Reinforcement Learning to Solve Merton's Portfolio Problem

By:

Jose Alejandro Hernandez Duenas
500553404

Thesis submitted for MTH 40 A/B course
Supervisor: Dr. Alexey Rubtsov
Ryerson department of Mathematics

Acknowledgements

Thank you to my professor and mentor Alexey Rubtsov for his help and guidance. I'd also like to extend my gratitude to my friend Buko for all his support and patience.

Introduction

One of the greatest motivations in modern quantitative finance is to streamline and optimize decision making. Reinforcement learning serves as an important tool in the modern investor's repetoire to handle analytically comlex decisions. Reinforcement learning is a type of machine learning in which an agent is trained by trial and error to optimize decisions taken in real time to maximize immediate and long term rewards. An important advantage of reinforcement learning is that the agent can be trained to make the best decisions without knowing its environment. This agent can be dropped into a market with many different scenarios and regulations and learn how to optimize for the desired result.

Merton's portfolio problem is a well known problem in which an investor must decide how to allocate their wealth to maximize utility. The investor makes decisions in distributing the wealth in a portfolio between a risky and risk-free asset while partially consuming the portfolio over time. There have been many adaptations to Merton's problem to increase the complexity to keep it in line with more contemporary financial markets and instruments as well as realistic scenarios[4][5]. We will attempt to solve the base problem with reinforcement learning and show that it can be easily adapted to encase any number of real life regulations.

Background Theory

Markov Decision Process

A finite Markov Decision Process (MDP) can be thought of as the formal mathematic representation of reinforcement learning. MDPs represent sequential decision making where actions affect immediate and future rewards. An MDP will include the role of an agent who operates in a statespace $\mathcal{S}$ and collects rewards R_t by the actions taken at each state. The agent operates in an environment underwhich actions are taken from the action space $\mathcal{A}$. After taking an action, the agent observes the next state generated in by the environment and collects the reward from the state the action resulted in. The goal of the agent is to create a policy π which will accumulate the greatest rewards total. The agent continues to take actions until reaching a terminal state or the end of an episode. Upon reaching the end, each state presents a value, this value is representative of how beneficial it is to be in that state. The policy will determine which action is taken at every state encountered and is written as $\pi(a|s)$. The optimal policy is the policy which will select the appropriate action at every state to maximize the total reward accumulated G_t[3]. The agent only comes to learn about the environment through experience. Over the course of multiple episodes, the agent will improve the policy it follows to maximize total rewards. The most used form to evaluate future rewards is the Bellman equation which allows the agent to look ahead.

MDP formulation and corresponding transition function:

$$P(S_t = s' | S_{t-1} = s,\ a_t = a) = T(s, a, s')$$

The Bellman equation[3]:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s], \qquad 0 \leq \gamma \leq 1 \tag{1}$$

Here v_π represents the value function with respect to the policy π. The value of the current state is equal to the expected value of the next reward and all future rewards given the state if the agent were to follow the policy π. The total of all future rewards may be discounted by the value of γ. At a discount rate of 0, the agent is only interested in maximizing the best next reward and is said to be myopic or nearsighted. At a rate of 1, the agent values future rewards as heavily as current rewards.

The optimal policy is defined as one in which[3]:

$$\pi \geq \pi' \iff v_p i \geq v'_\pi$$

That is, there will always be a policy which is greater than or equal to any other policy if and only if the value function of following that policy is greater than or equal to any other policy's value function. An optimal action value function $q^*(s, a)$ is therefore the value function whose policy maxes the total across all state action pairs[3].

Bellman's optimality equation:

$$v^*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi^*}(s, a) \qquad \forall : s \in \mathcal{S} \tag{2}$$

$$q^*(s, a) = \max_\pi q_\pi(s, a) \qquad \forall : [s \in \mathcal{S}, a \in \mathcal{A}]$$

$$v^*(s) = \max_a \sum_{s',\, r} p(s', r | s, a)[r + \gamma v^*(s')] \tag{3}$$

$$q^*(s, a) = \max_a \sum_{s',\, r} p(s', r | s, a)[r + \gamma \max_{a'} q^*(s', a')]$$

Dynamic Programming

The Bellman equation offers us a way to look ahead from the current state at all the possible states to get a better understanding of how valueable the current state is. From the current state s the actions the agent takes can present their immediate respective rewards and new resulting states. This is a one step look ahead, where one state's value is representative on how rewarding the next actions will be from that state. Similarly the Bellman equation lets us set a value to ensuing states by using the Bellman equation and doing a look ahead from those available states. The value of each state is therefore represented by performing the Bellman equation across all state-action pairs.

The Bellman equation therefore offers a system of equations which features all the actions probabilities and the states which they result in. A finite MDP contains all the state-action pairs with their respective rewards. Subject to a few constraints, the Bellman optimality equations

give a unique solution for the optimal value function. If the dynamics of the environment are known, then for n states there are n equations and n unknowns. This system can be solved in the same manner as any system of nonlinear equations. This would be akin to an exhaustive search however over every single state-action pair. Real world environments offer very large statespaces which would make this solution too computationally intensive to be logistically reasonable. Iterative and recursion methods allow for a numerical solution which approximates v^* and q^*. With the optimal v^* or q^* the agent would only need to do a one-step look ahead since by Bellman's optimality principals, if the next step was optimal every step stemming forward from it is optimal. The agent would only need to act greedily and select the highest value from v^* or the action that maximizes q^*.

Stochastic process

We will consider the model for the price of a risky asset which follows geometric brownian motion with log normal returns.

$$dX_t = \mu X_t \, dt + \sigma X_t \, dZ_t \tag{4}$$

μ is the mean return for the asset

σ is the volatility for the asset

X_t is the asset price at time t

Z_t is a Wiener process which denotes the random walk for the stochastic process

μX_t is the drift term for the process and N

The model for how a riskless asset evolves without an instantaneous volatility term

$$dY_t = \rho \, Y_t \, dt \tag{5}$$

ρ is the risk free rate

Y_t is the price of the riskless asset at time t

Merton's Portfolio Problem

An investor starts with current wealth $W_0 > 0$ and expects to hold it from time $0 \leq t \leq T$. They are free to allocate (π) the wealth a risky asset X — which could be: stock, real-estate, or a portfolio of different stocks — and $(1 - \pi)$ into a riskless asset Y. At time t they must decide what portion c to consume from the portfolio and how to reinvest the remainder between X and Y. The objective for the investor is to do the allocation while trying to accomodate the utility function which will be constant relative risk aversion:

$$U(x) = \begin{cases} \frac{x^{1-\gamma}}{1-\gamma} & \gamma \neq 1 \\ ln(x) & \gamma = 1 \end{cases} \tag{5}$$

The investor will therefore be attempting to maximize the utility of their consumption from t until T, as well as the utility value of their wealth W_t at time T:

$$\max \mathbb{E}\left[\int_t^T e^{\rho(s-t)}U(c_s)\,ds + e^{\rho(s-t)}U(W_T)\right]$$

$$\max \mathbb{E}\left[\int_t^T \frac{e^{\rho(s-t)}c_s^{1-\gamma}}{1-\gamma}\,ds + \frac{e^{\rho(T-t)}W_T^{1-\gamma}}{1-\gamma}\right] \tag{6}$$

The wealth of a portfolio can be described as a function of the number of shares n_x invested in X_t and n_Y in the riskless asset Y_t

$$W_t = n_X X_t + n_Y Y_t \tag{5}$$

The change in the value of the portfolio would therefore follow the update from the sum of the change in assets:

$$dW_t = n_X dX_t + n_Y dY_t \tag{6}$$

We can then substitute in the relation between W_T, Y_t, and X_t from (5)

$$dW_t = n_X X_t \frac{dX_t}{X_t} + n_Y Y_t \frac{dY_t}{Y_t}$$

$$dW_t = n_X X_t \frac{dX_t}{X_t} + (W_t - n_X X_t)\frac{dY_t}{Y_t}$$

$$dW_t = W_t\left[(\frac{n_X X_t}{W_t}\,\frac{dX_t}{X_t} + \frac{(W_t - n_X X_t)}{W_t}\,\frac{dY_t}{Y_t})\right] \tag{7}$$

We can simplify the terms in (7) if we create terms for the proportions of the wealth from the assets

$$\pi_t = \frac{n_X X_t}{W_t} \tag{8}$$
$$and$$
$$(1-\pi_t) = \frac{(W_t - n_X X_t)}{W_t}$$

Now plugging (8) back into (7):

$$dW_t = W_t\left[\pi_t \frac{dX_t}{X_t} + (1-\pi_t)\frac{dY_t}{Y_t})\right] \tag{9}$$

(3) and (4) into (9):

$$dW_t = W_t\left[\pi_t(\mu\,dt + \sigma\,dZ_t) + (1-\pi_t)r\,dt\right]$$
$$dW_t = W_t\left[\pi_t\mu + (r - \pi_t r)\,dt + \pi\sigma\,dZ_t)\right]$$

$$dW_t = W_t\left[(\pi_t(\mu - r) + r)\,dt + \pi\sigma\,dZ_t)\right] \tag{10}$$

Finally we add the term for consumption into the wealth dynamics to get the final dynamics for the evolution of W_t:

$$dW_t = (W_t(\pi_t(\mu - r) + r) - c_t)\,dt + \pi\sigma W_t\,dZ_t \tag{11}$$

Analytical solution to Merton's portfolio problem

The HJB equation:

$$0 = Max_{C(t),w(t)} \left[\int_t^T e^{-\rho s} U[C(s)] ds + \frac{\partial I_t}{\partial t} + \frac{\partial I_t}{\partial W} [(w(t)(\mu - r) + r)W(t) - C(t)] \right.$$
$$\left. + 1/2 \frac{\partial^2 I_t}{\partial W^2} \sigma^2 \pi^2(t) W^(t)] \right]$$

Here I_t is short for $I[W(t), t]$

[4][3]

Which Merton solves obtaining the following:

$$\pi^*(t) = \frac{\mu - r}{\sigma^2 \gamma}$$

$$v = \frac{\rho - (1-\gamma)(\frac{(\mu-r)^2}{2\sigma^2\gamma} + r)}{\gamma}$$

$$C^* = \begin{cases} [\frac{v}{1+(v\epsilon-1)exp(v(t-T))}]W(t)], & v \neq 0 \\ [\frac{1}{(T-t+\epsilon)}]W(t), & v = 0 \end{cases}$$

[4][6] This solution can be used to compare the agent to the continuous time model derived by Merton

Reinforcement Learning

MDP formulation for stochastic control

State:

The state at from the statespace $\mathcal{S}$ is the percentage of time elapsed t/T.

$$s \in \mathcal{S}$$

$$s(t) = \frac{100t}{T}$$

Action:

The agent will operate every 2 trading days or every 10 timesteps. This is equivalent to consuming every 2 weeks similair to a biweekly provision. The agents actions will be scaled to reflect 10 timesteps. The proportion of the assets allocated to the risky assets will remain the

same until the next action taken. Again, this is similair to a real world investor re-evaluating their asset allocation periodically vs continuously. Since the agent will operate in a discrete time setting, and since the statespace is a function of the percentage of time elapsed this helps keep generality across the state action pairs. For a consumption only model with no added investments we limit c_t to the set $c_t \in \{0.01, 0.10, 0.50, 1.00\}$. For faster convergence, and to avoid a curse of dimensionality problem we limit the values the agent can consume at every timestep.

$$a \in \mathcal{A},$$

$$a(\pi, c_t), \pi \in \{0.01, 0.02, 0.03, \ldots, 1\}$$
$$and$$
$$c_t \in \{0.01, 0.10, 0.50, 1.00\}$$

Reward:

The goal is to maximize the utility of consumption. Therefore the reward will be the utility for the consumption done at time t. The total return is the accumulated discounted reward up to time t.

$$r(a_t, s_t) = \begin{cases} \frac{W_t c_t^{1-\gamma}}{1-\gamma} & \gamma \neq 1 \\ ln(c_t W_t) & \gamma = 1 \end{cases}$$

The value function for this MDP is the following:

$$V_\pi(s) = E_\pi[G_t | S_t = s]$$

That is the value at state s according to the policy pi is the expected total return given the state from the state space.

Using the accumulated reward over time in discrete time steps we get the following:

$$V_\pi(s) = E_\pi[\sum_{i=0}^{T} (1+\rho)^{T-t} \frac{(W_t c_t)^{(1-\gamma)}}{1-\gamma} | S_t = s]$$

Using a Monte Carlo approach we can approximate the true value function by sampling each state action pair's return over a large number of visits. From the Law of Large Numbers we know this will eventually converge to the true value as the visits approach infinity. We will be using a running mean with the learning rate $\alpha = 0.01$ so that we can forget older episodes in the mean.

The simulation for the agent will run under the following parameters

$\mu = 0.1$

$\sigma = 0.2$

$r = 0.02$

$\gamma = 4$

$\rho = 0.05$

$W_0 = 50000$

$\alpha = 0.01$

$T = 5years$

$dt = 1/252$

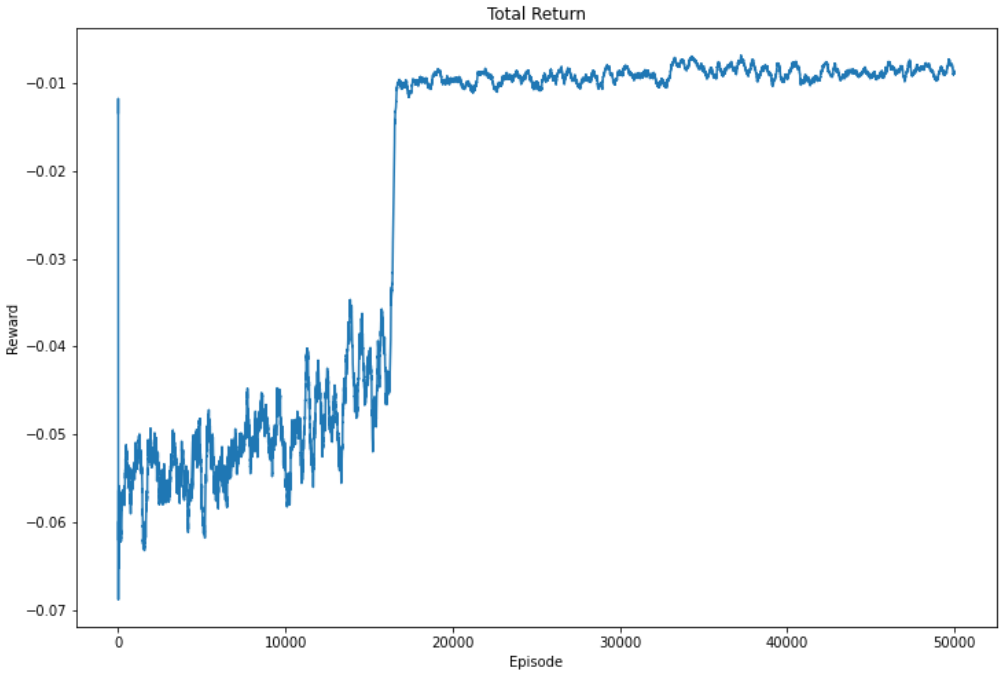


Fig. 1

The agent showed convergence, although slightly eradic, towards a more stable plateau. Fig. 1 represents a running average over a 300 episode window for the agents progress across episodes. The problem is not stationary and the agent is following an $\epsilon - greedy$ soft policy which will also add another stochastic element. Over time however, the progress becomes less eratic and stabilizes around a local maximum.

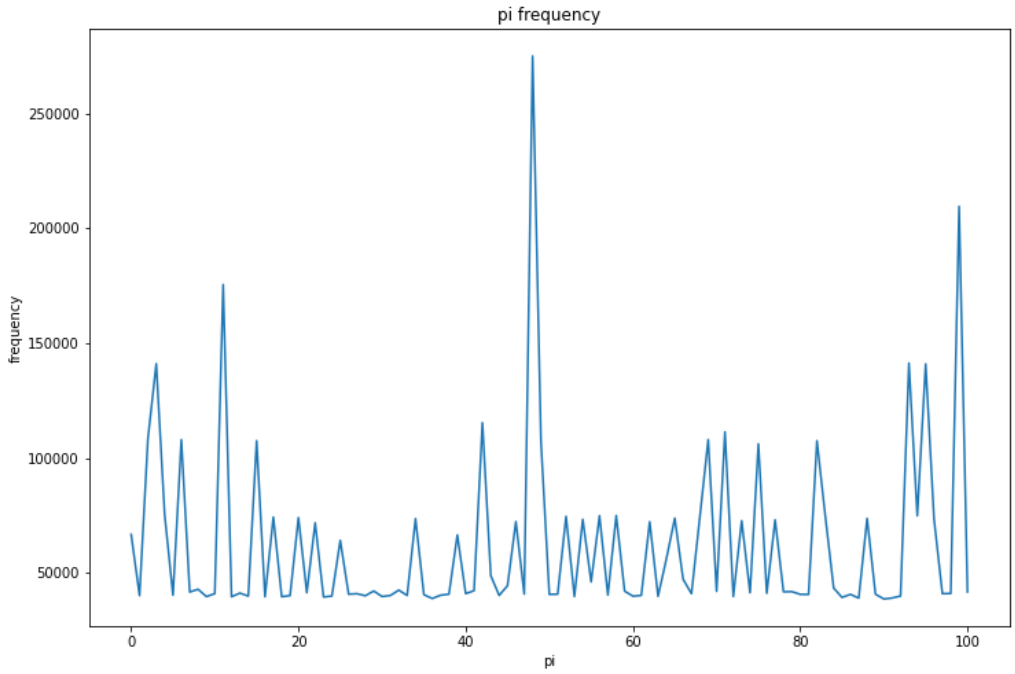


Fig. 2

The agent was able to approximate an optimal static allocation strategy similar to Merton's in which a certain proportion of wealth invested in a risky asset was favored. Merton's fraction for the parameters in this simulation called for a 49.99% investment into the risky asset.

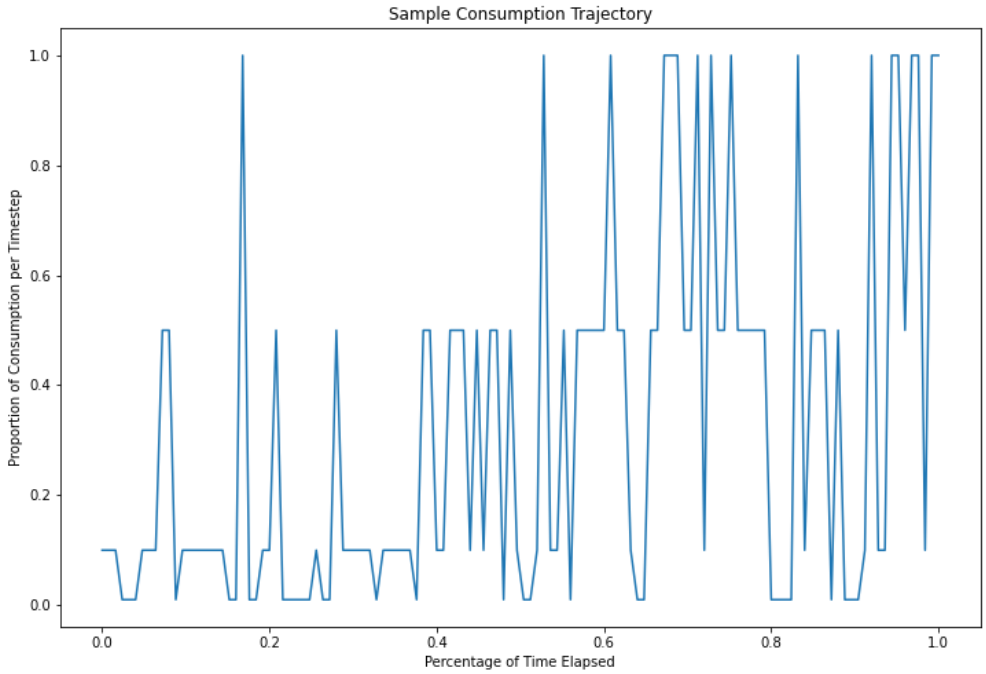


Fig. 3

The last consumption strategy for the last episode shows the agent values letting the investment grow by not consuming early and then rapidly consuming towards the end of the time horizon.

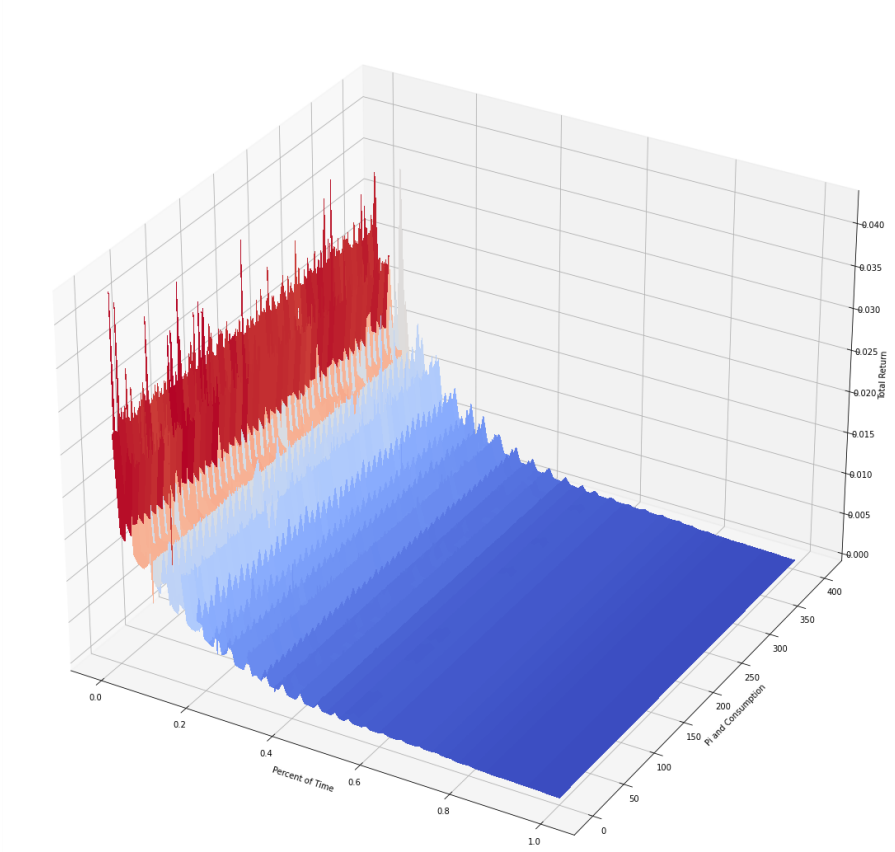


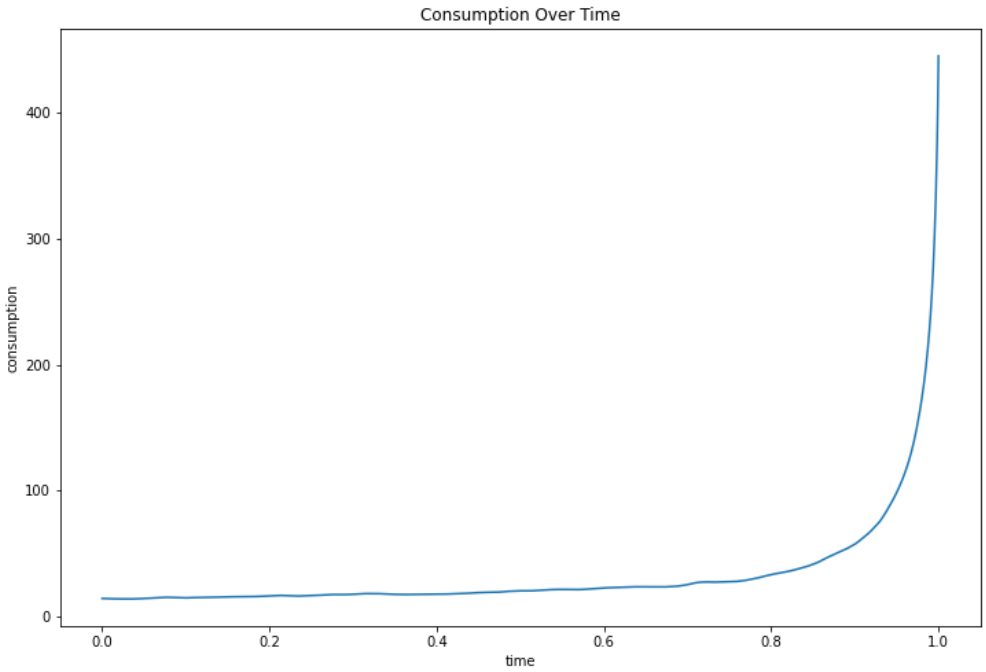
Fig. 4

The surface plot for the the Q action-value function of the agent. The return follows an exponential distribution from the final states toward the first states. This is in line with the exponential nature of discounting the rewards.

The agent averaged the following real world values

Total money consumed: 46161.88 Total money leftover: 13434.06 Mean Return: -0.0225

Merton's solution



Comparing Merton's analytical solution, the agent resembles the same strategy of low consumption at the start and letting the portfolio grow before rapidly consuming toward the end of the time horizon. This can also be inferred intuitively from the analytical solution. There is the (T-t) term in the denominator for the function dictation consumption, therefore as t approaches T the consumption increases.

Merton's solution was able to generate more money than the agent:

Total money consumed: 52653.72 Total money leftover: 10387.95

However the mean accumulated return was very similar to the agent:

Mean Return: -0.047

Conclusion:

The purpose of using reinforcement learning on Merton's Portfolio problem was mainly to overcome the difficult dynamic programming calculations and derivations. As a result it would be easier to change the parameters of the problem by altering the environment in the simulation. There is a variance that arises from using the MC method to approximate the value function. Although the agent was able to approximate a strategy to optimize a local max return there are some issues regarding the convergence. The main problem with the algorithm is the magnitude of the reward. With a gamma of 4, the utility function becomes an inverse relation with the consumption going to the denominator and being cubed. Therefore as the consumption increases the absolute magnitude of the reward changes at an exponential pace. The stochastic nature of the problem due to simulating the wealth process can mean that the change in value function can be influenced more by the random growth of the process. Over a

large enough sample the value function should converge to a stable point. However it should continue to oscillate because of the randomness associated with the rate of exploration as well as because of the agent being punished by an unfortunate brownian motion trajectory.

To improve the model and simulation, in the future more powerful algorithms should be implemented. An actor-critic model using a policy gradient could enable a larger action and state space allowing for a model with more control. These models are more subject to bias however, and are more computationally expensive to run than the simpler tabular methods used. Value function or policy gradient approximation using the parameters would be much easier to scale however. It would increase the generality and allow real world returns be used as parameters in the state space to create a much more dynamic agent.

One last consideration for this agent was the idea of creating a network of agents. Creating an agent for each task associated with the problem would simplify the parameters much more and allow for higher scalability without increasing exposure to the curse of dimensionality. One agent could be assigned to learning optimal consumption with the other agent learning optimal asset allocation. The agents would need to be able to communicate and consider the others actions. Following with contemporary machine learning, this would resemble the strategy used by humans of delegating tasks to teams for a combined effort. It would also be possible to assign multiple agents to each task and let either the strongest or a combination of the best strategies be the overarching strategy. Bootstrapping the agents in this manner would certainly help to reduce a lot of the variance associated with the MC method used.

As a simple but effective model however, the tabular MC method to approximate the optimal strategy gave a scalable model to simulate the problem.

Bibliography

[1] On the Theory of Dynamic Programming. Bellman

[2] Dynamic Programming. Bellman

[3] Reinforcement Learning. barto and sutton

[4] Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case. Merton

[5] Optimum Consumption and Portfolio Rules in a Continuous-time Model. Merton

[6] Foundations of Reinforcement Learning with Applications in Finance. Rao

[7] Multi-agent Reinforcement Learning: An Overview. Buşoniu, Babuška, and De Schutter