A Machine Learning Approach to Stochastic Optimal Control

Pablo Ever Avalos

The University of Texas at El Paso

Follow this and additional works at: https://scholarworks.utep.edu/open_etd

Part of the Computer Sciences Commons, Finance and Financial Management Commons, and the Statistics and Probability Commons

Recommended Citation

https://scholarworks.utep.edu/open_etd/3470

This is brought to you for free and open access by ScholarWorks@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.
A MACHINE LEARNING APPROACH TO STOCHASTIC OPTIMAL CONTROL

PABLO EVER AVALOS ROBLES

Master's Program in Statistics

APPROVED:

Michael Pokojovy, Ph.D., Chair

Ori Rosen, Ph.D.

Thomas M. Fullerton, Jr., Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School
A MACHINE LEARNING APPROACH TO STOCHASTIC OPTIMAL CONTROL

by

PABLO EVER AVALOS ROBLES

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

May 2022
Acknowledgements

Foremost, I would like to thank UTEP and the faculty of the mathematics department for giving me the opportunity to continue my education. I would like to express my gratitude to my thesis advisor, Dr. Michael Pokojovy, whom introduced me to this thesis topic and guided me throughout the process. I would also like to thank the rest of my thesis committee, Dr. Ori Rosen and Dr. Thomas M. Fullerton, Jr. for giving me the feedback needed to complete my thesis. I cannot see myself completing this thesis without the support from my advisor and the committee.

Lastly, I want to thank my family and friends that supported me throughout my graduate education. They were always there to reassure any doubts that I may have had. I will always be grateful of everyone that was there for me at any moment during my graduate degree.
Abstract

Merton’s portfolio optimization problem is a well-renowned problem in financial mathematics which seeks to optimize the investment decision for an investor. In the simplest situation, the market consists of a risk-less asset (i.e. a bond) that pays back a relatively low interest rate, and a risky asset (i.e. a stock) that follows a geometric Brownian motion. The optimal allocation strategy of the investor’s wealth is found by optimizing the expected utility along the stochastic evolution of the market. This thesis focuses on several different applications of this optimization problem. We look at pre-constructed analytical solutions and showcase the results. We formulate simulated allocation strategies and compare results. Lastly, we approach this optimization problem using machine learning, specifically, by training neural networks.
# Table of Contents

Acknowledgements .............................................................................................................. iv

Abstract ............................................................................................................................... v

Table of Contents .................................................................................................................. vi

List of Tables ........................................................................................................................ viii

List of Figures ..................................................................................................................... ix

Introduction .......................................................................................................................... 1

Theoretical Background on Bellman’s Dynamic Programming Principle ......................... 2
  Brownian Motion ............................................................................................................... 2
  Stochastic Control .......................................................................................................... 3
  Dynamic Programming ...................................................................................................... 5
  Model ............................................................................................................................. 7

Utility ............................................................................................................................... 8

Application using Dynamic Programming ........................................................................... 11
  Fads Model .................................................................................................................... 11
  Parameter Estimation ..................................................................................................... 17

Constant Allocation Strategy .......................................................................................... 21
  Logarithmic Utility ........................................................................................................ 21
  Simulation and Output Function ..................................................................................... 25
  Logarithmic Utility Results ........................................................................................... 27
  Power Utility .................................................................................................................. 30
  Power Utility Results ..................................................................................................... 31

Machine Learning ........................................................................................................... 33
  Neural Networks ........................................................................................................... 33
  Application ..................................................................................................................... 34
  Results .......................................................................................................................... 36
Conclusion .................................................................................................................................39
Future Research Topics ........................................................................................................40
References .............................................................................................................................41
Appendix ..................................................................................................................................43
Vita ..........................................................................................................................................70
List of Tables

Table 1.1: Expected Utility of Showcased Investor

Table 1.2: Expected Utility Comparison of Showcased Examples

Table 2.1: Expected Logarithmic Utility using Fads trajectories

Table 2.2: Expected Power Utility using Fads trajectories
List of Figures

Figure 1.1: Ornstein-Uhlenbeck Plots.................................................................................................14
Figure 1.2: Black-Scholes Plots of Risky Asset Prices....................................................................15
Figure 1.3: Fads Model of Risky Asset Prices using Ornstein-Uhlenbeck Processes .....................16
Figure 2.1: Daily Closing Prices for Northern Oil and Gas, Inc......................................................17
Figure 2.2: Residual Diagnostic for NOG stock..................................................................................18
Figure 2.3: NOG Stock Trajectory Statistics ......................................................................................20
Figure 3.1: Preview of Natural Logarithmic Utility ...........................................................................22
Figure 3.2: Wealth Using Optimal MR Allocation on Black-Scholes Market .................................23
Figure 3.3: Wealth Using Black-Scholes Trajectories with Monthly Consumption .......................24
Figure 3.4: Logarithmic Utility from Simulated Allocation.................................................................26
Figure 3.5: Preview of Power Utility ..................................................................................................31
Figure 4.1: Sample of a Generic Deep Neural Network .................................................................34
Figure 4.2: Sample Visual of Trained Neural Network ....................................................................37
Figure 4.3: Sigmoid Activation Function ..........................................................................................37
Figure 4.4: Output Results of Trained Neural Network .................................................................38
Introduction

Merton’s portfolio optimization problem is a well-renowned optimization problem in financial mathematics, named after Nobel laureate Robert Merton. In the simplest scenario, this continuous-time finance issue revolves around finding the optimal investment decision for the investor, who only has two possible options of investment. One is a risk-free asset (i.e. a savings account, or a bond) which usually pays back a relative low rate of interest. The other is a risky asset (i.e. a stock, or real estate) whose price is assumed to follow a geometric Brownian motion. Assuming the investor has a limited amount of time, the problem’s goal is to maximize the personal expected utility gained from consuming the portfolio’s wealth. The consumption pattern of the investor may vary. The process involves modeling a market and finding the optimal allocation of wealth between the risk-free asset (bond) and the risky asset (stock). We can then use a utility function to model the investor’s attitude towards the confronted risk of investment, which when optimized, will allow us to, both, maximize the final/consumed wealth, and simultaneously control the risk of losing money.

The model can be considered a continuous-time market model, meaning the capital can be re-balanced at any moment before time has run out. In other words, wealth allocation can be switched between the different assets without an additional cost. It is also assumed that the assets can be sold or bought arbitrarily at any time. Lastly, the investor only gets information on current prices.

Merton formulated this problem in 1969 and solved it in 1971 using a stochastic optimal control approach. The value function of this optimization problem could be solved using dynamic programming, by deriving a nonlinear partial differential equation referred to as the Hamilton-Jacobi-Bellman (HJB) equation. This equation, however, is impractical to solve analytically
considering its nonlinearity. For special cases, however, a utility function can be considered in the constant relative risk aversion (CRRA) class, such as logarithmic or power utility, which is useful, because, for these cases, the optimal strategy is to keep a constant fraction of the current wealth in risky assets.

The goal of this thesis is to understand the applications of Merton’s portfolio optimization problem and approach it using several different methods. We will look at pre-constructed analytical solutions and showcase the results. We will formulate simulated allocation strategies and compare results. Lastly, we approach this optimization problem using machine learning.

The machine learning algorithm that is considered is a simple neural network system. Neural network systems are trained algorithms that run input values through a series of hidden neuron layers which eventually lead to a predicted output value. This process will require stochastic gradient descent implemented in the backpropagation of the system training since we do not work with pre-labeled training data. The goal will be to predict the best allocation strategy from running current wealth through a trained neural network.

**Theoretical Background on Bellman’s Dynamic Programming Principle**

Before we can tackle the optimization problem, we must establish a framework where stochastic control is used. The primary contribution of this thesis will be in the application of the model, but we will take the time to construct the mathematical theory as similarly seen by [Tikosi, 2016] and [Aboagye, 2018]. This presentation does not contain any new developments but solely serves the purpose of presenting a concise review.

**Brownian Motion**

Let us start building the basic ideas of stochastic finance, by following the work of [Karatzas and Scheve, 1998] and [Steele, 2001].
Let us consider a probability space \((\Omega, \mathcal{F}, \mathbb{P})\), with filtration \((\mathcal{F}_t)_{t \in [0, T]}\). We assume the filtration is right-continuous and \(\mathcal{F}_0\) contains all the measure 0 sets. Let \(T > 0\) be a non-random terminal time, and \((W_t)_{t \in [0,T]}\) denote a Brownian motion, which is a stochastic process.

Definition: A process \(W = (W_t)_{t \in [0,T]}\) is a \(\mathbb{P}\)-Brownian motion if it is \(\mathcal{F}_t\)-adapted and it satisfies

1. \(W\) is continuous with \(W_0 = 0\),
2. \(W_t - W_s\) is independent of \(\mathcal{F}_s\), \(0 \leq s < t \leq T\),
3. For any finite \(0 \leq s < t \leq T\), \(W_t - W_s \sim N(0, t - s)\) under the probability measure \(\mathbb{P}\).

We can also denote a vector of higher dimensional Brownian motion:

\[ W_t = (W_t^1, ..., W_t^n)^\top \]

where the \(W^i\) are independent Brownian motions, all adapted to the same filtration \(\mathcal{F}\). This is essential in modeling volatility, or in our case, the fluctuation of how a stock would act in a market setting.

Remark: The expected value of Brownian motion \(W_t\) at any time \(t\) is zero, that is, \(\mathbb{E}[W_t] = 0\), and variance is \(t\), \(\text{Var}(W_t) = t\), since \(W_t \sim N(0, t)\) when \(s = 0\). Although not practically relevant for this thesis, we still note that if running time would be left indefinite, then the variance would also run to infinity.

Stochastic Control

We must then introduce basic stochastic control theory based on work by [Saß, 2006]. We will consider Itô processes, which satisfy stochastic differential equations driven by Brownian motion. We have the stochastic differential equation:

\[ dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t. \]
By stochastic Picard & Lindelöf theorem, this equation has a strong solution when drift
\( b: [0, \infty) \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) and diffusion coefficient \( \sigma: [0, \infty) \times \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m} \) satisfy the following for all \( 0 \leq s, t \) and \( x, y \in \mathbb{R}^n \)
\[
\| b(s, x) - b(t, y) \| + \| \sigma(s, x) - \sigma(t, y) \| \leq K (\| y - x \| + |t - s|)
\]
\[
\| b(t, x) \|^2 + \| \sigma(t, x) \|^2 \leq K^2 (1 + \| x \|^2)
\]
for some positive constant \( K \).

**Definition:** An \( \mathcal{F} \)-progressively measurable process \( (u_t)_{t \in [0, \tau]} \) with values in some set \( \mathcal{U} \subseteq \mathbb{R}^p \) is called a control process. An n-dimensional process \( (Y_t)_{t \in [0, \tau]} \) controlled by \( u_t \) if it is defined by
\[
dY_t = b(t, Y_t, u_t) dt + \sigma(t, Y_t, u_t) dW_t
\]
where \( Y_0 = y_0 \),
\[
b: [0, T] \times \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^n
\]
\[
\sigma: [0, T] \times \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^{n \times m}.
\]
and \( (W_t)_{t \in [0, \tau]} \) denotes the \( m \)-dimensional Brownian motion.

The optimization objective is
\[
J(t, x, u) = E \left[ \int_t^\tau \psi(s, X^u_s, u_s) ds + \Psi(T, X^u_T) | X^u_t = x \right]
\]
We can denote an admissible set of controls by \( \mathcal{A}(t, x) \) which contains all the controls that fulfill the following:

1. The control process \( u = (u_s)_{s \in [t, \tau]} \) is progressively measurable with values in \( \mathcal{U} \) and
\[
E \left[ \int_t^\tau |u_s|^2 ds \right] < \infty.
\]
2. The stochastic differential equation describing the controlled process has a unique strong solution \( (X_s)_{s \in [t, \tau]} \) with \( X_t = x \) and
3. The optimization criterion $J(t, x, u)$ is well defined.

Our goal is to maximize the value function of the control problem defined by

$$V(t, x) = \sup_{u \in A(t, x)} J(t, x, u).$$

We must find the optimal value $V(0, x_0)$ and the optimal control strategy $u^*$, from which this value is obtained.

**Dynamic Programming**

We will be using what we call dynamic programming to break down the optimization problem into smaller sub-problems to be able to achieve the best overall optimum. We will briefly go over how this method works and will apply it to the portfolio optimization problem later in the thesis. We construct the theory similarly done by [Tikosi, 2016]. To use dynamic programming, we must define a specific optimal substructure. We use the Bellman Principle. To this end, we introduce the value function:

$$V(t, x) = \sup_{u \in A(t, x)} E_{t, x} \left[ \int_t^{t_1} \psi(s, X_s^u, u_s) ds + V(t_1, X_{t_1}^u) \right]$$

The Bellman principle is used to solve optimal control problems by isolating part of the whole optimization problem. An optimal control on an interval $([t, t_1]$ in our case) will remain optimal if we continue optimally at $t_1$. We can then continue by applying Itô’s formula to $V(t_1, X_{t_1}^u)$ if the wealth process has sufficient smoothness properties. We then end up with:
\[ V(t, x) = \sup_{u \in A(t, x)} E_{t,x} \left[ \int_t^{t_1} \psi(s, X_s^u, u_s) \, ds + V(t, X_t) \right. \\
+ \left. \int_t^{t_1} V_t(s, X_s)(D_x V(s, X_s))^\top b(s, X_s, u_s) \, ds + \int_t^{t_1} (D_x V(s, X_s))^\top \sigma(s, X_s, u_s) \, dW_s \right. \\
+ \left. \frac{1}{2} \int_t^{t_1} \text{tr} \left( D_{xx} V(s, V_s) \right)^\top \sigma(s, X_s, u_s) \sigma(s, X_s, u_s)^\top \, dW_s \right] \]

The expectation of \[ \int_t^{t_1} (D_x V(s, X_s))^\top \sigma(s, X_s, u_s) \, dW_s \] is 0, considering it is a martingale. We can also use notation \( a(s, X_s, u_s) = \sigma(s, X_s, u_s) \sigma(s, X_s, u_s)^\top \) for the diffusion matrix. Applying this, we obtain

\[ V(t, x) = \sup_{u \in A(t, x)} E_{t,x} \left[ \int_t^{t_1} \psi(s, X_s^u, u_s) \, ds + V(t, X_t) \right. \\
+ \left. \int_t^{t_1} V_t(s, X_s)(D_x V(s, X_s))^\top b(s, X_s, u_s) \, ds \right. \\
+ \left. \frac{1}{2} \int_t^{t_1} \text{tr} \left( D_{xx} V(s, V_s) \right)^\top a(s, X_s, u_s) \, dW_s \right] \]

We then subtract \( V(t, x) \), divide by \( (t - t_1) \), and let \( t_1 \) tend to \( t \). We also take the supremum and limit after checking if taking the limit can be interchanged with the expectation. Taking the conditional expectation when \( X_t = x \), we know \( V(t, X_t) = V(t, x) \). Consequently, we end up with

\[ 0 = \sup_{u \in U} \left[ \psi(t, x, u) + V_t(t, x) + (D_x V(t, x))^\top b(t, x, u) + \frac{1}{2} \text{tr}((D_{xx} V(t, x))a(t, x, u)) \right] \]

From here, we can define a differential operator that depends on \( u \):

\[ \mathcal{L}^u f(t, x) = V_t(t, x) + (D_x f(t, x))^\top b(t, x, u) + \frac{1}{2} \text{tr}((D_{xx} V(t, x))a(t, x, u)) \]

Meaning we ultimately end up with

\[ 0 = \sup_{u \in U} [\psi(t, x, u) + \mathcal{L}^u V(t, x)] \]
We refer to this equation as the **Hamilton-Jacobi-Bellman equation** (HJB). With this, we derived a necessary condition for the value function $V$.

**Model**

We continue the work of [Karatzas and Sheve, 1998] and [Steele, 2001] to model an initial market. The initial market model will consider a simple portfolio, where the investor has a base initial wealth which can be partitioned and invested into a market with two separate types of investment assets. A model like the one being constructed is called the **Black-Scholes model**. One type of investment will be a single low-risk bond with evolving prices which can be denoted by

$$dB_t = B_t r dt$$

where $B_0 = 1$,

meaning $B_t = e^{rt}$.

The other type of investment will be a stock with fluctuating prices

$$dS_t = \text{Diag}(S_t)(\mu dt + \sigma dW_t)$$

where $S_0 = s_0 > 0$,

with drift $\mu \in \mathbb{R}$, volatility $\sigma > 0$,

and $W$ is an $m$-dimensional Brownian motion. We can then use Itô’s formula to derive the explicit solution to the stochastic differential equation:

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right).$$

We can also point out that this type of process is also said to follow a geometric Brownian motion and follows constant drift.

**Remark:** Considering the definition, $S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$, which is dependent on $W_t$, with the strength of the stochastic Picard-Lindelöf theorem, $S_t$ is considered an Itô process. This
means that with a probability of 1, the path is continuous with finite variance at time \( t \) provided \( S_0 \) has a finite variance. Like with the definition of Brownian motion, although not practically relevant in this thesis, we note that the variance of \( S_t \) goes to infinity as \( t \) goes to infinity, unless \( \mu < \frac{\sigma^2}{2} \).

If the investor begins with initial wealth \( x_0 \), then the wealth at time \( t \) can be said to be modeled by:

\[
\frac{dX_t}{X_0} = N_t^B dB_t + N_t^S dS_t
\]

where \( X_0 = x_0 > 0 \).

Here, \( N_t^B \) and \( N_t^S \) represent the possibly fractional number of assets (bonds and stocks respectively) that are held by the investor at time \( t \). \( N_t^B \) and \( N_t^S \) are non-negative and we assume that the wealth is also always non-negative, meaning

\[
X(t) \geq 0, \text{ for } 0 < t < T.
\]

Now that we have a general procedure as to how to build a market model, we introduce utility which allows us to find the optimal market allocation that will maximize the investor’s satisfaction.

**Utility**

Utility is an essential concept to define in Merton’s portfolio analysis. General economic utility is defined as the satisfaction gained from taking an action or consuming a product. In our case, utility represents the personal satisfaction of an investor from the outcome of their investment. This can be represented by utility functions that describe the pattern of utility gained from consuming wealth.

To begin constructing utility functions, we need define several assumptions. First, we shall assume that the investor is risk-averse, meaning the investor will act on investments which are in his favor, choosing a more predictable low-return investment over a risky high-return investment. This implies that we are looking at a strictly concave utility function, since the utility of an action will outweigh the demand of the action. Secondly, we assume that the investor will always benefit
from additional consumption, meaning they gain more satisfaction as they gain more wealth. This implies a monotone increasing utility function. Lastly, we will also be taking the assumption that the investor not only wants to maximize the utility gained from wealth, but also use the investment as a source of living by trying to generate more wealth with it while he consumes said utility from it. In essence, the goal is to maximize expected utility throughout the time period. We now follow the work of [Saß, 2006] once again to construct the utility of consumption.

**Definition**: For a subset $S \subseteq \mathbb{R}, U : S \to \mathbb{R}$ is a *utility function*, if $U$ is strictly increasing, strictly concave and continuous on $S$.

The goal would be to solve optimization model:

$$\max_u \mathbb{E}[U(X_T)|X_0 = x_0].$$

We will consider two different types utility functions: the logarithmic utility function and the power utility function. These two types of utility functions are hyperbolic absolute risk aversion functions, which are also referred to and known as constant relative risk aversion (CRRA) types, which are needed to analytically solve the optimization problem.

In the case of **logarithmic utility**, $U(x) = \ln(x)$. The optimal policy is

$$u_t^* = \frac{\mu - r}{\sigma^2} \text{ for all } t \in [0,T].$$

This is found by solving the following stochastic differential equation of wealth:

$$dX_t = X_t(r + u_t^*(\mu - r))dt + u_t^*\sigma dW_t$$

It is solved relatively easy, by assuming there is no consumption throughout the investment period. The stochastic differential equation has an exponential solution, meaning we can just take the logarithm and maximize it pointwise. Solving using this method would not work for other types of consumption. With this policy, the optimal strategy would be to keep the partition set aside for risky investments as a constant proportion of the total wealth (for example, if $u_t = 0.15$, then the
consumer should always keep 15% of the current wealth invested in the risky asset. However, this strategy implies a market where there is constant trading, which is not applicable in practice.

We can also consider \( n \)-stocks with prices \( (S_t)_{t \in [0,T]}, S_t = (S^1_t, ..., S^n_t)^T \) with similar dynamics

\[
dS_t = \text{diag}(S_t)(\mu dt + \sigma dW_t)
\]

where \( S_0 = s_0 \)

\[
s^i_0 > 0 \text{ for } i = 1,2,3, ..., n, \text{ with } \mu \in \mathbb{R}^n
\]

and \( \sigma \) being a non-singular symmetric, positive definite volatility matrix in \( \mathbb{R}^{n \times m} \).

If working with multiple stocks, stock \( i \) will evolve as such:

\[
dS^i_t = S^i_t(\mu_i dt + \sum_{j=1}^n \sigma_{ij} dW^j_t), \text{ for } i = 1,2, ..., n.
\]

We can take an \( n \)-dimensional process \( u \) as control at time \( t \), where \( u^i_t \) is the fraction of wealth which is invested in stock \( i \). This will give us optimal utility solution:

\[
u^*_t = \pi^* = (\sigma \sigma^T)^{-1}(\mu - r), \text{ for } t \in [0,T].
\]

However, for the time being, this thesis will focus on working with only one stock.

As mentioned, the other type of utility is the \textbf{power utility} function. \( U(x) = \frac{x^\gamma}{\gamma} \), where \( \gamma \in \mathbb{R}, \gamma < 1, \gamma \neq 0 \).

The Merton portfolio optimization problem realizes an HJB function for this power function. It follows, for \((t,x) \in \overline{Q_0} = [t_0, t_1] \times \mathbb{R}^n, p \in \mathbb{R}^n, A \in S^n_t, \) we have:

\[
H(t,x,p,A) = \sup_{\pi \epsilon [0,1], c \geq 0} \left[ -[(r + \pi)(\mu - r)x - c]p - \frac{1}{2}A \sigma_0^2 x^2 \pi^2 - e^{-\beta t} U(c) \right]
\]

We can then maximize the smooth function \( g(\pi, c) \) over a closed domain and the Lagrange principle is applied. Assuming \( p \geq 0 \), this gives us optimal pair:

\[
c^*(t,x,p,A) = (pe^{\theta t})^{\frac{1}{\gamma - 1}}
\]
\[ \pi^*(t, x, p, A) = -\frac{(\mu - r)p}{A\sigma_0^2 x} \]

along with

\[ V(t, x) = h(t)^{1-\gamma} x^\gamma \]

where \( h(t) \) is strictly positive for every \( t \). Plugging in this optimal pair, we get

\[ H(t, x, p, A) = -[(r + u)(\mu - r)x - c^*)p - \frac{1}{2}A\sigma_0^2 x^2(u^*)^2 - e^{-\rho t} U(c^*)] \]

which is our theoretical HJB function.

**Application using Dynamic Programming**

Now that we have presented the theory behind Merton’s portfolio optimization problem using HJB equations, we can take a look at direct applications of this strategy in a general market model. We must keep in mind that the best strategy for the model we constructed is to keep a constant portion of the current wealth invested in the risky asset. For this result to be achieved, we have to take the assumption that the risky asset price will follow a geometric Brownian motion.

**FADS Model**

“Fads” models were first introduced by [Shiller, 1984] and [Summers, 1986] in the 1980’s. They resemble Black-Scholes models that do not assume constant drift. We let \( Y_{\varepsilon} \) be an Ornstein-Uhlenbeck process for a positive constant \( \varepsilon \), defined by stochastic differential equation:

\[ dY_{\varepsilon}(t) = -\frac{1}{\varepsilon}Y_{\varepsilon}(t)dt + dW(t) \]

where \( Y_{\varepsilon}(0) = 0 \).

Like most Ornstein-Uhlenbeck processes, this equation has a mean reverting property, meaning the drift will pull the value back to the mean, which would be 0 in our case, considering the process is drawn by the white noise term \( dW(t) \). The plots on p. 14 show the strength of the drift to the
mean using different values of \( \varepsilon \). We will notice that the processes will tend more to zero the smaller the value of \( \varepsilon \). We can also find the explicit solution of the stochastic differential above:

\[
Y_\varepsilon(t) = \int_0^t e^{-\frac{(t-s)}{\varepsilon}} dB(s).
\]

At this moment, we can introduce the modified Black-Scholes model:

\[
dS(t) = S(t) \left( (\mu + Y_\varepsilon(t)) dt + \sigma \rho dW(t) + \sigma \sqrt{1 - \rho^2} dB(t) \right)
\]

where \( B(t) \) and \( W(t) \) are independent Brownian motions for \( 0 < \rho < 1 \).

We will now simulate the solution trajectories that will represent the asset prices of the risky asset. We let \( T \) be fixed as our time horizon, and let us take \( N \) discretization steps. We also let \( 0 < t_0 < t_1 < \cdots < t_n = T \), and \( dt = \frac{T}{N} \). We can then model the processes of the stochastic differential equations describing both geometric Brownian motion \( X_t \) and Ornstein-Uhlenbeck \( Y_t \), with their respective discretized versions:

\[
X_\varepsilon = X_0 \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma W_1(t) \right)
\]

\[
X(t_n) = X_0 \left( \exp \left( \sum_{j=0}^{n-1} \left( \mu - \frac{1}{2} \sigma^2 \right) dt + \sigma \sqrt{dt} Z_1(j) \right) \right)
\]

P. 15 will have several examples of this, plotting the Black-Scholes risky asset prices using different values for volatility \( \sigma \) and drift \( \mu \):

\[
Y_\varepsilon(t) = \int_0^t e^{-\frac{(t-s)}{\varepsilon}} dB(s)
\]

\[
Y_\varepsilon(t_n) = e^{-\frac{t_n}{\varepsilon}} \sum_{j=1}^{n-1} e^{\frac{t_j}{\varepsilon}} \sqrt{dt} Z_2(j)
\]

Lastly, we have the solution trajectories for the fads model:
\[ dS(t) = S(t) \left( (\mu + \epsilon(t))dt + \sigma \rho dW(t) + \sigma \sqrt{1 - \rho^2} dB(t) \right) \]

\[ S(t_{n+1}) = S(t_n) \left( (\mu + \epsilon(t))dt + \sigma \rho \sqrt{dt} Z_3(n) + \sigma \sqrt{1 - \rho^2} \sqrt{dt} Z_2(n) \right) \]

where \( W_1, W, \text{and } B \) are independent Brownian motions, meaning \( Z_1, Z_2, \text{and } Z_3 \) are independent \( N(0,1) \) random variables. Fads trajectory examples can be found on p. 16.
We use 200 discretization steps on 10 trajectories for different values of $\varepsilon$, the constant for the Ornstein-Uhlenbeck process.
Figure 1.2: Black-Scholes plots of risky asset prices

We use 1200 discretization steps on 10 trajectories of prices with an initial price of $100 and different values of $\sigma$ and $\mu$. 
Figure 1.3: Fads Model of Risky Asset Prices Using Ornstein-Uhlenbeck (OU) Processes

We use 1200 discretization steps on 10 trajectories of prices in each plot with an initial price of 100, $\rho = 0.4$, $\sigma = 0.4$ and $\mu = 0.12$, with different values of $\epsilon$, the constant for the OU process.
PARAMETER ESTIMATION

Now that we have been able to create risky asset (stock) trajectories using the both the Black-Scholes and fads modeling methods, we can take a look at parameter estimation for $\sigma$ and $\mu$, as similarly seen by [Teka, 2013]. We will showcase how this is done with real data provided by Yahoo! Finance, of the stock Northern Oil and Gas, Inc. (NOG) from the New York Stock Exchange – Nasdaq. We look at 254 daily closing prices of NOG ending on May 6th, 2022:

![Figure 2.1: Daily Closing Prices for Northern Oil and Gas, Inc.](image)

The data covers a full year; however, the total of 254 data points is due to the data being collected for the daily closing price between Monday and Friday, and it excludes weekends. This stock was chosen because of the visual upward trend over the past year, representing a stock that might interest an investor.

We treat this data set as time series data, and complete some residual diagnostics. We set $x(t_k)$ as the difference of logs:

$$x(t_k) = \ln(S(t_k)) - \ln(S(t_{k-1}))$$

We now look at the residual diagnostics of $x(t_k)$. 

17
Figure 2.2: Residual Diagnostic for NOG stock
We make sure that nothing significant stands out. We can observe that the standardized residuals resemble white noise. The ACF and p-values are within their respective significance boundaries, meaning they are insignificant. In the case that the investor runs into a data set like this one, that does not include weekend closing prices and find themselves with significant values for the 5th lag, the investor can also instead use average weekly closing prices of the data. Lastly, we observe the normal QQ plot, which although not perfect at the tails, is relatively normal without any extreme outliers.

From this, we can say that independence and normality follow \( x(t_k) \). We use this information to create a log-likelihood function:

\[
L(\theta) = \sum_{k=1}^{n} \ln \left( f_\theta \left( x(t_k) \right) \right)
\]

with probability density function:

\[
(f_\theta \left( x(t_k) \right)) = \frac{1}{S(t_k)\sigma \sqrt{2\pi(t_k - t_{k-1})}} \exp \left( -\frac{\left[ x(t_k) - \left( r - \frac{1}{2}\sigma^2 \right)(t_k - t_{k-1}) \right]^2}{2\sigma^2(t_k - t_{k-1})} \right)
\]

We then use mean and variance estimates computed using Black-Scholes model definition:

\[
\hat{m} = (\hat{\mu} - \frac{1}{2}\hat{\sigma}^2)\Delta t, \text{ and } \hat{\sigma} = \hat{\sigma}^2\Delta t, \text{ for } \Delta t = t_k - t_{k-1}
\]

After taking the derivative of the density function with respect to these parameters, we set it equal to zero to compute estimates:

\[
\hat{m} = \sum_{k=1}^{n} \frac{x(t_k)}{n} \text{ and } \hat{\sigma} = \sqrt{\frac{\sum_{k=1}^{n} \left( \frac{x(t_k) - \hat{m}}{n} \right)^2}{\Delta t}}.
\]

We can then, put it all together to solve for sample mean and variance:

\[
\hat{\sigma} = \sqrt{\frac{\sum_{k=1}^{n} \left( \frac{x(t_k) - \hat{m}}{n} \right)^2}{\Delta t}} \text{ and } \hat{\mu} = \left( \frac{\sum_{k=1}^{n} \frac{x(t_k)}{n}}{\Delta t} + \frac{1}{2}\sigma^2 \right) \text{ for } \Delta t = \frac{1}{253}.
\]
In our case, we find $\hat{\sigma} = 0.576$, and $\hat{\mu} = 0.808$. We use these parameters to create Black-Scholes trajectories for future SLB prices. We plot 10 sample trajectories out of 50,000 modeled Black-Scholes trajectories, along with their mean, median, and a 95% prediction interval.

![NOG Market Trajectories](image)

**Figure 2.3: NOG Stock Trajectory Statistics**

This is useful to showcase this risk vs. reward dilemma that an investor encounters when going into a market. Looking at the plot of the trajectories, with a beginning price of $29, we can see that an investor can expect a 50% chance of having the stock end up costing somewhere between $18 and $55, found using the end values of the prediction interval and median. This obviously does not benefit the investor as much, however, we can see that the final mean wealth is at $65, which is higher than the median, and if the stock trajectory ends up in the higher half of the projections, it can grow to be up to about $170. This is a representation of the dilemma because although an investor has 50% chance of either losing money or not making a significant profit margin, in the case that the price of the stock grows, the amount of profit gained for the investor outweighs the risk of losing money. This is the driving force of why an optimization strategy is
needed. Lastly, the confidence interval gives a range of where the price of a trajectory might end up at the end of the time horizon, meaning the investor has the chance to see the likely possible outcomes, and gain some insight of the future status of the market.

**CONSTANT ALLOCATION STRATEGY**

Once the price of a risky asset can be modeled, the next step would be to model an investor’s wealth over time while using an optimal allocation strategy on the risky asset. As previously mentioned, the optimal strategy for a Merton optimal control problem, like the one we have constructed, is to keep a constant percentage of your current wealth invested in the stock (risky asset) and keep the rest in a stable growing bond (risk-free asset). In this section we will model and showcase a couple of examples of what wealth might look like over time for an investor, and examine the personal utility gained by an investor from their final/total consumed wealth.

**Logarithmic Utility**

Under the logarithmic utility case, we are considering an investor whose ratio of personal utility gained per dollar, follows a logarithmic function $U(x) = \ln(x)$. Investors might have a modified version of logarithmic utility, but in this thesis, we will assume the investor will just follow a natural logarithmic utility like the one shown in the following page.
We can see from the graph that following a logarithmic utility meets the original preset assumptions, meaning the graph is strictly concave and monotone increasing.

For logarithmic utility, the optimal allocation policy is given by:

$$u_t^* = \frac{\mu - r}{\sigma^2}$$

The theoretical optimal value of utility is calculated using:

$$\mathbb{E}[\log(X_t)] = \log(V_0) + \left(r + \frac{\mu - r}{2\sigma^2}\right)$$

We now showcase an investor who may invest in a market of one risk-free asset (bond) with interest rate, $r = 0.05$, and of one risky asset (stock) with a price that follows a Black-Scholes trajectory. We use a Black-Scholes trajectory model, because although it is suboptimal in practice, it is good for modeling and showcasing consistent results. The independent variables used for this market will be initial stock price $S_0 = \$100$, drift $\mu = 0.12$, and volatility $\sigma = 0.4$. Initial wealth is normally given by the investor, but we will be using default initial wealth $V_0 = \$500$. Using these variables, we can find the theoretical optimal Merton allocation strategy for this investor, and the optimal value of personal utility gained from using this Merton ratio allocation strategy.
Table 1.1: Expected Utility of Showcased Investor

<table>
<thead>
<tr>
<th>Theoretical Merton Ratio Allocation</th>
<th>0.4375</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Value of Utility</td>
<td>6.4834</td>
</tr>
</tbody>
</table>

The following graph will show 10 trajectories of 1200 discretization steps showing what the wealth of an investor might look like over time in this market.

![Wealth using MR Allocation - BlackScholes](image)

Figure 3.2: Wealth Using Optimal MR Allocation on Black-Scholes Market

The model is flexible enough to incorporate intermediate consumption by the investor. Assuming the time horizon spans over a fiscal year, the investor might want to consume or withdraw a portion of their current wealth at given times, such as 25% of the portfolio’s value every quarter, or even a monthly set-amount withdrawal to cover personal expenses.

Let us showcase the wealth of an investor who has a unique consumption pattern. Let’s assume the market’s independent variables are the same as the previous example, with initial stock price $S_0 = $100, drift $\mu = 0.12$, and volatility $\sigma = 0.4$. The investor provides an initial wealth, $V_0 = $500, and mentions they want to consume/withdraw at the end of every month of a fiscal
year. The consumption rate will be 20% of the current portfolio wealth if they are at a profit margin (current wealth is greater than initial wealth), 10% otherwise, but will not consume at all if current wealth is below a minimum amount, $200, also set by the investor. We graph 10 trajectories with 1200 discretization steps showing the wealth over time of this investor.

Figure 3.3: Wealth Using Black-Scholes Trajectories with Monthly Consumption

The final and total consumed wealth is used to calculate the investor’s value of expected utility. We must assume that the utility gained from consuming wealth will still follow the logarithmic utility function regardless of when the wealth was consumed, that is, the investor will not gain or lose any excess utility from consuming early/within the time horizon. Addressing this assumption is something that can be revisited for future research. We may now take the time to compare the two investors from these last couple of showcased examples, and see who would receive more personal utility from their results. We calculate the expected utility of an investor by simulating 2,500 trajectories, for accuracy, and calculating expected logarithmic utility from the average final/consumed wealth of each investor.
Table 1.2: Expected Utility Comparison of Showcased Examples

<table>
<thead>
<tr>
<th>Showcased Client</th>
<th>Expected Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Intermediate Consumption</td>
<td>6.2877</td>
</tr>
<tr>
<td>Monthly Consumptions</td>
<td>6.2610</td>
</tr>
</tbody>
</table>

We can note that continuous trading is not possible, therefore the expected utility of an investor will always be lower than the theoretical optimal value of utility, 6.4834, in our case. We also note that Black-Scholes trajectories have a constant drift, therefore we can interpret the second showcased client’s lower expected utility due to the earlier consumption within the time horizon, losing possible wealth growth from the consumed amount.

**Simulation and Output Function**

As mentioned, we can also directly simulate the optimal allocation strategy. An R-code, as shown in the appendix, is written to check every allocation strategy of wealth for the investor. After establishing the market, the code checks and compares every tenth of a percent of allocation into the risky asset (stock) when modeling the investor’s wealth. The final/consumed wealth is then run through the utility function and outputs both the allocation strategy that achieved the largest expected utility, along with said expected utility value. We show graphically how different allocation strategies affect utility for a Black-Scholes trajectory market. We simulate allocation strategies for an investor who follows a logarithmic utility using 5,000 trajectories with variables $\sigma = 0.4$ and $\mu = 0.12$. 
The drawback with simulating in this way is that the processing time taken to run the simulation is proportional to the allocation strategies checked. Since we are checking every tenth of a percent of allocation, the simulation will take 1,000 times longer than just modeling wealth and utility once with a single pre-determined allocation strategy. This is important to consider because an investor might not always be able to use the theoretical allocation strategy in a real market.

We can bring all of this information together to make investment analysis, and portfolio optimization simple for the investor by using an output function. Provided simple information from the investor, such as the initial investment amount, the type of utility function the investor would like to follow, and their consumption patterns, the output function will return the following values:

- Theoretical Merton ratio allocation (MR) strategy
- Expected intermediate consumption amounts, if any
• Expected final wealth
• Utility gained from following the MR allocation strategy
• Simulated optimal allocation strategy
• Utility gained from following the simulated allocation strategy

This output function makes it very simple to compare different allocation strategies in different markets.

For the rest of this section, markets will be modeled using stock (risky-asset) trajectories that follow the modified fads trajectory model, which does not assume constant drift for larger values of $\epsilon$. This means we are assuming the investor will still want to use the Merton allocation strategy while being aware that the market does not follow Black-Scholes trajectories. This is important because real markets have no assumed drift, and when simulating allocation under Black-Scholes, the allocation strategy will tend towards the theoretical one.

Logarithmic Utility Results

We use the output function to create a table that compare different markets. We will look at examples of an investors logarithmic expected utility (EU) using both the theoretical allocation strategy (TMR) and the simulated allocation strategy (SMR). The investor begins with a starting wealth of $V_0 = $500 and some investors will intermediately consume (IC) following the rules of the showcased investor, 20% at profit margin, 0% if below $200, 10% otherwise. Each market is simulated using 5,000 trajectories, using an initial stock price of $100 with different independent market variables, $r$ interest rate, $\mu$ drift, $\sigma$ volatility, $\epsilon$ constant for Ornstein-Uhlenbeck process, and $\rho$ correlation constant of Brownian motion.
The green row will be used as our pseudo-control investor market, considering those variable values are relatively tame, while the gray cells will be the market values which are different from this control scenario. Addressing the markets by row number, from top to bottom, we can begin by taking a look at the first three rows. Similar to the showcased example, we can see that an investor receives less overall utility the more they consume intermediately. This, again, is explained by the assumption that earlier consumption does not grant excess/restrained utility. The investor is giving up possible total overall wealth by losing possible wealth gain from the

Table 2.1: Expected Logarithmic Utility using Fads trajectories

<table>
<thead>
<tr>
<th>IC</th>
<th>$r$</th>
<th>$\mu$</th>
<th>$\sigma$</th>
<th>$\epsilon$</th>
<th>$\rho$</th>
<th>TMR</th>
<th>TEU</th>
<th>SMR</th>
<th>SEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2714</td>
<td>0.275</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2653</td>
<td>0.331</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2544</td>
<td>0.459</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2658</td>
<td>0.229</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2601</td>
<td>0.269</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2511</td>
<td>0.392</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2643</td>
<td>0.217</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2566</td>
<td>0.213</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4375</td>
<td>6.2500</td>
<td>0.415</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.8</td>
<td>0.4375</td>
<td>6.2619</td>
<td>0.199</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0.05</td>
<td>0.36</td>
<td>0.8</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4844</td>
<td>6.2848</td>
<td>0.280</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0.09</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.1875</td>
<td>6.3049</td>
<td>0.100</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0.05</td>
<td>0.807</td>
<td>0.576</td>
<td>0.1</td>
<td>0.4</td>
<td>1.000</td>
<td>6.7583</td>
<td>1.000</td>
</tr>
</tbody>
</table>
wealth consumed intermediately. As mentioned, this may be considered a next step for future research options, by writing the model in a way where the consumption patterns affect the utility gained, that is, an investor whose goal is to consume intermediately, gains extra utility from consuming earlier than later, and can even consider losing utility if they overconsume. This assumption would also explain the simulated Merton-ratio allocation strategies. The simulated strategy increases as the number of times an investor consumes intermediately increases. This means the investor needs to allocate more of their current wealth into the stock (risky asset), since the amount of wealth invested will decrease over time because of intermediate consumption. We see similar patterns in rows 4 through 9.

Taking another look at rows 1 through 9, we can see that overall simulated allocation and utility decreases as the value of epsilon increases. Remember, from Figure 1.1, epsilon controls the strength of the drift from the mean in the Ornstein-Uhlenbeck process used in the fads trajectories. We can conclude that the decrease in simulated allocation strategy, and decrease in utility gained as epsilon grows, comes from the risk-adverse investor needing to invest less in a more uncertain stock (risky asset) trajectory. We also notice that the simulated values grant lower allocation strategies and higher utilities in comparison to the theoretical values, meaning in this type of market, it is better to allocate less current wealth into the stock (risky asset).

In rows 10 and 11, we see opposing results due the change in the markets. An increase of the value $\rho$, correlation of Brownian motion, in row 10 shows a slightly higher uncertainty, showing a lower expected utility, and simulated results, compared to our control. Row 11 shows a market with a higher $\mu$ and $\sigma$, drift and volatility of the model, resulting in a higher upward trend of the trajectories, resulting in higher values in both the theoretical and simulated amounts. In row 12, we see an increase in the interest rate of our bond (risk-free asset). This change significantly
decreases both the theoretical and simulated allocation strategies of our risk-adverse investor. This is expected, since a higher interest rate return should result in more wealth allocated into the bond (risk-free asset), and less wealth allocated into the stock (risky asset).

Lastly, for row 13, although originally created for the Black-Scholes model, we use the estimated parameters from the NOG stock. We can see that the relatively high $\mu$ and $\sigma$ directly affects our results, similar to row 11. For both the theoretical and simulated results, we are told that the investor should allocate all of their wealth into risky asset (stock). Both the theoretical and simulated allocation provide the same expected utility, which is also higher than the expected utility gained from other markets.

**Power Utility**

Under the power utility case, we are considering an investor whose ratio of personal utility gained per dollar, follows a power function $U(x) = \frac{x^\gamma}{\gamma}$. This type of utility will also meet the original requirements/assumptions made, by being both increasing and concave. Investors may select a value of $\gamma$ that is unique to the way they obtain utility. In this thesis, we will see examples of a variety of investors whom have selected different values of $\gamma$, but in the following preview we will show what power utility looks like with a value of 0.1 compared to a value of 0.5 for $\gamma$. We will see that for larger values of $\gamma$, the incline of the function is steeper meaning the increase in utility for every relative dollar is higher than other utilities, and the chance of gaining wealth will outweigh the risk of losing money. This will show interesting results when we compare different markets using this utility.
The optimal allocation strategy when using a power utility function is given by:

\[ u_t^* = \frac{\mu}{\sigma^2(1 - \gamma)}. \]

The optimal value of expected utility is calculated using:

\[ \mathbb{E} \left( \frac{X_T^{\gamma}}{\gamma} \right) = \frac{V_0^{\gamma}}{\gamma} \exp \left( \frac{\mu^2 \gamma}{2 \sigma^2(1 - \gamma)} \right). \]

**Power Utility Results**

For power utility, we will focus more in seeing examples of investors with different values of gamma (\( \gamma \)), who only consume at the end of the time period, instead of modeling investors who consume intermediately. Please note, under the same assumption that an investor does not gain excess/impaired utility from intermediate consumption, modeling intermediate consumption would result in similar patterns as the ones seen in Table 2.1 for logarithmic utility.

We can now, similarly to logarithmic utility, use the R output function to make a table showing a variety of investors in different markets, each of 5,000 fads trajectories.
Table 2.2: Expected Power Utility using Fads Trajectories

<table>
<thead>
<tr>
<th>γ</th>
<th>r</th>
<th>μ</th>
<th>σ</th>
<th>ε</th>
<th>ρ</th>
<th>TMR</th>
<th>TEU</th>
<th>SMR</th>
<th>SEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>0.758</td>
<td>106.45</td>
<td>0.300</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.758</td>
<td>106.44</td>
<td>0.273</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.758</td>
<td>106.44</td>
<td>0.278</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>0.833</td>
<td>18.670</td>
<td>0.330</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.833</td>
<td>18.670</td>
<td>0.311</td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.833</td>
<td>18.690</td>
<td>0.384</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.01</td>
<td>0.4</td>
<td>1.000</td>
<td>45.928</td>
<td>0.523</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>1.000</td>
<td>46.349</td>
<td>0.755</td>
</tr>
<tr>
<td>9</td>
<td>0.5</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>1.0</td>
<td>0.4</td>
<td>1.000</td>
<td>49.339</td>
<td>1.000</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.05</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.8</td>
<td>0.833</td>
<td>18.640</td>
<td>0.244</td>
</tr>
<tr>
<td>11</td>
<td>0.1</td>
<td>0.05</td>
<td>0.36</td>
<td>0.8</td>
<td>0.1</td>
<td>0.4</td>
<td>0.625</td>
<td>18.698</td>
<td>0.302</td>
</tr>
<tr>
<td>12</td>
<td>0.1</td>
<td>0.09</td>
<td>0.12</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.833</td>
<td>18.663</td>
<td>0.095</td>
</tr>
<tr>
<td>13</td>
<td>0.1</td>
<td>0.05</td>
<td>0.808</td>
<td>0.576</td>
<td>0.1</td>
<td>0.4</td>
<td>1.000</td>
<td>19.714</td>
<td>1.000</td>
</tr>
</tbody>
</table>

In general, we can make the observation that an investor with power utility tends to be less risk-adverse than an investor using logarithmic utility, gaining more utility from wealth, therefore, more personal utility from uncertainty in the trajectories. This makes the overall allocation strategies to be quite larger than an investor with logarithmic utility. Especially for larger values of $\gamma$, as shown by the steep plots from the previews of power utility, the utility gained per dollar heavily outweighs the risk of loss, therefore the allocation strategy would be to allocate all of your wealth into the stock (risky asset). We can, alternatively, also see that as $\gamma$ approaches zero, the
allocation strategy approaches a more reasonable ratio. The change in epsilon does not have much effect on the result, but we can notice that an increase in volatility does decrease the allocation strategy. Changing the interest rate of the bond has no effect on the theoretical strategy. Simulated result patterns are similar to the simulated results from the logarithmic utility results table. In conclusion, the utility gained is very sensitive to the value of $\gamma$ chosen by the investor.

**Machine Learning**

Let us approach Merton’s portfolio optimization problem from a different angle. Let us assume we were unable to analytically derive the optimal allocation strategy, meaning we did not know $u_t^*$. We may also say that simulating our allocation strategy takes a significantly long of a time to compute. A possible alternate approach would be to try to train a machine learning algorithm to find the best allocation strategy for us, and have it ready to be used when necessary. In this section we will review basic structure of a neural network, numerically find the training steps for the algorithm and show results of our trained system.

**Neural Networks**

Neural networks are a field of machine learning, typically used to predict certain outcomes. The name and structure of a neural network was inspired by the way a human brain works. In supervised learning, there usually consists three main layer types, an input layer, hidden layers, and an output layer. Although this thesis will work with a relatively small neural network, in general, neural networks may have several hidden layers stacked together and can be referred to as a deep neural net. The hidden layers can each have any number of neurons needed. Neurons contain weight values that are optimized for accuracy, which are then activated by an activation function, which ultimately lead us to the final output or prediction.
Figure 4.1: Sample of a Generic Deep Neural Network

The typical way of training a neural network is by updating the weights and biases in the nodes through a series of iterations, or epochs. Once the coder of the neural network initializes the weights and biases randomly, then training data can be used to calculate how inaccurate the model is with these random parameters. This is done using a loss function, which will then be used to update the parameters so that the system minimizes the prediction error of the neural network. This is repeated a number of times until the reduction in error flattens out.

**APPLICATION**

Assuming we do not know our allocation strategy $u^*_t$, using a neural network in our case is challenging since we do not have training data, or a loss function, considering we would not know how inaccurate the model is. We are unable to train the neural network directly in a typical manner and therefore must be trained another way.

Approaching this problem means we use a closed loop control function compared to the open loop control function used in dynamic programming. The difference in the type of model is...
that the dynamic programming’s open loop control function does not provide feedback, that is, once we have an allocation strategy, then it is implemented directly so that wealth and utility are modeled. Since we want our neural network to learn to provide the best allocation strategy for a given scenario, we must use feedback provided from a closed loop control function to correctly update the parameters in our system.

We define unknown feedback function $\Phi$:

$$u_t^* = \Phi(x_t, \theta)$$

dependent on current wealth and $\theta$ set of parameters representing the weights and biases of our neural network. Therefore, using the control process optimization objective:

$$J(t, x, u) = E \left[ \int_t^T \psi(t, X^u_t, u_t) dt + \Psi(T, X^u_T) | X^u_t = x \right],$$

we obtain new optimization objective:

$$J(t, x, \theta) = E \left[ \int_t^T \psi(t, X^\theta_t, \Phi(x_t; \theta)) dt + \Psi(T, X^\theta_T) | X^\theta_t = x \right].$$

This new optimization objective can be approximated numerically:

$$J(t, x, \theta) \approx -\frac{1}{N} \sum_{rep=1}^N \left[ \sum_{k=1}^{T/M} \psi(t_k, X^\theta_{tk}(\omega_{rep}), \Phi(x_{tk}; \theta)) \Delta t + \Psi(T, X^\theta_{T_k}) \right].$$

Our goal would be to, instead of maximizing utility, to minimize this approximation over the set of parameters $\theta$, throughout a set of iterations or epochs, denoted by ‘rep’.

This is ultimately done by applying gradient descent to the parameters. Gradient descent essentially creates a gradient from the derivatives of the objective with respect to the parameters, and finds the numerical minimum through a series of iterations. This is implemented in the back-
propagation of the system, so that we may update the parameters accordingly. The gradient is also estimated numerically, so for $\theta$, with $n$ parameters:

$$grad[k] \approx \frac{\text{objective}(\theta + \epsilon(V_k)) - \text{objective}(\theta)}{\epsilon}$$

for every $1 \leq k \leq n$,

$V_k$ is vector $(v_1, \ldots, v_k, \ldots, v_n)$, where $v_k = 1$, otherwise $v_k\epsilon = 0$, $\epsilon = 0.000001$

$V_k$ is used to approximate the partial derivative of each individual parameter in $\theta$, which is used to form the overall gradient. This is a bit abstract to visualize since the dimension of the gradient depends on the number of parameters in $\theta$. Our objective function is the negative utility of the investor, that is minimized over $\theta$.

RESULTS

The neural network showcased and trained for this thesis uses a market with fads trajectories using parameters, $\rho = 0.4, \sigma = 0.4$ and $\mu = 0.12$. It is trained for $t = 0.5$, meaning an investor can input the current wealth held at halfway through the time horizon to receive the optimal allocation strategy for that moment. The investor is taken as gaining logarithmic utility from wealth. We train using 500 iterations, or epochs, each creating a batch size of 100 fads trajectories, for a neural network with a single hidden layer of 10 nodes.
Figure 4.2: Sample Visual of Trained Neural Network

Each black line represents a parameter in $\theta$. We use the sigmoid activation function for our nodes since it limits the output to be between 0 and 1, which is necessary for our allocation.

Figure 4.3: Sigmoid Activation Function

As seen, the sigmoid function applies non-linearity and is what allows the neural network to learn complex response surfaces. We now look at a comparison between the trained neural network’s input and output.
Figure 4.4: Output Results of Trained Neural Network

We can see that the final result ends up being relatively constant. This is significant because the neural network learned to provide the same allocation strategy for any amount of wealth at time $t = 0.5$. This is especially significant because the optimal allocation strategy for a CRRA class of utility is to keep a constant part of your wealth invested in a risky asset (stock) regardless of wealth. That means that by using logarithmic utility for the investor, the machine learning algorithm decided that the best allocation strategy was to keep a constant part of wealth invested into the stock regardless of wealth, matching the theoretical results. This shows that using neural networks can be a sufficient alternative to approaching this optimization problem.
Conclusion

We have seen the analytical construction of Merton’s portfolio optimization problem, compared theoretical results to simulated results, and approached the problem using a machine learning algorithm. The largest contribution of this thesis is considered to be the machine learning approach to the problem which yielded significant results. It shows the power that machine learning can have when modeling unknown results, and can definitely be a viable method of approaching problems like Merton’s. This is especially relevant in scenarios where the utility gained by an investor potentially falls outside of the mentioned CRRA class types of utilities. This would make the problem extremely difficult to solve analytically, meaning the investor would have to resort to using other approaches, like using machine learning to find the optimal allocation.
Future Research Topics

1. Consider Merton’s portfolio optimization problem with multiple stocks and bonds.
2. Include transaction costs, proportional or fixed.
3. Increase the types of markets or utility modeled.
4. Approach the optimization problem using utilities outside of the CRRA class type.
5. Let intermediate consumption affect utility, i.e. the investor gains excess utility from consuming early, or loses utility from consuming too much.
6. Use other types of machine learning models, i.e. Random Forests, Deep Neural Models, or Support Vector Machines.
7. Explore scenarios where MR is not constant (depends on current wealth).
References


Historical Data of Northern Oil and Gas, Inc. (NOG). *Yahoo! Finance.*


Toprak, M. Activation Functions for Deep Learning, Medium Web Blog.

https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning-13d8b9b20e, 2020
R-Code:

#functions used for dynamic programming

#Ornstein-Uhlenbeck white noise process
#n discretization steps(N_t), m trajectories(N_x), epsilon OU constant,
#tx point in time(t_lat), dt discretization division

OU = function(n,m,epsilon,tx,dt){
  z1 = rnorm(n*m)
  z1 = matrix(z1,nrow = n)
  z1[1,] = 0
  c = exp((-1)/(epsilon)*(tx))
  p1 = exp((1/epsilon)*tx)
  p2 = matrix(p1,n,m)
  p3 = z1*sqrt(dt)
  p4 = p2 * p3
  OUP = c * apply(p4,2,cumsum)
  return(OUP)
}

#Black-Scholes trajectories of stock prices using geometric brownian motion

#n dicretization steps(N_t), m trajectories(N_x), mu drift, sigma volatility,
#TH time horizon, dt discretization division, s0 initial wealth

BlackScholes = function(n,m,mu,sigma,dt,s0){
  z = rnorm((n*m))
  z = matrix(z,nrow = n)
\begin{equation}
    BS = ((\mu - (\frac{1}{2}) \sigma^2) dt) + \sigma \sqrt{dt} \cdot z
\end{equation}

\begin{verbatim}
BS[1,] = 0
BS = s0 * exp(apply(BS, 2, cumsum))
BS[1,] = s0
return(BS)
\end{verbatim}

#construction of Fads trajectories using Ornstein-Uhlenbeck
#n discretization steps(N_t), m trajectories(N_x), epsilon OU constant,
#sigma volatility, mu drift, rho correlation of Brownian motion
#tx point in time(t_lat), dt discretization division, s0 initial wealth
Fads = function(n,m,epsilon,sigma,mu,rho,tx,dt,s0){

    #OU process
    z1 = rnorm(n*m)
    z1 = matrix(z1,nrow = n)
    z1[1,] = 0
    c = exp((-1)/(epsilon)*(tx))
    p1 = exp((1/epsilon)*tx)
    p2 = matrix(p1,n,m)
    p3 = z1*sqrt(dt)
    p4 = p2 * p3
    OUF = c * apply(p4,2,cumsum)

    #Fads
    z2 = rnorm(n*m)
    z2 = matrix(z2,nrow = n)
    St = matrix(0,n,m)
    St[1,] = s0

\end{verbatim}
for (nx in 1:(n-1)){
    St[(nx+1),] = St[nx,] + St[nx,]*(mu+OUF[nx,])*dt +  
    sigma*rho*sqrt(dt)*z2[nx,] + sigma*sqrt(1-rho^2)*sqrt(dt)*z1[nx,])
}
return(St)
}

#log theoretical utility functions

#true maximal expected utility and Merton ratio for log utility
#mu drift, sigma volatility, r interest rate, v0 initial wealth
tvlog = function(mu,sigma,r,v0){
    beta = (mu-r)/sigma
    op = beta/sigma
    if (op < 0){op = 0}
    if (op > 1){op = 1}
    ov = log(v0)+(r+((mu-r)/(2*(sigma^2))))
    loglist = list(ov,op)
    names(loglist) <- c("Theoretical Optimal Value of Utility","Theoretical Merton Ratio Allocation")
    return(loglist)
}

#expected utility of final wealth using log utility from theoretical merton-ratio without intermediate consumption
#S stock price process, mu drift, sigma volatility, m trajectories(N_x),
#n discretization steps(N_t), v0 initial wealth,
#tx point in time(t_lat), r interest rate
ExpUtLog = function(S,mu,sigma,m,n,v0,tx,r){
ovaop1 = tvlog(mu,sigma,r,v0)
op1 = ovop1[[2]]
x = matmerton(S,op1,v0,n,tx,r,m)[N_t,]
EUL = mean(log(x))
names(EUL) <- c("Simulated Expected Utility from Theory")
return(EUL)
}

#power theoretical utility functions

#true maximal expected utility and Merton ratio for power utility
#mu drift, sigma volatility, r interest rate, v0 initial wealth, alpha exponent
tvpow = function(mu,sigma,r,v0,alpha){
    op = (mu)/((1-alpha)*(sigma^2))
    if (op < 0){op = 0}
    if (op > 1){op = 1}
    cons = ((mu^2)*alpha)/(2*(sigma^2)*(1-alpha))
    ov = ((v0^alpha)/alpha)*(exp(cons))
    powlist = list(ov,op)
    names(powlist) <- c("Theoretical Optimal Value of Utility","Theoretical Merton Ratio Allocation")
    return(powlist)
}

#expected utility of final wealth using power utility from theoretical merton-ratio without intermediate consumption
#S stock price process, mu drift, sigma volatility, m trajectories(N_x),
#n discretization steps(N_t), v0 initial wealth,
#tx point in time(t_lat), r interest rate, alpha exponent

ExpUtPow = function(S,mu,sigma,m,n,v0,tx,r,alpha){
  ovop2 = tvpow(mu,sigma,r,v0,alpha)
  op2 = ovop2[[2]]
  x = matmerton(S,op2,v0,n,tx,r,m)[N_t,]
  EUP = mean(((x^alpha)/alpha))
  names(EUP) <- c("Simulated Expected Utility from Theory")
  return(EUP)
}

#returns wealth when the stock prices follows the process S, no intermediate consumption

#set S = BlackScholes/Fads function before calling to run process wanted
#to get final wealth, call merton[N_t]
#mr Merton ratio, r interest rate, v0 initial wealth
#n discretization steps(N_t), tx point in time(t_lat), r interest rate
merton = function(S,mr,v0,n,tx,r){

  B  = matrix(exp(r*tx))
  NB = (1-mr)*(v0/B[1])
  NS = mr*v0/S[1]
  wealth = matrix(1,n)
  wealth[1] = v0
  for (i in 1:n-1){
    wealth[i+1] = (NB*B[i+1])+(NS*S[i+1])
    NB[i+1] = ((1-mr)*wealth[i+1]) / B[i+1]
    NS[i+1] = (mr * wealth[i+1]) / S[i+1]
  }
  return(wealth)
# can input a matrix for S instead of a string
# m trajectories (N_x)
matmerton = function(S,mr,v0,n,tx,r,m){
  b = exp(r*tx)
  B  = matrix(b,n,m)  # bond value at t
  NB = (1-mr)*(v0/B[1])
  NS = mr*v0/S[1]
  wealth = matrix(1,n,m)
  wealth[1,] = v0
  for (i in 1:n-1){
    wealth[i+1,] = (NB*B[i+1,])+(NS*S[i+1,])  # wealth at time i
    NB = ((1-mr)*wealth[i+1,]) / B[i+1,]  # rebalancing bond
    NS = (mr * wealth[i+1,]) / S[i+1,]  # rebalancing stock
  }
  return(wealth)
}

# returns wealth over time when the stock prices follows the process S
# and intermediate consumptions are included
# N_w number of intermediate times client consumes during time horizon
# mr Merton ratio, r interest rate, v0 initial wealth, m trajectories (N_x)
# n discretization steps (N_t), tx point in time (t_lat), r interest rate
consmatmer = function(S,N_w,mr,v0,n,tx,r,m,sm,consr){
  b = exp(r*tx)
  B  = matrix(b,n,m)
  NB = (1-mr)*(v0/B[1])
  NS = mr*v0/S[1]
wealth = matrix(1,n,m)
wealth[1,] = v0
conval = matrix(0,N_w+1,N_x)

# partitions with intermediate consumptions
for (j in 1:N_w){
    for (i in ((n/(N_w+1))*(j-1)+1):((n/(N_w+1))*j)-1)){
        wealth[i+1,] = (NB*B[i+1,])+(NS*S[i+1,])
        NB = ((1-mr)*wealth[i+1,]) / B[i+1,]
        NS = (mr * wealth[i+1,]) / S[i+1,]
    }
    # intermediate consumptions
    wealth[((n/(N_w+1))*j)],] =
    ((NB*B[((n/(N_w+1))*j)+1),]+(NS*S[((n/(N_w+1))*j)+1),]))*(1-
    (consr(wealth[((n/(N_w+1))*j)],v0,sm)))
    conval[j,] =
    ((NB*B[((n/(N_w+1))*j)+1),]+(NS*S[((n/(N_w+1))*j)+1),))*(consr(wealth[((n/(N_w+1))*j)],v0,sm))
    NB = ((1-mr)*wealth[((n/(N_w+1))*j)+1],) / B[((n/(N_w+1))*j)+1],]
    NS = (mr * wealth[((n/(N_w+1))*j)+1],) / S[((n/(N_w+1))*j)+1],]
}
# final partition
for (i in (((n/(N_w+1))*N_w)+1):n-1){
    wealth[i+1,] = (NB*B[i+1,])+(NS*S[i+1,])
    NB = ((1-mr)*wealth[i+1,]) / B[i+1,]
    NS = (mr * wealth[i+1,]) / S[i+1,]
}
conval[N_w+1,] = wealth[n,]
conmat = list(wealth,conval)
names(conmat) <- c("Wealth","Consumption")
return(conmat)
}

#Simulation for both Power and Log utilities

#finds simulated optimal constant allocation strategy for log and power utility

#S stock price process, N_t discretization steps, N_x trajectories simulated,

#v0 initial wealth, r interest rate, alpha exponent, N_w number of intermediate consumptions

OptSCA = function(S,N_t,N_x,v0,r,alpha,N_w,sm,consr){
  ut1 = matrix(0,1000)
  ut2 = matrix(0,1000)
  s = c()
  if(N_w >= 1){
    for (i in 1:1000){
      a=i/1000
      x = consmatmer(S,N_w,a,v0,N_t,t_lat,r,N_x,sm,consr)[[2]]
      for(z in 1:N_x){
        s[z] = sum(x[,z])
      }
      ut1[i] = mean(log(s))
      ut2[i] = mean((s^alpha)/alpha)
    }
  }else{
    for (i in 1:1000){
      a=i/1000
      x = matmerton(S,a,v0,N_t,t_lat,r,N_x)
    }
  }
}
\[ s = x[N_t,] \]
\[ ut1[i] = \text{mean}(\log(s)) \]
\[ ut2[i] = \text{mean}((s^{\alpha})/\alpha) \]

\}
\}
\]
b = \text{which.max}(ut1)
const1 = b/1000
c = \text{which.max}(ut2)
const2 = c/1000
d = \text{list}(\text{max}(ut1),\text{const1}, \text{max}(ut2),\text{const2})

\text{names}(d) \text{<-} c("Sim. Value of Log Utility","Sim. Optimal Log Constant Allocation","Sim. Value of Power Utility","Sim. Optimal Power Constant Allocation")

\text{return}(d)
}

#function that returns several outputs
#finds expected utility from trajectories
#also returns theoretical MR, and simulated results

#investor provides basic personal investing information
#returns theoretical and simulated expected utility of consumed/final wealth

# U type of utility = {1 for Logarithmic, 2 for Power}
#S stock prices, v0 initial wealth, alpha exponent for power utility
#sm minimum consumption cutoff set by client, N_w number of intermediate consumptions
#consr consumption pattern
Utres = function(S,U,v0,alpha =0.1,sm =0,N_w=0,consr=0) {

c = c()
u = c()

#selecting theoretical optimal allocation
if(U == 1){
    ovop = tvlog(mu, sigma, r, v0)
} else{
    ovop = tvpow(mu, sigma, r, v0, alpha)
}

ov = ovop[[1]]
op = ovop[[2]]

#obtaining final wealth and consumption amounts from theoretical allocation
if(N_w >= 1){
    x = consmatmer(S, N_w, op, v0, N_t, t_lat, r, N_x, sm, consr)[[2]]
    for(z in 1:N_x){
        u[z] = sum(x[,z])
    }
    for(q in 1:(N_w+1)){
        c[q] = mean(x[q,])
    }
    AC = c[1:N_w]
    AFW = c[(N_w+1)]
} else{
    x = matmerton(S, op, v0, N_t, t_lat, r, N_x)
    u = x[N_t,]
    AFW = mean(x[N_t,])
}
#finding utility from consumed/final wealth

if(U == 1){
    EU = mean(log(u))
} else {
    EU = mean(((u^alpha)/alpha))
}

#simulated results
SIM = OptSCA(S, N_t, N_x, v0, r, alpha, N_w, sm, consr)
if(U == 1){
    SIM1 = SIM[[2]]
    SIM2 = SIM[[1]]
} else {
    SIM1 = SIM[[4]]
    SIM2 = SIM[[3]]
}

#Returns
if(N_w >= 1){
    EUL = list(op, AC, AFW, EU, SIM1, SIM2)
    return(EUL)
} else {
    EUL = list(op, AFW, EU, SIM1, SIM2)

return(EUL)

}

#independent variables for modeling

s0=100 #initial stock price
everpsilon=0.01 #constant for OU process
rho= 0.4 #correlation of Brownian motion
mu=0.12 #drift
sigma=.4 #volatility
r =0.05 #interest rate
TH=1 #time horizon

#discretization

N_x = 5000      #trajectories
N_t = 1200    #number of discretization steps, divisible by (N_w+1)
preferably
dt = TH / (N_t-1)
t_lat = seq(0,TH, by=dt)

#default variables given by client for graphing

v0=500 #initial wealth
sm = 200 #minimum savings
N_w = 11 #number of intermediate consumptions
alpha = 0.1 #exponent for power utility/ gamma in thesis

#standard intermediate consumption function given by client
consrdefault = function(wealth,v0,sm){
  #c_w = c(0:length(wealth))
  c_w = c()
  for (p in 1:length(wealth)){
    if (wealth[p] >= (v0)){
      c_w[p] = .2
    } else if(wealth[p] <= sm){
      c_w[p] = 0
    } else {
      c_w[p] = .1
    }
  }
  return(c_w)
}

#Ornstein-Uhlenbeck plot for white noise process
OU1 = OU(N_t,N_x,epsilon,t_lat,dt)
domain = t_lat
plot(domain,OU1[,1],type = "l",ylim = c(-1,1),
     xlab = "Time Horizon",ylab = "Brownian Motion", main = "Ornstein-Uhlenbeck")
for (p in 2:N_x){
  lines(domain,OU1[,p],type = "l", col=p)
}

#Black Scholes processes plots for stock prices
BS1 = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
domain = t_lat
plot(domain,BS1[,1],type = "l",ylim = c(0,s0*3),
    xlab = "Time Horizon",ylab = "Risky Asset Prices",
    main = "Black Scholes, sig = 0.2, mu = 0.24")
for (p in 2:N_x){
    lines(domain,BS1[,p],type = "l", col=p)
}

#Pads model plot for stock prices
FA = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
domain = t_lat
plot(domain,FA[,1],type = "l",ylim = c(0,s0*3),
    xlab = "Time Horizon",ylab = "Risky Asset Prices",
    main = "Fads, epsilon = 0.01")
for (p in 2:10){
    lines(domain,FA[,p],type = "l", col=p)
}

#parameter estimation
library(readxl)
library(tseries)
library(astsa)
stock <- read_excel("C:/Users/Pablo/Desktop/Thesis/stats/NOG.xlsx")
View(stock)
stock_close = ts(stock$Close)
plot(stock_close,xlab = "May 2021 to May 2022",ylab = "Price",
    main = "NOG Stock Price Over Previous Year, Ending in 05-06-2022")
acf2(stock_close)
xtk = diff(log(stock_close))
acf2(xtk)

#Model
f10 <- arima(xtk, order=c(0,0,0))
f10
#BIC
BIC(f10)
#standardized residuals
tsdia(f10)
qqn(f10$residuals)
qqline(f10$residuals)

mu_parameter = mean(xtk)
mu_parameter
sigma_parameter = mean(((xtk - mu_parameter)^2))
sigma_parameter

sigma_estimate = sqrt(sigma_parameter * (length(xtk)))
sigma_estimate
mu_estimate = (mu_parameter * (length(xtk))) + ((1/2) * ((sigma_estimate)^2))
mu_estimate

#Trajectory confidence interval
S = BlackScholes(N_t,N_x,mu_estimate,sigma_estimate,dt,42)
#S = Fads(N_t,N_x,epsilon,sigma_estimate,mu_estimate,rho,t_lat,dt,s0)
smean = c()
smed = c()
stop95 = c()
sbot95 = c()
for (i in 1:N_t){
smean[i] = mean(S[i,])
smed[i] = median(S[i,])
cint = ((.05 * N_x)/2) +1
ssort = sort(S[i,])
stop95[i] = tail(ssort,cint)[1]
sbot95[i] = head(ssort,cint)[cint]
}
domain = t_lat
ylimit = c(0,42*3)
plot(domain,smean,type = "l",ylim = ylimit,
  xlab = "Time Horizon",ylab = "Risky Asset Prices",
  main = "NOG Market Trajectories", lwd = 2.0)
lines(domain,smed,lty = 2, col=2, lwd = 2.0)
lines(domain,stop95,lty = 2, col=4, lwd = 2.0)
lines(domain,sbot95,lty = 2, col=4, lwd = 2.0)
legend(0, ylimit[2], legend=c("10 Stock Trajectories","Mean", "Median",
  "Prediction Interval"),
  col=c("green", "black","red","blue"), lty=c(1,1,2,2), cex=1.5)
for (p in 3:12){
  lines(domain,S[,p],type = "l", col=3)
}
time_check = 1 * TH * N_t
stop95[time_check]
smean[time_check]
smed[time_check]
sbot95[time_check]
# log utility sample

```r
x = 1:1000
plot(x,log(x),type = "l", xlab = "Wealth",ylab = "Utility from Wealth", main = "Logarithmic Utility Preview")
```

# Merton ratio plot for wealth over time

```r
# mr follows theoretical log utility
S = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
tvlog(mu,sigma,r,v0)

mr = tvlog(mu,sigma,r,v0)[[2]]
FW = matmerton(S,mr,v0,N_t,t_lat,r,N_x)
domain = t_lat
plot(domain,FW[,1],type = "l",ylim = c(0,v0*3),
     xlab = "Time Horizon",ylab = "Wealth",
     main = "Wealth using MR Allocation - BlackScholes")
for (p in 2:N_x){
    lines(domain,FW[,p],type = "l", col=p)
}
```

# MR plot with intermediate consumption

```r
S = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
mr = tvlog(mu,sigma,r,v0)[[2]]
FWC = consmatmer(S,N_w,mr,v0,N_t,t_lat,r,N_x,sm,consrdefault)[[1]]
domain = t_lat
plot(domain,FWC[,1],type = "l",ylim = c(0,v0*2),
     xlab = "Time Horizon",ylab = "Wealth",
```
main = "Wealth with Monthly Consumption - BlackScholes")

for (p in 2:N_x){
    lines(domain,FWC[,p],type = "l", col=p)
}

#plot of simulated allocation
S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
#S = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
utsa1 = matrix()
utsa2 = matrix()
for (i in 1:1000){
    a = i/1000
    x = matmerton(S,a,v0,N_t,t_lat,r,N_x)
    s = x[N_t,]
    utsa1[i] = mean(log(s))
    utsa2[i] = mean((s^alpha)/alpha)
}
par(mfrow = c(1,2))
plot(utsa1,type = "l", xlab = "Allocation Strategy, Index/1000",ylab = "Utility",
     main = "Logarithmic Utility from Simulated Allocation") #log utility
plot(utsa2,type = "l", xlab = "Allocation Strategy, Index/1000",ylab = "Power Utility",
     main = "Power Utility from Simulated Allocation") #power utility

#log utility clients
#S = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
Utres(S,1,500,,)

60
Utres(S,1,500,,200,3,consrdefault)
Utres(S,1,500,,200,11,consrdefault)

#power utility sample
x = 1:1000
alphasamp = 0.5
y= (x^alphasamp)/alphasamp
plot(x,y,type = "l", xlab = "Wealth",ylab = "Utility from Wealth",
     main = "Power Utility Preview, gamma = 0.5")

#power utility clients
S = BlackScholes(N_t,N_x,mu,sigma,dt,s0)
S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
Utres(S,2,500,0.01,,)
Utres(S,2,500,0.1,,)
Utres(S,2,500,0.5,,)

mu = mu_estimate
sigma = sigma_estimate
Utres(S,1,500,,,)
Utres(S,2,500,0.1,,)

#neural network code
library(neuralnet)
library(plotly)
library(sigmoid)
library(sgd)
# independent variables for modeling

s0=100 # initial stock price
epsilon=0.01 # constant for OU process
rho= 0.4 # correlation of Brownian motion
mu=0.12 # drift
sigma=.4 # volatility
r =0.05 # interest rate
TH=1 # time horizon

# discretization

N_x = 100 # trajectories
N_t = 500 # number of discretization steps, divisible by (N_w+1) preferably
dt = TH / (N_t-1)
t_lat = seq(0,TH, by=dt)

# default variables given by client for graphing

v0=500 # initial wealth
sm = 200 # minimum savings
N_w = 11 # number of intermediate consumptions
alpha = 0.1 # exponent for power utility/ gamma in thesis

# set neural network size
layer_size = c(1,10,1) # nodes: input, hidden, output

# randomly initialize parameters
initializeParameters <- function(list_layer_size){

n_x <- list_layer_size[1]
n_h <- list_layer_size[2]
n_y <- list_layer_size[3]

W1 <- matrix(runif(n_h * n_x), nrow = n_h, ncol = n_x, byrow = TRUE) * 0.01
b1 <- matrix(runif(n_h), nrow = n_h) * 0.01
W2 <- matrix(runif(n_y * n_h), nrow = n_y, ncol = n_h, byrow = TRUE) * 0.01
b2 <- matrix(runif(n_y), nrow = n_y) * 0.01

params <- list("W1" = W1,
                "b1" = b1,
                "W2" = W2,
                "b2" = b2)

return (params)
}

#transforms parameters into a vector
flatten = function(params){
  vec = c(params$W1, params$b1, params$W2, params$b2)
  return(vec)
}

#transform flat vector back into matrices
unflatten = function(vect, list_layer_size){

n_x <- list_layer_size[1]
n_h <- list_layer_size[2]
n_y <- list_layer_size[3]

W1 <- matrix(vect[1:(n_h * n_x)], nrow = n_h, ncol = n_x, byrow = TRUE)
step = (n_h * n_x)
b1 <- matrix(vect[(step+1):(step+n_h)], nrow = n_h)
step = step+n_h
W2 <- matrix(vect[(step+1):(step+(n_y * n_h))], nrow = n_y, ncol = n_h, byrow = TRUE)
step = step+(n_y * n_h)
b2 <- matrix(vect[(step+1):(step+n_y)], nrow = n_y)

params <- list("W1" = W1,
               "b1" = b1,
               "W2" = W2,
               "b2" = b2)

return (params)
}

# sigmoid function so values end up between 0 and 1
sigmoid <- function(x){
    return(1 / (1 + exp(-x)))
}

forwardPropagation <- function(X, params, list_layer_size){
#m = dim(X)[2]
n_h <- list_layer_size[2]
n_y <- list_layer_size[3]

W1 <- params$W1
b1 <- params$b1
W2 <- params$W2
b2 <- params$b2

#b1_new <- matrix(rep(b1, m), nrow = n_h) # if more than one input nodes
#b2_new <- matrix(rep(b2, m), nrow = n_y)

Z1 <- W1 %*% X + b1
A1 <- sigmoid(Z1)
Z2 <- W2 %*% A1 + b2
A2 <- sigmoid(Z2)

cache <- list("Z1" = Z1,
              "A1" = A1,
              "Z2" = Z2,
              "A2" = A2)

return (cache)
}

#provides utility from flattened parameters
NNobjective = function(S, flat_params, list_layer_size, Xt){

par = unflatten(flat_params, list_layer_size)
mr = as.numeric(forwardPropagation(Xt, par, list_layer_size)["A2"])  
x = matmerton(S, mr, v0, N_t, t_lat, r, N_x)[N_t,]  
x_mean_final = mean(x)  
    ut = -mean(log(x)) # N_t  
    return(list(ut, x_mean_final))

# Finds the derivatives of current parameters  
NNgrad = function(Xt, S, params, list_layer_size, epsilon){

    flat_params <- flatten(params)  
    grad = rep(0, length(flat_params))  
    for (k in 1:length(flat_params)){
        call_param = rep(0, length(flat_params))  
        call_param[k] = 1  
        eps_param = epsilon * call_param  
        grad_param = flat_params + eps_param  

        grad[k] = (as.numeric(NNobjective(S, grad_param, list_layer_size, Xt)[[1]]) -
                    as.numeric(NNobjective(S, flat_params, list_layer_size, Xt)[[1]]))/epsilon
    }
    grad2 = unflatten(grad, list_layer_size)  
    return(grad2)
}

updateParameters <- function(grads, params, learning_rate){

    W1 = params$W1  
    b1 = params$b1
W2 = params$W2  
b2 = params$b2

dW1 = grads$W1  
db1 = grads$b1  
dW2 = grads$W2  
db2 = grads$b2

W1 = W1 - learning_rate * dW1  
b1 = b1 - learning_rate * db1  
W2 = W2 - learning_rate * dW2  
b2 = b2 - learning_rate * db2

updated_params <- list("W1" = W1,  
                      "b1" = b1,  
                      "W2" = W2,  
                      "b2" = b2)

return (updated_params)
}

NNfunct = function(tp,v0,epochs,lr,list_layer_size,eps){

    init_params <- initializeParameters(layer_size)
    tp = tp * TH * N_t  #tp is t time period in time horizon
    Xt = v0
    VA2 = c()
    x_t = c()
    uti = c()
final_mean = c()

for (i in 1:epochs) {
    S = Fads(N_t,N_x,epsilon,sigma,mu,rho,t_lat,dt,s0)
    flat_p = flatten(init_params)
    obj = NNobjective(S,flat_p,list_layer_size,Xt)
    final_mean[i] = obj[[2]]
    uti[i] = -obj[[1]]
    fwd_prop <- forwardPropagation(Xt, init_params, layer_size)
    VA2[i] = as.numeric(fwd_prop["A2"])
    back_prop <- NNgrad(Xt,S,init_params,list_layer_size,eps)
    update_params <- updateParameters(back_prop, init_params,
                    learning_rate = lr)
    init_params <- update_params
    x_t[i] = Xt
    Xt = mean(matmerton(S,(VA2[i]),v0,N_t,t_lat,r,N_x)[tp,])
}

model_out <- list("Updated Parameters" = update_params,"Output over time"=VA2, "FINAL WEALTH at EPOCHS"=x_t, "Utility over EPOCHS" =uti)
return (model_out)

}

EPOCHS = 500
LEARNING_RATE = 0.2
eps = 0.000001

TrainedNN = NNfunct(.5,v0,EPOCHS, lr = LEARNING_RATE, layer_size,eps)
TrainedNN
merton = as.data.frame(TrainedNN["Output over time"])
wealth = as.data.frame(TrainedNN["FINAL WEALTH at EPOCHS"])
utility = as.data.frame(TrainedNN["Utility over EPOCHS"])

plot(1:EPOCHS,merton[,1], type = 'l', xlab = 'EPOCHS',ylab = 'MERTON RATIO', main = 'NN Merton Ratio', ylim = c(0,1))
ratio = rep(mean(merton[(EPOCHS/2):(EPOCHS),1]),EPOCHS)
lines(1:EPOCHS, ratio, col = 2, lty = 2)
ratio[1]

plot(1:EPOCHS,wealth[,1], type = 'l', xlab = 'EPOCHS',ylab = 'WEALTH', main = 'NN WEALTH')

plot(1:EPOCHS,utility[,1], type = 'l', xlab = 'EPOCHS',ylab = 'UTILITY', main = 'NN UTILITY')

trained_params = TrainedNN$'Updated Parameters'
trained_params
wmr = c()
for (i in 1:500){
    wmr[i] = forwardPropagation((300+i),trained_params,layer_size)["A2"]
}
wmr

plot(401:900,wmr, type = 'l', xlab = 'WEALTH',ylab = 'MERTON RATIO', main = 'NN INPUT VS OUTPUT')
plot(401:900,wmr,ylim = c(0,1), type = 'l', xlab = 'WEALTH',ylab = 'MERTON RATIO', main = 'NN INPUT VS OUTPUT')
Vita

Pablo Avalos migrated to El Paso, Texas, from Mexico, with his family, when he was four years old. Raised here in El Paso, Pablo focused on his education, hoping to make the most out of the opportunity offered to him.

He attended Mission Early College High School, where he was nominated to be a part of the Student Board of Directors in the Federal Reserve Bank for the class of 2014. He also participated in several extra-curricular activities during high school, such as competing for accounting UIL and completing economic research for Business Professionals of America. He completed dual-credit courses that allowed him to graduate with his Associate’s of Arts from El Paso Community College while he was still a junior in high school.

Pablo then moved to completing a Bachelor’s of Science degree at UTEP, majoring in actuarial mathematics with a minor in economics. During his undergraduate studies, he worked as a research assistant for the Bioinformatics program of UTEP, and as a mathematics tutor in an independent tutoring center.

After completing his undergraduate degree, he took some time to gain some work experience, working as an insurance representative for a life insurance firm, then as an account specialist in a finance company, eventually returning to UTEP to further his education.

Pablo’s professional interests range from mathematics and statistics to economics and finance. He hopes to use the knowledge of the Master’s degree of Science in Statistics to complete actuarial preliminary exams and go into a career of actuarial sciences.

Contact Information: contactPabloA@gmail.com