A Deep Reinforcement Learning Approach to Stock Portfolio Optimization

Undergraduate Research Project

Pavanpreet Singh Gandhi

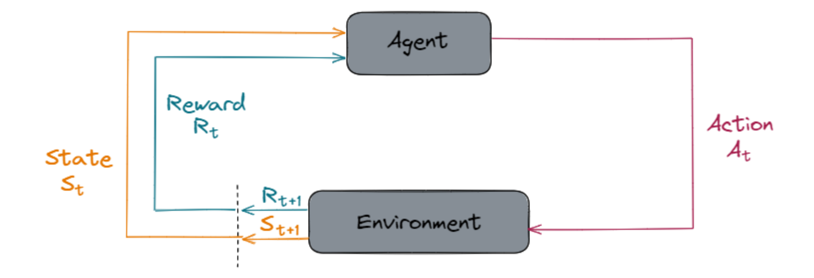
Mahidol University (MUIC)

Table of contents

What is Reinforcement Learning?

- Reinforcement learning is a **framework for learning** how to interact with a complex **environment** from **experience**.
- Reinforcement learning is a data science problem, also known as **self-supervised learning**.
- Reinforcement learning can be used to solve **sequential decision-making** problems.

Markov Decision Processes



Example Reward Sequence

$$s_0 \rightarrow a_1 \rightarrow s_1 \rightarrow a_2 \rightarrow s_2 \rightarrow a_3 \rightarrow s_3 \rightarrow r_3 \rightarrow a_4 \rightarrow s_4 \ldots \tag{1}$$

Policy/Value/Quality Functions

Policy functions map state-action pairs to probabilities.

$$\pi(s, a) = \Pr(\text{action} = a \mid \text{state} = s) \tag{2}$$

Value functions map states to expected reward values.

$$V_\pi(s) = \mathbb{E}\left(\sum_t \gamma^t r_t \mid s_0 = s\right) \tag{3}$$

Quality functions map state-action pairs to expected reward values.

$$Q_\pi(s, a) = \mathbb{E}\left(\sum_t \gamma^t r_t \mid s_0 = s, a_0 = a\right) \tag{4}$$

Taxonomy of Reinforcement Learning

- Reinforcement Learning
  - Model Based
  - Model Free
    - On Policy
    - Off Policy

Value Iteration

Overleaf

Value iteration was the **first ever reinforcement learning algorithm** [1].

Start by assuming we know the state transition probabilities

$$P(s' \mid s, a) = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{5}$$

and the reward structure

$$R(s', s, a) = \Pr(r_{t+1} \mid s_{t+1} = s', s_t = s, a_t = a) \tag{6}$$

These become our models for the reward and next state - hence value iteration is **model based**.

Value Iteration

Define the optimal value function as

$$V_\star(s) = \max_\pi V_\pi(s) = \max_\pi \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right), \forall s \tag{7}$$

Bellman showed that we could write this recursively as

$$V_\star(s) = \max_\pi \mathbb{E}\left(r + \sum_{t=1}^{\infty} \gamma^k r_t \mid s_1 = s'\right) = \max_\pi \mathbb{E}\left(r + \gamma V_\star(s')\right) \tag{8}$$

This is known as the **Bellman optimality equation**.

Value Iteration

Just change the **equality to an assignment**. Bellman [1] showed that this would eventually converge to the optimal value function.

$$V(s) \leftarrow \max_a \sum_{s'} P(s' \mid s, a) \left(R(s', s, a) + \gamma V(s')\right) \tag{9}$$

We can extract the optimal policy from the optimal value function by taking the action that results in the most *valuble* state.

Q-Learning

Q-learning was the **first ever model-free algorithm** [14].

In a similar manner to 7, write the optimal quality function recursively as

$$Q_\star(s,a) = \max_{\pi} \mathbb{E}\left(r + \gamma \max_{a'} Q_\star(s',a')\right). \tag{10}$$

Define the **TD-target estimate** as

$$R_\Sigma = r + \gamma \max_{a'} Q(s',a') \tag{11}$$

This is what we expect $Q(s,a)$ to be given some r, s'.

Q-Learning

The Q-learning algorithm just iterates through the MDP, collects **experience tuples** (s, a, r, s'), and updates the Q-values based on the following update equation

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R_\Sigma - Q(s, a)). \tag{12}$$

where α is some learning rate.

Watkins et al. [14] showed this process will eventually converge to the optimal Q-function.

Q-Learning

But how do we select the action when collecting experience tuples (s, a, r, s')?

- **SARSA** [10]: Select the best action always $\leftarrow$ On-policy
- **Q-Learning** [14]: Occasionally explore $\leftarrow$ Off-policy

Deep Q-Learning

Overleaf

Problem: Sometimes it is impossible to enumerate the Q-values for all the state-action pairs.

Solution: Use a neural network as a functional approximator of the Q-function.

$$Q(s, a) \approx Q(s, a; \theta) \tag{13}$$

where θ is a vector representing the parameters of the Q-network.

Deep Q-Learning

Overleaf

The loss function is derived from Q-learning12

$$\mathcal{L} = \left[Q(s, a; \theta) - (r + \gamma \max_{a'} Q(s', a'; \theta)\right]^2 \tag{14}$$

and the network parameters are adjusted using **backpropagation** and **experience replay**[1].

[1] In practice since deep Q-learning is off-policy, the training data set can be created on the go by playing the game and collecting experience tuples of the form (s, a, r, s'). We then randomly sample mini-batches of experience tuples for training and perform back-propagation on entire batches at a time. Simultaneously, we add new experience tuples based on our current Q-network and exploration strategy. This mechanism is called experience replay.

Finance Terminology

- A **security** is something that can be traded e.g stocks, bonds, options, futures, crypto, index funds, etc.
- A **portfolio** is a basket of tradeable securities. Investors hold portfolios with the hope that they will grow in value over time.
- The **rate of return** on an investment is the percentage change in price for the given unit of time, for example the daily rate of return.

$$r_t = \left(\frac{p_1 - p_0}{p_0}\right) \tag{15}$$

Risk and Return of a Security

Overleaf

Investing in a security for n days results in a sequence of daily returns.

$$\{r_1, r_2, r_3, \ldots, r_n\} \tag{16}$$

The **annualized mean historical return** is

$$\mu = \frac{252}{n} \sum_{i=1}^{n} r_i \tag{17}$$

and the **annualized historical risk** *(or standard deviation)* is

$$\sigma = \sqrt{\frac{252}{n} \sum_{i=1}^{n} (\mu - r_i)^2} \tag{18}$$

where there are 252 trading days in a year.

Risk and Return of a Portfolio

Overleaf

Suppose an investor holds a portfolio of k securities for n days. Each security will have its own sequence of daily returns.

The portfolio can be characterized by a sequence of weights.

$$\{w_1, w_2, \ldots, w_k\} \tag{19}$$

The **annualized mean historical return** of the portfolio is

$$\mu_{\text{port}} = \sum_{i=1}^{k} w_i \cdot \mu_i \tag{20}$$

and the **annualized historical risk** of the portfolio is

$$\sigma_{\text{port}} = \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{k} w_i \cdot w_j \cdot \sigma_{ij}} \tag{21}$$

where σ_{ij} is the covariance between the returns of securities i and j.

Mean-Variance Optimization

Overleaf

In 1952, Harry Markowitz proposed a systematic approach to constructing portfolios using **mean-variance analysis** [7], which involves solving the following **nonlinear bi-objective optimization problem**.

$$\begin{gathered}
\max \sum_{i=1}^{k} w_i \cdot \mu_i \\
\min \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{k} w_i \cdot w_j \cdot \sigma_{ij}} \\
\text{subject to} \\
\sum_{i=1}^{k} w_i = 1 \\
0 \leq w_1 \leq 1
\end{gathered} \tag{22}$$

Sharpe Ratio Optimization

The **Sharpe ratio** is a metric introduced by William F Sharpe to evaluate the performance of mutual funds according to their expected risk and return [11].

$$\max \left(\frac{\sum_{i=1}^{k} w_i \cdot \mu_i - R_f}{\sqrt{\sum_{i=1}^{k} \sum_{j=1}^{k} w_i \cdot w_j \cdot \sigma_{ij}}} \right)$$

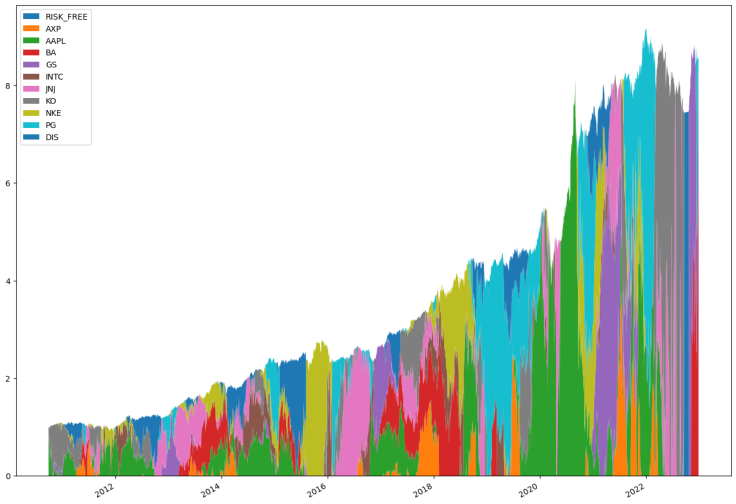
subject to

$$\sum_{i=1}^{k} w_i = 1$$

$$0 \leq w_1 \leq 1 \tag{23}$$

Maximum Sharpe Portfolio Visualized

Overleaf



Data

Overleaf

- Ten stocks[2] from the **Dow Jones Industrial Average** were arbitrarily selected.
- Price data was downloaded from *Yahoo Finance* and technical analysis-inspired features were generated e.g daily returns, rolling returns, rolling standard deviations, volume percent changes, etc.
- Data was split into training, validation, and test sets.

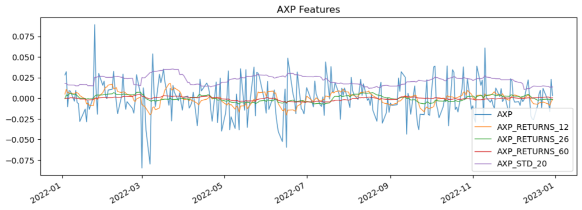


Figure: Visualization of features for one stock over one year

[2] American Express Company, Apple Inc, Boeing Co, Goldman Sachs Group Inc, Intel Corporation, Johnson & Johnson, Coca-Cola Co, Nike Inc, Procter & Gamble Co, and Walt Disney Co.

Environment Details

State and Observation

- The **state** is the current market situation.
- The **observation** is a 10 day history of the features.

Action

Actions change portfolio weights. There are 2 actions per stock - **buy or sell** - and a hold action which does nothing. Buying and selling occur in **10% increments**.

Reward Structure

A good **reward structure** is essential for success since we want to *reinforce* the right behavior.

1. Return of the portfolio ✗
2. Cumulative return of the portfolio ✗
3. Weighted cumulative return of the portfolio ✓

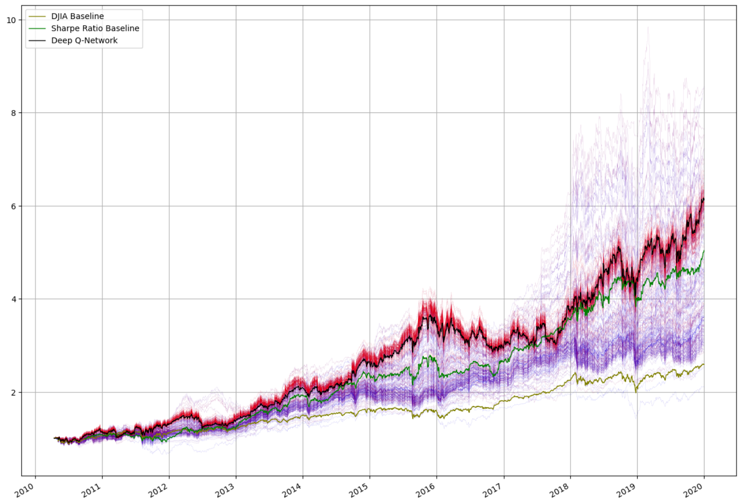
Model

Overleaf

- **Deep Q-learning model** with **experience replay**.
- Default hyper-parameters[3].
- Default network architecture (2 hidden layers, each with 64 nodes).
- Trained over $3,000,000$ time-steps.

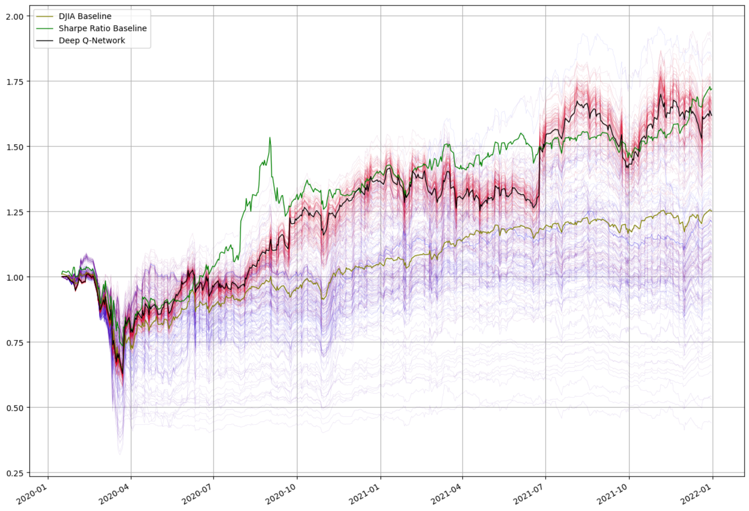
[3] except for *learning rate* being set to 0.0003 and the *batch size* being set to 64

Performance on Training Data

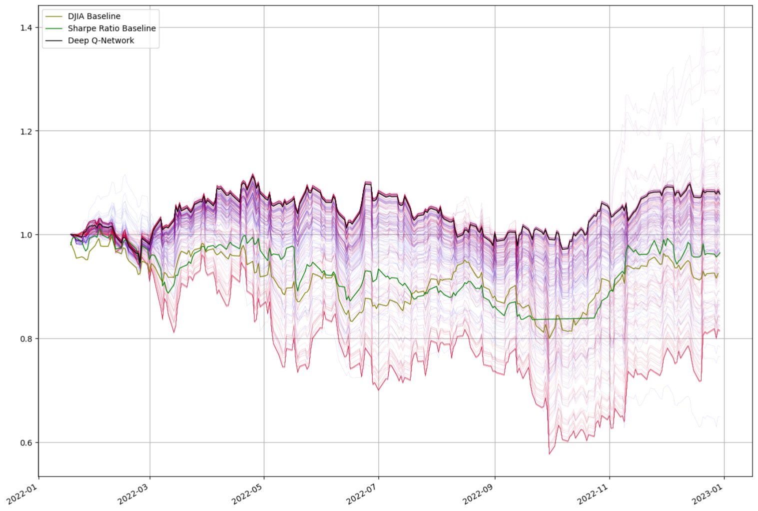


Performance on Validation Data

Overleaf



Performance on Test Data



Performance

Overleaf

Training Data	Mean Return	Risk	Sharpe Ratio
DJIA	0.103	0.141	0.730
Max Sharpe	0.191	0.170	1.129
Deep Q-Network	0.206	0.216	0.951

Validation Data	Mean Return	Risk	Sharpe Ratio
DJIA	0.121	0.277	0.438
Max Sharpe	0.318	0.319	0.996
Deep Q-Network	0.278	0.333	0.835

Test Data	Mean Return	Risk	Sharpe Ratio
DJIA	-0.079	0.202	-0.389
Max Sharpe	-0.037	0.196	-0.189
Deep Q-Network	0.082	0.172	0.474

Limitations and Future Work

- Hyper-parameter tuning
- Verify with different underlying securities
- More informative features (e.g market sentiment, news headlines)
- Different model (e.g Actor-Critic, DDPG)
- Different network architecture (e.g LSTM)

References I

[1] R. Bellman.
A markovian decision process.
Journal of mathematics and mechanics, pages 679–684, 1957.

[2] S. L. Brunton and J. N. Kutz.
Data-driven science and engineering: Machine learning, dynamical systems, and control.
Cambridge University Press, 2022.

[3] H. Hasselt.
Double q-learning.
Advances in neural information processing systems, 23, 2010.

[4] R. A. Howard.
Dynamic programming and markov processes.
1960.

References II

[5] V. Konda and J. Tsitsiklis.
Actor-critic algorithms.
Advances in neural information processing systems, 12, 1999.

[6] D. Lee, H. Seo, and M. W. Jung.
Neural basis of reinforcement learning and decision making.
Annual review of neuroscience, 35:287–308, 2012.

[7] H. Markowitz.
Portfolio selection the journal of finance, vol. 7, no. 1, 1952.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller.
Playing atari with deep reinforcement learning.
arXiv preprint arXiv:1312.5602, 2013.

References III

[9] G. V. Pai.
Metaheuristics for portfolio optimization: an introduction using MATLAB.
John Wiley & Sons, 2017.

[10] G. A. Rummery and M. Niranjan.
***On-line Q-learning using connectionist systems*, volume 37.**
University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[11] W. F. Sharpe.
The sharpe ratio.
Streetwise–the Best of the Journal of Portfolio Management, 3:169–185, 1998.

[12] R. S. Sutton and A. G. Barto.
Reinforcement learning: An introduction.
MIT press, 2018.

References IV

[13] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour.
Policy gradient methods for reinforcement learning with function approximation.
Advances in neural information processing systems, 12, 1999.

[14] C. J. Watkins and P. Dayan.
Q-learning.
Machine learning, 8:279–292, 1992.