The Application of Reinforcement Learning to the Portfolio Management Problem

Final Project Paper
Introduction to Deep Reinforcement Learning Seminar (WIHN0033)

Reviewer:

Maotong Sun (M.Sc.), Doctoral Candidate
Chair of Business Analytics
TUM School of Management
Technical University of Munich

composed by:

Manuel Schreiber (03700297)
Marc Henneberger (03744170)
Emmanuel Boateng (03741449)

Study programs:

M.Sc. in Management and Technology
(Technology Major in Computer Science)

M.Sc. in Management

submitted on:

March 02, 2023

The Application of Reinforcement Learning to the Portfolio Management Problem

Manuel A. Schreiber
Technical University of Munich
manuelalex.schreiber@gmail.com

Marc Henneberger
Technical University of Munich
mail@marc-henneberger.de

Abstract

The portfolio management problem which deals with reallocating capital to a collection of financial assets in order to maximize the expected return on investment is a common financial application case for reinforcement learning. Using daily price data for six exchange traded funds (ETFs) for a five year period that includes the Corona pandemic, we train a reinforcement learning (RL) agent through an actor critic algorithm to maximize the reward in form of the portfolio profit by dynamically changing the weight allocation among the portfolio positions (actions) in each time step. The states consist of a price history window for the financial assets in the portfolio and the benchmark is a static equal weight heuristic. We find that the RL agent outperforms the heuristic in terms of a lower portfolio standard deviation, a higher Sharpe ratio as well as higher cumulative returns.

1 Introduction

The portfolio management problem can be thought of as the process of reallocating capital to a collection of financial assets, such as exchange traded funds (ETFs), stocks or cryptocurrencies, in order to enhance the expected return on investment over a certain investment horizon [1]. The prices of such financial assets can be characterized by the stochastic, non-linear and non-stationary nature of their movements. Key determinants of the return on such a portfolio are the correlations among the individual positions as well as their respective voltatility. Snow [2] separates the field of Financial Machine Learning Research into two main streams: the first one is concerned with the creation of trading strategies based on e.g. the prediction of future values of financial assets or the prediction of financial events such as corporate defaults. The second stream deals with the use of machine learning techniques to solve optimization and simulation problems in Finance, one of which is the portfolio management problem outlined above. In this particular problem, the allocation of capital to the financial assets that form a portfolio can be formalized as a weight optimization problem. Reinforcement Learning allows to conduct online portfolio management by changing the weights associated with the assets in a portfolio and receiving as a reward the new portfolio value in the respective time step. According to Snow [3], RL permits to take the long-term consequences of actions into account that more standard models, which recombine single period predictions, cannot capture. To sum up, the optimal policy we are seeking to extract from the actor-critic algorithm employed in this paper consists of a state to action mapping outputting a sequence of portfolio weight vectors for each state that maximizes the cumulative return of the portfolio over the 5 years worth of price data for each trading day considered.

In this paper, the research objective is to train a reinforcement learning (RL) agent with an actor-critic algorithm to maximize the cumulative portfolio profit over time (objective) through determining a weight allocation to the 6 positions (action) which make up the portfolio for every time step. As a reward, the portfolio profit for the respective time step is experienced by the agent. The states in this application are the price histories of the assets for a fixed time window. For details, see Section 3.

The data used in this paper are the end of day closing ETF share prices for 5 world regions of the stock market (North America, Developed Europe, Japan, Developed Asia and Emerging Markets) and for a money market fund for the period from 2018 to 2022.

2 Related Work

This section highlights related work regarding the use of Reinforcement Learning techniques within the context of the portfolio management problem in Finance.

Snow [3] argues that allocation decisions in portfolio (PF) management are complicated by the noisy, nonstationary and partially observed nature of financial data. Thus, investor biases can occur. Huang and Tanaka [4] investigate whether RL agents exhibit the same degree of bias as humans in the regard to the disposition effect and to the narrow framing effect. The former bias, the disposition effect, refers to the situation in which profits are realized too soon while losses are kept too long and the latter, the narrow framing effect, is a term for neglecting the big picture of the portfolio and making isolated suboptimal decisions [4]. They find that agents exhibit fewer biases in addition to delivering outstanding capital returns. Guan and Liu [5] investigate explainability in deep reinforcement learning but also employ a proximal policy optimization (PPO) and a synchronous advantage actor critic (A2C) algorithm for the portfolio management task using the stocks from the Dow Jones 30 index for the time period from 2020 to 2021. They find that the PPO algorithm produces the highest cumulative portfolio returns and the highest sharpe ratio even though the performance of the A2C model is close.

3 Problem Definition

In this section, the portfolio management problem is formalized with respect to the portfolio weight vector, which represents the action of the agent, the portfolio profit, which is the reward received by the agent as well as the states of the environment, which is a collection of vectors with prices over a fixed observation window.

Let N be the number of assets in the portfolio. According to Snow [3] the **closing price vector** is given by

$$\boldsymbol{\gamma}_t = (\frac{S_{m,t,close}}{S_{m,t-1,close}}, \frac{S_{1,t,close}}{S_{1,t-1,close}}, \ldots, \frac{S_{N,t,close}}{S_{N,t-1,close}}) \in \mathbb{R}^{N+1}. \tag{1}$$

$S_{1,t,close}$ is the closing price of asset 1 at time t. $S_{m,t,close}$ is the money market fund price. The **portfolio weight vector** can be defined as follows:

$$\boldsymbol{w}_t = (w_{m,t}, w_{1,t}, \ldots, w_{N,t}) \in \mathbb{R}^{N+1}, \tag{2}$$

where $w_{m,t}$ is the money market position. $\boldsymbol{w}_t$ is initialized with $\boldsymbol{w}_0 = (1, 0, \ldots, 0)$ as all starting capital is initially invested in S_m, the money market fund. The **portfolio profit after T time periods** is then

$$P_T = \prod_{t=1}^{T} \boldsymbol{\gamma}_t^\intercal \boldsymbol{w}_{t-1}. \tag{3}$$

Following the notation of Snow [3], the **state** o_t in this RL application problem can be formalized using the **observation window price history** $\boldsymbol{S}_{i,t}$ $\forall$ $N+1$ assets as follows:

$$o_t = (\boldsymbol{S}_{m,t,close}, \boldsymbol{S}_{1,t,close}, \ldots, \boldsymbol{S}_{N,t,close}) \in \mathbb{R}^{(N+1)\times W}, \tag{4}$$

where $\boldsymbol{S}_{i,t,close} = (S_{i,t-W}, S_{i,t-W+1}, \ldots, S_{i,t}) \in \mathbb{R}^W$ and W is the observation window size.

4 Environment and Data

Having clarified the problem definition in the preceding section, this section describes the environment used for building and training the agent. First, the available data is discussed which serves as a foundation for the research problem. After that, it is described how actions are performed and how the environment reacts to them.

The data representing the situation of the financial market the agent is trained in consists of 5 Vanguard equity ETFs which represent the worldwide stock market and a money market fund to represent the money market. The global stock market is represented by the following ETFs which are chosen to reflect the following geographical areas of the world and their share of the overall equity markets:

- North America
- Developed Europe
- Japan
- Developed Asia ex Japan
- Emerging Markets.

The share price data is retrieved for the time period between 01/2018 and 12/2022. For each day the closing prices of the ETFs are considered. Figure 2 shows a time plot of the series of ETF share prices over the time period considered for each position in the portfolio.

At the beginning of each episode, the environment is reset to its basic state. During an episode, data is stored in local arrays which are initialised within reset. In the array *asset_allocation*, the weight allocation for each asset is stored for each time step. In *prices_df*, the closing prices of all assets are stored for the dates within the episode. *portfolio_value* is an array in which the resulting portfolio value after each time step is stored. Similarly, in the list *profitslist*, the profits are stored for each day.

In each timestep, an interaction between the agent and the environment occurs. The procedure is defined in the method *_step* within the environment. First, it is assessed whether the episode has already ended. If not, an action is taken. During an action, the weights are normalised so that their sum is equal to one. This is to make sure that 100% of the agent's money is being distributed among the available funds. After that, the action is stored within the array *asset_allocation*. Using the closing prices of the current and next day, the closing price vector is calculated. The closing price vector is then used to calculate the profit for the current timestep by taking the dot product of the closing price vector and the current asset allocation. The portfolio value of the next day is then calculated by multiplying the profit with the portfolio value of the present day. After taking the action, the next state is retrieved by shifting the observation window one step further down. An overview of the interaction between the RL agent and the environment is shown in Figure 1.

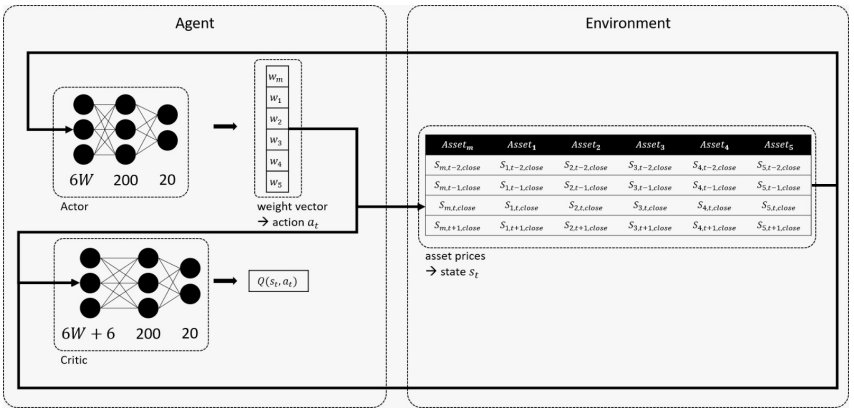

Figure 1: **Agent environment interaction in the context of the portfolio management task.** The RL agent can be broken down into an actor which learns a parametrized policy that outputs a weight vector for each input state and the critic takes a state-action pair to approximate a state-action value function.

5 RL framework and algorithms

This section provides a brief characterization of actor-critic algorithms, pseudocode of the basic actor critic algorithm as well as the implementation details of the actor and the critic network architectures for this particular project.

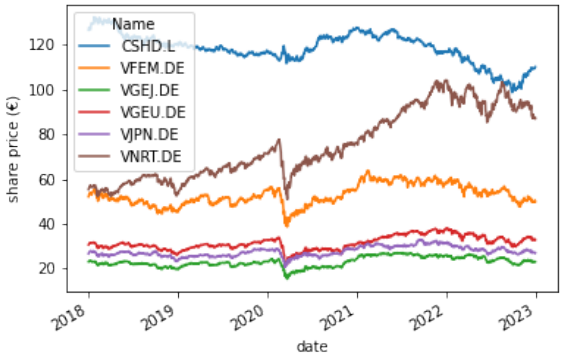

Figure 2: **Time plots of the relative closing prices for each portfolio position.** The color legend contains the acronyms of the 5 Vanguard stock ETFs which track FTSE Russell indices that correspond to the following regions: VFEM - Emerging Markets, VGEJ - Developed Asia Pacific Ex-Japan, VGEU - Developed Europe, VJPN - Japan, VNRT - North America. CSHD stands for the Lyxor (Euro) Overnight return money market ETF. The y-axis contains the price and the x-axis contains the price dates spanning a period of 5 years from 01/01/2018 to 12/31/2023.

Actor-critic algorithms count towards the policy-based category of RL algorithms and they are also model-free, which means that the explicit transition function $T(s, a, s')$ and the reward function $R(s, a, s')$ are assumed to be unknown, where s, a, s' denote the current state, the current action and the future state, respectively. One advantage of such policy-based RL algorithms is that they can handle continuous action spaces and that the exploration is carried out on-policy. One disadvantage is the low-sample efficiency, which refers to the fact that experienced transitions are only used once. Optimization of such actor-critic algorithms is commonly done through the gradient ascent algorithm, which is used to find a maximum by iteratively taking steps into the direction of the gradient towards the steepest ascent. The estimated policy gradient by the actor-critic algorithm is

$$\nabla_\theta J(\theta) \approx \mathbb{E}_\pi[\nabla_\theta log \underbrace{\pi(s, a, \theta)}_{\text{actor (states, opt. action)}} * \underbrace{Q(s, a, w)}_{\text{critic (action, Q(a,s),V(s) or A(a,s))}}], \tag{5}$$

where $\pi(s, a, \theta)$ is the policy function, $Q(a, s)$ denotes the estimated Q-value, θ denotes the policy weights of the parametrized policy that is estimated by the actor network and w represents the weights of the Q-value approximated by the critic network. Note that the actor takes the current state as an input and outputs the optimal action for that time step and the critic evaluates actions and estimates either the Q-value, the V-value or the Advantage value. In the version used in this paper, the Q(a,s) value is estimated.

Both the actor network's and the critic network's architecture are structured as a 3 layer feedforward neural network with 200 output neurons in the input layer and 20 in the hidden layer. A leaky rectified linear unit activation function is used in the input and in the hidden layer in order to permit a small slope for negative input values to avoid the vanishing gradient problem. The trained heads of the two networks differ. While the actor network outputs 6 values within the interval $[0, 1]$, which represent the positional weights, the critic network outputs a scalar value for the respective input state, the array of current share prices, and for the provided action, which is the latest weight vector provided by the actor. Moreover, the actor network uses a sigmoid activation function on each of the 6 outputs separately in order to take the multi-output or multi-label classification problem into account, which means that multiple non-exclusive positional weight labels can be assigned to each input state of the actor network.

Algorithm 1 shows the pseudocode of the Q-value actor critic algorithm that is employed in this paper. Line 7 shows the update of the policy weights through the actor network and line 8 shows the update of the weights of the parametrized Q-function through the critic network. Note that both

weight parameter vectors are updated once after each episode in our implementation in contrast to the pseudocode.

Algorithm 1 Actor-critic algorithm

Require: policy and Q-function weights θ, w
for ($\forall$ episode) **do**:
  initialize s
  sample action from policy $\pi(s, \cdot, \theta)$
  repeat(for each step of the current episode) :
    take action a, observe r, s'
    sample a' acc. to policy $\pi(s', \cdot, \theta)$
    $\theta \leftarrow \theta + \alpha_\theta * \nabla_\theta log\pi(s, a, \theta) * Q(s, a, w)$ ▷ update θ
    $w \leftarrow w + \alpha_w * (r + \gamma Q(s', a', w) - Q(s, a, w)) * \nabla_w Q(s, a, w)$ ▷ update w
    a $\leftarrow$ a', s $\leftarrow$ s'
  until s is terminal
end for

6 Experiment Results and Discussion

In this section, several portfolio evaluation metrics as well as the evolution of the portfolio value over time are compared for the dynamic weight allocation of the RL agent and the static equal weight allocation across the six financial assets in the portfolio, that is used as a benchmark heuristic.

Figure 3 shows the development of the portfolio value over time during a complete episode, i.e. 1,245 trading days over a span of 5 years, both for the agent and an heuristic. It is clearly visible that the trained agent (green) is able to outperform the heuristic (blue). The agent's portfolio value is equal to or higher than the benchmark's during most timesteps over the episode. The impact of the corona crisis at around 500 trading days is clearly visible. Both, the agent and the heuristic are greatly affected by plummeting asset prices. However, the agent is able to recover more quickly from this bear market phase.

Table 1 shows performance metrics to quantitatively compare the portfolio returns produced by the agent and the heuristic. The sharpe ratio $\frac{\mathbb{E}[R_p]-\mathbb{E}[R_f]}{\sigma_{R_p}}$ gives an indication how much excess return over the return of a risk-free rate, which is the return of the money market fund in this case, is expected from the portfolio in relation to the volatility of the portfolio. With a sharpe ratio of 2.72, the agent outperforms the benchmark, meaning that a higher excess return is expected for the same amount of risk. It further exhibits a lower standard deviation, higher cumulative return and also higher average returns per annum.

Table 1: **Portfolio performance metrics**
This table shows performance metrics for the portfolio returns produced by the RL agent and by the equal weight benchmark heuristic. SR stands for Sharpe Ratio, MeanRet represents the mean return per annum, SD stands for the annualized standard deviation, which is obtained as follows: $\sigma_p^{p.a.} = \sqrt{252} * \sigma_p^{daily}$., where 252 is the number of trading days in a year.

	Metrics			
Portfolio	**Sharpe Ratio**	**Std.dev. (p.a.)**	**Cum. Ret. (5 years)**	**Mean Ret.(p.a.)**
RL Agent	2.72	11.96%	15.96%	5.53%
Benchmark	2.19	14.12%	8.42%	3.90%

7 Conclusion and possible extensions

In this paper, the architecture and practical application of a RL algorithm for portfolio management have been shown and discussed. By using an actor-critic framework, agent and environment have been programmed and connected. It was shown that the agent is able to clearly outperform an equal

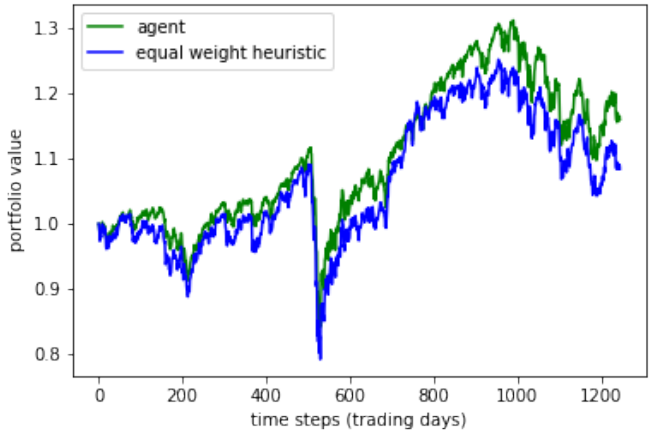

Figure 3: **Portfolio value over time - RL agent vs equal weight benchmark.**
This plot shows the evolution of the portfolio values for the dynamic weight allocation chosen by the RL agent in comparison to the static equal weight allocation used for the benchmark.

weight heuristic. Future research could concentrate on modifying the agent's underlying neural network architecture e.g. by using convolutional layers instead of fully connected layers to capture patterns in the time series of price data. Also, additional heuristics could be used to benchmark the agent's performance.

Two main limitations of basic actor-critic algorithms are identified and resolved by Ciosek, Vuong, Loftin, and Hofmann [6]: pessimistic underexploration and directional uninformedness. The first concept refers to a greedy exploration by the actor combined with a pessimistic estimate of the critic and the second concept states that actions in opposite directions from the current mean are explored with equal probabilities, even though regions in one direction of the action space are likely to have been explored by past policies previously. Thus, Ciosek, Vuong, Loftin, and Hofmann [6] propose the optimistic actor-critic algorithm, which is a modification based on an upper and lower bound on the state-action value function. Direct exploration can be done using the upper bound and the lower bound is used to avoid overestimation. Further research on the portfolio management application problem could be done by deploying and comparing this version of the actor-critic algorithm to the regular actor-critic that is utilized in this paper.

Another possibility is to employ the soft actor-critic algorithm suggested by Haarnoja, Zhou, Abbeel, and Levine [7]. This variant comes with an entropy term that is maximized along with the reward term in order to ensure random off-policy exploration while learning a stochastic policy. This approach comes with the advantages of better stability and sample efficiency. Thus, adding an entropy term and controlling exploration of the state space instead of using an on-policy actor-critic algorithm as implemented in this paper, would be a viable option for future research as well.

References

[1] F. Soleymani and E. Paquet. "Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath". In: *Expert Systems with Applications* 156 (2020), p. 113456. DOI: https://doi.org/10.1016/j.eswa.2020.113456.

[2] D. Snow. "Machine learning in asset management—Part 1: Portfolio construction—Trading strategies". In: *The Journal of Financial Data Science* 2.1 (2020), pp. 10–23. DOI: https://doi.org/10.3905/jfds.2019.1.021.

[3] D. Snow. "Machine learning in asset management part 2: Portfolio construction weight optimization". In: *The Journal of Financial Data Science* 2.2 (2020), pp. 17–24. DOI: https://doi.org/10.3905/jfds.2020.1.029.

[4] Z. Huang and F. Tanaka. "Investment Biases in Reinforcement Learning-based Financial Portfolio Management". In: *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*. IEEE. 2022, pp. 494–501. DOI: 10.23919/SICE56594.2022.9905789.

[5] M. Guan and X.-Y. Liu. "Explainable deep reinforcement learning for portfolio management: an empirical approach". In: *Proceedings of the Second ACM International Conference on AI in Finance*. 2021, pp. 1–9.

[6] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann. "Better exploration with optimistic actor critic". In: *Advances in Neural Information Processing Systems* 32 (2019). URL: https://proceedings.neurips.cc/paper/2019/file/a34bacf839b923770b2c360eefa26748-Paper.pdf.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.