Combining (Deep) Reinforcement Learning with Goal-Based Investment

Eric Sun, Ning Cai, Ziming Mao
New York University Tandon School of Engineering
Department of Finance and Risk Engineering

Advisor: Dr. Cristian Homescu
Bank of America

Abstract

Recent advancements in the efficient implementation of Deep Learning (DL) models provided a new methodology in solving problems in finance. In particular, using Deep Reinforcement Learning (DRL) in portfolio management proved to be a reliable and efficient approach. Goal-Based Investment (GBI) differs from the traditional portfolio management tasks due to the unique investment objective. In this paper, we formulate a GBI problem into a DRL optimization problem and create a custom reward function to be implemented in the Deep Deterministic Policy Gradient (DDPG) model. Finally, we discuss the performance of the model against different benchmarks in Monte Carlo simulation and conclude the GBI model have the highest probability of achieving the investment goal under any market situations.

Keywords: Goal-Based Investment, Deep Reinforcement Learning, Portfolio optimization

1 Introduction

Goal-Based Investing (GBI) is an investment approach where performance is measured by the success of investments in meeting the investor's financial goals (e.g., retirement, education, or vacation home). The objective is to invest systematically in a consistent manner with the investor's risk profile and time horizon of the goals.

Portfolio optimization for GBI can be viewed as an optimal control problem performed within a data-driven world. GBI can be solved via reinforcement learning (RL), a paradigm of learning by trial-and-error, solely from rewards or punishments. The software agent attempts to learn appropriate actions that maximize reward within the constraints. It was shown that RL is good at decision-making, especially at sequential decision problem, and can solve financial applications of intertemporal choice.

Within a deep reinforcement learning (DRL) framework, the RL agents construct and learn their knowledge through deep learning of neural networks. As pointed out by Arulkumaran (2017), implementing deep learning architectures with reinforcement learning algorithms is capable of scaling to previously unsolvable problems. That is because DRL can learn from raw sensors directly as input. Therefore, using DRL in investment management could be an efficient solution to the traditional portfolio management problem.

This research aims to apply RL and DRL to GBI by setting up an investment framework that transforms requests of clients into practical portfolio constructions and automated investment executions. Common problems in RL such as the selection of agents and dimension reduction of action space will also be addressed within the scope of GBI.

2 Literature review

Since the inception of modern portfolio theory devised in the 1950s by Harry Markowitz, portfolio managers have been constructing different portfolios corresponding to different financial needs. As GBI gains popularity after the 2008 subprime mortgage crisis, many pieces of research are done

to examine both the theoretical and practical aspects of GBI. Brunel (2011) discusses how to conduct GBI in practice. The author highlights the change in clients' perception of risk and how to set up an investment framework while keeping the goal and business challenges in mind. In particular, he argues that any investment process should start with a thorough understanding of the investor problem. Similarly, Kim et al. (2019) formulate the GBI problem into a multi-stage stochastic programming problem and demonstrates portfolio rebalancing techniques under the GBI framework. We aim to recast the GBI ideas in these articles under a reinforcement learning framework to automate the portfolio optimization process.

RL is also a hot topic in both academia and the industry. As illustrated by Das et al. (2018), a widely used dynamic programming known in RL literature is solving the "planning problem", which works backward from all values of mean-variance pairs to achieve the best outcomes. However, despite much recent interest in RL, little work has been done to apply these techniques in GBI. In the finance industry, the most common application of RL is to identify trading signals. Ponomareva et al. (2019) use the asynchronous advantage actor-critic (A3C) (Actor-Critic) model to construct an algorithmic trading framework. This research uses a model-based approach by setting the state space as a Long Short-Term Memory (LSTM) model while limiting the action space to be discrete and narrow. Backtesting of this model suggests some handsome returns. The methodology of this research, especially the model-based approach and limiting the action space, could be helpful for our research. However, as the RL agents in this research only address the position but not amount, this approach could not be implemented directly into GBI. Similar research done by Qi (2018) also suggests that by replacing the penultimate layer of Deep Q-Network with an LSTM layer, a Deep Recurrent Q-Network could be created and implemented in trading financial products.

Another topic in finance where RL and DRL have been applied is portfolio management. While this topic ties in closely with our research of DRL in GBI, most current research only deals with the rudimentary portfolio constructions, which makes the risk management portion of the problem trivial. The models in most researches are constructed to find the optimal portfolio that attempts to maximize return, while the priority of GBI might be different. Qi (2018) applies Deep Deterministic Policy Gradient (DDPG) with Actor-Critic which concurrently learns a Q-function

and a policy. This model diversifies risk by using dropouts and limiting the weight of individual stock. Dixon et al. (2020) suggest applying a G-Learner to the setting of Inverse Reinforcement Learning (IRL) where rewards collected by the agent are not observed but inferred. These researches provide a solid ground for our paper.

Two common problems faced by researchers in RL are setting the action space and reward function. When applying RL to GBI, these problems magnifies due to the increase in model complexity. To avoid the explosion of action space, efforts must be made to reduce its dimension. Dulac-Arnold et al. (2016) suggested methods such as approximate nearest-neighbor, bucketing of actions, and action generalization to deal with large action spaces. The also highlights methods of transforming between discrete and continuous action spaces, which might be useful for our research. On the other hand, the reward function of an RL model in GBI is more complicated than normal portfolio management because of the inclusion of multiple constraints and sub-goals. Shelton (2001) examines methods of balancing multiple sources of reward in RL. Although this is a theory-heavy paper, it is possible to apply some of the ideas presented in our research.

In section 3 of this paper, we will break down the components of GBI and identify assets classes, benchmarks, and risk management measures. In sections 4 and 5 of this paper, we will construct a reinforcement learning framework and test the feasibility of the above-mentioned models.

3 Goal-Based Investment

3.1 Comparing traditional portfolio management with GBI

Developed by Harry Markowitz in 1950s, the mean-variance framework is one of the most famous and commonly used wealth management tools. Portfolio targets are expressed in terms of expected returns and covariances of risky assets in the portfolio and it constructs a portfolio for risk-averse investors to maximize return given level of market risk. While logic and normative, this approach takes no explicit account of whether the portfolio reaches the goal. It is appropriate for investors who seek to achieve all their goals by investing in a single mean-variance efficient portfolio.

Goal-based Investing offers a more valuable perspective to solve wealth management problems. As a more intuitive method, investment principles are redefined from the viewpoint of the investor rather than the practitioner. We define portfolio efficiency in terms of clients' goals, such as solving children's education or building a retirement plan, rather than focusing on generating the highest possible portfolio return or beating the market. In GBI, we seek to maximize the probability of reaching the goal, portfolio value W will achieve a desired level C at maturity T. This can be seen as a binary option on final wealth of W on strike price C, $\boldsymbol{P}(W > C)$ = E[$\mathbf{1}(W > C)$]. Risk measurement is also based on clients' goals, which is the probability of failing to attain the goal at maturity in this formula. Based on our custom measures of portfolio efficiency and risk, we then create investment solutions by matching each goal with an appropriate strategy rather than creating a single overall portfolio. The investment solution is reevaluated over time, maintaining consistency with new circumstances and changing goals.

While this approach is certainly more complex than traditional methods of wealth management, it offers valuable benefits. Investors should have more confidence in strategies that are explicitly aligned with their own objectives. They should also have a clearer understanding of their risk exposure, which is expressed in terms that relate more directly to the achievement of goals. Even though GBI, probably most tangibly, increases the likelihood of achieving goals, greater confidence and clarity will not prevent disappointment when markets fall; however, the investor should be better prepared for bear markets and more likely to maintain perspective and discipline.

3.2 Selection of assets

As a start, we have 32 indices available to us. The full list is shown in appendix. The data source is Bloomberg. We used data from January 2000 to December 2019 as the training set. To select the assets from the pool of 32, we rank all assets by best return, best Sharpe ratio, and minimum volatility over the entire period. For each of the three indicators, the top two performers are selected for our portfolio. This allows our GBI portfolio to have some low-risk assets and some high-return assets. Along with cash, there are a total of seven asset categories in our portfolio. The final selection is shown in Table 1.

Table 1. Assets selected

Cash	
LBUSTRUU	Bloomberg Barclay US Aggregate Bond Index
LMBITR	Bloomberg Barclays Municipal Bond Index Total Return Index Value Unhedged USD
NDUEEGF	SPDR MSCI Emerging Market UCITS ETF
RU10GRTR	iShares Russell 1000 Growth ETF
S5IFT	S&P 500 Information Technology Index
NDDUPXJ	MSCI Pacific ex Japan UCITS ETF

3.3 Benchmark and risk management measures

We choose two common portfolios as comparisons to our GBI portfolio during the evaluation phase. The benchmark is an equally weighted portfolio based on asset value, which means we invest an equal amount of cash in all assets. This type of portfolio construction is more diversified than others, and, therefore, carries less risk.

The reference portfolio is a Profit and Loss (PnL) based portfolio, where the aim is to generate as much profit as possible. We will generate the PnL portfolio using the same DRL framework as the GBI portfolio, except removing a term in the GBI reward function that encourages the RL agent to beat the goal. This will be explained in detail in section 4.

For risk management purposes, we use two indicators, maximum drawdown (MDD) and Sharpe ratio. MDD is the maximum observed loss from the peak of a portfolio before a new peak is attained. MDD is an indicator of downside risk over a specified period. The standard formula is shown below, obtained from Burghardt et al. (2005).

$$D(T) = \max\left[\max_{t\in(0,T)} X(t) - X(T), 0\right]$$

$$MDD(T) = \max_{\tau\in(0,T)} D(\tau) = \max_{\tau\in(0,T)}\left[\max_{t\in(0,\tau)} X(t) - X(\tau)\right]$$

Sharpe ratio is a common performance measure in investment management, introduced by William Sharpe (1994). It represents the additional amount of return that an investor receives per unit of increase in risk. The formula is obtained from the original paper by Sharpe (1994).

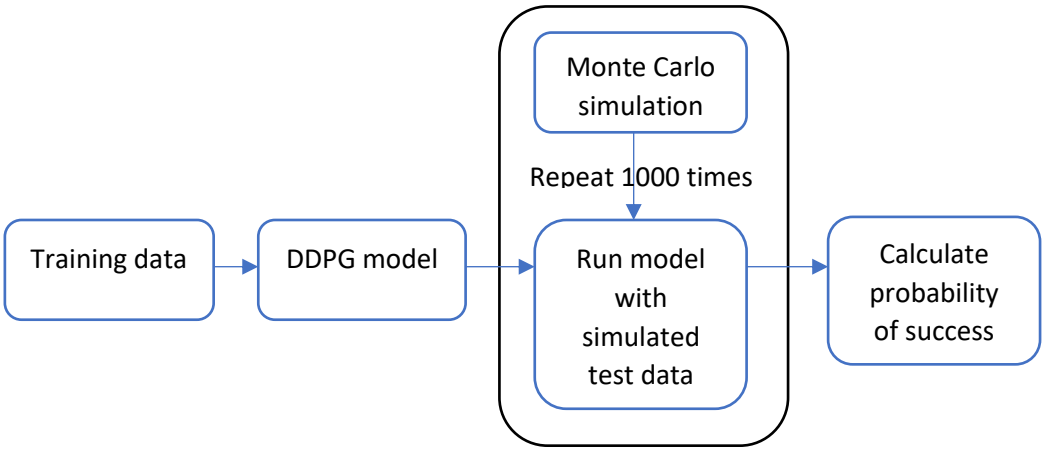
$$S_a = \frac{E[R_a - R_{rf}]}{\sigma_a} = \frac{E[R_a - R_{rf}]}{\sqrt{var[R_a - R_{rf}]}}$$

Both the MDD and the Sharpe ratio will be added in the reward function as risk management terms. MDD will serve as a penalizing term; Sharpe ratio will serve as a reward term. More details will be explained in section 4.

3.4 Monte Carlo simulation

The success of the GBI portfolio is judged by the probability of completing the investment goal. For this project, the goal will be reaching a single investment target, which is defined as the terminal portfolio value at a predetermined time T. As each run of the model only provides one terminal portfolio value, to find out the probability of reaching the goal, we use Monte Carlo simulation to generate large amount of market data and test the performance of the model. For each simulation, we record the binary variable of success/failure to achieve the goal and calculate the probability of reaching the goal using the percentage of simulated results that surpasses the investment goal. The flow is shown in Figure 1.

Figure 1. Simulation process



4 Reinforcement Learning

4.1 Introduction

Reinforcement learning(RL) is a specialized application of machine and deep learning techniques that encourages computer agents to take actions driven by a reward function in an environment that resembles a problem Kaelbling et al.. In the field of finance, the first implementation was made by Moody et al. (1998). Since then, RL is used in many financial applications such as identifying trading signals and portfolio optimization.

4.2 Basic structure

In our GBI problem, we define the following components of RL algorithms: State $s(t)$ is the current value of variables on which action is based. Action $a\big(s(t)\big)$ is the decision making from state $s(t)$. In the field RL in GBI, policy $\pi(s)$ is a path that take actions A from set $\{(\mu_1, \sigma_1), ..., (\mu_K, \sigma_K)\}$, where μ_i and σ_i stand for the return and volatility of underlying assets. The optimal policy $\pi^*(s)$ is the one that maximizes the total expected reward, which in this case is the probability of the final value $W_i(T)$ exceeding initial goal H. And transition probability $p[s(t+h)|\{s(t), a(t)\}]$ is defined as the likelihood of moving to a probabilistic state conditional on the current state $s(t)$ and action $a(t)$. The value function V(s(t)) is defined over the same grid. For our problem, $V(s(T))$ is binary—that is, if $W(T) \geq H$, then $V(s(T)) = 1$, else it is equal to 0. Then, we can use backward iteration to calculate the value function at time $T-h$, since we have populated the value function at time T.

Solving Markov decision process (MDP) problem is one way to identify this optimal policy. A MDP problem can be solved by a Model-based algorithm, where assumes that the state transition probabilities are known. For a given policy π, the transition probabilities are based on State Value Function $V_\pi(s)$, and State-Action Value Function $Q_\pi(s,a)$, $a \in A$. The expectation of total reward starting from state s under policy π is Value Function $V_\pi(s)$, and the expectation of total reward starting form state s and taking action a is State-Action Value Function $Q_\pi(s,a)$.

The MDP can be solved by collecting sample-paths of the state transitions. If the state transition probabilities are not known in advance, and the corresponding rewards can be estimating the optimal State–Action Value Function $Q_\pi(s,a)$ using statistical averaging. Note that the state value function $\mathrm{V}(s(T))$ is no longer useful in the model-free case, since even if it were known, the calculation of the optimal policy still requires knowledge of the state dynamics. On the other hand, once $Q_\pi(s,a)$ is known, the optimal policy can be obtained by doing a simple maximization $\pi(s) = argmax_a\ Q_\pi(s,a)$.

One model free algorithms is Monte Carlo (MC) Learning: This algorithm aims to estimate $\mathrm{Q}(s,a)$ by averaging the total future rewards. Thus, it is an unbiased estimate but subject to a large variance due to the large number of sample paths.

$$\mathrm{Q}(s,a) \leftarrow \mathrm{Q}(s,a) + \ \alpha[R + \gamma Q(s',a') - Q(s,a)]$$

Where the reward R, state s, and action a are seen at time t, and the next period state and action denoted as s' and a', repectively. The parameter α proxies for the 'learning rate' and is usually chosen to be a small value. The 'discount rate' is $\gamma \leq 1$ and it suffices to tradeoff later rewards against earlier ones in an episode. It also helps to set a horizon on the importance of rewards when episodes do not terminate in a short horizon.

We seek to obtain estimates of the optimal State-Action Value Function $Q_\pi(s,a)$ using the generated sample paths. When in state s, the action a is generated according to what is known as an 'epsilon-greedy' policy π.

Using the current policy is known as 'exploitation' and using the random policy generates 'exploration' behavior. Exploration is a key ingredient in RL because it enables better coverage of the state space. The current action a is chosen based on the current policy with probability $(1-\epsilon)$ but with probability ϵ, a random action is chosen. The equation above can, therefore, also be written as:

$$\mathrm{Q}(s,a) \leftarrow \mathrm{Q}(s,a)(1-\alpha) + \ \alpha(\gamma Q(s',a'))$$

where we note that this update equation sets the new value of $\mathrm{Q}(s,a)$ to a weighted average of the current value function and the value function in the next period, and when α is small, the learning is of courses low, but convergence is more stable.

This formula uses the following sample path transitions: Start from state s and take action a (using the ϵ-greedy policy) to generate reward R, followed by a probabilistic transition to state s' from where the action a' is taken, again under the ϵ-greedy policy. This is followed by a version of the policy iteration algorithm, to progressively refine the policy, until it converges to the optimal policy $\pi(s)$.

$$\mathrm{Q}(s,a) \leftarrow \mathrm{Q}(s,a) + \alpha[R + max_{a'}[\gamma Q(s',a') - Q(s,a)]$$

In this equation, Q learning announces that action a is chosen according to the behavior policy while a' is chosen according to the target policy. See that in Q-Learning the policy a' is chosen optimally from highest value function in state s'. Let $\hat{R}_t(x_t, a_t)$ be a random reward collected by the agent for taking action a_t at time t when the state of the environment is x_t. Assume that all future actions a_t for future time steps are determined according to a policy $\pi(a_t|x_t)$ which specifies which action a_t to take when the environment is in state x_t.

For a given policy π, the expected value of cumulative reward with a discount factor γ, conditioned on the current state x_t, defines the value function:

$$\boldsymbol{\acute{V}_t^{\pi}}(x_t) := \mathbb{E}_t^{\boldsymbol{\pi}}[\sum_{t'=t}^{T-1} \gamma^{t'-t} \hat{R}_{t'}(x_t, a_t) | \, x_t]$$

Here $\mathbb{E}_t^{\pi}$ stands for the expectation of future states and actions, conditioned on the current state x_t and policy π.

Let π^* be the optimal policy, i.e. the policy that maximizes the total reward. This policy corresponds to the optimal value function, denoted $\boldsymbol{\acute{V}_t^{\pi}}(x_t)$. The latter satisfies the Bellman Optimality Equation (see e.g. (Sutton and Barto, 2018))

$$\acute{\boldsymbol{V}}_{\boldsymbol{t}}^{\boldsymbol{\pi}}(x_t) = \boldsymbol{max}_{\boldsymbol{a}}\acute{\boldsymbol{\pi}}(x_t|a_t) = \boldsymbol{max}_{\boldsymbol{a}}(\widehat{\boldsymbol{R}}_{\boldsymbol{t}}(x_t|a_t) + \gamma\mathbb{E}_t^{\boldsymbol{\pi}}[\acute{\boldsymbol{V}}_{\boldsymbol{t+1}}^{\boldsymbol{\pi}}(x_{t+1})]$$

$$\acute{\boldsymbol{\pi}}(x_t|a_t) = \widehat{\boldsymbol{R}}_{\boldsymbol{t}}(x_t|a_t) + \gamma\mathbb{E}_t^{\boldsymbol{\pi}}[\acute{\boldsymbol{V}}_{\boldsymbol{t+1}}^{\boldsymbol{\pi}}(x_{t+1})$$

$\acute{\boldsymbol{V}}_{\boldsymbol{t}}^{\boldsymbol{\pi}}(x_t)$ describes the long-term optimization value of a state, that is, the value of this state when all possible subsequent actions are considered and the optimal actions are selected to perform; $\acute{\boldsymbol{\pi}}(x_t|a_t)$ describes the long-term optimal value brought by the performance of an action in a state, that is, after a specific action take place in this state, all possible states in the future are considered, and in these states, the optimal action is always selected to perform the long-term value.

4.3 Objective function

Our objective for the project is to meet client's consumption goal. Therefore, the objective function can be expressed as:

$$MaxProb[W_t \geq C_t]$$

where W_t = cumulative wealth at time t, and C_t = consumption goal at t. There might be multiple consumption goals at different t, but for this project, we are going to evaluate a basic base that there is only one consumption goal at the end of the investment period, time T. This transform the equation into:

$$MaxProb[W_T \geq C_T]$$

To achieve this objective, we alter the reward function to make the RL agent learn in different ways.

4.4 Reward function for GBI portfolio

There are two common ways to construct a reward function in RL. The first approach is setting a simple reward function such as maximizing return and include constraints such as risk factors separately. The second approach is adding constraints directly in the reward function. In this project, we are going to use the second approach.

Now we consider a simplified model for retirement planning which is a research hot spot and direction. What we assume is that the whole process can be divided into *T* steps, which is time horizon. The investment manager keeps the wealth in *N* assets, which contains one risk-free bond and $(N-1)$ risky assets. Without loss of generality, we assume n_1 is the bond and $\boldsymbol{r}_{n1}(t) = r_f$, which is the risk-free rate. With x_t being the vector of investor's hold positions at time t, u_t being the vector of changes in these positions, and r_t being the return of our assets, our goal is the target value $\bar{P}_{t+1}$ at step *t* exceeds the next step value $V_{t+1} = (1+\boldsymbol{r}_t)(\boldsymbol{x}_t + \boldsymbol{u}_t)$ of our portfolio. To achieve the goal, there should be a penalty function for those under-performance ones relative to this target. Considering the infusions, we introduce I_t standing for the installment of money to our capital pool at the beginning of time t. Thus, we can consider the following expected reward for time step *t*:

$$R_t(\boldsymbol{x}_t, \boldsymbol{u}_t, c_t) = -a^{\min(\mathrm{P}_{t+1} - \mathrm{GOAL},\ 0)} + \beta(\mathrm{P}_{t+1} - \mathrm{B}_{t+1}) + \gamma SR + \delta W \times \frac{t}{T} - \theta MDD - \mu I_t \quad (1)$$

where $\alpha, \beta, \gamma, \delta, \theta, \mu$ are constants,

$$\mathrm{P}_{t+1} = \sum_{i=1}^{N} w_{t+1}^i \, x_{t+1}^i$$

$$\mathrm{B}_{t+1} = \frac{1}{N} \sum_{i}^{N} x_{t+1}^i$$

$$SR = Sharpe\ Ratio = \frac{E[P] - R_f}{\sigma_P} = \frac{\frac{1}{t}\sum_{\tau=1}^{t}\sum_{i=1}^{N} w_\tau^i \, x_\tau^i - R_f}{\sqrt{\frac{1}{t}\sum_{\zeta}^{t}(\sum_{i=1}^{N} w_\zeta^i \, x_\zeta^i - \frac{1}{t}\sum_{\tau=1}^{t}\sum_{i=1}^{N} w_\tau^i \, x_\tau^i)^2}}$$

$$W_t = \sum_i w_t^i \, , i^{th} \; asset \; is \; low \; risk \; asset$$

$$MDD = \max_{\tau \in (0,T)} \left[\max_{t \in (0,\tau)} P_t - P_\tau \right] = \max_{\tau \in (0,T)} \left[\max_{t \in (0,\tau)} \sum_{i=1}^{N} w_t^i \, x_t^i - \sum_{i=1}^{N} w_\tau^i \, x_\tau^i \right]$$

P_t: $Portfolio \; return \; from \; step \; t-1 \; to \; t$

B_t: $Benchmark \; return \; from \; step \; t-1 \; to \; t$

W_t: $Summation \; of \; low-risk \; assets' \; weights$

w_t: $Vector \; of \; every \; asset's \; weight \; at \; step \; t$

x_t: $Vector \; of \; every \; asset's \; return \; from \; step \; t-1 \; to \; t$

N: $Total \; number \; of \; assets$

I_t: $Additional \; investment \; at \; step \; t$

R_f: $Risk-free \; rate$

The first term, with constant α, penalizes when the portfolio value is below the goal. We aim to emphasize the penalization when the portfolio is too much below the goal by setting the term as exponentia. As shown in Figure 3, the penalization is huge during the first few time steps, and quickly diminish when portfolio value increases.

Figure 2. $y = \alpha^x, \; \alpha = 0.9$

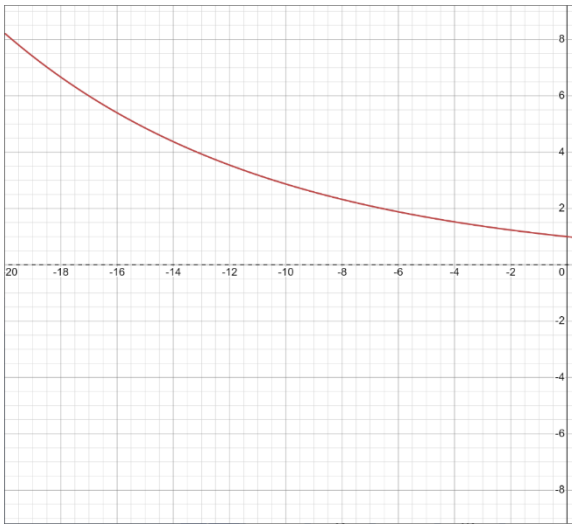
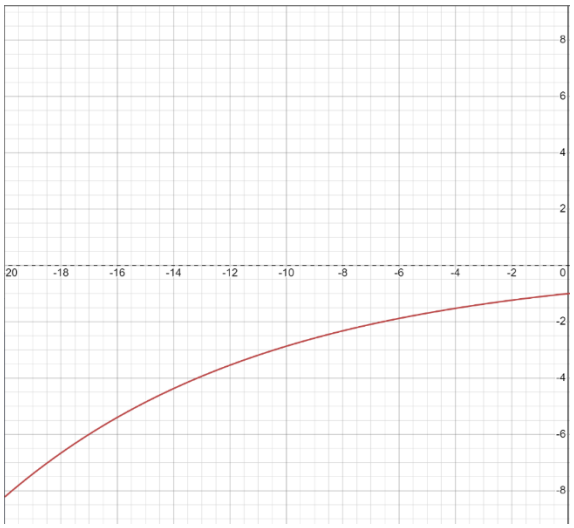


Figure 3. $y = -\alpha^x, \; \alpha = 0.9$



The second term, with constant β, encourages profit. In particular, it encourages extra profit above the benchmark, which is the equally weighted portfolio. The third term, with constant γ, is the Sharpe ratio from time 0 to time t, which encourages profit while keeping a certain level of systematic risk and idiosyncratic risk. This term serves as a risk control term in the reward function. The fourth term, with constant δ, encourages the weight of low-risk assets in the portfolio when the time is near the end of the investment period. In our portfolio, low-risk assets are cash and two bond indices as mentioned in section 3.2. We add this term to lower the volatility near the completion date. The fifth term, with constant θ, is the maximum drawdown from time 0 to time t. Same as Sharpe ratio, this term serves as risk control term to prevent the RL agent making decisions that are too aggressive. When the market is bad, this term will encourage the RL agent to switch to low-volatility assets such as bonds. The final term, with constant μ, is the installment of money, which is also the total amount of changes position in our portfolio. This is a penalization term because we do not want the RL agent to rely on new cash flow.

Reward function (1) is in a simplified form. To show the full mathematical formula, we expend it into formula (2) below.

$$R_t(\boldsymbol{x}_t, \boldsymbol{w}_t, t) = -a^{\min\left(\sum_{i=1}^{N} w_{t+1}^i x_{t+1}^i - \mathrm{GOAL},\ 0\right)} + \beta\left(\sum_{i=1}^{N} w_{t+1}^i\, x_{t+1}^i - \frac{1}{N}\sum_{i}^{N} x_{t+1}^i\right) +$$

$$\gamma \frac{\frac{1}{t}\sum_{\tau=1}^{t}\sum_{i=1}^{N} w_\tau^i x_\tau^i - R_f}{\sqrt{\frac{1}{t}\sum_{\zeta}^{t}\left(\sum_{i=1}^{N} w_\zeta^i x_\zeta^i - \frac{1}{t}\sum_{\tau=1}^{t}\sum_{i=1}^{N} w_\tau^i x_\tau^i\right)^2}} + \delta W \times \frac{t}{T} -$$

$$\theta \underset{\tau \in (0,T)}{max}\left[\underset{t \in (0,\tau)}{max}\sum_{i=1}^{N} w_t^i\, x_t^i - \sum_{i=1}^{N} w_\tau^i\, x_\tau^i\right] - \mu I_t \qquad (2)$$

Our model combines the advantages of unconstrained conditions and lower dimensionality in the optimization problem because of the dollar-measured actions. One former example is provided by Matthew et al. (2020) who developed a G-learner and GIRL method for goal-based wealth management problems. Another approach provided by Lin et al. (2019) developed a similar squared loss function for dynamic optimization. However, due to quadratic function symmetricity

property, our model would penalize whether our portfolio reaches the target or not. Hereby, we introduce a benchmark B_t which is initially equally weighted portfolio strategy:

$$\bar{P}_{t+1} = (1-\rho)B_t + \rho\eta \mathbf{1}^{\boldsymbol{T}} \boldsymbol{x}_t \tag{3}$$

where $0 \leq \rho \leq 1$ is a linearity weight parameter combined with portfolio-independent and portfolio-dependent terms. η defines the growth rate of our current portfolio. If we control our parameters η and find appropriate benchmark B_t, we can set our target to be well above. Thus, our penalty becomes asymmetric and suitable for our contribution retirement plan.

For computing optimal stochastic consumption-investment policies for a retirement plan — the method is sufficiently general for either a cumulation or de-cumulation phase. For other specifications of rewards, numerical optimization, and function approximations (e.g. neural networks) would be required.

4.5 Reward function for PnL portfolio

To prove the GBI method is superior in terms of maximizing the probability of attaining clients' investment goal, apart from comparing the GBI portfolio with the equally weighted portfolio, we also add a PnL based portfolio as a reference. To be as fair as possible, both the GBI and PnL portfolio will be generated using the same settings. Our reward function for PnL is very similar to the GBI reward function. All terms are the same except removing the GOAL related term in the PnL reward function, which is the first term in the GBI reward function. Details are shown below.

$$R_t(\boldsymbol{x}_t, \boldsymbol{u}_t, c_t) = \beta(\mathrm{P}_{t+1} - \mathrm{B}_{t+1}) + \gamma SR + \delta W \times \frac{t}{T} - \theta MDD - \mu I_t \tag{4}$$

5 Deep Reinforcement Learning

5.1 DDPG model

To make the learning process more efficient and to capture more non-linear characteristics, we implement the RL model discussed in Section in a deep learning framework. The DRL model we use in this project is DDPG. According to Qi (2018) and Xiong (2018), DDPG is an improved version of Deterministic Policy Gradient (DPG) algorithm. DDPG concurrently learns a Q-function and a policy, which improves the learning rate under a continuous circumstances.

Since the focus of this project is applying DRL in GBI, we are not going to modify the original DDPG model. Hence, we will not build a DDPG framework from scratch, but use an external package "Stable Baseline" written by Hill at el (2020). The DDPG function in the stable baseline package resembles a basic DDPG structure, which is sufficient for this project. The algorithm of DDPG is shown in Figure 4, credit to Lillicrap et al. (2015).

Figure 4. pseudo-code of the DDPG algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

5.2 DRL settings

To run the DRL environment, we first initialized some parameters in the reward function and the environment. The list is shown in Table 2. Some of the parameters in this table were selected based on trial-and-error, such as the coefficients of the reward function terms; Some parameters were selected because of common industry standards such as steps in the training phase.

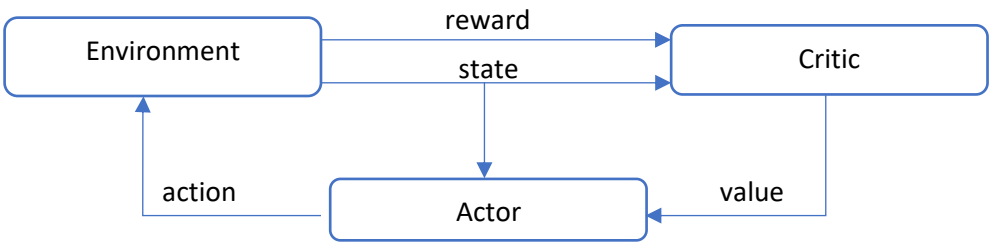
Table 2. settings of parameters

α	0.95	Training steps	10000
β	0.5	Test steps	120
γ	0.9	Rebalance interval	1 month
δ	1	Rounds of simulation	1000
θ	1	Initial investment	1 (unit)
μ	0.1	Additional investment	0.1 (unit) per step

5.3 Model construction

The DDPG code framework is shown in Figure 5. The Actor – Critic sections are achieved using the Stable_Baseline package mentioned in section 5.1, the Environment section has to be programed separately. The main function will initiate an environment and send the environment along with the settings and data to the DDPG framework from the Stable_Baseline package.

Figure 5. Flow of DDPG



5.4 Test run results and analysis

Using the settings mentioned in section 5.2, we run the DDPG model. Figure 6 shows the cumulative GBI portfolio value verses the benchmark with goal = 18; Figure 7 shows the cumulative GBI portfolio value verses the benchmark with goal = 20; Figure 8 Shows the cumulative PnL portfolio value verses the benchmark. All these results are from a single simulation.

Figure 6. GBI portfolio value, GOAL = 18

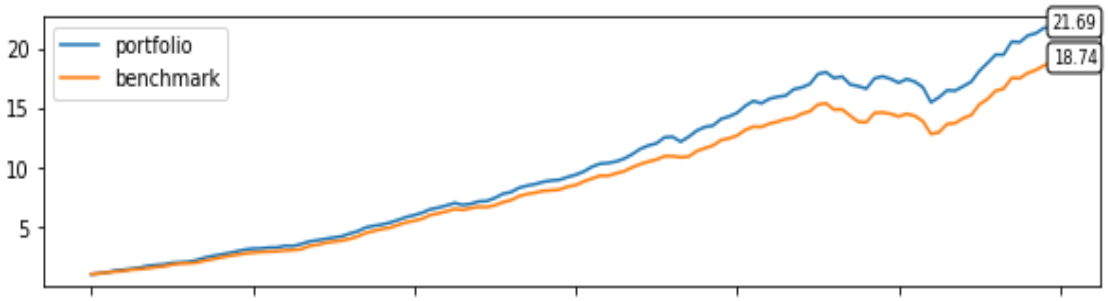


Figure 7. GBI portfolio value, GOAL = 20

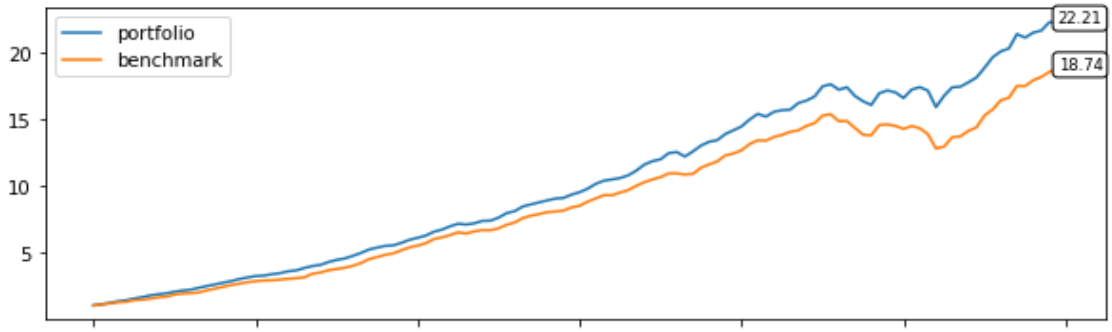
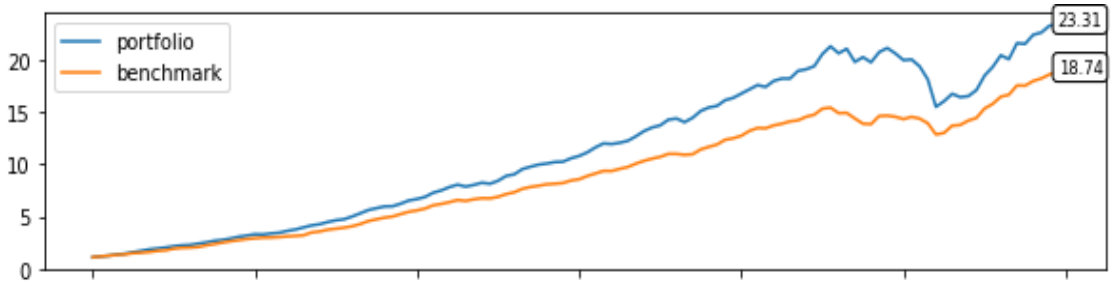


Figure 8. PnL portfolio value



From Figure 6 and Figure 7, we can see that the GBI portfolio with two different goals both had stable increase in portfolio value. When the market is not as good near the end of the investment period, the GBI portfolios also showed stable return. In contrast, the PnL portfolio in Figure 8 had a higher return at the cost of higher volatility. The MDD was 25% even though we added the MDD term as a penalization in the reward function.

While all the above simulations reached the investment goal, it is unclear how the model performs in all market conditions. We also have to retrieve the probability of attaining the goal with more simulations. Using the Monte Carlo framework presented in section 3.4, we simulated each type of the portfolio 1000 times. A sample simulation is shown in Figure 9, and the results of all simulation is shown in Table 3.

Figure 9. Simulated market return over 120 months

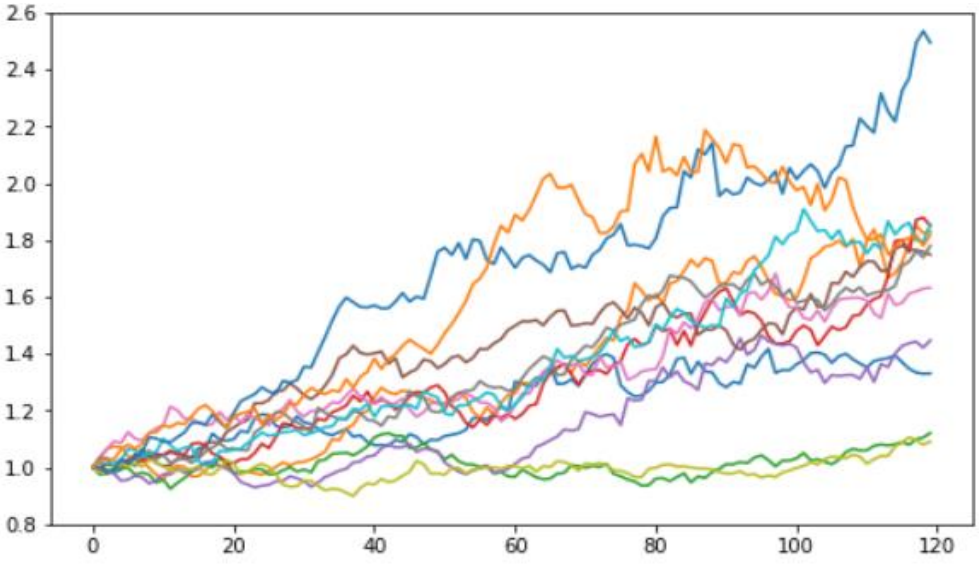


Table 3. Simulation results, Goal = 18

Portfolio	Probability of beating the goal
GBI	76.8%
PnL	59.2%
Equally weighted	57.1%
PnL w/o risk management terms	52.2%

From Table 3, GBI has the highest probability of beating the goal at the end of the investment period. To our surprise, the performance of PnL portfolio is not much better than the benchmark, which is equally weighted. This might be the result of underperformance of the PnL approach during bad times. To justify of hypothesis, we run the simulation using a model built from a simple PnL reward function with all risk management terms removed. The result is very poor, indicating that the PnL approach is not wise under all market situations.

Although the result of GBI simulation seems a little far from ideal, we believe the actual probability of attaining the goal is higher than the simulated 76.8% because of two reasons. First, the Monte Carlo simulation included many extremely bad market conditions that is rare in the real world (see bottom two simulated trends in Figure 9). Even if a bearish market appears, the condition will unlikely to stay for 10 years, which means in the actual world, our GBI portfolio's probability of attaining the goal should be much higher. Second, we simulated 10 years of portfolio management actions in one run. In the later part of the investment period, the model might be already outdated. In the real world, if we refit the model more frequently throughout the investment period using more updated market data, the result should be better than the simulated result shown above.

6 Conclusion

This project used a DRL framework to solve a GBI problem and proved GBI could perform better than a PnL portfolio under different market situations. Some future research directions might include, testing other DRL models, using more asset classes, and change the details of Monte Carlo simulation to obtain more realistic simulated market data.

Reference

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34(6), pp. 26-38.

Browne, S. N. (1997). “Reaching Goals by a Deadline: Digital Options and Continuous-Time Active Portfolio Management”. In: *SSRN Electronic Journal*. doi:10.2139/ssrn.703

Brunel, J. L. P. (2011). “Goal-Based Wealth Management in Practice”. In: *The Journal of Wealth Management* 14(3), pp. 17-26.

Das, S.R., D. Ostrov, A. Radhakrishnan, D. Srivastav (2018). “Goals-Based Wealth Management: A New Approach”. In: *Journal of Investment Management* 16(3), pp. 1-27.

Dasa, S.R., Varmaa, S. (2020). “Dynamic goals-based wealth management using reinforcement learning”. In: *Journal Of Investment Management* 18(2), pp. 1-20

Dixon, M. F., & Halperin, I. (2020). “G-Learner and GIRL: Goal Based Wealth Management with Reinforcement Learning”. In: *SSRN Electronic Journal*. doi:10.2139/ssrn.3543852

Dulac-Arnold, G. (2016). “Deep Reinforcement Learning in Large Discrete Action Spaces”. arXiv:1512.07679

Hu, Y.-J. and Lin, S.-J. (2019). “Deep reinforcement learning for optimizing finance portfolio management”. In: *2019 Amity International Conference on Artificial Intelligence (AICAI)*. IEEE, pp. 14-20.

Kim, W. C., Kwon, D.-G., Lee, Y., Kim, J. H., and Lin, C. (2019). “Personalized goal-based investing via multi-stage stochastic goal programming”. In: *Quantitative Finance*, pp. 1-12.

Nevin, D. (2004). “Goals-Based Investing: Integrating Traditional and Behavioral Finance”. In: *The Journal of Wealth Management* 6(4), pp. 8-23.

Ponomarev, E. S., Oseledets, I. V., & Cichocki, A. S. (2019). “Using Reinforcement Learning in the Algorithmic Trading Problem”. In: *Journal of Communications Technology and Electronics*, 64(12), pp. 1450–1457.

Qi, Y., Huang, S. (2018). “Portfolio Management Based on DDPG Algorithm of Deep Reinforcement Learning”. In: *Computer and Modernization*, doi: 10.3969/j.issn.1006-2475.2018.05.019.

Shelton, C.R., (2001). “Balancing multiple sources of reward in reinforcement learning”. In: *Advances in Neural Information Processing Systems.*

Wang, H., Suri, A., Laster, D., and Almadi, H. (2011). “Portfolio Selection in Goals-Based Wealth Management”. In: *The Journal of Wealth Management* 14(1), pp. 55-65.

Appendix

Software and packages used:

Python 3.7.0

Stable Baselines 2.10.1

Tensorflow 1.8.0

Datasets:

Name	Description	Name	Description
BCOMTR	Bloomberg Commodity Index Total Return	RU20VATR	iShares Russell 2000 Value ETF
HFRIFWI	HFRI Fund Weighted Composite Index	RUMCINTR	iShares Russell Mid-Cap ETF
LBUSTRUU	Bloomberg Barclays US Aggregate Bond Index	RUMRINTR	iShares Micro-Cap ETF
LG30TRUU	Bloomberg Barclays Global High Yield Total Return Index Value Unhedge	RUTPINTR	iShares Russell Top 200 ETF
LMBITR	Bloomberg Barclays Municipal Bond Index Total Return Index Value Unhedged USD	S5COND	S&P 500 Consumer Discretionary Index
NDDUE15X	Amundi MSCI Europe Ex UK Ucits ETF Dr	S5CONS	S&P 500 Consumer Staples Index
NDDUJN	MSCI Japan Index	S5ENRS	S&P 500 Energy Index
NDDUNA	iShares MSCI North America UCITS ETF	S5FINL	S&P 500 Financials Sector GICS Level 1 Index
NDDUPXJ	MSCI Pacific ex Japan UCITS ETF	S5HLTH	S&P 500 Health Care Index
NDDUUK	iShares MSCI UK ETF	S5INDU	S&P 500 Industrials Index
NDDUWXUS	MSCI World ex USA total net return	S5INFT	S&P 500 Information Technology Index
NDUEEGF	SPDR MSCI Emerging Markets UCITS ETF	S5MATR	S&P 500 Materials Index
RU10GRTR	iShares Russell 1000 Growth ETF	S5RLST	S&P 500 Real Estate Index
RU10VATR	iShares Russell 1000 Value ETF	S5TELS	S&P 500 Communication Services Index
RU20GRTR	iShares Russell 2000 Growth ETF	S5UTIL	S&P 500 Utilities Index
RU20INTR	Russell 2000 Total Return	SPXT	Proshares S&P 500 EX Technology ETF