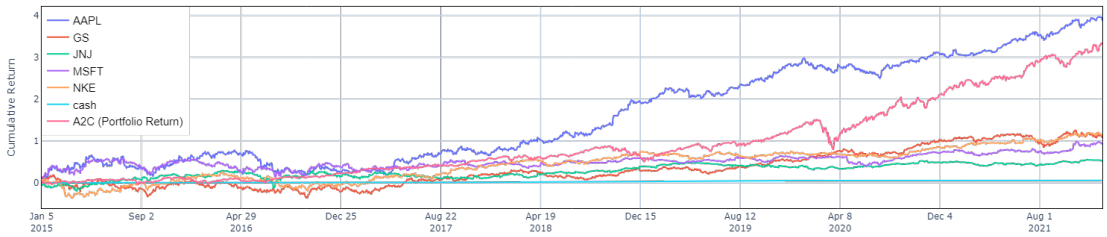

Figure 7: Stocks backtesting

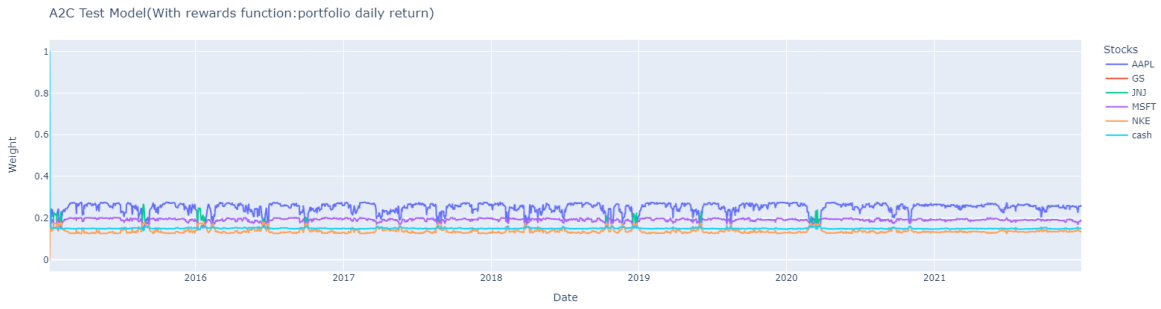

Figure 8: Weights distribution

5 Discussion and Conclusion

We successfully build a reinforcement learning model using the advantage actor-critic algorithm, and we tried two different types of reward functions under the condition that we can only long stocks, the model with reward function daily return not only outperforms all baseline models in the problem of portfolio optimization but also have better convergence property. The risk-free asset we add also significantly reduces the volatility of the portfolio. Deep reinforcement learning is a dynamic optimization model based on data-driven thinking, which can effectively avoid the subjective arbitrariness of model users. I think that the dynamic optimization idea contained in the method is very consistent with the actual transaction process, so it is very suitable for optimizing the assets in the investment portfolio configuration.
Further improvement should consider:

1. The state s_t we consider here is only a 2D array, we may consider 3D time series price and technical indicators input because each days price may have time correlation

2. This leads us to apply advanced deep neural networks like VGG Simonyan and Zisserman (2014).

3. Following improvement 1, we can also construct a reward function like the rolling Sharpe ratio, using rolling 5 days daily returns of the portfolio. Because many financial econometric analysis methods are using the Sharpe ratio as the main performance measurement.

The application of deep reinforcement learning in the financial field covers the comprehensive aspects of statistics, mathematics, finance, dynamic optimization and computer science. The progress in each direction will promote effective application in the financial field. There is still a lot of room for development in optimal asset allocation.

References

Gao, Ziming, Yuan Gao, Yi Hu, Zhengyong Jiang, Jionglong Su. 2020. Application of deep q-network in portfolio management. *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 268–275.

Hakansson, Nils H., William T Ziemba. 2011. Capital Growth Theory. Leonard C MacLean, Edward O Thorp, William T Ziemba, eds., *THE KELLY CAPITAL GROWTH INVESTMENT CRITERION THEORY and PRACTICE*, chap. 41. World Scientific Book Chapters, World Scientific Publishing Co. Pte. Ltd., 577–598. URL `https://ideas.repec.org/h/wsi/wschap/9789814293501_0041.html`.

Li, Bin, Steven C. H. Hoi. 2012. Online portfolio selection: A survey. Papers 1212.2129, arXiv.org.

Lillicrap, Timothy P, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Markowitz, Harry. 1952. Portfolio Selection. *Journal of Finance* **7**(1) 77–91.

Munikoti, Sai, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar, Balasubramaniam Natarajan. 2022. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *arXiv preprint arXiv:2206.07922* .

Sen, Jaydip, Abhishek Dutta, Sidra Mehtab. 2021. Stock portfolio optimization using a deep learning lstm model. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*. IEEE, 263–271.

Shiller, Robert J. 2015. Irrational exuberance. *Irrational exuberance*. Princeton university press.

Simonyan, Karen, Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .