Foundations of Reinforcement Learning with Applications in Finance

Ashwin Rao, Tikhon Jelvis

1 Dynamic Asset-Allocation and Consumption

This chapter covers the first of five financial applications of Stochastic Control covered in this book. This financial application deals with the topic of investment management for not just a financial company, but more broadly for any corporation or for any individual. The nuances for specific companies and individuals can vary considerably but what is common across these entities is the need to:

- Periodically decide how one's investment portfolio should be split across various choices of investment assets - the key being how much money to invest in more risky assets (which have potential for high returns on investment) versus less risky assets (that tend to yield modest returns on investment). This problem of optimally allocating capital across investment assets of varying risk-return profiles relates to the topic of Utility Theory we covered in Chapter **??**. However, in this chapter, we deal with the further challenge of adjusting one's allocation of capital across assets, as time progresses. We refer to this feature as *Dynamic Asset Allocation* (the word dynamic refers to the adjustment of capital allocation to adapt to changing circumstances)

- Periodically decide how much capital to leave in one's investment portfolio versus how much money to consume for one's personal needs/pleasures (or for a corporation's operational requirements) by extracting money from one's investment portfolio. Extracting money from one's investment portfolio can mean potentially losing out on investment growth opportunities, but the flip side of this is the Utility of Consumption that a corporation/individual desires. Noting that ultimately our goal is to maximize total utility of consumption over a certain time horizon, this decision of investing versus consuming really amounts to the timing of consumption of one's money over the given time horizon.

Thus, this problem constitutes the dual and dynamic decisioning of asset-allocation and consumption. To gain an intuitive understanding of the challenge of this dual dynamic decisioning problem, let us consider this problem from the perspective of personal finance in a simplified setting.

1.1 Optimization of Personal Finance

Personal Finances can be very simple for some people (earn a monthly salary, spend the entire salary) and can be very complicated for some other people (eg: those who own multiple businesses in multiple countries and have complex assets and liabilities). Here we shall consider a situation that is relatively simple but includes sufficient nuances to provide you with the essential elements of the general problem of dynamic asset-allocation and consumption. Let's say your personal finances consist of the following aspects:

- *Receiving money*: This could include your periodic salary, which typically remains constant for a period of time, but can change if you get a promotion or if you get a new job. This also includes money you liquidate from your investment portfolio, eg: if you sell some stock, and decide not to re-invest in other investment assets. This also includes interest you earn from your savings account or from some bonds you might own. There are many other ways one can *receive money*, some fixed regular payments and some uncertain in terms of payment quantity and timing, and we won't enumerate all the different ways of *receiving money*. We just want to highlight here that *receiving money* at various points in time is one of the key financial aspects in one's life.
- *Consuming money*: The word "consume" refers to ""spending." Note that one needs to *consume money* periodically to satisfy basic needs like shelter, food and clothing. The rent or mortgage you pay on your house is one example - it may be a fixed amount every month, but if your mortgage rate is a floating rate, it is subject to variation. Moreover, if you move to a new house, the rent or mortgage can be different. The money you spend on food and clothing also constitutes *consuming money*. This can often be fairly stable from one month to the next, but if you have a newborn baby, it might require additional expenses of the baby's food, clothing and perhaps also toys. Then there is *consumption of money* that are beyond the "necessities" - things like eating out at a fancy restaurant on the weekend, taking a summer vacation, buying a luxury car or an expensive watch etc. One gains "satisfaction"/"happiness" (i.e., *Utility*) from this *consumption of money*. The key point here is that we need to periodically make a decision on how much to spend (*consume money*) on a weekly or monthly basis. One faces a tension in the dynamic decision between *consuming money* (that gives us *Consumption Utility*) and *saving money* (which is the money we put in our investment portfolio in the hope of the money growing, so we can consume potentially larger amounts of money in the future).
- *Investing Money*: Let us suppose there are a variety of investment assets you can invest in - simple savings account giving small interest, exchange-traded stocks (ranging from value stocks to growth stocks, with their respective risk-return tradeoffs), real-estate (the house you bought and live in is indeed considered an investment asset), commodities such as gold, paintings etc. We call the composition of money invested in these assets as one's investment portfolio (see Appendix **??** for a quick introduction to Portfolio Theory). Periodically, we need to decide if one should play safe by putting most of one's money in a savings account, or if we should allocate investment capital mostly in stocks, or if we should be more speculative and invest in an early-stage startup or in a rare painting. Reviewing the composition and potentially re-allocating capital (refered to as re-balancing one's portfolio) is the problem of dynamic asset-allocation. Note also that we can put some of our *received money* into our investment portfolio (meaning we choose to not consume that money right away). Likewise, we can extract some money out of our investment portfolio so we can *consume money*. The decisions of insertion and extraction of money into/from our investment portfolio is essentially the dynamic money-consumption decision we make, which goes together with the dynamic asset-allocation decision.

The above description has hopefully given you a flavor of the dual and dynamic decisioning of asset-allocation and consumption. Ultimately, our personal goal is to maximize the Expected Aggregated Utility of Consumption of Money over our lifetime (and perhaps, also include the Utility of Consumption of Money for one's spouse and children,

after one dies). Since investment portfolios are stochastic in nature and since we have to periodically make decisions on asset-allocation and consumption, you can see that this has all the ingredients of a Stochastic Control problem, and hence can be modeled as a Markov Decision Process (albeit typically fairly complicated, since real-life finances have plenty of nuances). Here's a rough and informal sketch of what that MDP might look like (bear in mind that we will formalize the MDP for simplified cases later in this chapter):

- States: The *State* can be quite complex in general, but mainly it consists of one's age (to keep track of the time to reach the MDP horizon), the quantities of money invested in each investment asset, the valuation of the assets invested in, and potentially also other aspects like one's job/career situation (required to make predictions of future salary possibilities).
- Actions: The *Action* is two-fold. Firstly, it's the vector of investment amounts one chooses to make at each time step (the time steps are at the periodicity at which we review our investment portfolio for potential re-allocation of capital across assets). Secondly, it's the quantity of money one chooses to consume that is *flexible/optional* (i.e., beyond the fixed payments like rent that we are committed to make).
- Rewards: The *Reward* is the Utility of Consumption of Money that we deemed as flexible/optional - it corresponds to the second part of the *Action*.
- Model: The *Model* (probabilities of next state and reward, given current state and action) can be fairly complex in most real-life situations. The hardest aspect is the prediction of what might happen tomorrow in our life and career (we need this prediction since it determines our future likelihood to receive money, consume money and invest money). Moreover, the uncertain movements of investment assets would need to be captured by our model.

Since our goal here was to simply do a rough and informal sketch, the above coverage of the MDP is very hazy but we hope you get a sense for what the MDP might look like. Now we are ready to take a simple special case of this MDP which does away with many of the real-world frictions and complexities, yet retains the key features (in particular, the dual dynamic decisioning aspect). This simple special case was the subject of Merton's Portfolio Problem (Merton 1969) which he formulated and solved in 1969 in a landmark paper. A key feature of his formulation was that time is continuous and so, *state* (based on asset prices) evolves as a continuous-time stochastic process, and actions (asset-allocation and consumption) are made continuously. We cover the important parts of his paper in the next section. Note that our coverage below requires some familiarity with Stochastic Calculus (covered in Appendix **??**) and with the Hamilton-Jacobi-Bellman Equation (covered in Appendix **??**), which is the continuous-time analog of Bellman's Optimality Equation.

1.2 Merton's Portfolio Problem and Solution

Now we describe Merton's Portfolio problem and derive its analytical solution, which is one of the most elegant solutions in Mathematical Economics. The solution structure will provide tremendous intuition for how the asset-allocation and consumption decisions depend on not just the state variables but also on the problem inputs.

We denote time as t and say that current time is $t = 0$. Assume that you have just retired (meaning you won't be earning any money for the rest of your life) and that you are going to live for T more years (T is a fixed real number). So, in the language of the previous section, you will not be *receiving money* for the rest of your life, other than the option of extracting money from your investment portfolio. Also assume that you have no fixed payments to make like mortgage, subscriptions etc. (assume that you have already paid for a retirement service that provides you with your essential food, clothing and other services). This means all of your *money consumption* is flexible/optional, i.e., you have a choice of consuming any real non-negative number at any point in time. All of the above are big (and honestly, unreasonable) assumptions but they help keep the problem simple enough for analytical tractability. In spite of these over-simplified assumptions, the problem formulation still captures the salient aspects of dual dynamic decisioning of asset-allocation and consumption while eliminating the clutter of A) *receiving money* from external sources and B) *consuming money* that is of a non-optional nature.

We define wealth at any time t (denoted W_t) as the aggregate market value of your investment assets. Note that since no external money is received and since all consumption is optional, W_t is your "net-worth." Assume there are a fixed number n of risky assets and a single riskless asset. Assume that each risky asset has a known normal distribution of returns. Now we make a couple of big assumptions for analytical tractability:

- You are allowed to buy or sell any fractional quantities of assets at any point in time (i.e., in continuous time).
- There are no transaction costs with any of the buy or sell transactions in any of the assets.

You start with wealth W_0 at time $t = 0$. As mentioned earlier, the goal is to maximize your expected lifetime-aggregated Utility of Consumption of money with the actions at any point in time being two-fold: Asset-Allocation and Consumption (Consumption being equal to the capital extracted from the investment portfolio at any point in time). Note that since there is no external source of money and since all capital extracted from the investment portfolio at any point in time is immediately consumed, you are never adding capital to your investment portfolio. The growth of the investment portfolio can happen only from growth in the market value of assets in your investment portfolio. Lastly, we assume that the Consumption Utility function is Constant Relative Risk-Aversion (CRRA), which we covered in Chapter **??**.

For ease of exposition, we formalize the problem setting and derive Merton's beautiful analytical solution for the case of $n = 1$ (i.e., only 1 risky asset). The solution generalizes in a straightforward manner to the case of $n > 1$ risky assets, so it pays to keep the notation and explanations simple, emphasizing intuition rather than heavy technical details.

Since we are operating in continuous-time, the risky asset follows a stochastic process (denoted S) - specifically an Ito process (introductory background on Ito processes and Ito's Lemma covered in Appendix **??**), as follows:

$$dS_t = \mu \cdot S_t \cdot dt + \sigma \cdot S_t \cdot dz_t$$

where $\mu \in \mathbb{R}, \sigma \in \mathbb{R}^+$ are fixed constants (note that for n assets, we would instead work with a vector for μ and a matrix for σ).

The riskless asset has no uncertainty associated with it and has a fixed rate of growth in continuous-time, so the valuation of the riskless asset R_t at time t is given by:

$$dR_t = r \cdot R_t \cdot dt$$

Assume $r \in \mathbb{R}$ is a fixed constant, representing the instantaneous riskless growth of money. We denote the consumption of wealth (equal to extraction of money from the investment portfolio) per unit time (at time t) as $c(t, W_t) \geq 0$ to make it clear that the consumption (our decision at any time t) will in general depend on both time t and wealth W_t. Note that we talk about "rate of consumption in time" because consumption is assumed to be continuous in time. As mentioned earlier, we denote wealth at time t as W_t (note that W is a stochastic process too). We assume that $W_t > 0$ for all $t \geq 0$. This is a reasonable assumption to make as it manifests in constraining the consumption (extraction from investment portfolio) to ensure wealth remains positive. We denote the fraction of wealth allocated to the risky asset at time t as $\pi(t, W_t)$. Just like consumption c, risky-asset allocation fraction π is a function of time t and wealth W_t. Since there is only one risky asset, the fraction of wealth allocated to the riskless asset at time t is $1 - \pi(t, W_t)$. Unlike the constraint $c(t, W_t) \geq 0$, $\pi(t, W_t)$ is assumed to be unconstrained. Note that $c(t, W_t)$ and $\pi(t, W_t)$ together constitute the decision (MDP action) at time t. To keep our notation light, we shall write c_t for $c(t, W_t)$ and π_t for $\pi(t, W_t)$, but please do recognize throughout the derivation that both are functions of wealth W_t at time t as well as of time t itself. Finally, we assume that the Utility of Consumption function is defined as:

$$U(x) = \frac{x^{1-\gamma}}{1-\gamma}$$

for a risk-aversion parameter $\gamma \neq 1$. This Utility function is essentially the CRRA Utility function (ignoring the constant term $\frac{-1}{1-\gamma}$) that we covered in Chapter **??** for $\gamma \neq 1$. γ is the Coefficient of CRRA equal to $\frac{-x \cdot U''(x)}{U'(x)}$. We will not cover the case of CRRA Utility function for $\gamma = 1$ (i.e., $U(x) = \log(x)$), but we encourage you to work out the derivation for $U(x) = \log(x)$ as an exercise.

Due to our assumption of no addition of money to our investment portfolio of the risky asset S_t and riskless asset R_t and due to our assumption of no transaction costs of buying/selling any fractional quantities of risky as well as riskless assets, the time-evolution for wealth should be conceptualized as a continuous adjustment of the allocation π_t and continuous extraction from the portfolio (equal to continuous consumption c_t).

Since the value of the risky asset investment at time t is $\pi_t \cdot W_t$, the change in the value of the risky asset investment from time t to time $t + dt$ is:

$$\mu \cdot \pi_t \cdot W_t \cdot dt + \sigma \cdot \pi_t \cdot W_t \cdot dz_t$$

Likewise, since the value of the riskless asset investment at time t is $(1 - \pi_t) \cdot W_t$, the change in the value of the riskless asset investment from time t to time $t + dt$ is:

$$r \cdot (1 - \pi_t) \cdot W_t \cdot dt$$

Therefore, the infinitesimal change in wealth dW_t from time t to time $t + dt$ is given by:

$$dW_t = ((r + \pi_t \cdot (\mu - r)) \cdot W_t - c_t) \cdot dt + \pi_t \cdot \sigma \cdot W_t \cdot dz_t \tag{1.1}$$

Note that this is an Ito process defining the stochastic evolution of wealth.

Our goal is to determine optimal $(\pi(t, W_t), c(t, W_t))$ at any time t to maximize:

$$\mathbb{E}[\int_t^T \frac{e^{-\rho(s-t)} \cdot c_s^{1-\gamma}}{1-\gamma} \cdot ds + \frac{e^{-\rho(T-t)} \cdot B(T) \cdot W_T^{1-\gamma}}{1-\gamma} \mid W_t]$$

where $\rho \geq 0$ is the utility discount rate to account for the fact that future utility of consumption might be less than current utility of consumption, and $B(\cdot)$ is known as the "bequest" function (think of this as the money you will leave for your family when you die at time T). We can solve this problem for arbitrary bequest $B(T)$ but for simplicity, we shall consider $B(T) = \epsilon^\gamma$ where $0 < \epsilon \ll 1$, meaning "no bequest." We require the bequest to be ϵ^γ rather than 0 for technical reasons, that will become apparent later.

We should think of this problem as a continuous-time Stochastic Control problem where the MDP is defined as below:

- The *State* at time t is (t, W_t)
- The *Action* at time t is (π_t, c_t)
- The *Reward* per unit time at time $t < T$ is:

$$U(c_t) = \frac{c_t^{1-\gamma}}{1-\gamma}$$

  and the *Reward* at time T is:

$$B(T) \cdot U(W_T) = \epsilon^\gamma \cdot \frac{W_T^{1-\gamma}}{1-\gamma}$$

The *Return* at time t is the accumulated discounted *Reward*:

$$\int_t^T e^{-\rho(s-t)} \cdot \frac{c_s^{1-\gamma}}{1-\gamma} \cdot ds + \frac{e^{-\rho(T-t)} \cdot \epsilon^\gamma \cdot W_T^{1-\gamma}}{1-\gamma}$$

Our goal is to find the *Policy* : $(t, W_t) \to (\pi_t, c_t)$ that maximizes the *Expected Return*. Note the important constraint that $c_t \geq 0$, but π_t is unconstrained.

Our first step is to write out the Hamilton-Jacobi-Bellman (HJB) Equation (the analog of the Bellman Optimality Equation in continuous-time). We denote the Optimal Value Function as V^* such that the Optimal Value for wealth W_t at time t is $V^*(t, W_t)$. Note that unlike Section **??** in Chapter **??** where we denoted the Optimal Value Function as a time-indexed sequence $V_t^*(\cdot)$, here we make t an explicit functional argument of V^*. This is because in the continuous-time setting, we are interested in the time-differential of the Optimal Value Function. Appendix **??** provides the derivation of the general HJB formulation (Equation (**??**) in Appendix **??**) - this general HJB Equation specializes here to the following:

$$\max_{\pi_t, c_t} \{\mathbb{E}_t[dV^*(t, W_t) + \frac{c_t^{1-\gamma}}{1-\gamma} \cdot dt\} = \rho \cdot V^*(t, W_t) \cdot dt \tag{1.2}$$

Now use Ito's Lemma on dV^*, remove the dz_t term since it's a martingale, and divide throughout by dt to produce the HJB Equation in partial-differential form for any $0 \leq t < T$, as follows (the general form of this transformation appears as Equation (**??**) in Appendix **??**):

$$\max_{\pi_t, c_t} \{\frac{\partial V^*}{\partial t} + \frac{\partial V^*}{\partial W_t} \cdot ((\pi_t(\mu - r) + r)W_t - c_t) + \frac{\partial^2 V^*}{\partial W_t^2} \cdot \frac{\pi_t^2 \cdot \sigma^2 \cdot W_t^2}{2} + \frac{c_t^{1-\gamma}}{1-\gamma}\} = \rho \cdot V^*(t, W_t) \tag{1.3}$$

This HJB Equation is subject to the terminal condition:

$$V^*(T, W_T) = \epsilon^\gamma \cdot \frac{W_T^{1-\gamma}}{1-\gamma}$$

Let us write Equation (1.3) more succinctly as:

$$\max_{\pi_t, c_t} \Phi(t, W_t; \pi_t, c_t) = \rho \cdot V^*(t, W_t) \tag{1.4}$$

It pays to emphasize again that we are working with the constraints $W_t > 0, c_t \geq 0$ for $0 \leq t < T$

To find optimal π_t^*, c_t^*, we take the partial derivatives of $\Phi(t, W_t; \pi_t, c_t)$ with respect to π_t and c_t, and equate to 0 (first-order conditions for Φ). The partial derivative of Φ with respect to π_t is:

$$(\mu - r) \cdot \frac{\partial V^*}{\partial W_t} + \frac{\partial^2 V^*}{\partial W_t^2} \cdot \pi_t \cdot \sigma^2 \cdot W_t = 0$$

$$\Rightarrow \pi_t^* = \frac{-\frac{\partial V^*}{\partial W_t} \cdot (\mu - r)}{\frac{\partial^2 V^*}{\partial W_t^2} \cdot \sigma^2 \cdot W_t} \tag{1.5}$$

The partial derivative of Φ with respect to c_t is:

$$-\frac{\partial V^*}{\partial W_t} + (c_t^*)^{-\gamma} = 0$$

$$\Rightarrow c_t^* = (\frac{\partial V^*}{\partial W_t})^{\frac{-1}{\gamma}} \tag{1.6}$$

Now substitute π_t^* (from Equation (1.5))and c_t^* (from Equation (1.6)) in $\Phi(t, W_t; \pi_t, c_t)$ (in Equation (1.3)) and equate to $\rho \cdot V^*(t, W_t)$. This gives us the Optimal Value Function Partial Differential Equation (PDE):

$$\frac{\partial V^*}{\partial t} - \frac{(\mu - r)^2}{2\sigma^2} \cdot \frac{(\frac{\partial V^*}{\partial W_t})^2}{\frac{\partial^2 V^*}{\partial W_t^2}} + \frac{\partial V^*}{\partial W_t} \cdot r \cdot W_t + \frac{\gamma}{1-\gamma} \cdot (\frac{\partial V^*}{\partial W_t})^{\frac{\gamma-1}{\gamma}} = \rho \cdot V^*(t, W_t) \tag{1.7}$$

The boundary condition for this PDE is:

$$V^*(T, W_T) = \epsilon^\gamma \cdot \frac{W_T^{1-\gamma}}{1-\gamma}$$

The second-order conditions for Φ are satisfied under the assumptions: $c_t^* > 0, W_t > 0$, $\frac{\partial^2 V^*}{\partial W_t^2} < 0$ for all $0 \leq t < T$ (we will later show that these are all satisfied in the solution we derive), and for concave $U(\cdot)$, i.e., $\gamma > 0$

Next, we want to reduce the PDE (1.7) to an Ordinary Differential Equation (ODE) so we can solve the (simpler) ODE. Towards this goal, we surmise with a guess solution in terms of a deterministic function (f) of time:

$$V^*(t, W_t) = f(t)^\gamma \cdot \frac{W_t^{1-\gamma}}{1-\gamma} \tag{1.8}$$

Then,

$$\frac{\partial V^*}{\partial t} = \gamma \cdot f(t)^{\gamma-1} \cdot f'(t) \cdot \frac{W_t^{1-\gamma}}{1-\gamma} \tag{1.9}$$

$$\frac{\partial V^*}{\partial W_t} = f(t)^\gamma \cdot W_t^{-\gamma} \tag{1.10}$$

$$\frac{\partial^2 V^*}{\partial W_t^2} = -f(t)^\gamma \cdot \gamma \cdot W_t^{-\gamma-1} \tag{1.11}$$

Substituting the guess solution in the PDE, we get the simple ODE:

$$f'(t) = \nu \cdot f(t) - 1 \tag{1.12}$$

where

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

We note that that the bequest function $B(T) = \epsilon^\gamma$ proves to be convenient in order to fit the guess solution for $t = T$. This means the boundary condition for this ODE is: $f(T) = \epsilon$. Consequently, this ODE together with this boundary condition has a simple enough solution, as follows:

$$f(t) = \begin{cases} \frac{1+(\nu\epsilon-1)\cdot e^{-\nu(T-t)}}{\nu} & \text{for } \nu \neq 0 \\ T - t + \epsilon & \text{for } \nu = 0 \end{cases} \tag{1.13}$$

Substituting V^* (from Equation (1.8)) and its partial derivatives (from Equations (1.9), (1.10) and (1.11)) in Equations (1.5) and (1.6), we get:

$$\pi^*(t, W_t) = \frac{\mu - r}{\sigma^2\gamma} \tag{1.14}$$

$$c^*(t, W_t) = \frac{W_t}{f(t)} = \begin{cases} \frac{\nu \cdot W_t}{1+(\nu\epsilon-1)\cdot e^{-\nu(T-t)}} & \text{for } \nu \neq 0 \\ \frac{W_t}{T-t+\epsilon} & \text{for } \nu = 0 \end{cases} \tag{1.15}$$

Finally, substituting the solution for $f(t)$ (Equation (1.13)) in Equation (1.8), we get:

$$V^*(t, W_t) = \begin{cases} \frac{(1+(\nu\epsilon-1)\cdot e^{-\nu(T-t)})^\gamma}{\nu^\gamma} \cdot \frac{W_t^{1-\gamma}}{1-\gamma} & \text{for } \nu \neq 0 \\ \frac{(T-t+\epsilon)^\gamma \cdot W_t^{1-\gamma}}{1-\gamma} & \text{for } \nu = 0 \end{cases} \tag{1.16}$$

Note that $f(t) > 0$ for all $0 \leq t < T$ (for all ν) ensures \$W_t > 0, $c_t^* > 0$, $\frac{\partial^2 V^*}{\partial W_t^2} < 0$. This ensures the constraints $W_t > 0$ and $c_t \geq 0$ are satisfied and the second-order conditions for Φ are also satisfied. A very important lesson in solving Merton's Portfolio problem is the fact that the HJB Formulation is key and that this solution approach provides a template for similar continuous-time stochastic control problems.

1.3 Developing Intuition for the Solution to Merton's Portfolio Problem

The solution for $\pi^*(t, W_t)$ and $c^*(t, W_t)$ are surprisingly simple. $\pi^*(t, W_t)$ is a constant, i.e., it is independent of both of the state variables t and W_t. This means that no matter what wealth we carry and no matter how close we are to the end of the horizon (i.e., no matter what our age is), we should invest the same fraction of our wealth in the risky asset (likewise for the case of n risky assets). The simplifying assumptions in Merton's Portfolio problem statement did play a part in the simplicity of the solution, but the fact that $\pi^*(t, W_t)$ is a constant is still rather surprising. The simplicity of the solution means

that asset allocation is straightforward - we just need to keep re-balancing to maintain this constant fraction of our wealth in the risky asset. We expect our wealth to grow over time and so, the capital in the risky asset would also grow proportionately.

The form of the solution for $c^*(t, W_t)$ is extremely intuitive - the excess return of the risky asset $(\mu - r)$ shows up in the numerator, which makes sense, since one would expect to invest a higher fraction of one's wealth in the risky asset if it gives us a higher excess return. It also makes sense that the volatility σ of the risky asset (squared) shows up in the denominator (the greater the volatility, the less we'd allocate to the risky asset, since we are typically risk-averse, i.e., $\gamma > 0$). Likewise, it makes since that the coefficient of CRRA γ shows up in the denominator since a more risk-averse individual (greater value of γ) will want to invest less in the risky asset.

The Optimal Consumption Rate $c^*(t, W_t)$ should be conceptualized in terms of the *Optimal Fractional Consumption Rate*, i.e., the Optimal Consumption Rate $c^*(t, W_t)$ as a fraction of the Wealth W_t. Note that the Optimal Fractional Consumption Rate depends only on t (it is equal to $\frac{1}{f(t)}$). This means no matter what our wealth is, we should be extracting a fraction of our wealth on a daily/monthly/yearly basis that is only dependent on our age. Note also that if $\epsilon < \frac{1}{\nu}$, the Optimal Fractional Consumption Rate increases as time progresses. This makes intuitive sense because when we have many more years to live, we'd want to consume less and invest more to give the portfolio more ability to grow, and when we get close to our death, we increase our consumption (since the optimal is "to die broke," assuming no bequest).

Now let us understand how the Wealth process evolves. Let us substitute for $\pi^*(t, W_t)$ (from Equation (1.14)) and $c^*(t, W_t)$ (from Equation (1.15)) in the Wealth process defined in Equation (1.1). This yields the following Wealth process W^* when we asset-allocate optimally and consume optimally:

$$dW_t^* = (r + \frac{(\mu - r)^2}{\sigma^2 \gamma} - \frac{1}{f(t)}) \cdot W_t^* \cdot dt + \frac{\mu - r}{\sigma \gamma} \cdot W_t^* \cdot dz_t \tag{1.17}$$

The first thing to note about this Wealth process is that it is a lognormal process of the form covered in Section **??** of Appendix **??**. The lognormal volatility (fractional dispersion) of this wealth process is constant $(= \frac{\mu - r}{\sigma \gamma})$. The lognormal drift (fractional drift) is independent of the wealth but is dependent on time $(= r + \frac{(\mu - r)^2}{\sigma^2 \gamma} - \frac{1}{f(t)})$. From the solution of the general lognormal process derived in Section **??** of Appendix **??**, we conclude that:

$$\mathbb{E}[W_t^*] = W_0 \cdot e^{(r + \frac{(\mu - r)^2}{\sigma^2 \gamma})t} \cdot e^{-\int_0^t \frac{du}{f(u)}} = \begin{cases} W_0 \cdot e^{(r + \frac{(\mu - r)^2}{\sigma^2 \gamma})t} \cdot (1 - \frac{1 - e^{-\nu t}}{1 + (\nu \epsilon - 1) \cdot e^{-\nu T}}) & \text{if } \nu \neq 0 \\ W_0 \cdot e^{(r + \frac{(\mu - r)^2}{\sigma^2 \gamma})t} \cdot (1 - \frac{t}{T + \epsilon}) & \text{if } \nu = 0 \end{cases} \tag{1.18}$$

Since we assume no bequest, we should expect the Wealth process to keep growing up to some point in time and then fall all the way down to 0 when time runs out (i.e., when $t = T$). We shall soon write the code for Equation (1.18) and plot the graph for this rise and fall. An important point to note is that although the wealth process growth varies in time (expected wealth growth rate = $r + \frac{(\mu - r)^2}{\sigma^2 \gamma} - \frac{1}{f(t)}$ as seen from Equation (1.17)), the variation (in time) of the wealth process growth is only due to the fractional consumption rate varying in time. If we ignore the fractional consumption rate $(= \frac{1}{f(t)})$, then what we get is the Expected Portfolio Annual Return of $r + \frac{(\mu - r)^2}{\sigma^2 \gamma}$ which is a constant (does not depend on either time t or on Wealth W_t^*). Now let us write some code to calculate

the time-trajectories of Expected Wealth, Fractional Consumption Rate, Expected Wealth Growth Rate and Expected Portfolio Annual Return.

The code should be pretty self-explanatory. We will just provide a few explanations of variables in the code that may not be entirely obvious: `portfolio_return` calculates the Expected Portfolio Annual Return, `nu` calculates the value of ν, `f` represents the function $f(t)$, `wealth_growth_rate` calculates the Expected Wealth Growth Rate as a function of time t. The `expected_wealth` method assumes $W_0 = 1$.

```
@dataclass(frozen=True)
class MertonPortfolio:
    mu: float
    sigma: float
    r: float
    rho: float
    horizon: float
    gamma: float
    epsilon: float = 1e-6

    def excess(self) -> float:
        return self.mu - self.r

    def variance(self) -> float:
        return self.sigma * self.sigma

    def allocation(self) -> float:
        return self.excess() / (self.gamma * self.variance())

    def portfolio_return(self) -> float:
        return self.r + self.allocation() * self.excess()

    def nu(self) -> float:
        return (self.rho - (1 - self.gamma) * self.portfolio_return()) / \
            self.gamma

    def f(self, time: float) -> float:
        remaining: float = self.horizon - time
        nu = self.nu()
        if nu == 0:
            ret = remaining + self.epsilon
        else:
            ret = (1 + (nu * self.epsilon - 1) * exp(-nu * remaining)) / nu
        return ret

    def fractional_consumption_rate(self, time: float) -> float:
        return 1 / self.f(time)

    def wealth_growth_rate(self, time: float) -> float:
        return self.portfolio_return() - self.fractional_consumption_rate(time)

    def expected_wealth(self, time: float) -> float:
        base: float = exp(self.portfolio_return() * time)
        nu = self.nu()
        if nu == 0:
            ret = base * (1 - (1 - exp(-nu * time)) /
                          (1 + (nu * self.epsilon - 1) *
                           exp(-nu * self.horizon)))
        else:
            ret = base * (1 - time / (self.horizon + self.epsilon))
        return ret
```

The above code is in the file rl/chapter7/merton_solution_graph.py. We highly encourage you to experiment by changing the various inputs in this code $(T, \mu, \sigma, r, \rho, \gamma)$ and visualize how the results change. Doing this will help build tremendous intuition.

A rather interesting observation is that if $r + \frac{(\mu-r)^2}{\sigma^2\gamma} > \frac{1}{f(0)}$ and $\epsilon < \frac{1}{\nu}$, then the Fractional Consumption Rate is initially less than the Expected Portfolio Annual Return and

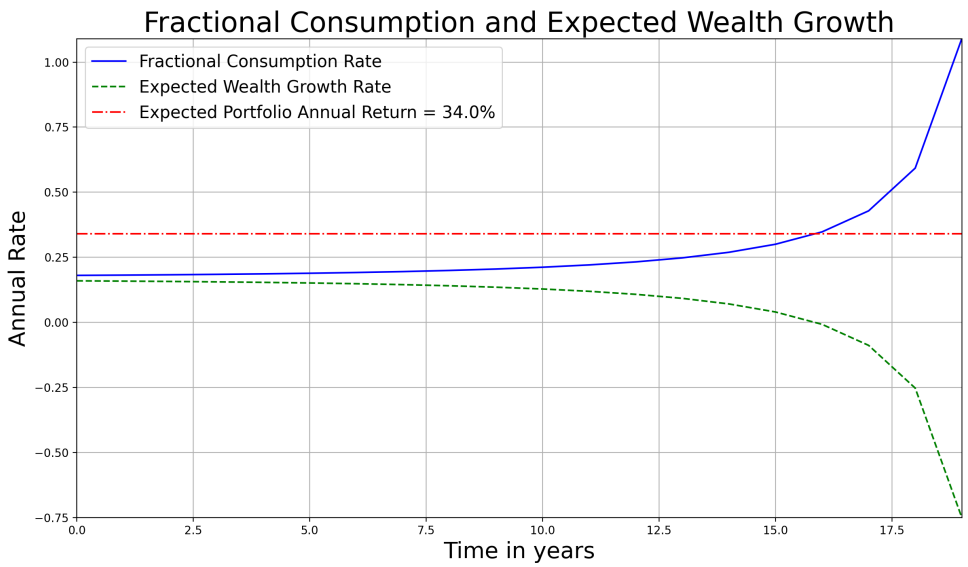

Figure 1.1: Portfolio Return and Consumption Rate

over time, the Fractional Consumption Rate becomes greater than the Expected Portfolio Annual Return. This illustrates how the optimal behavior is to consume modestly and invest more when one is younger, then to gradually increase the consumption as one ages, and finally to ramp up the consumption sharply when one is close to the end of one's life. Figure 1.1 shows the visual for this (along with the Expected Wealth Growth Rate) using the above code for input values of: $T = 20, \mu = 10\%, \sigma = 10\%, r = 2\%, \rho = 1\%, \gamma = 2.0$.

Figure 1.2 shows the time-trajectory of the expected wealth based on Equation (1.18) for the same input values as listed above. Notice how the Expected Wealth rises in a convex shape for several years since the consumption during all these years is quite modest, and then the shape of the Expected Wealth curve turns concave at about 12 years, peaks at about 16 years (when Fractional Consumption Rate rises to equal Expected Portfolio Annual Return), and then falls precipitously in the last couple of years (as the Consumption increasingly drains the Wealth down to 0).

1.4 A Discrete-Time Asset-Allocation Example

In this section, we cover a discrete-time version of the problem that lends itself to analytical tractability, much like Merton's Portfolio Problem in continuous-time. We are given wealth W_0 at time 0. At each of discrete time steps labeled $t = 0, 1, \ldots, T-1$, we are allowed to allocate the wealth W_t at time t to a portfolio of a risky asset and a riskless asset in an unconstrained manner with no transaction costs. The risky asset yields a random return $\sim \mathcal{N}(\mu, \sigma^2)$ over each single time step (for a given $\mu \in \mathbb{R}$ and a given $\sigma \in \mathbb{R}^+$). The riskless asset yields a constant return denoted by r over each single time step (for a given $r \in \mathbb{R}$). We assume that there is no consumption of wealth at any time $t < T$, and that we liquidate and consume the wealth W_T at time T. So our goal is simply to maximize the Expected Utility of Wealth at the final time step $t = T$ by dynamically allocating $x_t \in \mathbb{R}$ in the risky asset and the remaining $W_t - x_t$ in the riskless asset for each $t = 0, 1, \ldots, T-1$. Assume the single-time-step discount factor is γ and that the Utility of Wealth at the final time step

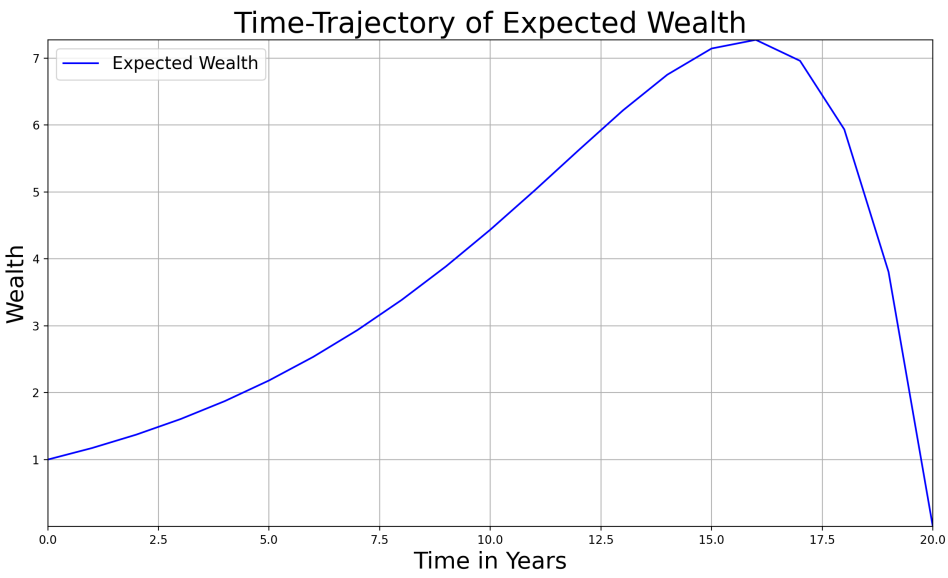

Figure 1.2: Expected Wealth Time-Trajectory

$t = T$ is given by the following CARA function:

$$U(W_T) = \frac{1 - e^{-aW_T}}{a} \text{ for some fixed } a \neq 0$$

Thus, the problem is to maximize, for each $t = 0, 1, \ldots, T-1$, over choices of $x_t \in \mathbb{R}$, the value:

$$\mathbb{E}[\gamma^{T-t} \cdot \frac{1 - e^{-aW_T}}{a} | (t, W_t)]$$

Since γ^{T-t} and a are constants, this is equivalent to maximizing, for each $t = 0, 1, \ldots, T-1$, over choices of $x_t \in \mathbb{R}$, the value:

$$\mathbb{E}[\frac{-e^{-aW_T}}{a} | (t, W_t)] \tag{1.19}$$

We formulate this problem as a *Continuous States* and *Continuous Actions* discrete-time finite-horizon MDP by specifying it's *State Transitions*, *Rewards* and *Discount Factor* precisely. The problem then is to solve the MDP's Control problem to find the Optimal Policy.

The terminal time for the finite-horizon MDP is T and hence, all the states at time $t = T$ are terminal states. We shall follow the notation of finite-horizon MDPs that we had covered in Section **??** of Chapter **??**. The *State* $s_t \in \mathcal{S}_t$ at any time step $t = 0, 1, \ldots, T$ consists of the wealth W_t. The decision (*Action*) $a_t \in \mathcal{A}_t$ at any time step $t = 0, 1, \ldots, T-1$ is the quantity of investment in the risky asset ($= x_t$). Hence, the quantity of investment in the riskless asset at time t will be $W_t - x_t$. A deterministic policy at time t (for all $t = 0, 1, \ldots T-1$) is denoted as π_t, and hence, we write: $\pi_t(W_t) = x_t$. Likewise, an optimal deterministic policy at time t (for all $t = 0, 1, \ldots, T-1$) is denoted as π_t^*, and hence, we write: $\pi_t^*(W_t) = x_t^*$.

Denote the random variable for the single-time-step return of the risky asset from time t to time $t+1$ as $Y_t \sim \mathcal{N}(\mu, \sigma^2)$ for all $t = 0, 1, \ldots T-1$. So,

$$W_{t+1} = x_t \cdot (1 + Y_t) + (W_t - x_t) \cdot (1 + r) = x_t \cdot (Y_t - r) + W_t \cdot (1 + r) \tag{1.20}$$

for all $t = 0, 1, \ldots, T - 1$.

The MDP *Reward* is 0 for all $t = 0, 1, \ldots, T - 1$. As a result of the simplified objective (1.19) above, the MDP *Reward* for $t = T$ is the following random quantity:

$$\frac{-e^{-aW_T}}{a}$$

We set the MDP discount factor to be $\gamma = 1$ (again, because of the simplified objective (1.19) above).

We denote the Value Function at time t (for all $t = 0, 1, \ldots, T - 1$) for a given policy $\pi = (\pi_0, \pi_1, \ldots, \pi_{T-1})$ as:

$$V_t^{\pi}(W_t) = \mathbb{E}_{\pi}[\frac{-e^{-aW_T}}{a} | (t, W_t)]$$

We denote the Optimal Value Function at time t (for all $t = 0, 1, \ldots, T - 1$) as:

$$V_t^*(W_t) = \max_{\pi} V_t^{\pi}(W_t) = \max_{\pi} \{\mathbb{E}_{\pi}[\frac{-e^{-aW_T}}{a} | (t, W_t)]\}$$

The Bellman Optimality Equation is:

$$V_t^*(W_t) = \max_{x_t} Q_t^*(W_t, x_t) = \max_{x_t} \{\mathbb{E}_{Y_t \sim \mathcal{N}(\mu, \sigma^2)}[V_{t+1}^*(W_{t+1})]\}$$

for all $t = 0, 1, \ldots, T - 2$, and

$$V_{T-1}^*(W_{T-1}) = \max_{x_{T-1}} Q_{T-1}^*(W_{T-1}, x_{T-1}) = \max_{x_{T-1}} \{\mathbb{E}_{Y_{T-1} \sim \mathcal{N}(\mu, \sigma^2)}[\frac{-e^{-aW_T}}{a}]\}$$

where Q_t^* is the Optimal Action-Value Function at time t for all $t = 0, 1, \ldots, T - 1$.

We make an educated guess for the functional form of the Optimal Value Function as:

$$V_t^*(W_t) = -b_t \cdot e^{-c_t \cdot W_t} \tag{1.21}$$

where b_t, c_t are independent of the wealth W_t for all $t = 0, 1, \ldots, T-1$. Next, we express the Bellman Optimality Equation using this functional form for the Optimal Value Function:

$$V_t^*(W_t) = \max_{x_t} \{\mathbb{E}_{Y_t \sim \mathcal{N}(\mu, \sigma^2)}[-b_{t+1} \cdot e^{-c_{t+1} \cdot W_{t+1}}]\}$$

Using Equation (1.20), we can write this as:

$$V_t^*(W_t) = \max_{x_t} \{\mathbb{E}_{Y_t \sim \mathcal{N}(\mu, \sigma^2)}[-b_{t+1} \cdot e^{-c_{t+1} \cdot (x_t \cdot (Y_t - r) + W_t \cdot (1 + r))}]\}$$

The expectation of this exponential form (under the normal distribution) evaluates to:

$$V_t^*(W_t) = \max_{x_t} \{-b_{t+1} \cdot e^{-c_{t+1} \cdot (1+r) \cdot W_t - c_{t+1} \cdot (\mu - r) \cdot x_t + c_{t+1}^2 \cdot \frac{\sigma^2}{2} \cdot x_t^2}\} \tag{1.22}$$

Since $V_t^*(W_t) = \max_{x_t} Q_t^*(W_t, x_t)$, from Equation (1.22), we can infer the functional form for $Q_t^*(W_t, x_t)$ in terms of b_{t+1} and c_{t+1}:

$$Q_t^*(W_t, x_t) = -b_{t+1} \cdot e^{-c_{t+1} \cdot (1+r) \cdot W_t - c_{t+1} \cdot (\mu - r) \cdot x_t + c_{t+1}^2 \cdot \frac{\sigma^2}{2} \cdot x_t^2} \tag{1.23}$$

Since the right-hand-side of the Bellman Optimality Equation (1.22) involves a max over x_t, we can say that the partial derivative of the term inside the max with respect to x_t is 0. This enables us to write the Optimal Allocation x_t^* in terms of c_{t+1}, as follows:

$$-c_{t+1} \cdot (\mu - r) + \sigma^2 \cdot c_{t+1}^2 \cdot x_t^* = 0$$

$$\Rightarrow x_t^* = \frac{\mu - r}{\sigma^2 \cdot c_{t+1}} \tag{1.24}$$

Next we substitute this maximizing x_t^* in the Bellman Optimality Equation (Equation (1.22)):

$$V_t^*(W_t) = -b_{t+1} \cdot e^{-c_{t+1} \cdot (1+r) \cdot W_t - \frac{(\mu-r)^2}{2\sigma^2}}$$

But since

$$V_t^*(W_t) = -b_t \cdot e^{-c_t \cdot W_t}$$

we can write the following recursive equations for b_t and c_t:

$$b_t = b_{t+1} \cdot e^{-\frac{(\mu-r)^2}{2\sigma^2}}$$

$$c_t = c_{t+1} \cdot (1 + r)$$

We can calculate b_{T-1} and c_{T-1} from the knowledge of the MDP *Reward* $\frac{-e^{-aW_T}}{a}$ (Utility of Terminal Wealth) at time $t = T$, which will enable us to unroll the above recursions for b_t and c_t for all $t = 0, 1, \ldots, T-2$.

$$V_{T-1}^*(W_{T-1}) = \max_{x_{T-1}} \{\mathbb{E}_{Y_{T-1} \sim \mathcal{N}(\mu, \sigma^2)} [\frac{-e^{-aW_T}}{a}]\}$$

From Equation (1.20), we can write this as:

$$V_{T-1}^*(W_{T-1}) = \max_{x_{T-1}} \{\mathbb{E}_{Y_{T-1} \sim \mathcal{N}(\mu, \sigma^2)} [\frac{-e^{-a(x_{T-1} \cdot (Y_{T-1} - r) + W_{T-1} \cdot (1+r))}}{a}]\}$$

Using the result in Equation (**??**) in Appendix **??**, we can write this as:

$$V_{T-1}^*(W_{T-1}) = \frac{-e^{-\frac{(\mu-r)^2}{2\sigma^2} - a \cdot (1+r) \cdot W_{T-1}}}{a}$$

Therefore,

$$b_{T-1} = \frac{e^{-\frac{(\mu-r)^2}{2\sigma^2}}}{a}$$

$$c_{T-1} = a \cdot (1 + r)$$

Now we can unroll the above recursions for b_t and c_t for all $t = 0, 1, \ldots T-2$ as:

$$b_t = \frac{e^{-\frac{(\mu-r)^2 \cdot (T-t)}{2\sigma^2}}}{a}$$

$$c_t = a \cdot (1 + r)^{T-t}$$

Substituting the solution for c_{t+1} in Equation (1.24) gives us the solution for the Optimal Policy:

$$\pi_t^*(W_t) = x_t^* = \frac{\mu - r}{\sigma^2 \cdot a \cdot (1+r)^{T-t-1}} \tag{1.25}$$

for all $t = 0, 1, \ldots, T-1$. Note that the optimal action at time step t (for all $t = 0, 1, \ldots, T-1$) does not depend on the state W_t at time t (it only depends on the time t). Hence, the optimal policy $\pi_t^*(\cdot)$ for a fixed time t is a constant deterministic policy function.

Substituting the solutions for b_t and c_t in Equation (1.21) gives us the solution for the Optimal Value Function:

$$V_t^*(W_t) = \frac{-e^{-\frac{(\mu-r)^2(T-t)}{2\sigma^2}}}{a} \cdot e^{-a(1+r)^{T-t} \cdot W_t} \tag{1.26}$$

for all $t = 0, 1, \ldots, T-1$.

Substituting the solutions for b_{t+1} and c_{t+1} in Equation (1.23) gives us the solution for the Optimal Action-Value Function:

$$Q_t^*(W_t, x_t) = \frac{-e^{-\frac{(\mu-r)^2(T-t-1)}{2\sigma^2}}}{a} \cdot e^{-a(1+r)^{T-t} \cdot W_t - a(\mu-r)(1+r)^{T-t-1} \cdot x_t + \frac{(a\sigma(1+r)^{T-t-1})^2}{2} \cdot x_t^2} \tag{1.27}$$

for all $t = 0, 1, \ldots, T-1$.

1.5 Porting to Real-World

We have covered a continuous-time setting and a discrete-time setting with simplifying assumptions that provide analytical tractability. The specific simplifying assumptions that enabled analytical tractability were:

- Normal distribution of asset returns
- CRRA/CARA assumptions
- Frictionless markets/trading (no transaction costs, unconstrained and continuous prices/allocation amounts/consumption)

But real-world problems involving dynamic asset-allocation and consumption are not so simple and clean. We have arbitrary, more complex asset price movements. Utility functions don't fit into simple CRRA/CARA formulas. In practice, trading often occurs in discrete space - asset prices, allocation amounts and consumption are often discrete quantities. Moreover, when we change our asset allocations or liquidate a portion of our portfolio to consume, we incur transaction costs. Furthermore, trading doesn't always happen in continuous-time - there are typically specific windows of time where one is locked-out from trading or there are trading restrictions. Lastly, many investments are illiquid (eg: real-estate) or simply not allowed to be liquidated until a certain horizon (eg: retirement funds), which poses major constraints on extracting money from one's portfolio for consumption. So even though prices/allocation amounts/consumption might be close to being continuous-variables, the other above-mentioned frictions mean that we don't get the benefits of calculus that we obtained in the simple examples we covered.

With the above real-world considerations, we need to tap into Dynamic Programming - more specifically, Approximate Dynamic Programming since real-world problems have large state spaces and large action spaces (even if these spaces are not continuous, they

tend to be close to continuous). Appropriate function approximation of the Value Function is key to solving these problems. Implementing a full-blown real-world investment and consumption management system is beyond the scope of this book, but let us implement an illustrative example that provides sufficient understanding of how a full-blown real-world example would be implemented. We have to keep things simple enough and yet sufficiently general. So here is the setting we will implement for:

- One risky asset and one riskless asset.
- Finite number of time steps (discrete-time setting akin to Section 1.4).
- No consumption (i.e., no extraction from the investment portfolio) until the end of the finite horizon, and hence, without loss of generality, we set the discount factor equal to 1.
- Arbitrary distribution of return for the risky asset, allowing the distribution of returns to vary over time (`risky_return_distributions: Sequence[Distribution[float]]` in the code below).
- The return on the riskless asset, varying in time (`riskless_returns: Sequence[float]` in the code below).
- Arbitrary Utility Function (`utility_func: Callable[[float], float]` in the code below).
- Finite number of choices of investment amounts in the risky asset at each time step (`risky_alloc_choices: Sequence[float]` in the code below).
- Arbitrary distribution of initial wealth W_0 (`initial_wealth_distribution: Distribution[float]` in the code below).

The code in the class `AssetAllocDiscrete` below is fairly self-explanatory. We use the function `back_opt_qvf` covered in Section **??** of Chapter **??** to perform backward induction on the optimal Q-Value Function. Since the state space is continuous, the optimal Q-Value Function is represented as a `QValueFunctionApprox` (specifically, as a `DNNApprox`). Moreover, since we are working with a generic distribution of returns that govern the state transitions of this MDP, we need to work with the methods of the abstract class `MarkovDecisionProcess` (and not the class `FiniteMarkovDecisionProcess`). The method `backward_induction_qvf` below makes the call to `back_opt_qvf`. Since the risky returns distribution is arbitrary and since the utility function is arbitrary, we don't have prior knowledge of the functional form of the Q-Value function. Hence, the user of the class `AssetAllocDiscrete` also needs to provide the set of feature functions (`feature_functions` in the code below) and the specification of a deep neural network to represent the Q-Value function (`dnn_spec` in the code below). The rest of the code below is mainly about preparing the input `mdp_f0_mu_triples` to be passed to `back_opt_qvf`. As was explained in Section **??** of Chapter **??**, `mdp_f0_mu_triples` is a sequence (for each time step) of the following triples:

- A `MarkovDecisionProcess[float, float]` object, which in the code below is prepared by the method `get_mdp`. *State* is the portfolio wealth (`float` type) and *Action* is the quantity of investment in the risky asset (also of `float` type). `get_mdp` creates a class `AssetAllocMDP` that implements the abstract class `MarkovDecisionProcess`. To do so, we need to implement the `step` method and the `actions` method. The `step` method returns an instance of `SampledDistribution`, which is based on the `sr_sampler_func` that returns a sample of the pair of next state (next time step's wealth) and reward, given the current state (current wealth) and action (current time step's quantity of investment in the risky asset).

- A `QValueFunctionApprox[float], float]` object, prepared by `get_qvf_func_approx`. This method sets up a `DNNApprox[Tuple[NonTerminal[float], float]]` object that represents a neural-network function approximation for the optimal Q-Value Function. So the input to this neural network would be a `Tuple[NonTerminal[float], float]` representing a (state, action) pair.
- An `NTStateDistribution[float]` object prepared by `get_states_distribution`, which returns a `SampledDistribution[NonTerminal[float]]` representing the distribution of non-terminal states (distribution of portfolio wealth) at each time step.

The `SampledDistribution[NonTerminal[float]]` is prepared by `states_sampler_func` that generates a sampling trace by sampling the state-transitions (portfolio wealth transitions) from time 0 to the given time step in a time-incremental manner (invoking the `sample` method of the risky asset's return `Distributions` and the `sample` method of a uniform distribution over the action choices specified by `risky_alloc_choices`).

```
from rl.distribution import Distribution, SampledDistribution, Choose
from rl.function_approx import DNNSpec, AdamGradient, DNNApprox
from rl.approximate_dynamic_programming import back_opt_qvf, QValueFunctionApprox
from operator import itemgetter
import numpy as np

@dataclass(frozen=True)
class AssetAllocDiscrete:
    risky_return_distributions: Sequence[Distribution[float]]
    riskless_returns: Sequence[float]
    utility_func: Callable[[float], float]
    risky_alloc_choices: Sequence[float]
    feature_functions: Sequence[Callable[[Tuple[float, float]], float]]
    dnn_spec: DNNSpec
    initial_wealth_distribution: Distribution[float]

    def time_steps(self) -> int:
        return len(self.risky_return_distributions)

    def uniform_actions(self) -> Choose[float]:
        return Choose(self.risky_alloc_choices)

    def get_mdp(self, t: int) -> MarkovDecisionProcess[float, float]:
        distr: Distribution[float] = self.risky_return_distributions[t]
        rate: float = self.riskless_returns[t]
        alloc_choices: Sequence[float] = self.risky_alloc_choices
        steps: int = self.time_steps()
        utility_f: Callable[[float], float] = self.utility_func

        class AssetAllocMDP(MarkovDecisionProcess[float, float]):

            def step(
                self,
                wealth: NonTerminal[float],
                alloc: float
            ) -> SampledDistribution[Tuple[State[float], float]]:

                def sr_sampler_func(
                    wealth=wealth,
                    alloc=alloc
                ) -> Tuple[State[float], float]:
                    next_wealth: float = alloc * (1 + distr.sample()) \
                        + (wealth.state - alloc) * (1 + rate)
                    reward: float = utility_f(next_wealth) \
                        if t == steps - 1 else 0.
                    next_state: State[float] = Terminal(next_wealth) \
                        if t == steps - 1 else NonTerminal(next_wealth)
                    return (next_state, reward)

                return SampledDistribution(
```

```
                sampler=sr_sampler_func,
                expectation_samples=1000
            )

        def actions(self, wealth: NonTerminal[float]) -> Sequence[float]:
            return alloc_choices

    return AssetAllocMDP()

def get_qvf_func_approx(self) -> \
        DNNApprox[Tuple[NonTerminal[float], float]]:

    adam_gradient: AdamGradient = AdamGradient(
        learning_rate=0.1,
        decay1=0.9,
        decay2=0.999
    )
    ffs: List[Callable[[Tuple[NonTerminal[float], float]], float]] = []
    for f in self.feature_functions:
        def this_f(pair: Tuple[NonTerminal[float], float], f=f) -> float:
            return f((pair[0].state, pair[1]))
        ffs.append(this_f)

    return DNNApprox.create(
        feature_functions=ffs,
        dnn_spec=self.dnn_spec,
        adam_gradient=adam_gradient
    )

def get_states_distribution(self, t: int) -> \
        SampledDistribution[NonTerminal[float]]:

    actions_distr: Choose[float] = self.uniform_actions()

    def states_sampler_func() -> NonTerminal[float]:
        wealth: float = self.initial_wealth_distribution.sample()
        for i in range(t):
            distr: Distribution[float] = self.risky_return_distributions[i]
            rate: float = self.riskless_returns[i]
            alloc: float = actions_distr.sample()
            wealth = alloc * (1 + distr.sample()) + \
                (wealth - alloc) * (1 + rate)
        return NonTerminal(wealth)

    return SampledDistribution(states_sampler_func)

def backward_induction_qvf(self) -> \
        Iterator[QValueFunctionApprox[float, float]]:

    init_fa: DNNApprox[Tuple[NonTerminal[float], float]] = \
        self.get_qvf_func_approx()

    mdp_f0_mu_triples: Sequence[Tuple[
        MarkovDecisionProcess[float, float],
        DNNApprox[Tuple[NonTerminal[float], float]],
        SampledDistribution[NonTerminal[float]]
    ]] = [(
        self.get_mdp(i),
        init_fa,
        self.get_states_distribution(i)
    ) for i in range(self.time_steps())]

    num_state_samples: int = 300
    error_tolerance: float = 1e-6

    return back_opt_qvf(
        mdp_f0_mu_triples=mdp_f0_mu_triples,
        gamma=1.0,
        num_state_samples=num_state_samples,
        error_tolerance=error_tolerance
    )
```

The above code is in the file rl/chapter7/asset_alloc_discrete.py. We encourage you to create a few different instances of `AssetAllocDiscrete` by varying it's inputs (try different return distributions, different utility functions, different action spaces). But how do we know the code above is correct? We need a way to test it. A good test is to specialize the inputs to fit the setting of Section 1.4 for which we have a closed-form solution to compare against. So let us write some code to specialize the inputs to fit this setting. Since the above code has been written with an educational motivation rather than an efficient-computation motivation, the convergence of the backward induction ADP algorithm is going to be slow. So we shall test it on a small number of time steps and provide some assistance for fast convergence (using limited knowledge from the closed-form solution in specifying the function approximation). We write code below to create an instance of `AssetAllocDiscrete` with time steps $T = 4$, $\mu = 13\%, \sigma = 20\%, r = 7\%$, coefficient of CARA $a = 1.0$. We set up `risky_return_distributions` as a sequence of identical `Gaussian` distributions, `riskless_returns` as a sequence of identical riskless rate of returns, and `utility_func` as a `lambda` parameterized by the coefficient of CARA a. We know from the closed-form solution that the optimal allocation to the risky asset for each of time steps $t = 0, 1, 2, 3$ is given by:

$$x_t^* = \frac{1.5}{1.07^{4-t}}$$

Therefore, we set `risky_alloc_choices` (action choices) in the range [1.0, 2.0] in increments of 0.1 to see if our code can hit the correct values within the 0.1 granularity of action choices.

To specify `feature_functions` and `dnn_spec`, we need to leverage the functional form of the closed-form solution for the Action-Value function (i.e., Equation (1.27)). We observe that we can write this as:

$$Q_t^*(W_t, x_t) = -sign(a) \cdot e^{-(\alpha_0 + \alpha_1 \cdot W_t + \alpha_2 \cdot x_t + \alpha_3 \cdot x_t^2)}$$

where

$$\alpha_0 = \frac{(\mu - r)^2 (T - t - 1)}{2\sigma^2} + \log(|a|)$$
$$\alpha_1 = a(1 + r)^{T-t}$$
$$\alpha_2 = a(\mu - r)(1 + r)^{T-t-1}$$
$$\alpha_3 = -\frac{(a\sigma(1 + r)^{T-t-1})^2}{2}$$

This means, the function approximation for Q_t^* can be set up with a neural network with no hidden layers, with the output layer activation function as $g(S) = -sign(a) \cdot e^{-S}$, and with the feature functions as:

$$\phi_1((W_t, x_t)) = 1$$
$$\phi_2((W_t, x_t)) = W_t$$
$$\phi_3((W_t, x_t)) = x_t$$
$$\phi_4((W_t, x_t)) = x_t^2$$

We set `initial_wealth_distribution` to be a normal distribution with a mean of `init_wealth` (set equal to 1.0 below) and a standard distribution of `init_wealth_stdev` (set equal to a small value of 0.1 below).

```
from rl.distribution import Gaussian

steps: int = 4
mu: float = 0.13
sigma: float = 0.2
r: float = 0.07
a: float = 1.0
init_wealth: float = 1.0
init_wealth_stdev: float = 0.1

excess: float = mu - r
var: float = sigma * sigma
base_alloc: float = excess / (a * var)

risky_ret: Sequence[Gaussian] = [Gaussian(mu=mu, sigma=sigma)
                                 for _ in range(steps)]
riskless_ret: Sequence[float] = [r for _ in range(steps)]
utility_function: Callable[[float], float] = lambda x: - np.exp(-a * x) / a
alloc_choices: Sequence[float] = np.linspace(
    2 / 3 * base_alloc,
    4 / 3 * base_alloc,
    11
)
feature_funcs: Sequence[Callable[[Tuple[float, float]], float]] = \
    [
        lambda _: 1.,
        lambda w_x: w_x[0],
        lambda w_x: w_x[1],
        lambda w_x: w_x[1] * w_x[1]
    ]
dnn: DNNSpec = DNNSpec(
    neurons=[],
    bias=False,
    hidden_activation=lambda x: x,
    hidden_activation_deriv=lambda y: np.ones_like(y),
    output_activation=lambda x: - np.sign(a) * np.exp(-x),
    output_activation_deriv=lambda y: -y
)
init_wealth_distr: Gaussian = Gaussian(
    mu=init_wealth,
    sigma=init_wealth_stdev
)

aad: AssetAllocDiscrete = AssetAllocDiscrete(
    risky_return_distributions=risky_ret,
    riskless_returns=riskless_ret,
    utility_func=utility_function,
    risky_alloc_choices=alloc_choices,
    feature_functions=feature_funcs,
    dnn_spec=dnn,
    initial_wealth_distribution=init_wealth_distr
)
```

Next, we perform the Q-Value backward induction, step through the returned iterator (fetching the Q-Value function for each time step from $t = 0$ to $t = T - 1$), and evaluate the Q-values at the `init_wealth` (for each time step) for all `alloc_choices`. Performing a max and arg max over the `alloc_choices` at the `init_wealth` gives us the Optimal Value function and the Optimal Policy for each time step for wealth equal to `init_wealth`.

```
from pprint import pprint

it_qvf: Iterator[QValueFunctionApprox[float, float]] = \
    aad.backward_induction_qvf()

for t, q in enumerate(it_qvf):
    print(f"Time {t:d}")
```

```
    print()
    opt_alloc: float = max(
        ((q((NonTerminal(init_wealth), ac)), ac) for ac in alloc_choices),
        key=itemgetter(0)
    )[1]
    val: float = max(q((NonTerminal(init_wealth), ac))
                     for ac in alloc_choices)
    print(f"Opt Risky Allocation = {opt_alloc:.3f}, Opt Val = {val:.3f}")
    print("Optimal Weights below:")
    for wts in q.weights:
        pprint(wts.weights)
    print()
```

This prints the following:

```
Time 0

Opt Risky Allocation = 1.200, Opt Val = -0.225
Optimal Weights below:
array([[ 0.13318188,  1.31299678,  0.07327264, -0.03000281]])

Time 1

Opt Risky Allocation = 1.300, Opt Val = -0.257
Optimal Weights below:
array([[ 0.08912411,  1.22479503,  0.07002802, -0.02645654]])

Time 2

Opt Risky Allocation = 1.400, Opt Val = -0.291
Optimal Weights below:
array([[ 0.03772409,  1.144612  ,  0.07373166, -0.02566819]])

Time 3

Opt Risky Allocation = 1.500, Opt Val = -0.328
Optimal Weights below:
array([[ 0.00126822,  1.0700996 ,  0.05798272, -0.01924149]])
```

Now let's compare these results against the closed-form solution.

```
for t in range(steps):
    print(f"Time {t:d}")
    print()
    left: int = steps - t
    growth: float = (1 + r) ** (left - 1)
    alloc: float = base_alloc / growth
    val: float = - np.exp(- excess * excess * left / (2 * var)
                          - a * growth * (1 + r) * init_wealth) / a
    bias_wt: float = excess * excess * (left - 1) / (2 * var) + \
        np.log(np.abs(a))
    w_t_wt: float = a * growth * (1 + r)
    x_t_wt: float = a * excess * growth
    x_t2_wt: float = - var * (a * growth) ** 2 / 2
    print(f"Opt Risky Allocation = {alloc:.3f}, Opt Val = {val:.3f}")
    print(f"Bias Weight = {bias_wt:.3f}")
```

```
    print(f"W_t Weight = {w_t_wt:.3f}")
    print(f"x_t Weight = {x_t_wt:.3f}")
    print(f"x_t^2 Weight = {x_t2_wt:.3f}")
    print()
```

This prints the following:

```
Time 0

Opt Risky Allocation = 1.224, Opt Val = -0.225
Bias Weight = 0.135
W_t Weight = 1.311
x_t Weight = 0.074
x_t^2 Weight = -0.030

Time 1

Opt Risky Allocation = 1.310, Opt Val = -0.257
Bias Weight = 0.090
W_t Weight = 1.225
x_t Weight = 0.069
x_t^2 Weight = -0.026

Time 2

Opt Risky Allocation = 1.402, Opt Val = -0.291
Bias Weight = 0.045
W_t Weight = 1.145
x_t Weight = 0.064
x_t^2 Weight = -0.023

Time 3

Opt Risky Allocation = 1.500, Opt Val = -0.328
Bias Weight = 0.000
W_t Weight = 1.070
x_t Weight = 0.060
x_t^2 Weight = -0.020
```

As mentioned previously, this serves as a good test for the correctness of the implementation of `AssetAllocDiscrete`.
We need to point out here that the general case of dynamic asset allocation and consumption for a large number of risky assets will involve a continuous-valued action space of high dimension. This means ADP algorithms will have challenges in performing the max / arg max calculation across this large and continuous action space. Even many of the RL algorithms find it challenging to deal with very large action spaces. Sometimes we can take advantage of the specifics of the control problem to overcome this challenge. But in a general setting, these large/continuous action space require special types of RL algorithms that are well suited to tackle such action spaces. One such class of RL algorithms is Policy Gradient Algorithms that we shall learn in Chapter **??**.

1.6 Key Takeaways from this Chapter

- A fundamental problem in Mathematical Finance is that of jointly deciding on A) optimal investment allocation (among risky and riskless investment assets) and B) optimal consumption, over a finite horizon. Merton, in his landmark paper from 1969, provided an elegant closed-form solution under assumptions of continuous-time, normal distribution of returns on the assets, CRRA utility, and frictionless transactions.
- In a more general setting of the above problem, we need to model it as an MDP. If the MDP is not too large and if the asset return distributions are known, we can employ finite-horizon ADP algorithms to solve it. However, in typical real-world situations, the action space can be quite large and the asset return distributions are unknown. This points to RL, and specifically RL algorithms that are well suited to tackle large action spaces (such as Policy Gradient Algorithms).

Bibliography

Merton, Robert C. 1969. "Lifetime Portfolio Selection Under Uncertainty: The Continuous-Time Case." *The Review of Economics and Statistics* 51 (3): 247–57. https://doi.org/10.2307/1926560.