Combining (deep) reinforcement learning and goals based investing

Sijie Wen, Yamin Tang
Advisor: Dr. Cristian Homescu
School of Business
Stevens Institute of Technology

May 15, 2020

Abstract

Recently there has been an efficient exponential methodology, Reinforcement learning (RL), increasing the use of the financial markets such as trading and investment strategy. Goal-Based Investing (GBI) is an investment approach where performance is measured by the success of investments in meeting the investor's financial goals. Portfolio optimization for GBI can be viewed as an optimal control problem performed within a data-driven world, and be solved via RL. We propose a goal-based investment model that is suitable for personalized retirement management and be implemented by Q-learning and DDPG, which are RL algorithms, in our paper. Compared with traditional portfolio investment, our model shows statistically significant profitability and better performance.

Keywords: Goal-based Investing; Reinforcement Learning; Portfolio Management; Retirement planning

1 Introduction

Goal-Based Investing (GBI) refers to the management of an investor's portfolios to meet long-term financial goals, as opposed to only optimizing a risk-return tradeoff, and the performance is measured by the success of investments in meeting the goals. The objective of GBI is to invest symmetrically in a consistent investor's risk profile and time horizon of the goals. A dynamic programming approach is present and analyzed for GBI that is optimal in meeting long-term goals via reinforcement learning (RL), a paradigm of learning by trial-and-error, solely from rewards or punishments.

In this paper, we set the retirement saving as our goal, construct and learn the trading knowledge utilizing two kinds of (deep) reinforcement learning algorithms, Q-learning, a value-based method, and deep deterministic policy gradient (DDPG), adapt to the continuous action domain. Results are consistent with what is expected: the performance of the RL algorithm is better than the traditional portfolio investment. This paper is to prove that RL is valuable to use in the GBI portfolio, which can avoid the factors of human behavior in the traditional approaches. Our codes can be viewed on GitHub[1].

The remaining structure of this paper is as follows: In Section 2, we will summarize some related work and construct basic knowledge. Section 3 will introduce the definition of goal-based investing. Then Section 4 is about the RL algorithms used. Although most RL algorithms focus on game playing and robot control, we will build our reward function and introduce personalized models specified to the financial portfolio management problem. Section 5 consists of our experimental setup. Finally, Section 6 includes conclusion and future work in goal-based investing via deep reinforcement learning in portfolio management.

[1] https://github.com/SijieWen/RL_GBI

2 Literature review

From the Goal-based portfolio theory, investors can define their tolerance of risk and are concerned with the dangers which are not hitting their goals. Therefore, Wang et al. (2011) mentioned that it was a significant risk for goal-based investors under the time of market losses since risk and return are separate. We can get a conclusion that in a goal-based investment set, the timing of downside variance is more important than upside variance, making us focus on avoiding downside risk rather than variance (a measure of both upside and downside risk). Besides, we can understand the risk further referred to Parker (2016a)and the importance of other factors when choosing a portfolio to accomplish a goal through various ratio referred to Parker (2016b).

Reinforcement learning has been proposed as a way to learn by interacting with the environment and gradually improving its performance via trial-and-error. Mnih et al. (2015) suggested that Q-learning as one of the algorithms we choose for our portfolio is using a neural network as an approximation of Q value function and replay for learning, gains remarkable performance in playing different games without changing network structure and hyperparameters. Pendharkar and Cusatis (2018) showed that a predetermined probability in the process of the Q-learning algorithm that chose the action to maximize the reward at the next step to learn what were the rewards or actions. Another algorithm, Deep Deterministic Policy Gradient (DDPG), uses an actor-critic framework to stabilize the training process and achieve higher efficiency (Lillicrap et al. (2015)). Currently, most reinforcement learning algorithms focus on game playing and robot control, such as AlphaGo, the most famous and successful story of RL (Li (2019)). However, deep reinforcement learning can figure out patterns of the market movements at some level, because the data in the financial market is complicated, nonlinear patterns, and low signal-noise ratio. Generalized the idea of Zhao et al. (2019) and Plappert et al. (2018) to the financial domain, we will show the critical characters in portfolio management after modification. Liang et al. (2018)applies RL algorithms with continuous action space to asset allocation, summarizing that strategy better than the conventional strategy. Fowler and De Vassal (2006) presents a conceptual framework without considering changes in wealth based on income, saving, and goal achievements assuming normally distributed asset returns. With its simplicity, flexibility, and computational efficiency, the proposed multistage stochastic goal programming model, mentioned in Kim et al. (2020), provides a new GBI framework for au-

tomated investment management services that figures out the optimal financial plan for an individual aiming to achieve consumption goals and presents the features of simple language and real-time interaction. Besides, the personalized objective function and reward function in this paper incorporate a few scenario-based constraints. The result of Fontoura et al. (2019) report shows that the RL framework is a reliable alternative to the more usual multistage stochastic programming approach and fitted to solve the asset-liability management without scenario discretization. Therefore, we learn to combine goal programming approaches and RL algorithms that optimize the portfolio investment in the period.

In the next part, we will demonstrate the definition of GBI, reinforcement learning, and how to apply to the retirement goal via RL algorithms.

3 Goal-based investing

Modern portfolio theory (MPT) was proposed by Markowitz (1952) and is viewed as the fundamental approach for investors in the field of quantitative investment management. A portfolio optimization problem can be solved in two methods: (1) minimize the risk measure given an expected return, or (2) maximize the expected performance at a particular risk level.

Goal-based investing (GBI) is a new approach to wealth management, which focuses on investing in attaining financial goals based on the investor profile (including risk preference) instead of seeking the highest portfolio return while taking an increased risk. Investors can set up their goals and manage their funds for specific purposes, such as for children's college tuition or house's down payment. So, an essential aspect of a target-date fund, such as pension funds, is the adjustment of asset allocation in line with the remaining of a fixed investment horizon. They are managing the retirement fund assets to ensure that the investor receives promised retirement benefits and meet the special requirement of consumption after retirement. For instance, some can bear more volatility and expect more returns than risk-aversion investors.

With the reduction of decision-making based on market fluctuations, GBI has the advantage compared to the conventional portfolio. Modified by the dynamic

portfolio strategy provided by R et al. (2019) and Das and Varma (2019) and, the essence of our problem can clarify as follows. Investors set a retirement account given wealth W_t at any point in time t. Mathematically, this is expressed as an objective function:

$$maximize Prob\left[W_t \geq C_t\right] \tag{1}$$

where W_t is the expected wealth at time t, C_t is the consumption goal at time t, and $Prob\left[\cdot\right]$ is the probability that the bracketed condition is met. In the above objective function, an investor to a retirement plan may demand that the final wealth level at the age of his or her retirement, at maturity day, be at least equal to, or preferably larger than, some target consumption goal C_t . In other words, they want to maximize the probability that the final wealth level is more than the total consumption at the end of the whole investment horizon.
This described procedure is a concrete step forward for goals-based investors. We agree with the redefinition of risk as to the probability of failing to reach a goal. However, this goal-based strategy still overlooks the main disadvantage of a timing issue. There is a specific time when a specified amount of assets is needed. Therefore, we propose some basic adjustments described in Section 4.4.

4 Reinforcement learning

4.1 Definition

In 1998, reinforcement learning was first introduced and implemented by Moody et al. (1998) in the field of the financial market. It is a forward iteration approach that is repeated interactions between the agent and the environment, thus learns about the optimal timing of trades, strategy of investment and financial planning management through new information, such as managing stocks and bonds in a portfolio and managing the foreign exchange market (Arulkumaran et al. (2017)). Basic reinforcement is modeled as Markov Decision Process (MDP), a model to achieve a goal via a straightforward framing of the problem that learns from the interaction. An initial state s is observed by an agent in the process. a is a finite set of actions. The environment and the agent interact at each stage.

$P_a(s, s^{'}) = Pr(s_{t+1} = s^{'}|s_t = s, a_t = a)$ is a probability, and action a in state s at time t will lead to state $s^{'}$ at time $t + 1$. $R_a(s, s^{'})$ is the reward. The agent carries on these actions and the environment responding and presenting new actions and situations to the agent. Then, looking for a policy π that can maximize the expected total rewards is the objective in the process (Sutton and Barto (2018)).

One important concept is the trajectory in algorithms for control learning in RL. The environment gives a reward $r_{t+1}$and a new state s_{t+1} based on state-action pair. A trajectory is a sequence generated via all interactions. The sum of expected rewards equals to the return of a trajectory. A discount factor $\gamma \in (0, 1)$ is a hyperparameter tuned by the user, which represents how much future events lose their value according to how far away in time they are (Fontoura et al. (2019)). The culmulative discounted rewards G of a trajectory τ is given by:

$$G_t(\tau) = \sum_{t=1}^{T} \gamma^t r_t$$

The value function is another important concept. The approaches of value function tend to find the policy maximizing the return by a set of expected returns for some policy. Define the value of a policy π by

$$V^{\pi}(s_t) = E[G_t|s_t]$$

It is also useful to define action-values, which is similar to the value function. Given a state s_t, an action a_t and a policy π , the action-value of the pair (s, a) under π is defined by

$$Q^{\pi}(s_t, a_t) = E[G_t|s_t, a_t]$$

The purpose of the RL algorithm tends to find the policy maximizing the expectation of the discounted reward given an initial state s_1.

4.2 Q-learning

In portfolio optimization, we choose the Q-learning algorithm in our paper as the first algorithm. The tabular method is what we used in our paper. The tabular approach is better suited for action spaces and discrete states. Arrays or tables represent the approximate value functions. Bellman equations (Bellman (1954)): $V^{\pi}(s_{t+1}) = r_t + \gamma V^{\pi}(s_t)$ and $Q^{\pi}(s_{t+1}, a_{t+1}) = r_t + \gamma Q^{\pi}(s_t, a_t)$

to calculate optimal values. The optimal policy is that the action taken at any state tends to maximize V^{π} or Q^{π}.

Algorithm 1 Q-learning

1. Initialize $Q(s,a)$ arbitrarily
2. Repeat (for each episode):
3. Initialize s
4. Repeat (for each step of the episode):
5. Choose $a$from $s$using policy derived from Q (e.g., ε-greedy)
6. Take action a, observe r, $s^{'}$
7. $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma max_{a'} Q\left(s^{'}, a^{'}\right) - Q(s,a)\right]$
8. $s \leftarrow s^{'}$;
9. until s is terminal

The above summarizes all steps of the Q-learning algorithms. Every update we use Q reality and Q estimation, and the part of Q-learning is in $Q(s_1, a_2)$ reality, It also includes a maximum estimate of $Q(s_2)$. The process uses the maximum evaluation of the next attenuation and the current reward as the reality of this step. At last, the meaning of parameters in this algorithm, like epsilon greedy, is a strategy used in decision-making. For example, it means that in 90% of the cases when $\varepsilon = 0.9$, we will choose the behavior according to the optimal value of the Q table, and use the random selection behavior 10% of the time. α is a learning rate, to determine how much error this time is to be learned, α is a number less than 1. γ is the attenuation value of the future reward. When $\gamma = 1$, the Q at s_1 is a reward without any decay in the future; that is, the machine can clearly see all the steps after it. However, the full value of, when $\gamma = 0$, the machine can only touch the reward in front of it. It also only cares about the recent big reward. If γ changes from 0 to 1, the machine is gradually becoming optimally, not only looking at the benefits in front of it but also thinking about its future.

Extended to the retirement problem, investors put a certain amount of wealth at each point and take action to train the model via repeated investing in portfolio optimization sequences. There is a single optimization portfolio in a reward of 1 or 0 after a series of state-action pairs via time t . Increase the likelihood of investment that leads to rewards of 1 and reduce the 0 rewards are the purpose of the whole process. Also, actions only focus on the current state instead of preceding states so that only a certain number known as “episodes”. The

machine plays the portfolio optimization game forward and assesses the final reward.

4.3 Deep Deterministic Policy Gradient (DDPG)

We choose Deep Deterministic Policy Gradient algorithm in our paper as another algorithm (Lillicrap et al. (2015)).
DDPG is a combination of Q-learning and policy gradient, improving the convergence to the optimal policy. DDPG uses a neural network as the Q-function approximator to minimize the loss function:

$$L\left(\theta^{Q}\right)=E_{s_{t}\sim\rho^{\beta},a_{t}\sim\beta,r_{t}\sim}\left[\left(Q\left(s_{t},a_{t}|\theta^{Q}\right)-y_{t}\right)^{2}\right]$$

where

$$y_{t}=r\left(s_{t,}a_{t}\right)+\gamma Q\left(s_{t+1},\mu\left(s_{t+1}\right)|\theta^{Q}\right)$$

An experience replay buffer is used for training a deep neural network to approximate $Q^{*}\left(s,a\right)$, which is a set of previous experiences. It may not always be sufficient to keep everything, although the replay buffer is large enough to maintain the experiences. Thus, we have to choose the appropriate amount of experience to learn and to get right. The policy learning in DDPG consists that actors can directly output continuous action. Because the action space is ongoing, and we assume the Q-function is differentiable concerning an action, we can perform gradient ascent to solve:

$$J\left(\pi_{\theta}\right)=E_{\tau\sim p_{\theta}(\tau)}\left[\sum_{t}\gamma^{t}r\left(s_{t},a_{t}\right)\right]$$

In DDPG, online actor, critic, target actor, and target cretic are required. The actor is the function μ and critic is the Q-value function combining Q-learning and policy gradient. The online critic evaluates the actor's action and updates the online actor. Moreover, an online critic can be updated by a target actor and target critic. Also, DDPG trains a deterministic policy in an off-policy way. It would not try a wide variety of actions to find the signals at the beginning since the policy is deterministic if the agent is to explore policy.

Algorithm 2 DDPG

Randomly initialize actor $\mu(s|\theta^\mu)$ and critic $Q(s,a|\theta^Q)$
Create Q' and μ' by $\theta^{Q'} \rightarrow \theta^Q$,$\theta^{\mu'} \rightarrow \theta^\mu$
Initialize replay buffer *R*
for $i = 1$ to M **do**
  Initialize a UO process $\mathcal{N}$
  Receive initial observation state s_1
  for $t = 1$ to T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$
    Execute action a_t and observe r_t and s_{t+1}
    Save 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}) in *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 actor policy by policy gradient:

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

    Update the target networks:

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

  end for
end for

4.4 Reward function

4.4.1 Parameters

We firstly introduce the set of parameters used in the function. The first asset ($i = 1$) is assumed to represent cash (or cash equivalent). An investor's initial wealth is fully allocated in cash.

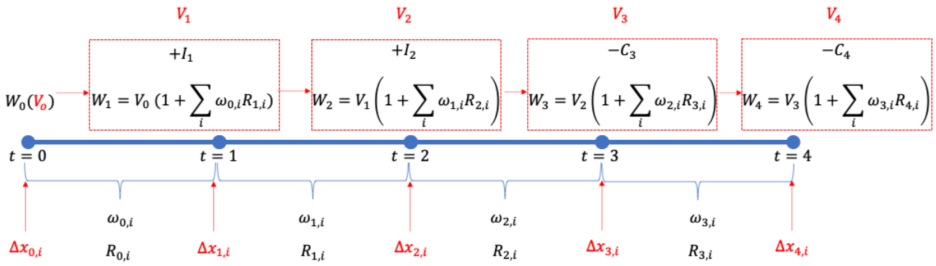


Figure 1: An example of whole investment process

W_t	total wealth before additional investment and consumption at stage t
V_t	total value after additional investment and consumption at stage t
I_t	additional investment at stage t
C_t	consumption goal at stage t
$R_{t,i}$	expected return of asset i from stage t to stage $t+1$
$\omega_{t,i}$	weights of asset i from stage t to stage $t+1$
ρ	risk exposure percentage
α	the probability of the success rate
s	transaction cost percentage corresponding to the expense of buying or selling the assets
$\triangle x_{t,i}$	changes of (purchase or selling) value of asset i at stage t

4.4.2 Objective function

Recall Eq.[1] brought up in Section 3, the objective function of the retirement optimization problem is to maximize the probability that the final wealth level is more than the consumption goal at the end of the investment horizon.

$$maximize Prob\,[W_t \geq C_t] \tag{1}$$

Eq.[2] means there is no additional investment and consumption at stage 0:

$$subject\ to\ V_0 = W_0 \tag{2}$$

Eq.[3] reflects the total value V_t after additional investment and consumption at stage t equal to the total wealth W_t at stage t (including compounding returns) plus additional investments and deducts the consumption:

$$V_t = W_t + I_t - C_t, \qquad t = 1, \ldots, T \tag{3}$$

As shown in Eq.[4, 5] at stage 0, we use all cash or cash equivalent we hold to buy a portfolio, to buy different assets i with different weights $\omega_{1,i}$:

$$\Delta x_{0,i} = \omega_{0,i} W_0 = \omega_{0,i} V_0 \tag{4}$$

$$\sum_i \Delta x_{0,i} = W_0 = V_0 \tag{5}$$

Eq.[6] represents the changes in the value of asset i at stage t equal the difference between the asset value after adjusting the weight and before adjusting the weight. $\Delta x_{t-1,i} > 0$ means asset i is purchased at stage t. Otherwise, it is sold at stage t:

$$\Delta x_{t,i} = \omega_{t,i} V_t - (1 + R_{t-1,i})\, \omega_{t-1,i} V_{t-1}, \qquad t = 1, \ldots, T-1 \tag{6}$$

Eq.[7] expresses that at every time step, the sum of the change in each asset position should equal the net additional cash investment at this time:

$$\sum_i \Delta x_{t,i} = I_t - C_t, \qquad t = 1, \ldots, T-1 \tag{7}$$

We subtract the transaction fee percentage in Eq.[8] corresponding to the expense from the return at this stage, to get the net return, the total wealth W_t before additional investment and consumption:

$$\left(V_t - s \sum_i |\Delta x_{t,i}| \right) \left(1 + \sum_i \omega_{t,i} R_{t,i} \right) = W_{t+1}, \qquad t = 0, \ldots, T-1 \tag{8}$$

Total weights in Eq.[9] equal 1 at any point of time based on our portfolio:

$$\sum_{i=1}^{n} \omega_{t,i} = 1, \qquad t = 0, \ldots, T-1 \tag{9}$$

In Eq.[10], we want to consider the downside risk and constrain the average underachievement of the goal in stage t. Here we choose a conditional value at risk $CVaR$ to measure this risk and define the loss $L_t = C_t - W_t$. In the worst (1- α) x 100%, the downside risk is less than or equal to ρC_t, where $0 \leq \rho \leq 1$:

$$CVaR_{\alpha}\left(C_t - W_t\right) \leq \rho C_t, \qquad 0 \leq \rho \leq 1 \tag{10}$$

In Eq.[11], we set the range of $\omega_{t,i}, W, C_t, I_t$, which are not negtive. And the portfolio has N-1 assets with $i = 1$ is cash or cash equivalent.

$$\omega_{t,i},\ W_t,\ C_t,\ I_t \geq 0$$

$$i = 1, \ldots, N \tag{11}$$

4.4.3 RL Settings

To apply RL to real life applications, first, we need to have an RL problem formulation (defined in Section 4.4.2), then defining the environment, states, actions, and rewards.

- Environments: The customized environment is constructed using OpenAI gym Jiang et al. (2017), and the RL agents are trained using some RL libraries. Accordingly, we define the following terminologies for this specific kind of goal-based scenarios.

- State s: Agent's state has two components. The first component is the current portfolio value before adjusting additional investment and consumption W_t , including previous open, closing, high, low price in a fixed window. The second component is a vector of the current portfolio status $\omega_{\mathrm{t,i}}$, which are the weights of different assets in the portfolio, with $\sum_{i=1}^{n} \omega_{t,i} = 1$.

- Action a: the vector of changes of value of these assets $\Delta x_{t,i}$, with $\Delta x_{t,i} = I_t - C_t$ (exception of $\Delta x_{0,i}$).

- Terminal state : we define the terminal state from two aspects. The first one is where the current portfolio value is less than 90 percent of the total cost (initial wealth plus additional investment), because of some inevitable market fluctuation. In addition, I impose maximum weight allocation, preventing each weight of assets more than 50 percent, to limit aggressive investment. When achieving the terminal state, the learning process will be done.

- Reward r: based on the article provided by Dixon and Halperin (2020), we can conclude the one-step reward function at time t as:

$$r_t\left(s_t, a_t\right) = -\lambda E_t |C_{t+1} - W_{t+1}| \left(\frac{t}{T}\right) - \lambda I_t \tag{12}$$

The first term is the expected negative reward for underachievement of the target goal, incorporating transaction costs, and the second term is due to additional investment at the beginning of this time period. To solve the time issue as we mentioned in Section 3, I incorporate the time modifier (the current step divide by total steps) into the first term. In this way, the underachievement of the target in the early period will cause smaller punishment; on the contrary, the underachievement closer to the final target goal will cause more punishment. Moreover, to only penalize the downside risk ($W_{t+1} < C_{t+1}$), provided by Lin et al. (2019) and Dixon and Halperin (2020), we set the target value as a linear combination of a portfolio-independent benchmark B_t and the current portfolio growing with a fixed rate η:

$$C_{t+1} = (1 - \rho) B_t + \rho \eta V_t \tag{13}$$

where $0 \leqslant \rho \leqslant 1$ is a relative weight of the portfolio-independent and portfolio-dependent terms, and $\eta > 1$ is a parameter that defines the desired growth rate of the current portfolio V_t.
In the next section, we will use Q-learning and DDPG to implement these RL ideas.

5 Practical Info

In this section, we illustrate how our proposed model can be utilized in some numerical examples in the practical problem of optimization of a retirement plan. An investor will make periodic investments in the portfolio while being employed, seeking to achieve his or her retirement goal.

To illustrate the Q-learning and DDPG algorithms for goal-based investing, we consider an investor's initial wealth is fully allocated in cash, and we will withdraw all the money if the total wealth is less than 90% of the initial investment. The transaction cost is 0.15% for each transaction. The benchmark portfolio is initially set equal to the initial value of the portfolio, and is continuously compounded at a constant rate based on final wealth goal. We set the investment horizon as ten years and assume investment in six asset classes that cover stock and bond index from 2010 to 2019 :

- S&P 500 Index
- iShares Russell Top 200 ETF
- iShares Russell Mid-Cap ETF
- SPDR MSCI Emerging Markets UCITS ETF
- Bloomberg Barclays Global High Yield Total Return Index
- Bloomberg Barclays Municipal Bond Index Total Return Index

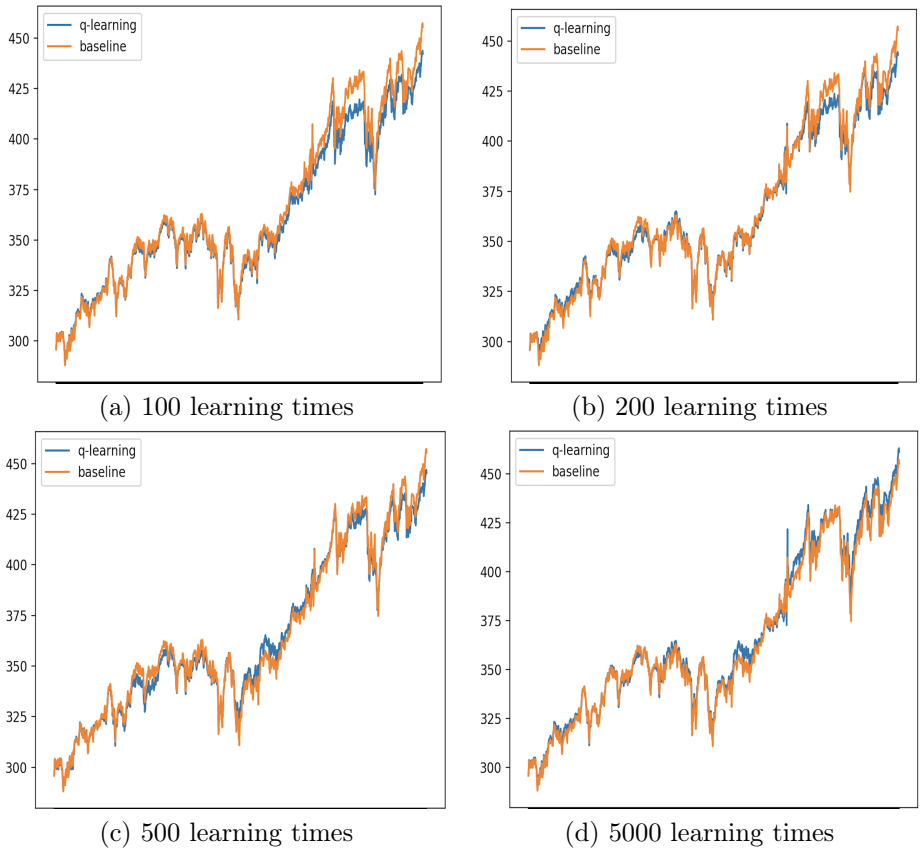
Components of portfolio	Expected Return (%)*	Standard Deviation (%)*
S&P 500	10.50	10.14
iShares Russell Top 200	14.05	11.99
iShares Russell Mid-Cap	13.88	14.52
SPDR MSCI EM	4.26	18.13
Bloomberg Barclays High Yield	4.20	4.78
Bloomberg Barclays Municipal Bond	6.60	7.98

* Expected returns and standard deviations are estimated from 2010.01 to 2019.12 and annualized

Table 1: Portfolio Construction

Table 1summarized the statistic characters of these asset classes, but the asset universe is not restricted to this list in practice.
We also publish and share following results online with the interactive visualizer[2].

5.1 Q-learning algorithm



(a) 100 learning times (b) 200 learning times

(c) 500 learning times (d) 5000 learning times

Figure 2: Q-learning result

In our paper, we choose some suitable components in our portfolio to meet the requirements of risk-aversion investors optimally. At each time stage, with an absolute wealth that is the total amount to be invested, and a set of investments, then the incomes are increasing with the initial investment after some time. We

[2]https://swen1.shinyapps.io/Viz_Shiny_Modules/

assume retirement saving as a single goal that final wealth level is more than the total consumption by the end of the period. In the beginning, we invest $300K into six assets equally and set the different learning times for the computer.
We can see Figure 2 that with the increase of learning times from 100 to 5000. The performance of the Q-learning portfolio is not better compared to the orange baseline. Also, it is not a significant performance under the Q-learning algorithm. In Figure 2, we find it has no big improvement even when we raise the learning times to five thousand. Therefore, the Q-learning algorithm is not a suitable algorithm in our GBI portfolio.

5.2 DDPG algorithm

To set up the agent, we design the code with Stable_Baselines[3], a library as a big improvement upon OpenAI Baselines, and combine with action, state, and reward we have defined to a specified environment.
To illustrate a DDPG agent, we consider the first asset ($i = 1$) is assumed to represent cash (or cash equivalent), which is a risk-free asset. An investor's initial wealth is fully allocated in cash and allocated between cash, stocks, and bonds later. To see the result more directly and easily, I normalized the initial value as 1.
We simply set the initial parameter and some constraints as Table 2.

Parameters	
ρ	0.4
λ	0.01
η	1.01
Maximum weight allocation	50% for each asset
Minimum weight allocation	0% for each asset

Table 2: Initial parameters used for a portfolio of DDPG agent

[3]https://github.com/hill-a/stable-baselines

It's worth noting that the number of window sizes is relative to the accuracy of the model. Window size is how much past data we let the agent observe. The smaller windows perform better because of the temporary relationship among prices in short. In this paper, we choose an initial window size of 5.
From Figure 3, we show the effect of DDPG without any cash installment during the investment horizon, to compare with the effect of Q-learning in Section 5.1, which is observed to be significant.

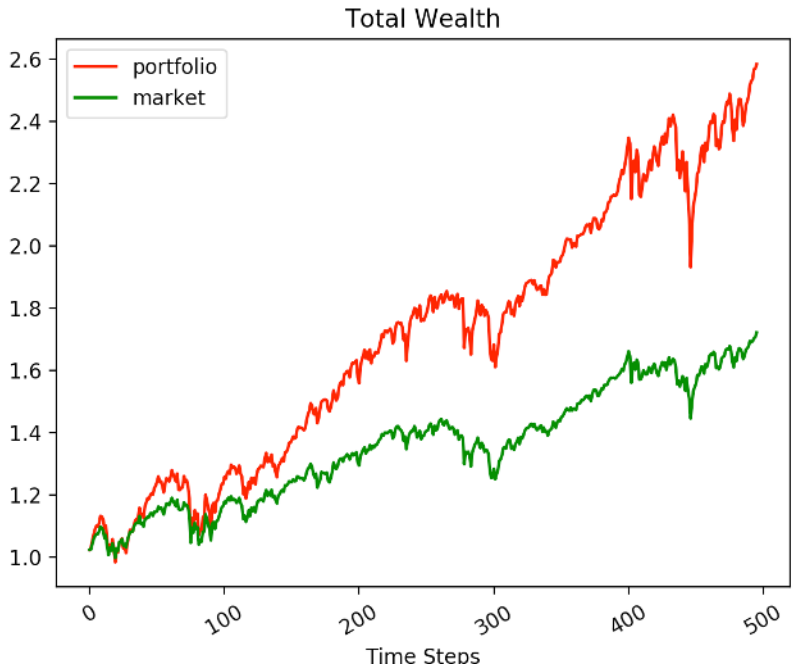


Figure 3: Portfolio value vs market value
without cash installment and adjust weights daily

Next, we set an assumed scenario as following: an investor is at age 40 and will retire at his 50. We assume the investor adds 0.1 cash every month in the ten years. Generalize to practical, we could consider the initial value as 10 thousand dollars and add 1 thousand to the portfolio monthly. Figure 4 demonstrate goal achievement and asset allocation outcomes when the problem is solved for a goal of 18(180 thousand dollars) and 22(220 thousand dollars). The DDPG model is overperformed than the baseline, which is initially set equal to the initial value as 1 and equally weighted. The weight of each asset is reasonable and satisfied

the constraint: total weights equal to 1 at any point in time. When the investor has a conservative portfolio like a goal of 18, the model continues to invest in safe assets in all stages (municipal bond account for the large part). On the other hand, if we improve the goal to 22, the model has no choice but to allocate more to risky assets and reduce the proportion of risk-free assets like cash. In this way, the investor could get 19.16(23.44) successfully if we set the final wealth goal after ten years as 18(22).

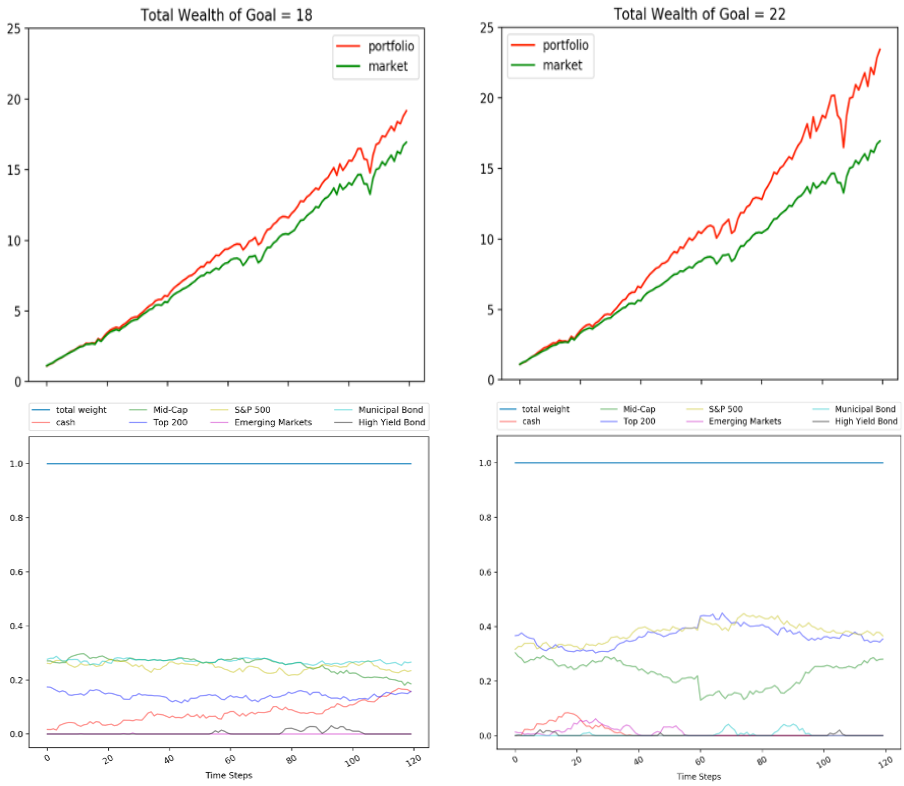


Figure 4: Total wealth and weights of
incremental goal achievement and adjust weights monthly

Therefore, we could find a reasonable goal is vital for the result of the experiment. The goal at the end of the whole investment horizon will affect the success of wealth management directly. More specifically, on this issue, we do another experiment to achieve a final target of 50, as shown in Figure 5. Such a highly aggressive goal is hardly achievable, because we have given a maximum

weight allocation constraint in terminal state to limit the aggressive investment strategy and the reward will be negative at each step. The learning effect is not significant with such wrong reward, even worse than a smaller goal. The model fails.

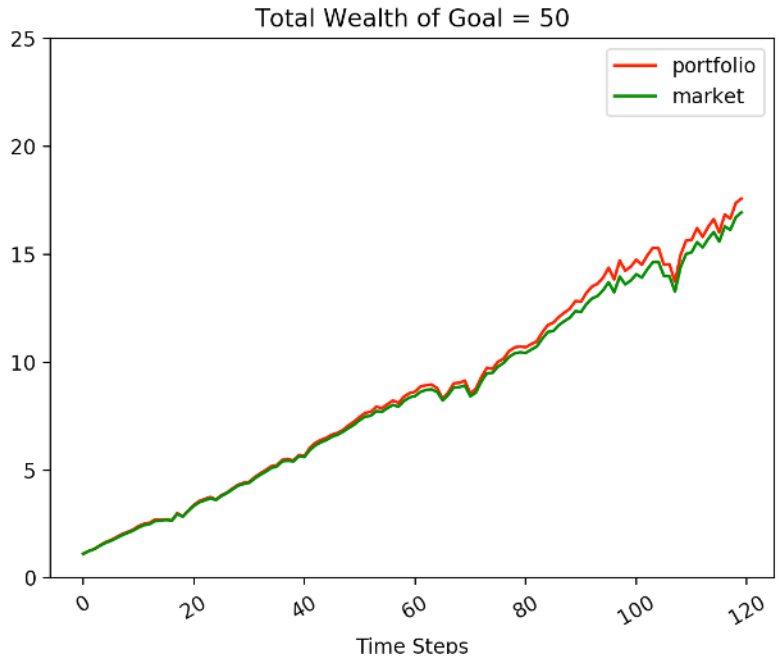


Figure 5: Incremental goal achievement of the goal of 50

6 Conclusion

To summarize, we proposed a reinforcement learning and goal based approach to the retirement optimization problem. Two algorithms, Q-learning and DDPG, are used in this study. Compared with previous works of portfolio management using reinforcement learning, we set our agents with a series of constraints to prevent aggressive investment strategy for the capital-preservation client for retirement goal, and a clear final wealth goal, to which the model is sensitive. A reasonable target will decide the success of a specified investment strategy. Our study shows the strategy obtained by DDPG algorithm can outperform the

conventional strategies in assets allocation. Furthermore, our proposed method only requires inputs of initial wealth, future investment, and goals at the end of the investment horizon, which is applicable for automated financial advising services, also known as robo-advisors.

There are still some improvements could be done in future research. Now we choose the parameters used in the model in Table 2 artificially, so we cannot guarantee the numerical value is optimal for our problem. Using inverse reinforcement learning (IRL) to learn these parameters could help us set preferably and improve numerical accuracy (Krishnan et al. (2016)).

Another deficiency is not involved in alternative investment in the portfolio, primarily with REITs or some Precious Metals or Gold (Shalett (2014)). There are some challenges here due to multiple considerations: for example, one would want that index to be available for a period of time which incorporates as many market regimes as possible. Thus, indices with data since the early 1990s would be strongly preferred, while indices with data only for the last ten years would not be useful enough. Also, it's hard because of the recent inception date of some index.

Given that the generated data is noisy (as it also happens in real financial markets, the time series data is not always stationary), a success of such an endeavor could not be guaranteed beforehand, at either stage. One common idea is to use differencing and transformation techniques to produce a more normal distribution. Or there are many other algorithms in RL, such as G-learning (Dixon and Halperin (2020)), generally produces stochastic policies in contrast to Q-learning, and Hindsight Experience Replay (Andrychowicz et al. (2017)), a method wrapper that works with off-policy methods (DDPG for example).

ACKNOWLEDGMENT

We would like to say thanks to Dr. Cristian Homescu from Bank of America, Professor Zhenyu Cui, from Stevens Institute of Technology for their generous guidance and all the professors who have taught us in our master studying. We could not overcome so many challenges during this project without their support.

References

Andrychowicz, M., F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba (2017). Hindsight experience replay. In *Advances in neural information processing systems*, pp. 5048–5058.

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

Bellman, R. (1954, 11). The theory of dynamic programming. *Bull. Amer. Math. Soc. 60*(6), 503–515.

Das, S. R. and S. Varma (2019). Dynamic goals-based wealth management using reinforcement learning.

Dixon, M. and I. Halperin (2020). G-learner and girl: Goal based wealth management with reinforcement learning. *arXiv preprint arXiv:2002.10990*.

Fontoura, A., D. Haddad, and E. Bezerra (2019). A deep reinforcement learning approach to asset-liability management. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 216–221. IEEE.

Fowler, G. B. and V. De Vassal (2006). Holistic asset allocation for private individuals: Simultaneously solving for an optimal asset allocation given the multiple goals and multiple locations of private clients. *The Journal of Wealth Management 9*(1), 18–30.

Jiang, Z., D. Xu, and J. Liang (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Kim, W. C., D.-G. Kwon, Y. Lee, J. H. Kim, and C. Lin (2020). Personalized goal-based investing via multi-stage stochastic goal programming. *Quantitative Finance 20*(3), 515–526.

Krishnan, S., A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg (2016). Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. *arXiv preprint arXiv:1604.06508*.

Li, Y. (2019). Reinforcement learning applications. *arXiv preprint arXiv:1908.06973*.

Liang, Z., K. Jiang, H. Chen, J. Zhu, and Y. Li (2018). Deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, C., L. Zeng, and H. Wu (2019). Multi-period portfolio optimization in a defined contribution pension plan during the decumulation phase. *Journal of Industrial & Management Optimization 15*(1), 401.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance Vol. 7*(No. 1).

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015). Human-level control through deep reinforcement learning. *Nature 518*(7540), 529–533.

Moody, J., L. Wu, Y. Liao, and M. Saffell (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting 17*(5-6), 441–470.

Parker, F. J. (2016a). Goal-based portfolio optimization. *The Journal of Wealth Management 19*(3), 22–30.

Parker, F. J. (2016b). Portfolio selection in a goal-based setting. *The Journal of Wealth Management 19*(2), 41–46.

Pendharkar, P. C. and P. Cusatis (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications 103*, 1–13.

Plappert, M., M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

R, S., D. Ostrov, A. Radhakrishnan, and D. Srivastav (2019). Dynamic portfolio allocation in goals-based wealth management. *Computational Management Science*, 1–28.

Shalett, L. (2014, february). An outcomes-oriented approach to alternatives. *Tech.rep.Morgan Stanley Wealth Management.*.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Wang, H., A. Suri, D. Laster, and H. Almadi (2011). Portfolio selection in goals-based wealth management. *The Journal of Wealth Management 14*(1), 55–65.

Zhao, R., X. Sun, and V. Tresp (2019). Maximum entropy-regularized multi-goal reinforcement learning. *arXiv preprint arXiv:1905.08786*.