TDF Glide Path Optimization with Reinforcement Learning

Besides supervised and unsupervised learning, reinforcement learning is also an exciting area of research for financial problems. Reinforcement learning allows us to solve complex dynamic optimization problems in an almost model-free way, by optimizing its reward under environment and actor spaces (Kolm & Ritter, 2020). In this section, we apply reinforcement learning (Policy Gradient) for constructing optimal TDF glide paths based on market scenarios. First of all, we reviewed the related research and the basic concepts of reinforcement learning. Secondly, the reinforcement learning model of this research is constructed. Thirdly, we implement Monte Carlo simulation for evaluating our model, compared with the benchmark of Vanguard glide path. Finally, we conclude this section with our training and simulating results, and the future improvements of our model.

Literature Review

1. Relevant Research

Reinforcement learning has many applications on finance, including hedging, pricing, asset allocation, asset liability management, etc. For instance, Buehler et al. (2019), Kolm & Ritter (2020) and Cao (2021) applied reinforcement learning on derivatives hedging; Fontoura et al. (2019) implemented Deep Deterministic Policy Gradient in Asset-Liability Management, while Abe et al. (2010) develop a framework on Markov Decision Process to solve the debt collections problem.

Asset allocation applications, where the goal of reinforcement learning is to obtain the weights of the assets that maximize the rewards in a given state of the market considering risk and transaction costs, is more relevant to our glide path optimization problem. Moody & Saffell (2001) use Recurrent Reinforcement Learning (RRL) on optimizing portfolio. In their research, risk-adjusted returns such as maximum drawdowns and differential Sharpe ratio is included as a part of their reward function. Almahdi & Yang (2017) also adopt RRL in their portfolio optimization research. They show that the maximum drawdown risk based objective function yields superior return performance compared to previously proposed RRL objective functions (i.e., the Sharpe ratio and the Sterling ratio).

Besides maximizing return-based reward function, a Q-learning approach is used to maximize the expected utility of consumption in Weissensteiner (2009), to optimize consumption and asset allocation decisions. The objective function in Irlam (2020) is also designed to maximize the expected utility of consumption with Proximal Policy Optimization (PPO). Using Monte Carlo simulation, Irlam (2020) simulates stock prices, bond prices, and client's financial scenarios under different relative risk aversion (RRA) assumptions for lifetime portfolio selection. However, although Irlam (2020) provides the performance comparison between traditional glide path and the reinforcement learning result under Monte Carlo simulation, they do not guarantee the shape of their lifetime asset allocation conforms the basic rule of traditional glide path.

Lastly, Wang et al. (2021) optimize their asset scoring unit and market scoring unit with policy gradient to construct risk-return balanced portfolio. Also based on policy gradient, i Alonso & Srivastava (2020) provides a simple deep reinforcement learning method with portfolio vector memory to optimize portfolio within 24 US equity securities. Convolutional neural network (CNN), recurrent neural network (RNN), and Long Short Term Memory Neural Network (LSTM) were included as a component of their actor network. The LSTM model outperforms other models in terms of total returns, Sharpe ratio, and maximum drawdown.

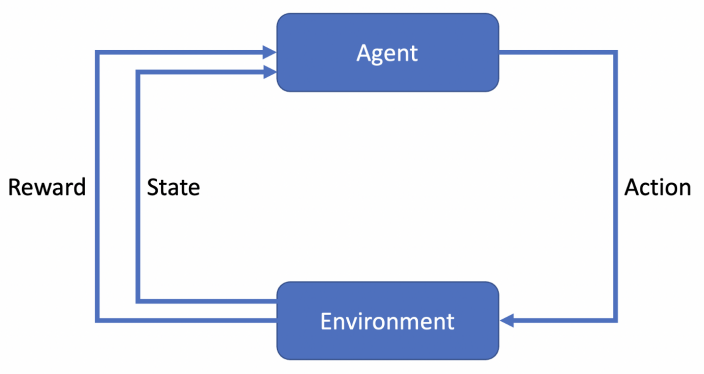
In this research, following the approach by i Alonso & Srivastava (2020), we implement deep policy gradient with the LSTM actor network. To take the portfolio volatility into account, returns and downside

risk are both adopted in our reward function. Moreover, to deal with the glide path problem discussed earlier, a special glide path term is included in our reward function to make sure the decreasing asset allocation path.

2. A Brief Introduction to Reinforcement Learning

To make readers comprehend our model more easily, we briefly introduce the basic concepts of reinforcement learning algorithm here. As we mentioned above, reinforcement learning is a machine learning approach concerned with solving dynamic optimization problems in an almost model-free way by maximizing a reward function in state and action spaces (I Alonso & Srivastava, 2020). That is to say, reinforcement learning has three key components: Agent (Policy), Environment (State) and Reward. The goal of a reinforcement learning algorithm is to maximize the total reward, using the data generated by its agent, environment, and reward function. Environment of a reinforcement learning algorithm could be a game, a simulation, or a series of finance data which are usually undetermined (a black box). As shown in Fig. 1., agent receives states that generated by environment and transforms the received states into actions. After each action, environment responses by another state and reward generated by the reward function (according to the action in environment, for example, killing a monster in a game might gain positive rewards). Hence, the sum of each reward after each action until the episode end (being killed or win), the total reward, is the objective to be maximized. Different from supervised learning, target labels are not needed in reinforcement learning. And similar to supervised learning of which the goal is to minimize its loss function, the goal of reinforcement learning is to maximize its objective function (total reward), which can be considered as minimizing a negative loss.

Fig. 1.



Source: Toward Data Science

Methodology

In this research, our goal is to construct optimal TDF glide paths based on given market scenarios. We follow the approach by i Alonso & Srivastava (2020) to construct the actor network and train the model with Deep Policy Gradient algorithm. To design a new glide path for TDF under market scenarios, the input data of our model includes bond returns, stock returns and the investor's age. The contribution of this research is that we propose a reward function for applying reinforcement learning on designing TDF glide path. Following is the concept of our environment, agent, reward function, and the training algorithm.

1. Environment

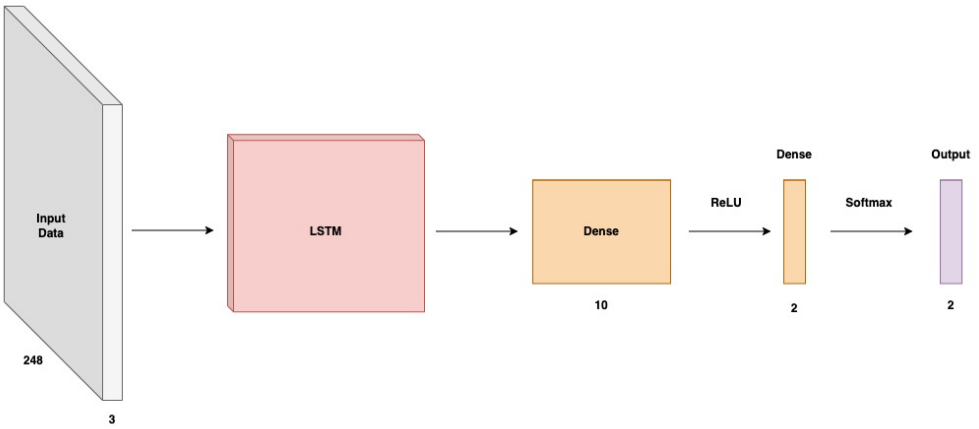
As mentioned above, we adopt market scenarios as our environment. We use 41 years (from 1981 to 2021) daily bond returns of Fidelity Investment Grade Bond Fund, and stock index returns of S&P 500

index as our training environment to train our reinforcement learning model as the simulation of TDF from 25-year-old to the target date of retirement (65-year-old). Furthermore, to make sure the input data of agent network in the same shape, we adopt the least yearly trading date of 41 years as our yearly trading date, which is 248 days during the data period.

2. Agent

The function of agent in reinforcement learning is to transfer information given by environment (states) into actions. In our case, we need to transfer the market scenarios, which is the returns of stock index and bond, into asset allocation strategies. In order to construct the agent that behaves differently under different age of client, we also add a feature that composed by the client's current age as the input environment data. Similar to the agent network of i Alonso & Srivastava (2020), as shown in Fig. 2., our agent network is composed by a LSTM layer and two dense layers. The output of our agent network is a 1*2 array, represents the weight we are going to invest in the equity market and the bond market next year. Note that the sum of two weights is equal to one (no cash position) in our case.

Fig. 2.



3. Reward Function

Reward function plays the most important role in our research. Our reward function is composed by three components: return, risk, and the special term for designing glide path. First, as proposed in previous papers, returns of assets is the most popular feature being used in asset allocation reinforcement learning research. Hence, the first term of our reward function is the cumulative return R_c^i of the year i (we have 40 years in an episode). Secondly, to consider the portfolio volatility, risk-adjusted return is also very famous in reinforcement learning reward. Considering previous research, we decided to adopt drawdown risk:

$$DD^i = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \min\{R_t, 0\}^2}$$

where R_t is the daily return at time t of the year i, and T equals 248 in our case. Lastly, we design a term for the reward function in order to make sure the asset allocation path of our training result conforms the basic glide path rule: begin with 80% to 90% of equity position and remains 30% to 50% of equity position at the target date. The glide path term is:

$$G^i = \sqrt{Age^i} \times W_S^i$$

where Age^i is the age of the investor at year i, and W_s^i is the weight of the equity position at year i. The term G^i increase when the investor's age increase, hence the algorithm will automatically decrease W_S^i to find the expected decreasing glide path. To conclude, our reward function at year i is:

$$Reward^i = R_c^i - \alpha DD^i - \beta G^i$$

where α and β are hyperparameters. Downside risk and the glide path term negatively impact the reward of the year. In our model, α is set as 0.01 and β is 0.0045. Larger β makes the model prefer greater bond position to avoid the punishment from the glide path term, and smaller β makes the model prefer equity position for greater returns. Also, to make the model converge more easily to the expected glide path, we limit the minimum of the weight of the positions to 10%, and maximum to 90%. Furthermore, a gaussian random is added to the weight to prevent the algorithm from sticking on a specific strategy.

4. Training Algorithm

Combining the environment, agent and reward function presented above, here is our training algorithm for constructing optimal TDF glide paths. The first step of the algorithm is to setup the input data for agent network, that is, the $State^0$ generated by environment. The $State^0$ in the training process is the year 0 (age 24) stock return data and bond return data with investor's age of 24. The data $State^0$ is the input data of agent network, and the network will generate the weight of equity position next year, W_S^i, and the weight of bond position next year, W_B^i. After we get the asset allocation weights from agent network, we can calculate the reward using the $State^1$ (the stock returns and the bond returns of year 1 (age 25)) and the weights. The process continues until we reach the investors retirement (age 65). Hence, at the end of this episode, we calculate the reward using $State^{40}$ and weights generated by $State^{39}$ and agent network. Summing up all 40 rewards, we get the total reward of this episode which can be used to calculate gradients and update our agent network. Using Adam Optimizer to update the parameters in our agent network, this is our first training epoch. Repeat the steps for multiple times, the model for generating optimal TDF glide path in prepared. To avoid overfitting and converging to an abnormal glide path, we do not set the training number too large. We only trained the model for approximately 60 epochs. Fig. 3. shows our complete reinforcement learning training algorithm.

Fig. 3.

Algorithm 1 Reinforcement Learning Training Algorithm

while $n < TotalEpoch$ **do**
  Initialize Environment
  $i = 0$
  while $i < 40$ **do**
    $State^i = \{StockPrice^i, BondPrice^i, Age^i\}$
    $State^{i+1} = \{StockPrice^{i+1}, BondPrice^{i+1}, Age^{i+1}\}$
    Input $State^i$ into Agent Network
    Agent output W_S^i, W_B^i
    Input $State^{i+1}$ and W_S^i, W_B^i into Reward Function
    Calculate $Reward^{i+1} = R_C^{i+1} - \alpha DD^{i+1} - \beta G^{i+1}$
  end while
  Calculate $TotalReward = \sum_{i=1}^{40} Reward^i$
  Update Parameters in Agent Network
end while

Monte Carlo Simulation

To validate our model, we implement the Monte Carlo simulation on both traditional benchmark glide path and the glide paths generated by our reinforcement learning model.

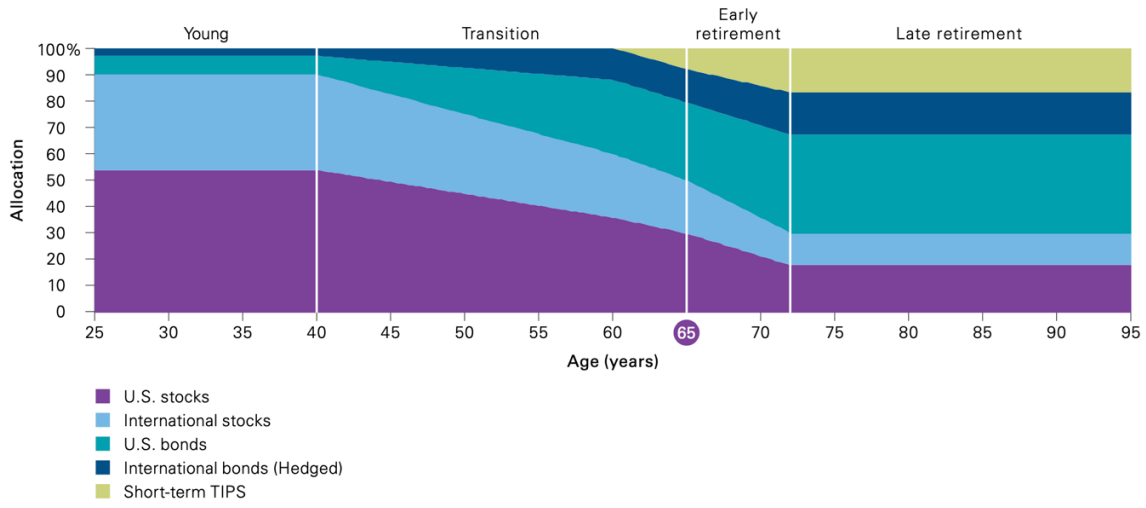
1. Monte Carlo Method

To implement the Monte Carlo simulation on our research, the simulation period and features of simulated data must be equal to our training data. Hence, our simulation period is also 41 years per simulation, and the features of simulations are also extracted from our training data, S&P 500 index and Fidelity Investment Grade Bond Fund from 1981 to 2021. The mean of returns and covariance of equity and bond data are extracted to implement the Monte Carlo simulation. The simulation processes are similar to the algorithm presented in Fig. 3. We first generate the 40-year equity and bond return data by Monte Carlo method and input the data into our trained reinforcement learning model. A series of weights will be generated by our agent network, which compose our optimal glide path for the simulation. We then calculate the return and risk-adjusted returns that symbolize the performance of the simulation. We repeat the simulation for 1,000 times and average the results of each simulation as the performance of our model.

2. Vanguard Glide Path

Vanguard is well-known for its variety of low-cost index mutual funds and ETFs and captured most investor money in its target-date funds in 2021 relative to other asset managers. According to Morningstar, retirement savers invested a net $55 billion in Vanguard's Target Retirement Funds last year. To choose a benchmark to compare with our model, the Vanguard glide path is selected as the traditional benchmark glide path. According to the report by Vanguard Research in 2015, the glide path for Vanguard TDF is designed beginning with 90% equity position from 25-year-old to 40-year-old and decrease to 50% at the target retirement date (65-year-old) as shown in Fig 4. The same 1,000 simulations are also executed on Vanguard glide path.

Fig. 4.



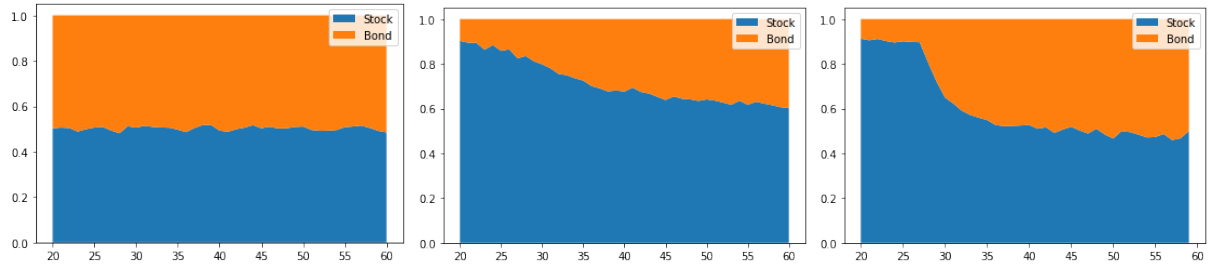
Source: Vanguard Research

Training and Simulation Results

1. Training Result

The initial output of agent network is designed as uniformly distributed. Hence, on Fig. 5. we can observe that the initial weights of the glide path are approximately 50% equity and 50% bond. After we trained the model for several epochs, the decreasing asset allocation path along with the increasing age of the investor demonstrates the effect of the glide path term in our reward function. At the end of the training process, an incurve appears between 25 to 35-year-old. In terms of rewards, the total reward increases from 2.93 to 3.12 during the training process. Fig. 5. shows the evolution of reinforcement learning glide path.

Fig. 5.



2. Monte Carlo Result

After each simulation of Vanguard glide path and the optimal glide path, we calculate Sharpe ratio, Sortino ratio, Maximum Drawdown, Calmar ratio and average yearly return. After 1,000 simulations, we calculate arithmetic average of the ratios. As shown in Tab. 1. the result shows that, our model outperforms the Vanguard glide path in terms of all risk-adjusted returns, and slightly underperforms in terms of average yearly return.

Tab. 1.

	Sharpe Ratio	Sortino Ratio	Maximum Drawdown	Calmar Ratio	Average Yearly Return
Vanguard Glide Path	0.589	1.654	-0.263	0.466	**0.101**
Reinforcement Learning	**0.679**	**1.963**	**-0.200**	**0.618**	0.095

Conclusion

1. Result Discussion

First of all, although the risk-adjusted returns of our model outperform the Vanguard glide path, the average return is lower. The one reason for lower average return is the shape of our final model. Compared to the Vanguard glide path, our model holds more bond position when the investor's age increases. While the Vanguard glide path holds approximately 80% equity position at 45, our model only holds 50% of equity position. Hence, even if the initial and final positions of both models are close, the shape of our model has a significant disadvantage concerning returns.

Secondly, as we mentioned in previous paragraph, our model has an unusual concave appears between 25 to 35-year-old. During our training processes, the drop of investment ratio appears more rapidly as we train the model with more epochs. At the end, if we train the model for 1,000 epochs, the model will converge to the extreme value we set (10% and 90%) with a single cliff appears depends on the β we set. The larger the β is, the earlier the cliff appears. Changes in hyperparameters and the reward function do not influence the appearance of the cliff, as the training times increase. Hence, our solution is to not train the model for too many epochs to avoid the problem.

Moreover, unlike most asset allocation research that focus on short-term and mid-term investment, TDF requires at least tens of years of data for one training episode. To make sure the veracity of the training data, we can only train the model with the same data set again and again. It may easily cause overfitting of our model. Although we can assume that the future price paths have the same features as our training data and apply the Monte Carlo simulation for model testing, it does not solve the problem of lacking in training data.

Lastly, the model is designed to change its asset allocation strategy based on different market scenarios. Whereas the result of Monte Carlo simulation shows that the optimal glide paths for every simulation are close. Considering the varying simulated market scenarios, the only fixed term in the reward function is the increasing punishment from the glide path term. Hence, weaken the influence of the glide path term may be necessary to increase the influence of market scenarios.

2. Future improvements

Besides the points discussed above, there are some directions of potential improvements of this model. First, in our research, we only consider the returns of equity and bond. To provide the model with more market information, other features like trading volume could be added into the model once the problem of lacking data being addressed. Also, we only consider an equity index and a bond in our asset allocation of glide paths. Whereas most of TDF providers include multiple assets in a single TDF. Hence, considering multiple assets will not only make the research closer to reality, but also bring the power of reinforcement learning into play. Finally, again, although our reward function provides a way for applying reinforcement leaning on designing TDF glide path, there are many problems occur while training the model. Hence, a redesigned reward function that could deal with the above problems will be a significant improvement.

3. Conclusion

In this section, we implemented Deep Policy Gradient to design optimal TDF glide paths under market scenarios and the investor's age. A new reward function was developed to make sure the decreasing asset allocation paths that conforms the basic rules of traditional TDF glide path. Using Monte Carlo simulation, our optimal glide path model outperformed the Vanguard glide path in terms of all risk-adjusted returns, but slightly underperformed in terms of the average yearly return.

References

Kolm, P. N., & Ritter, G. (2020). Modern perspectives on reinforcement learning in finance. *Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance*, *1*(1).

Kolm, P. N., & Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, *1*(1), 159-171.

Cao, J., Chen, J., Hull, J., & Poulos, Z. (2021). Deep hedging of derivatives using reinforcement learning. *The Journal of Financial Data Science*, *3*(1), 10-27.

I Alonso, M. N., & Srivastava, S. (2020). *Deep Reinforcement Learning for Asset Allocation in US Equities* (No. 2010.04404).

Kolm, P. N., & Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, *1*(1), 159-171.

Buehler, H., Gonon, L., Teichmann, J., Wood, B., Mohan, B., & Kochems, J. (2019). Deep hedging: hedging derivatives under generic market frictions using reinforcement learning. *Swiss Finance Institute Research Paper*, (19-80).

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

Abe, N., Melville, P., Pendus, C., Reddy, C. K., Jensen, D. L., Thomas, V. P., ... & Gardinier, T. (2010, July). Optimizing debt collections using constrained reinforcement learning. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 75-84).

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, *12*(4), 875-889.

Irlam, G. (2020). Lifetime Portfolio Selection: Using Machine Learning.

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, *87*, 267-279.

Weissensteiner, A. (2009). A $ Q $-Learning Approach to Derive Optimal Consumption and Investment Strategies. *IEEE transactions on neural networks*, *20*(8), 1234-1243.

Wang, Z., Huang, B., Tu, S., Zhang, K., & Xu, L. (2021, May). DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 1, pp. 643-650).