Deep reinforcement learning in stock portfolios

Yue Quan

Boston University, 1 Silber Way Boston, MA 02215

yuequan@bu.edu

Abstract. This paper investigates stock portfolios by application of Deep Reinforcement Learning (DRL) Models to achieve an optimal tactical asset allocation. The research problem is described as an optimization scenario that seeks to maximize the portfolio risk adjusted returns for a given portfolio asset allocation. The problem is set up with an initial capital investment which is invested in a set of assets. The initial strategic allocation is determined, which in our case is the equal weight allocation, and all of the capital is invested in the set of assets. At each point in time, the assets are reallocated according to the allocation which will increase the portfolio value. Two DRL models are implemented. The performance of the DRL models is compared with the uniform weights portfolio. The results show that, generally, two DRL models have higher cumulative returns.

Keywords: Deep Reinforcement Learning, Portfolio, Uniform Weights, Cumulative Return, Tactical Asset Allocation.

1. Introduction

Deep learning is divided into a variety of ways, including supervised learning, semi supervised learning, unsupervised learning as well as reinforcement learning. Reinforcement learning is different from the other three learning methods. When strengthening learning training, the environment needs to give feedback and the corresponding value. It is mainly to guide the training objects to make decisions at each step. In addition, it guides what actions can be taken to achieve specific goals. The inspiration of reinforcement learning comes from behaviorism theory in psychology [1]. Reinforcement learning has some characteristics:

(1) Delay feedback. In the process of reinforcement learning and training, the trial and error behavior of the training object gets environmental feedback. Sometimes, it may need to wait until the whole training is over before getting a feedback.

(2) Time is an important factor in strengthening learning. A series of environmental state changes and feedback of reinforcement learning are strongly linked with time. The whole training process of reinforcement learning changes with time. State feedback is also changing.

(3) The current behavior affects the subsequently received data. In supervised learning and semi supervised learning, each training data is independent. They have nothing to do with each other. However, in reinforcement learning, the current state and the actions taken will work in concert with the state received in the next step. In other words, there is a correlation among data [2].

Reinforcement learning can be applied in many fields. Tactical Asset Allocation (TAA) is an active research area with a number of research focusing on optimizing the tactical allocation of assets in a

© 2023 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

portfolio to take advantage and profit from market movements and anomalies. The main focus of many research works has to do with development of models that can accurately capture market movements and predict the expected future returns so as to rebalance the portfolio optimally based on the current and future market trends.

2. Literature review

The optimization problem of Tactical Asset Allocation (TAA) is presented as a Markov Decision Process (MDP) which can be solved by Dynamic Programming (DP) [3]. In his paper, Neuneier formalizes this representation and proposes the use of DP or Reinforcement Learning algorithms in solving such an optimization problem. Yang, Liu,Zhong and Walid have proposed a Deep Reinforcement Learning (DRL) Library that uses various DRL algorithms for portfolio construction and stock trading [4]. The methodology in this paper adopts this approach and make use of DRL models based on the Stable Baselines library which is a library for DRL algorithms.

Chakravorty et al., have proposed in their paper that Deep Learning based Global Tactical Asset Allocation shows the use of deep neural networks with macro-economic data in a walk-forward setting to implement TAA [5]. They used macroeconomic indicators and price-volume features to perform Tactical Asset Allocation and optimizing the weights using a custom utility function of a single metric. Further research in this regard can be done by using a utility function which takes into consideration a number of matrices for optimization. Their strategy showed a significant performance improvement across different runs. This demonstrates the advantage of using RL models for TAA as the model leans iteratively and improve the policy for each run.

Obeidat et. al., used a methodology for TAA based on prediction of future returns using the Long Short Term Memory (LSTM) Neural Network [6]. The predicted future returns where then used to optimize the asset allocation. As the main challenge with any asset allocation problem is accuracy with which expected returns can be estimated, their model showed better performance compared to the traditional passive approaches. However, the model performed worse in certain market trends. Also, the rebalancing period was fixed on a monthly basis. This can be optimized by use of RL in which we include the optimization period as part of the cost function.

3. Methodology

Deep Reinforcement Learning (DRL) model framework is shown.

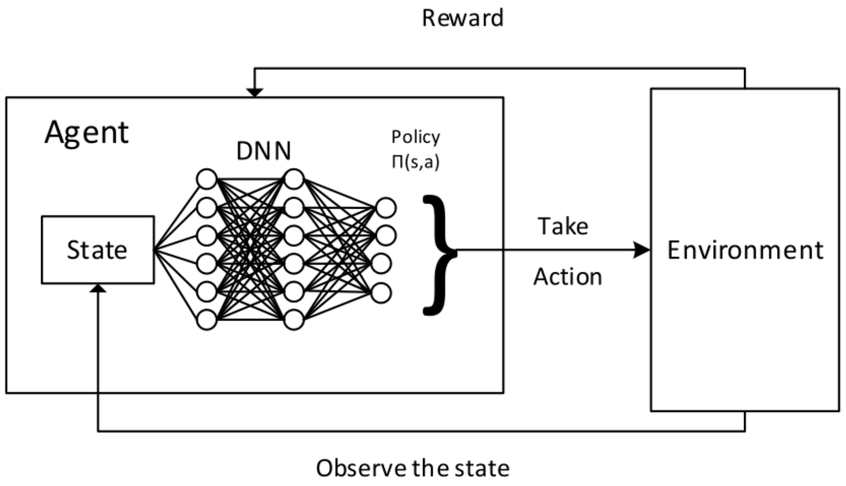

Figure 1. Representation of a deep reinforcement learning framework.

A Neural Network is proposed as the agent which finds an optimal policy that acts on the state of the environment to produce actions that maximize the reward function. We will consider a portfolio of n number of assets with a vector of close prices at time t given by V_t. The Normalized vector of the price is given by element-wide division of vector at time t by the vector at time t − 1:

$$Y_t = \frac{V_t}{V_{t-1}} \tag{1}$$

$$Y_t = \left[1, \frac{V_{1,t}}{V_{1,t-1}}, \frac{V_{2,t}}{V_{2,t-1}}, \ldots, \frac{V_{n,t}}{V_{n,t-1}}\right]^T \tag{2}$$

The portfolio vector at time t is determined by the element-wise product of the normalized price vector and the weights at time t − 1 divide by their dot product.

$$\mathrm{W}_\mathrm{t}' = \frac{\mathrm{Y}_\mathrm{t} \cdot \mathrm{W}_{\mathrm{t}-1}}{\mathrm{Y}_\mathrm{t}\mathrm{W}_{\mathrm{t}-1}} \tag{3}$$

The portfolio weight will evolve from Wt′ to Wt due to transaction costs. Then the price at time t will be P_t = =$\mu P_t'$ where $\mu \in [0,1]$ the transaction costs factor.

The portfolio value at time t is given by:

$$P_t = \mu_t P_{t-1} Y_t w_{t-1} \tag{4}$$

The portfolio log return is given by:

$$R_t = \ln \frac{P_t}{P_{t-1}} = \ln \mu_t Y_t w_{t-1} - 1 \tag{5}$$

The final portfolio value at time T is given by:

$$\mathrm{P_T} = \mathrm{P_0} \exp\left(\sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{R_t}\right) = \mathrm{P_0} \prod_{\mathrm{t}=1}^{\mathrm{T}} \mu_\mathrm{t} \mathrm{Y_t} \mathrm{w}_{\mathrm{t}-1} \tag{6}$$

P_0 is the investment at time zero that we start with. The objective function in the optimization problem will be to maximize this cumulative portfolio value at time T.

We define the Reinforcement Learning Configuration of our model as follows:

1. The state (S) consists of the high, low, and close. The covariance matrix of close prices and the market characteristics serve as input.

2. The action (A) is the desired weights allocation. The action at time t is supposed to be denoted by the weights vectors at time t − 1: $A_{t-1} = W_{t-1}$. As a consequence of the action A_{t-1} and the state inputs, we will have an action or weights allocation at time t which is $A_t = W_t$. Taking into consideration the transaction costs, our model should make a decision on how the weights are rebalanced from W_{t-1} weights W_t. This is done to maximize the accumulative portfolio value.

3. The reward function is given based on the total portfolio value adjusted for the transaction costs. The reward function adjusted for transaction costs is given by:

$$R_t(S_{t-1}, A_{t-1}) = ln\left(A_{t-1}Y_{t-1} - \mu \sum_{i=1}^{n} \left|A_{i,t-1} - W_{i,t-1}\right|\right) \tag{7}$$

The policy Π(s,a) determines the action to maximize the reward at each state. In our case the policy is made by a DNN using DRL.

In implementation of the model the following assumptions are taken into consideration:

• The market is liquid and it is possible to trade at any time;

• Transactions are small enough not to affect the market price of the assets which means the number of assets traded is small enough compared with the general market liquidity;

• Transaction costs have been assumed at 0.1% of each trade total value.

The methodology proposed for the asset allocation problem follows the following steps:

1. Stock Selection. We have considered the 30 stocks of Dow Jones Industrial Average (DJIA) to form part of our portfolio. Perform stock selection from the total stock constituent. This process ensures that the stocks considered for the portfolio are less volatile. Neural Network Auto Encoders are proposed to perform Stock Selection.

2. Feature Selection and Reduction. Technical indicators are added to the selected stocks to aid in the performance of the agent. In addition, we perform dimensionally reduction of the added features (technical indicators) to optimize the learning process of the model. Neural Network Autoencoders are used to perform feature reduction.

3. Benchmark Portfolio. We construct portfolio based on equal weights allocation, which is used to benchmark the DRL model.

4. Deep Reinforcement Learning Models. We develop two DRL models to perform asset allocation which are based on the A2C and PPO RL algorithms.

5. Backtesting and Evaluation. We test the performance of the DRL models against the uniform weights portfolio.

4. Results

4.1. Stock Selection

We use the data for the 30 stocks of the DJIA for the period from 01 January 2010 to 30 December 2020 giving us a total of 2768 data points. From the 30 stocks, 20 with lower volatility are selected using Autoencoders. We construct an Autoencoder with the total number of stocks in our input layer, we have an encoder layer with dimension 5 and a decoder layer with dimension 30. The stock data is reconstructed by passing it through the Autoencoder and then the reconstruction error of each stock is calculated. We then pick the 20 stocks with the lowest reconstruction errors.

Below is the summary of the Autoencoder model:

Table 1. Model summary for stock selection.

Layer (type)	Output Shape	Param #
Input_1 (InputLayer)	[(None, 30)]	0
Encoder_Input (Dense)	(None, 5)	155
Decoder_Input (Dense)	(None, 30)	180
Decoder_Activation_function (Activation)	(None, 30)	0
Total: 335		
Trainable: 335		
Non-trainable: 0		

The selected stocks are 'KO', 'PG', 'JNJ', 'PFE', 'MMM', 'VZ', 'MRK', 'JPM', 'XOM', 'V', 'AXP', 'IBM', 'MCD', 'CAT', 'HD', 'WMT', 'NKE', 'MSFT', 'DIS', and 'TRV'.

4.2. Feature Selection and Dimensionality Reduction

The following technical indicators are generated to be chosen as features [7].

1. Volatility Average True Range (ATR). It is an N-day exponential moving average of the true range values. It reflects the intensity of price fluctuations. Low ATR indicates a relatively cold market trading atmosphere. A high ATR indicates a relatively prosperous market trading atmosphere. Extreme high ATR or low ATR values can be seen as the reversal of the price trend or the beginning of the next trend.

2. Volatility Bollinger Band Width (BBW). It draws the distance between the upper and lower Bollinger Bands. The graph line represents the contraction and expansion of the bands on a basis of recent volatility. A high width usually indicates a slowing trend, while a low width indicates a forming trend.

3. Volume On-balance Volume (OBV). It is the average volume of each transaction. Its increase indicates that there is a large amount of business, which may be the main force or large customers entering the market. Its decrease indicates that most of the traders are small and medium-sized retail investors.

4. Volume Chaikin Money Flow (CMF). It is the sum of close location value volume in a certain time period divided by the total transaction volume. It indicates the strong or weak trend of a stock.

5. Trend Moving Average Convergence Divergence (MACD). It measures the aggregation and separation between short-term index average indicators and long-term index average indicators.

6. Trend Average Directional Index (ADX). it can show an investor if a trend is gathering steam or beginning to fade.

7. Trend Fast Simple Moving Average (SMA). It simply averages the closing price.

8. Trend Fast Exponential Moving Average (EMA). It weights the average closing price.

9. Trend Commodity Channel Index (CCI). It measures whether the stock price is beyond the normal distribution range.

10. Momentum Relative Strength Index (RSI). It calculates the comparison of market buying and selling power through the change of stock price, and speculates the future change direction of stock price.

Dimension reduction with the Autoencoders is implemented to reduce the number of input features from ten to four. Autoencoders consists of two parts which are the encoder which reduces the dimension of the input data in the latent hidden layer and the decoder which reconstructs the data from the hidden layer.

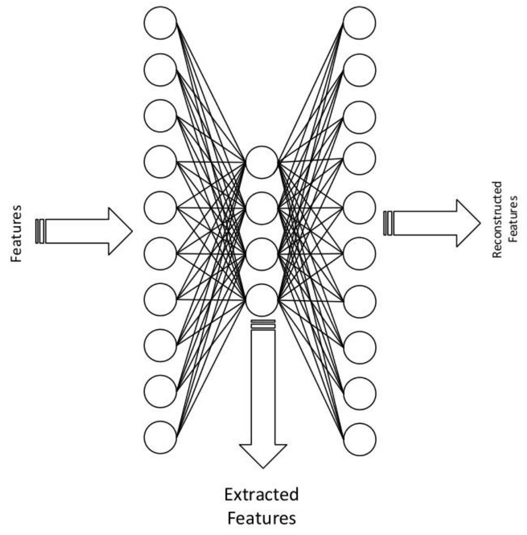

Figure 2. Structure of the Autoencoder.

The following is a graphical representation of an Autoencoder which we propose to use. It contains of One Input Layer, Two Hidden Layers, and One Output Layer.

Table 2. Model summary for stock selection.

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 4)	240
repeat_vector_1 (RepearVector)	(None, 20, 4)	0
lstm_3 (LSTM)	(None, 20, 100)	42000
time_distributed_1 (TimeDistributed)	(None, 20, 10)	1010
Total: 43250		
Trainable: 43250		
Non-trainable: 0		

4.3. Benchmark Portfolio

Uniform weights portfolio serves as a benchmark. An equally weighed portfolio of assets with the capital invested equally among the total number of assets is constructed. With n number of assets in a portfolio, the weight for each asset in the portfolio is given by:

$$w_1 = w_2 = \ldots = w_n = \frac{1}{n}$$

4.4. Deep Reinforcement Learning Models

Two DRL models are implemented using the following two RL algorithms [8].

1. Advantage Actor Critic (A2C) Algorithm. The training process of reinforcement learning is mainly divided into two processes. On the one hand, in the process of sample collection, it interacts with the environment with behavioral strategies to generate training samples. On the other hand, it uses the collected training samples to update the strategy. The above two processes are repeated in A2C implementation. It collects samples first. After collecting a certain number of training samples, it uses the gradient update formula to conduct a gradient update and generate a new strategy. The algorithm consists of the policy which is the actor and acts on the environment. The critic calculates the value function based on a set reward function and this helps the actor to determine the optimal policy.

2. Proximal Policy optimization (PPO). The PPO algorithm is an on-policy method that can be used for discrete or continuous action space environments.It solves the reinforcement learning problem of discrete and continuous action space. In PPO, the policy accepts the state and outputs the probability distribution of action. It samples actions in the action probability distribution, executes actions, gets returns, and jumps to the next state. In this process, it can use the strategy to gather a batch of samples, and then use the gradient descent method to train these samples. However, when the policy parameters are updated, these samples can no longer be used. Reusing policies to interact with the environment to gather data takes time. Therefore, it uses importance sampling so that these samples are reused.

We split the data into train-data and test-data, which is prepared to train the model and test its performance respectively. The environment setup within which we implement the algorithms is made up of the stock constituents, the prices, the technical indicators (features) and the covariance matrix derived from the one-year period data at each time step. The DRL agent interacts with the environment in an exploration-exploitation way. This involves making decisions which will maximize rewards by considering whether to make previous good decisions (exploitation) or to try new decisions with potential for greater rewards (exploitation). We make the assumptions that there are no short sales and all of our capital is used to invest in the portfolio stocks. The following algorithm gives a summary of the interaction within our environment.

Table 3. Algorithm: Portfolio Strategy By DRL.

Algorithm: Portfolio Strategy By DRL	
Input	s, state space which contains covariance matrix for stocks and features.
Output	Final portfolio value
Initialize	P_0 = 1,000,000 is the initial portfolio value. $w_0 = (1/m,...,1/m)$ is the initial weights and m is the number of assets.
for	$t = 1,2,...,n$
do	
	Observe the state s and output portfolio weights vector w_t at time t.
	Normalize the weights w_t whose sum is 100%.
	Calculate stock returns vector $r_t = ((v_{1,t}-v_{1,t-1})/v_{1,t-1},...,(v_{m,t}-v_{m,t-1})/v_{m,t-1})$, v is the closing price.
	Portfolio returns are $w_t^T r_t$
	Update portfolio value $P_t = P_{t-1}(1+w_t^T r_t)$
End	

The actions which the agents make are the portfolio weights and the policy is learned at each step that maximizes the portfolio value (reward function).

4.5. Backtesting and Evaluation

The weights for the equal weight portfolio is determined using the train data and these are applied on the test data. The DRL models, however, learns a policy for allocation and the train data and continuously rebalances the portfolio based on the state of the portfolio at every time space. The DRL portfolios rebalances the weights at each time step using a policy that would maximize the portfolio cumulative value. The figure below shows the cumulative returns of the DRL models benchmarked against the uniform weight and mean-variance portfolios using the training data.

We can see that the two DRL models have better performance than uniform weights portfolio. PPO model outperforms all the other portfolios on the training data. The cumulative return of A2C model ranks second. From 2011 to 2014, the cumulative return of the three kinds of portfolio are similar. Since 2015, the cumulative return of two DRL models has far exceeded the cumulative return of the uniform weights portfolio.

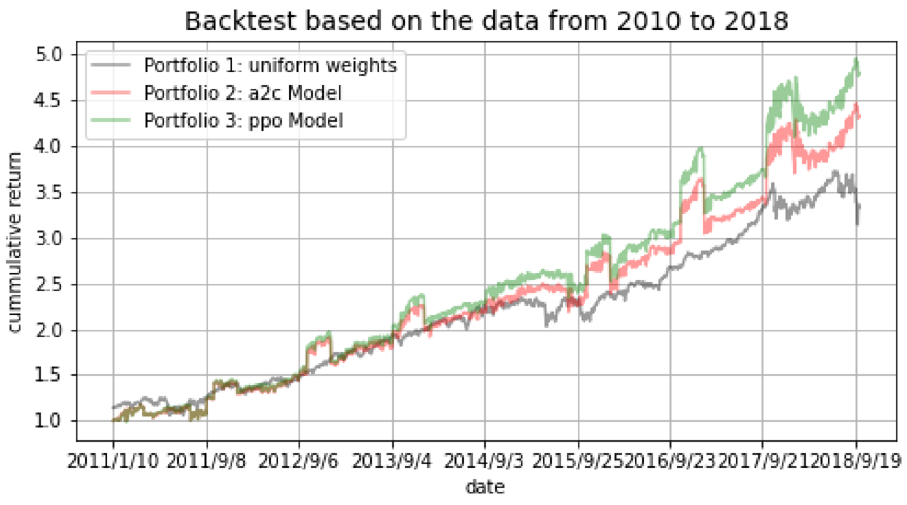

Figure 3. Cumulative return for train data.

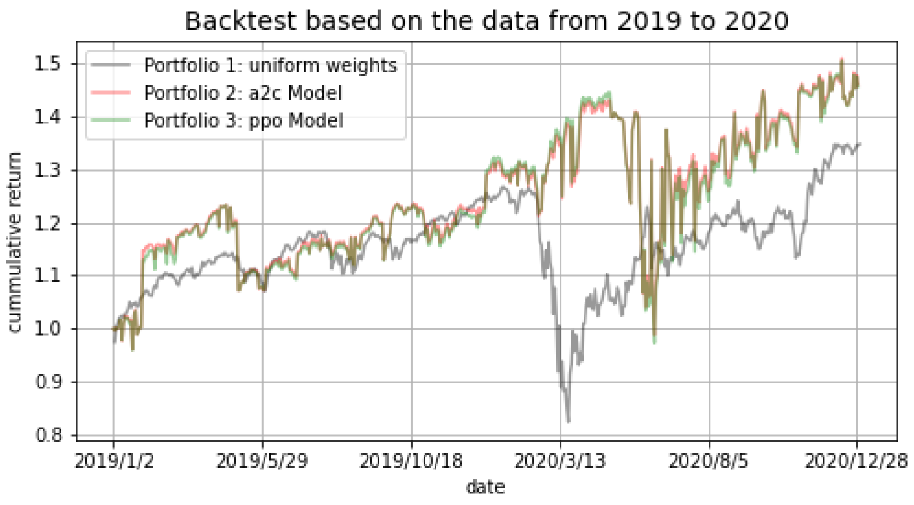

Figure 4. Cumulative return for test data.

With the test data, we see that, mostly, the two DRL models have better performance than uniform weights portfolio. A2C model and PPO model have similar performance. The under-performance of the DRL models during the period in Q2 2020 could be attributed to the market fall during the period of the COVID-19 pandemic. After Q2, the cumulative return of two DRL models increase a lot, and exceed the cumulative return of uniform portfolio.

5. Conclusion and future works

The two implemented DRL models do perform relatively better than the uniform weights portfolio, especially A2C model. We can conclude that DRL helps investor increase cumulative return. In addition, an analysis of the performance of the model during market down times as observed during the 2020 market crash shows that the DRL model has the ability to recover at a relatively quicker rate than the benchmark index. This is an indication of the Reinforcement Learning nature of the model to be able to explore and exploit the prevailing market environment.

Further works should be explored to improve the performance of the DRL models to be able to significantly outperform the uniform weights portfolio in all market conditions. We propose the following further works to be considered in refining the DRL model:

1. Explore the implementation of differential Sharpe ratio as the reward function for the DRL model;
2. Application of the DRL model on a bigger stock constituent;
3. Consideration of a number of technical indicators;
4. Consider using different portfolio initialization strategies apart from the equal weights initialization.

References

[1] Duryea, E. , Ganger, M. , & Wei, H. . (2016). Deep Reinforcement Learning with Double Q-learning.

[2] Zhixiong, X. U. , Cao, L. , Zhang, Y. , Chen, X. , & Chenxi, L. I. . (2019). Research on deep reinforcement learning algorithm based on dynamic fusion target. Computer Engineering and Applications.

[3] Neuneier, R., 1996. Optimal asset allocation using adaptive dynamic programming. In Advances in Neural Information Processing Systems. pp. 952-958.

[4] Yang, H., Liu, X., Zhong, S. and Walid, A., 2020. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. SSRN Electronic Journal.

[5] Chakravorty, G., Awasthi, A., Da Silva, B. and Singhal, M., 2018. Deep learning based global tactical asset allocation. SSRN Electronic Journal.

[6] Obeidat, S., Shapiro, D., Lemay, M., MacPherson, M.K. and Bolic, M., 2018. Adaptive portfolio asset allocation optimization with deep learning. International Journal on Advances in Intelligent Systems, 11(1), pp.25-34.

[7] Taghian, M. , Asadi, A. , & Safabakhsh, R. . (2022). Learning financial asset-specific trading rules via deep reinforcement learning. Expert Systems with Application (Jun.), pp. 195.

[8] Hirsa, A. , Osterrieder, J. , Hadji-Misheva, B. , & Posth, J. A. . (2021). Deep reinforcement learning on a multi-asset environment for trading. arXiv e-prints.