# Naive implementation of Reinforcement learning in Portfolio Management and its interpretation

Laurens Weijs
366515lw@eur.nl
Personal: *https://sites.google.com/view/laurensweijs*
Project: *https://laurenswe.github.io*

November 12, 2017

Master Thesis Quantitative Finance

Professor: Rutger-Jan Lange

**Abstract**

All machine learning methods commonly in use today are viewed as black boxes, the goal of this paper is to make one of these methods transparent in the context of Portfolio management. I will interpret the strategies implied by Reinforcement learning problem solving and relate them to the strategies implied by academic portfolio advice with the help of their classical portfolio management models.
Because Reinforcement learning is approximate Dynamic Programming, it is perfectly suited for the volatile Dynamic Programming environment of portfolio management compared to other machine learning methods in use. In terms of performance, this naive implementation is able to receive the same average terminal wealth with the classical portfolio management model with a low risk-aversion, while maintaining the standard deviation of the model with a high risk-aversion and improving on the total turnover. While strategy-wise this can mostly be explained by the conservative investing of the reinforcement learning methods in times of economic crises.

Erasmus School of Economics

# 1 Introduction

The correct Long-term portfolio management decision is the most important decision for large institutional investors such as mutual funds or pension funds. But the right trade-off of returns and risk in highly volatile markets still is one of the least understood topics. With the arrival of sophisticated quantitative modeling techniques, the stock market became more predictable and long-term portfolio management was revived again. Now with the arrival of Machine Learning models which do not need any knowledge of financial markets or the sophisticated models in use a new era of long-term portfolio management has begun.

Because the large institutional investors do not want to put faith in a model which has no theoretical background this research will help them generate the same results of the machine learning methods while knowing exactly what the model does. The need to adapt for pension and mutual funds is high because of several large hedge funds [Metz], which already made the step to embrace artificial intelligence for their portfolio management decisions.

In this paper, I will investigate what the underlying dynamics are of a naively implemented reinforcement learning method based on Sutton and Barto [1998], in an out of sample portfolio management context. And how the current portfolio management, called classical in this paper, can be improved with the knowledge of the reinforcement learning methods. The emphasis of this paper is mainly on understanding in which cases one of the two methods outperforms the other and where the possible improvements lie for both models.

Two methods are taken into consideration, first the classical portfolio management (CP) method. This method models stock returns with a Vector Autoregressive (VAR) model. Based on Monte Carlo simulations from this VAR model the average utility over all the future paths of stock returns for each portfolio weight is calculated, with the average utilities over time and over the portfolio weights, the weight in the next step will be chosen by the maximum average utility.

The second method is the reinforcement learning (RL) method, this method, in operation research often referred to as approximate dynamic programming, learns the optimal state value function in order to make a decision on what portfolio weight to take each period. This state value function represents the value of the next state given the action to take, the actor can than chose the state with maximum utility and chose the respective action. The state value function will be estimated with a neural network which gradually learns the optimal state value function.

Reinforcement learning methods are sometimes used in the context of finance, but not yet fully investigated in terms of their strategies in an out of sample context. Most research focuses on proving that it can be used to replace dynamic programming with a therefore known environment like shown in Hens and Woehrmann [2007], which is therefore not usable in this context. Or in Jiang and Liang [2017] and Jin and El-Saawy [2017] which show as a proof of concept that Deep reinforcement learning methods can work, but lack the further investigation of their inner workings. This out of sample context and the investigation on the strategies used by the reinforcement learning method is provided by this paper in an enough bounded environment.

The reinforcement learning method which optimizes simply over the maximum reward, instead of utility, shows resemblance with the classical portfolio method for a risk aversion of $\gamma = 2$, which stands equal to a low risk-aversion of the investor. While having a lower standard deviation and a lower turnover, this is mainly due to the relative conservativity of the RL method in investing. This conservativity translated into action is that the RL rarely takes a full weight in one of the weights, while it does when it thinks its completely sure given the historical returns. This holds true in the simulations and also in the real case.

Compared to other literature this paper compares RL methods to state-of-the-art econometric portfolio management techniques, and together with these methods tries to give more insights into the RL methods. One could conclude that given the sophisticated tools there is not more information

in the dataset currently at hand. While assets according to the Efficient Market Hypothesis (EMH), stated in Fama [1970], suggests that prices reflect all available information, one should consider larger datasets of more diverse information for the methods to gather from the datasets.

### 1.1 Research Questions

*How does the reinforcement learning perform in the a simulated environment with known parameters compared to classical portfolio management?*

Especially because reinforcement learning methods are widely seen as a black box, it would certainly help to investigate the method in a simulated and controlled environment. The results can help to understand the RL methods and possible improve on the classical portfolio management techniques.

*How can reinforcement learning improve on classical portfolio management or the naive portfolio diversification out-of-sample?*

Diris et al. [2015] recognizes that the true data generating process is not known and therefore the VAR model is prone to misspecification and to parameter estimation errors. This leads to the fact that dynamic portfolio is the same as the repeated myopic portfolio, only looking one step at each time and is not able to beat the naive portfolio diversification of equal weights. Reinforcement learning is tries not to estimate the dynamics of assets but just the dynamics of the state value function and therefore it can be able to improve on estimation or even the naive diversification.

## 2 Data

This research will be based on the monthly stock and bond market of the United States. Because this research considers three assets classes to choose from as an investor, I need to gather these three classes from various data sources. Please see the Table 1 for a description of the data and its source.

**Table 1** – Data used in this research combined with the data source.

| **Stock class** | **US Asset** | **source** |
|---|---|---|
| Equity | Weighted average of NYSE, NASDAQ, and AMEX | CRSP |
| Long-term nominal bonds | Nominal 5-year T-Note | FRED |
| Short-term nominally risk-free T-bills | Nominal 3-month T-Bill | FRED |

The summary statistics stated in Table 2 shows typical market behavior, the safest asset(with the lowest volatility but also a low return is the Ex-post T-Bill rate $r$. The stocks, on the other hand, are more volatile but on average have a higher return.

**Table 2** – Summary statistics of the three assets taken into consideration, the ex-post real T-bill, Value-weighted stock returns, Excess-bond returns. The data set starts in February 1954 and ends in December 2016 and are notated in monthly returns.

| | $r$ | $x_s$ | $x_b$ |
|---|---|---|---|
| Avg. | 0.0008 | 0.0057 | 0.0013 |
| Std. dev. | 0.0033 | 0.0432 | 0.0146 |
| Min | -0.0108 | -0.2305 | -0.0687 |
| Max | 0.0193 | 0.1594 | 0.0951 |
| AR(1) | 0.4456 | 0.0907 | 0.1193 |

In this research, three models are considered to simulate the asset returns from the Constant Ex-

pected Return(CER) [1] model, Vector Autoregressive(VAR) model, and a Bayesian Vector Autoregressive(BVAR) model. The models are shown respectively in Equations 1,2, and 3. Where $y$ denotes the vector of asset returns $(r, x_s, x_b)'$.

$$y_t = \mu + \varepsilon_t, \text{ with } \varepsilon_t \sim N(0, \sigma^2) \tag{1}$$

$$y_t = \hat{A} + \hat{B} y_{t-1} + \varepsilon_t, \text{ with } \varepsilon_t \sim N(0, \sigma^2) \tag{2}$$

$$y_t = A + B y_{t-1} + \varepsilon_t, \tag{3}$$

$$\text{with } \varepsilon_t \sim N(0, \sigma_i^2),\ A_i, B_i \sim N(\hat{A} \text{ or } \hat{B}, \frac{\sigma_i^2}{T}), \text{ and } \sigma_i^2 \sim iGamma2(SSE, T-1)$$

Where for each simulation $i$ the uncertainty parameter of Equation 3 has a different draw of the inverse Gamma distribution with as parameters the sum squared residual of the shrinkage model stated in Equation 2 and $T-1$.

**Table 3** – Different assumptions involved in the asset allocation problem.

| **Distribution returns** | Constant parameters | Time-varying parameters |
|---|---|---|
| Known parameters | 1 | 2 |
| Unknown parameters | 3 | 4 |

Based on these assets, simulations are made for four different scenarios based on the knowledge of the distribution of returns. See Table 3 for the different scenarios. In scenario 1 and 3 the CER and VAR model are simulated and in scenario 1 one can solve the optimal weights allocation analytically for the CER model and approximate analytically for the VAR model. In scenario 2 and 4 however, I simulate from the BVAR model in order to get time varying parameters. Scenario 2 can again be solved approximate analytically but this time for the known BVAR model. Classical portfolio management assumes that by estimating a model of the returns they have found the true Data Generating Process of the returns and thus operate in the domain of scenario 1 and 2. This research, however, doesn't assume it knows the model of the underlying assets and operates in scenario 3 and 4 therefore. The parameters for the simulations are shown in Table 4

**Table 4** – Parameters of the simulation models in the control experiment estimated on the real data described in the beginning of the data section and also available on https://laurenswe.github.io. $y_t$ is denoted as a vector of the three assets: $(r, x_s, x_b)'$. *The Bayesian VAR is simulated with the help of the Gibbs sampler and the two probability density functions given in the second and third row with a thinning of 100 and burn-in of 1000.

| **Model:** | Simulation equation |
|---|---|
| CER | $y_{t+1} = \begin{pmatrix} 0.0008 \\ 0.0057 \\ 0.0013 \end{pmatrix} + \epsilon_{t+1}, \text{ with } \epsilon_{t+1} \sim N\left(0, \begin{pmatrix} 0.0000 \\ 0.0019 \\ 0.0002 \end{pmatrix}\right)$ |
| VAR | $y_{t+1} = \begin{pmatrix} 0.0004 \\ 0.0047 \\ 0.0012 \end{pmatrix} + \begin{pmatrix} 0.46 & -0.01 & 0.01 \\ 0.10 & 0.07 & 0.36 \\ 0.45 & -0.06 & 0.11 \end{pmatrix} y_t + \epsilon_{t+1}, \text{ with } \epsilon_{t+1} \sim N\left(0, \begin{pmatrix} 0.003 \\ 0.043 \\ 0.014 \end{pmatrix}\right)$ |
| BVAR* | $y_{t+1} = B_0 + B_1 y_t + \varepsilon_{t+1}, \varepsilon, \text{ with}$<br>$P(\Sigma \mid B, Y) = iWishart\left((Y - XB')'(Y - XB'), T\right)$<br>$P(B \mid \Sigma, Y) = N_{trunc}\left(\hat{B}, \Sigma \otimes (X'X)^{-1}\right)$ |

[1] The CER model is introduced because it has a clear analytical solution, the myopic solution.

# 3 Methodology

## 3.1 Problem Statement

This research focuses on a world where an investor can choose between three assets: equity, long-term nominal T-Notes, and short-term nominal T-Bills. The problem stated for a long-term investor in portfolio management is a dynamic intertemporal weight optimization problem with uncertainty about the future states. The objective function of such an investor with intertemporal utility $U(\cdot)$ is given by:

$$\max_{w_t,\dots,w_{t+K-1}} \mathbb{E}_t[U(W_{t+K})] \text{ s.t.} \tag{4}$$

$$W_{s+1} = W_s(w_s' r_{s+1} + R^f_{s+1}) \quad \text{, for } s = t,\dots,t+K-1 \tag{5}$$

$$w_s' \iota = 1 \quad \text{, with } \iota \text{ being a vector of ones} \tag{6}$$

With the second equation being the budget restriction, including the restriction that the weights should count to one. Your wealth one period ahead can only change by the change in the assets times the weights in the assets. The other parameters in these equations are described as follows: $K$ is the number of periods to optimize over in the future, $W_s$ is the current wealth at time $s$, not to be confused with $w_s$ which is the weight per asset class at time $s$, $r_{s+1}$ is a vector of excess returns on the asset classes one period in the future, $R^f_{s+1}$ is the current Risk-free asset at time $s+1$, and $U(\cdot)$ is the utility function, which in this research is the power utility function defined by $\frac{(\cdot)^{1-\gamma}}{1-\gamma}$, with $\gamma$ being the risk aversion of the investor. The risk aversions taken into account in this research are $\gamma = 0, 2, 5, 10$, with 0 being completely risk neutral and 10 being risk averse.

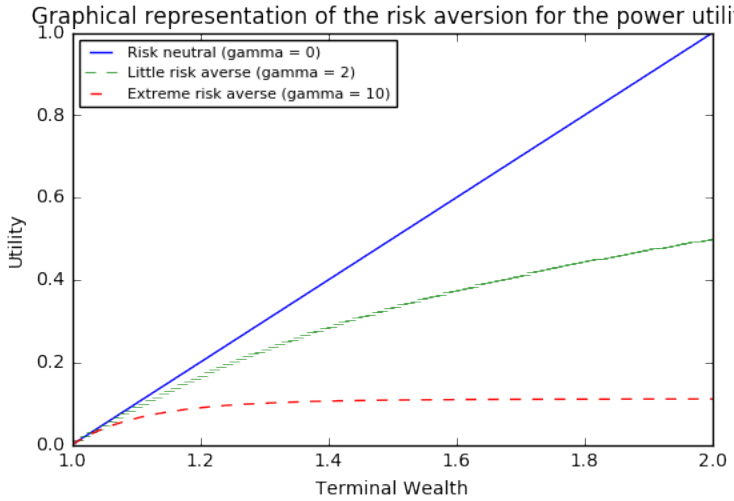



**Figure 1** – Utility gained for units of Terminal Wealth with the power utility for three different values of $\gamma$.

This dynamic problem can be transformed into the following Bellman equation to show the recursive nature of this dynamic optimal strategy:

$$V_{t+K}(W_t, \theta) = \max_{w_t,\dots,w_{t+K-1}} \mathbb{E}_t[U(W_{t+K})] \tag{7}$$

$$= \max_{w_t} \mathbb{E}_t \left[ \max_{w_t+1,\dots,w_{t+K-1}} \mathbb{E}_{t+1}[U(W_{t+K})] \right] \tag{8}$$

$$= \max_{w_t} \mathbb{E}_t \left[ V_{t+1}(W_t(w_t' r_{t+1} + R^f_{t+1}, \theta)) \right] \tag{9}$$

$$\text{with terminal condition:} \quad V_t(W_t) = U(W_t) \tag{10}$$

With $\theta$ representing the parameters of the model from the assets classes stated in the different scenarios of Table 3. When the parameters are unknown it is therefore not possible to estimate the expectation of the right side immediately.
When you now write the budget constraint of 5 in terms of the starting wealth and terminal wealth you will get:

$$W_{t+K} = W_t \prod_{s=t}^{t+K-1} (w_s' r_{s+1} + R_{s+1}^f) \tag{11}$$

When you substitute Equation 11 into the Bellman equation, work out $W_t$ from the expectation, and use the power utility function as utility you will get the following equation:

$$V_{t+K}(W_t, \theta) = \max_{w_t} \mathbb{E}_t \left[ \frac{\left( W_t (w_t' r_{t+1} + R_{t+1}^f) \right)^{1-\gamma}}{1-\gamma} \max_{w_{t+1} \cdots w_{t+K-1}} \mathbb{E}_{t+1} \left[ \left( \prod_{s=t+1}^{t+K-1} (w_s' r_{s+1} + R_{s+1}^f) \right)^{1-\gamma} \right] \right] \tag{12}$$

From this point, you can solve the Bellman equation, which is a necessary condition for optimality, in four different ways with differing assumptions about the distribution of returns see Table 3. I state here, that the classical way in Portfolio Management is the solution method performed by Diris et al. [2015] with the assumption that the distribution is known and covers Scenario 1 and 2 of this table. And the proposed way, which is raised by the Reinforcement Learning community, has no assumptions about the distribution of returns and cover scenario 3 and 4. All four cases will be simulated in this research and as an extension, the methods will be tested against the real data.

## 3.2 Classical portfolio management

In the classical way, I assume that one knows a model of the returns and are therefore is able to calculate the estimation of the wealth in the next period ( $\mathbb{E}_{t+1}[U(W_{t+K})]$ ). This model is first estimated from the true sample of the historical returns of asset prices and the mathematical estimation is then retrieved from the simulation of this estimated model of returns. The industry workhorse is a VAR model, this can be shown mathematically as:

$$y_t = A + B y_{t-1} + \varepsilon_t \tag{13}$$

$A$ is a vector of intercepts, $B$ is an $(n \times n)$ of the slope of the equation, and $\varepsilon$ is a vector of idiosyncratic errors.

With this VAR model stated before, the probability distribution of the returns is known. According to Bayesian statistics, you can approximate the expectation of any distribution by the simulated sample averages.

$$\mathbb{E}[f(\theta)] \approx \frac{1}{N} \sum_{i=1}^{N} f(\theta_i) \tag{14}$$

This is exactly what I will do with the help of the MCMC algorithm, an extensive reversible Markov Chain (MC) is sampled from the estimated VAR model and is assumed to be a true representation of the real underlying dynamics of the asset returns. This Markov Process (MP) can be seen graphically in Figure 2, where each transition is chosen by random by the stated transition probability. Please note that in classical portfolio theory the focus is not on the Markov Chains, but in this research, the focus is on MC's due to the clear link with Reinforcement learning methods.

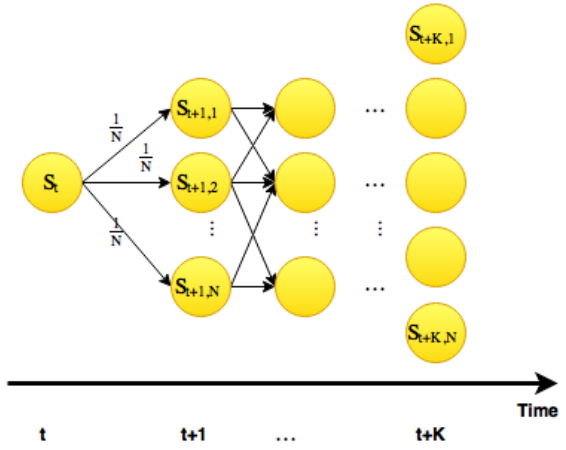



**Figure 2** – Graphical representation of the Markov Process of the asset prices, transitions are given in transition probabilities and determined by nature.

Given that you can construct the Markov Process[2] of the asset returns, one can also construct a Markov Decision Process (MDP) in which you make the whole problem statement of the optimal portfolio weight a Markov Process. See Figure 3 for the graphical representation of this decision process. In each node, the value of the objective function is shown and an action should be executed at each time period, this action is the weights in the corresponding asset prices. After that the weights have been chosen, nature decides based on the Markov process of the prices in Figure 2 what the price in the next period will be and therefore the new value of the objective function.

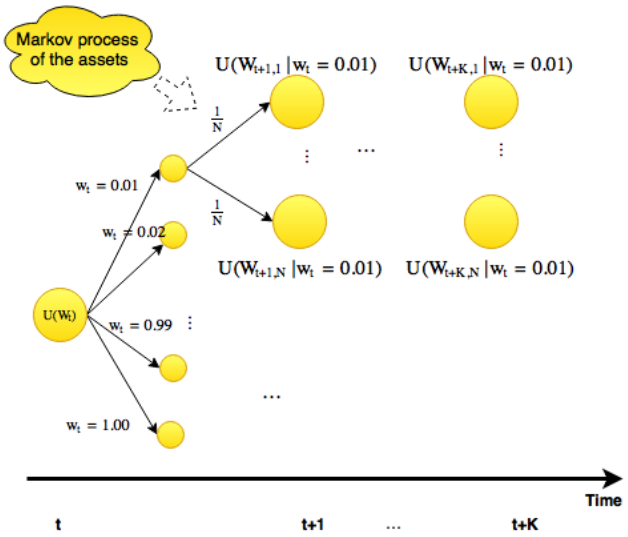



**Figure 3** – Graphical representation of a Markov Decision Progress representing the Portfolio problem statement of Equation 5. Small nodes are actions nodes, where the weights should be determined and large nodes represent the state of the objective function.

Now that the objective function is transformed into a fully deterministic Markov Decision Process you can now solve the problem by Backward induction, this is actually the same as the method of Value iteration in Reinforcement learning. Working from the back you calculate the Expectation of utility for each weight at time $t + K - 1$ by taking the average over the utilities acquired by each $w_{t+K-1}$. By maximizing each period over the expected utility and starting at the end of the period you maximize the Bellman Equation stated in Equation 9.

[2] For the ease of consistency in literature I write here Markov Processes instead of Markov Chains, while they are equivalent in discrete state spaces.

## 3.3 Extensions

**Predictability of returns**

Diris et al. [2015] not only implements the base case stated in the section before, it also takes the Bayesian estimation of the probability distribution of the assets into account and predictability in the returns. The latter one now implies that instead of $\mathbb{E}[f(\theta)], \mathbb{E}[f(\theta|z_t)]$ needs to be estimated. You can approximate the conditional expectation by the fitted values of the across path regression, that is the fitted values of the regression of the simulated utilities on the state variables.

**Bayesian inference**

When considering a Bayesian inference, top right cell of the scenario table in Table 3, on the asset prices, one can still make use of the MCMC algorithm to transform the assets into a known Markov Process. Now each path that is sampled by the MCMC has different parameters of the probability distribution. This will in most cases result in a higher variance in the values of the Markov Process.

**Transaction costs**

By adding transaction costs according to Gârleanu and Pedersen [2013] the Bellman equation in Equation 12 changes to:

$$\max_{w_t} \mathbb{E}_t \left[ \frac{\left( W_t(w_t' r_{t+1} + R_{t+1}^f - \Delta w_t TC) \right)^{1-\gamma}}{1-\gamma} \max_{w_{t+1} \cdots w_{t+K-1}} \mathbb{E}_{t+1} \left[ \left( \prod_{s=t+1}^{t+K-1} (w_s' r_{s+1} + R_{s+1}^f - \Delta w_s TC) \right)^{1-\gamma} \right] \right] \tag{15}$$

The term added to the equation is the transaction costs $\Delta w_t TC$, $\Delta w_t$ is the difference between the weights of time $t$ and $t-1$ and and TC are the constant transaction costs per unit of weight. It is harder to simulate with the numerical solution proposed in Diris et al. [2015], therefore the closed-form solution from Gârleanu and Pedersen [2013] is taken. The optimal weight is the weighted average of the current weight and the aim portfolio (the weighted average of the current and future expected Markowitz portfolios),

$$w_t = \left(1 - \frac{a}{TC}\right) w_{t-1} + \frac{a}{TC} aim_t \tag{16}$$

With $a$ being the optimal weighting scheme futher specified in Gârleanu and Pedersen [2013] For the further specification, I would like to refer to to that paper. The aim of this paper is not about this the dynamic strategy with transaction costs. It is more important for the RL method to include transaction costs because of the fluctuate nature of the machine learning methods.

## 3.4 Reinforcement learning in portfolio management

In reality, the underlying Markov Process of the asset returns is unknown and also volatile in terms of its model parameters. Therefore, I make use of a model free RL method. Please note, that you need to take the natural log of the objective function and the budget constraint in order to be able to solve the problem by reinforcement learning. This is needed so you can rewrite the objective function in the summation of intermediate functions/rewards, in order for our RL agent to learn from each step in the future. The main difference between the classical method and the reinforcement method is that the latter solves the problem from the beginning propagating forward to the MDP in comparison to the value iteration by backward induction which calculates the optimal path backward. The main problem one faces with RL is that one is unable to get a completely correct estimation of $\mathbb{E}_{t+1}[U(W_{t+K})]$.

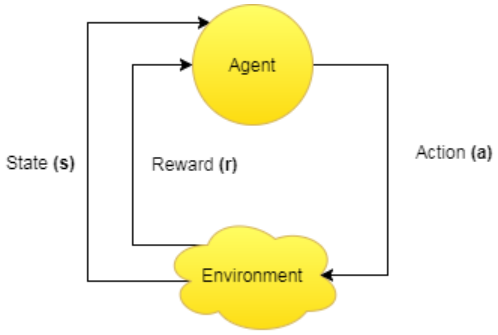



**Figure 4** – High-level description of an reinforcement learning agent and how it interacts with the environment, in this case the assets.

A reinforcement learning agent can be represented like in Figure 4. Each point in time the agent takes an action based on its current knowledge of the problem stated and based on its action the environment will give a reward to the agent and the new state of the environment. In this research the state $(s)$ is represented as the current level of the assets, the action $(a)$ is the weights of the assets, and the reward $(r)$ is the log utility gained from moving from state $s_t$ to $s_{t+1}$ with action $a$(not to be confused with the returns).

Instead of trying to estimate the expectation of utility next period, RL makes use of the Bellman equation and the only known parameter $R_{t+1}$, the immediate reward from the transition of time period $t$ to $t+1$. Because it is more interesting which actions to take in order to receive the maximum value function, the value function of Equation 9 needs to be decomposed into the action-value function. This is simply the value function given a certain action, these functions are defined in Equation 17 and 18. The value function is actually the same as the intermediate Bellman equation of Equation 7.

$$V_t(s) = \mathbb{E}_t[U(W_{t+K})|S_t = s] \tag{17}$$
$$Q_t(s,a) = \mathbb{E}_t[U(W_{t+K})|S_t = s, A_t = a] \tag{18}$$

This value function can be rewritten to an immediate reward, directly resulting from a specific action performed in a given state, plus the value function of its successor state. This Bellman Equation is shown in Equation 19 and also the corresponding Bellman Equation of the action-value function in Equation 20. With also the power utility, written in the equation, so that you have an analytical term for the immediate return $R_{t+1}$.

$$\begin{aligned} V_t(s) &= \mathbb{E}_t[R_{t+1} + V_{t+1}(S_{t+1})|S_t = s] \\ &= \mathbb{E}_t\left[\log\left(\frac{\left(W_t(w'_{t,optimal}r_{t+1,S_t} + R^f_{t+1,S_t})\right)^{1-\gamma}}{1-\gamma}\right) + \log(V_{t+1}(S_{t+1}))\middle| S_t = s\right] \end{aligned} \tag{19}$$

$$\begin{aligned} Q_t(s,a) &= \mathbb{E}_t[R_{t+1} + Q_{t+1}(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \\ &= \mathbb{E}_t\left[\log\left(\frac{\left(W_t(w'_{t,A_t}r_{t+1,S_t} + R^f_{t+1,S_t})\right)^{1-\gamma}}{1-\gamma}\right) + \log(Q_{t+1}(S_{t+1}, A_{t+1}))\middle| S_t = s, A_t = a\right] \end{aligned} \tag{20}$$

In summary, in RL you assume that the expectations of assets and therefore its respective MP is unknown. By construction, the value function only exists out of the intermediate value functions and is characterized by Equation 19 and 20. If one is in the last period and knows the reward given a certain action, one can back propagate the last action-value function into the rest of the action-value functions, by value iteration. This could also be done for the value function, but I am more interested in which action to take given the state. This method would be an approximate dynamic

programming solution and can solve scenario 1 and 3 from Table 3.

#### Unknown Markov Decision Process / Q-learning

Now I assume that the MDP made with the help of the econometric model is not known, this results that the right side of Equation 20 can not be estimated at all times. Now you need to estimate $Q(s,a)$ at specific states in order to be able to solve for the optimal action-value function at the end of the period. You would, therefore, learn the optimal Q values, in the discrete case a Matrix representing the knowledge of the MDP describing the trading system. First, the Q matrix initialized at zero and updated iteratively with the reward function and the next Q value in the matrix. Because the reward function is known at each state and action pair you can iterate w.r.t. the adjusted greedy policy ($\varepsilon$-greedy heuristic) to update the values of the Q matrix. This heuristic chooses the next action based on the maximal Q function and chooses a random action by the parameter $\varepsilon$. This updating rule follows directly from Equation 20 and is given in Equation 21.

$$q(s_t,a_t) = q(s_t,a_t) + \alpha \left[ r + \max_a q(s_{t+1},a_{t+1}) - q(s_t,a_t) \right], \tag{21}$$

with $\alpha$ being the learning rate, the rate to which extent the value is updated with the estimated value of the Q function. And the action $a$, here written as the action leading to the highest action-value function in the next period, chosen greedily but also some random deviation included, otherwise only one path will be updated continuously without learning rest of the Q matrix. By iteratively updating $q(s,a)$ it converges to the optimal action-value function $q_*(s,a)$ [Melo, 2001], within a deterministic framework but also within a stochastic framework as stated by Jaakkola et al. [1994]. And from the optimal Q table, one can calculate by value iteration or backward induction the optimal policy/weights for the assets.

### 3.5 Extensions Reinforcement Learning

#### Transactions costs

Adding transaction costs to the RL method would mean that the immediate Reward should be reduced in Equation 20 relative to a number of assets traded which has a direct relation of the action to take by the method. The new reward function would, therefore, become equal to Equation 22.

$$R_{t+1} = \log \left( \frac{\left( W_t (w'_{t,A_t} r_{t+1,S_t} + R^f_{t+1,S_t}) \right)^{1-\gamma}}{1-\gamma} - \Delta w_t TC \right) \tag{22}$$

#### Function approximation

Because the states are actually not representable in discrete values because the assets have a continuous state space and in this research, even three continuous variables are taken into consideration the Q table would become large. This makes it not viable in terms of memory but also not in terms of learning time. One solution for this would be to estimate the value function by function approximation: like a linear combination of the assets, an econometric model, or a neural network.

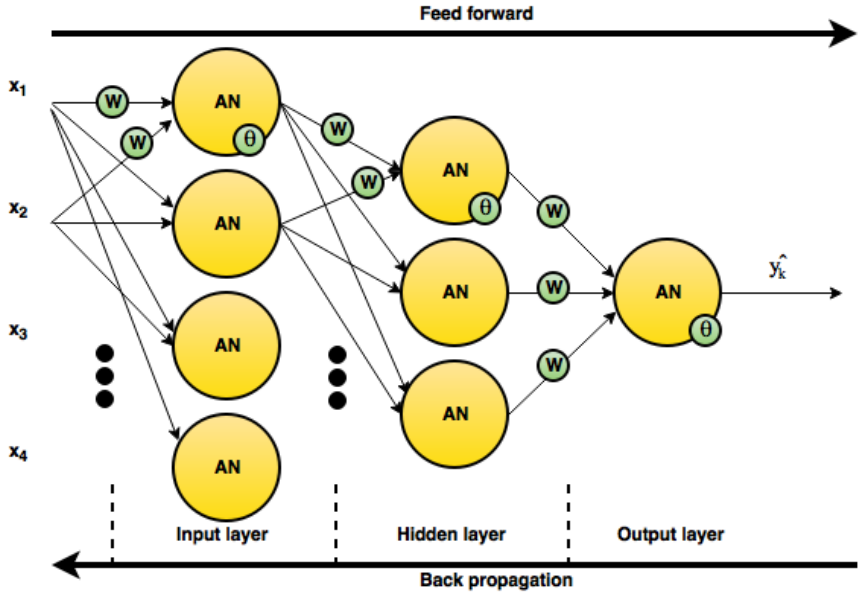
$$\hat{Q}(s,a,\theta) \approx Q_t(s,a) \tag{23}$$

This is more efficient because this function could generalize from states already encountered to new states and therefore reduce the memory usage. Although it never converges to the true action-value function in practice it tends to oscillate around and approximate the action-value function closely. In this research, it would not be viable to construct a table of all the states and the infinite actions when considering continuous weights.

To optimize the parameters of $\theta$ from the function approximation I make use of the stochastic gradient descent method for Neural Networks[3]. For the ease of generality, Neural Networks are used so that you do not have to optimize over the functional form of the function approximator because you can form many different kinds of functions with Neural Networks. Keep in mind that Neural Networks can represent any abstract relationship between two variables of interest, from linear to nonlinear, from low dimensional to high dimensional relationships. The loss function that would be used to propagate back through the Neural Network is defined in Equation 24. Only the loss function is stated because that is the sole instrument needed for the Neural Network to update.

$$Loss = \sum_{a} \left( R + \max_{a_{t+1}} (q(s_{t+1}, a_{t+1})) - q(s_t, a_t) \right) \tag{24}$$

**Figure 5** – Graphical representation of a Neural Network




### Clipping of rewards

One should take special notice in the clipping of the rewards for the Reinforcement Learning to properly function. In order for the model to be flexible with multiple economic regimes, and therefore probably different data generating functions the rewards should be rescaled to the range $[-1, 1]$ [Hasselt et al., 2016]. This is especially the case in this data set because returns are mostly positive for the bonds and stock index, if the rewards are mainly positive it would result in an almost continuous grow in the Q-values and, therefore, never converge.

## 4 Results

This section is split up into three parts: the results of the different models in the three simulated cases, the different models applied to the real data and the interpretation of the strategies in terms of portfolio advice.[4] For reference, also another strategy has been added to the tables, the max strategy this would be the maximum possible to get in the 60 periods to take into account. This

[3]The package this paper uses for Neural Networks is Tensorflow for Python.

[4]For the code and the data used for this Master Thesis please have a look at my Github pages via https://laurenswe.github.io

is simply the summation over the max of the two stocks, this would be the upper limit of what is possible with the models in terms of terminal wealth.

## 4.1 Simulations

In this section the results are shown of the models run on simulated data specified in Equations 1, 2, and 3 for the classical methods by specific $\gamma$ denoted by CP and the reinforcement learning methods denoted by RL-NN (the abbreviation after the hyphen denotes the function approximation method, in this case, NN represents a neural network).

Table 5 shows the results for the model-independent strategies, special notice should be given to the perfect foresight. This model takes a full stance in the bonds or the stocks while knowing what comes the next period, this model, therefore, serves as the perfect timing benchmark although unrealistic. Also, it should be mentioned that the terminal wealth for the VAR and BVAR simulated values are relatively low and will result in a losing portfolio.

**Table 5** – This table shows the the average weight in the stock $x_s$ ($w_s$), the average terminal wealth ($\bar{TW}$), and the standard deviation of terminal wealth ($\sigma(TW)$) for three model-free strategies and one reference strategy for three different samples of the return models stated in Equations 1, 2, and 3.

| Method: | CER | | VAR | | BVAR | |
|---|---|---|---|---|---|---|
| | $\bar{TW}$ | $\sigma_{TW}$ | $\bar{TW}$ | $\sigma_{TW}$ | $\bar{TW}$ | $\sigma_{TW}$ |
| 1/N each asset | 1.17 | 0.00 | 1.04 | 0.11 | 0.92 | 0.13 |
| Full r | 1.05 | 0.00 | 1.00 | 0.04 | 1.00 | 0.03 |
| Full stock | 1.40 | 0.02 | 1.19 | 0.27 | 0.82 | 0.28 |
| Full bond | 1.08 | 0.00 | 0.98 | 0.12 | 0.99 | 0.10 |
| Perfect foresight | 1.40 | 0.02 | 3.31 | 0.36 | 2.61 | 0.65 |

For the constructed models stated in this paper, the simulated values were principally executed as a bare minimum for the RL method. The results of the CP and RL methods for the simulated values are shown in 6. Especially with the CER simulated returns because the stock returns are for 95 % strictly greater than the bond returns, here the main drawback of the RL in consideration is exposed. Because all the input is randomized, instead of chronological order to improve the training of the model, the cases where a few of the returns for the bonds were higher than the stocks were in the last training sets and, therefore, have great influence on the prediction. This, however, does not break the confidence in the model but should be taken into account when constructing the model.

**Table 6** – This table shows the the average weight in the stock $x_s$ ($w_s$), the average terminal wealth ($\bar{TW}$), the standard deviation of terminal wealth ($\sigma(TW)$), the average Turnover ($\bar{TO}$), and the average realized utility ($\bar{RU}$) for the five models. Three models are the classical portfolio management methods $(CP)$ with different risk aversion $\gamma$, and the reinforcement methods denoted by $(RL)$ followed by the method of function approximation. In this table the results are shown of runs on **simulated** values.

| Method: | CER | | | | | VAR | | | | | BVAR | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\bar{w}_s$ | $\bar{TW}$ | $\sigma$(TW) | $\bar{TO}$ | $\bar{RU}$ | $\bar{w}_s$ | $\bar{TW}$ | $\sigma$(TW) | $\bar{TO}$ | $\bar{RU}$ | $\bar{w}_s$ | $\bar{TW}$ | $\sigma$(TW) | $\bar{TO}$ | $\bar{RU}$ |
| CP ($\gamma = 2$) | 1.00 | 1.40 | 0.02 | 59.72 | -0.71 | 0.43 | 1.07 | 0.21 | 65.49 | -0.62 | 0.45 | 0.92 | 0.21 | 70.93 | -1.15 |
| CP ($\gamma = 5$) | 1.00 | 1.40 | 0.02 | 59.75 | -0.06 | 0.38 | 1.06 | 0.18 | 65.82 | -0.27 | 0.30 | 0.94 | 0.18 | 69.32 | -0.47 |
| CP ($\gamma = 10$) | 1.00 | 1.40 | 0.02 | 59.71 | -0.01 | 0.28 | 1.03 | 0.15 | -0.25 | -0.25 | 0.21 | 0.95 | 0.15 | 68.48 | -0.66 |
| RL-NN | 0.95 | 1.38 | 0.03 | 58.09 | - | 0.19 | 1.07 | 0.19 | 64.50 | - | 0.54 | 0.91 | 0.24 | 70.24 | - |

Note: Realized Utility for the Reinforcement Learning methods is nonexisting due to the absence of gamma when considering a reward function as just the wealth increase per period.

## 4.2 Real data

Table 7 shows the results for the model-free benchmarks and the perfect foresight benchmark in case of the real data for benchmark purposes, like stated in Table 1 and also the results of the model-dependent methods. Graphically the benchmark models are shown in Figure 6, you can see that with a perfect timing, the Perfect Foresight line in the figure, you can significantly gain in this problem statement.

**Table 7** – This table shows the the average weight in the stock $x_s$ ($w_s$), the average terminal wealth ($T\bar{W}$), the standard deviation of terminal wealth ($\sigma(TW)$), the average Turnover ($T\bar{O}$), and the average realized utility ($R\bar{U}$) for the five modeled methods and the three model-free methods. Three models are the classical portfolio management methods $(CP)$ with different risk aversion $\gamma$, and the reinforcement methods denoted by $(RL)$ followed by the method of function approximation. In this table the results are shown of runs on **real** values.

| Method: | $\bar{w}_s$ | $T\bar{W}$ | | | |
|---|---|---|---|---|---|
| 1/N each asset | 1.21 | 0.14 | | | |
| Full r | 1.06 | 0.14 | | | |
| Full stock | 1.48 | 0.42 | | | |
| Full bond | 1.14 | 0.11 | | | |
| Perfect foresight | 3.77 | 0.83 | | | |
| | $\bar{w}_s$ | $T\bar{W}$ | $\sigma$(TW) | $T\bar{O}$ | $R\bar{U}$ |
| CP ($\gamma = 0$) | 0.61 | 1.32 | 0.35 | 65.89 | 1.32 |
| CP ($\gamma = 2$) | 0.64 | 1.34 | 0.35 | 65.13 | -0.79 |
| CP ($\gamma = 5$) | 0.61 | 1.31 | 0.33 | 66.16 | -0.15 |
| CP ($\gamma = 10$) | 0.48 | 1.26 | 0.25 | 69.98 | -0.06 |
| RL-NN | 0.57 | 1.33 | 0.25 | 63.37 | - |

From Table 7 you can conclude that the RL-NN model combines the best results of a risk-neutral and risk-averse investor. RL-NN is able to achieve a comparable average Terminal Wealth of 1.33 with the risk neutral investor and a low standard deviation of 0.25 compared to the high risk-averse investor. Together with these striking results, the turnover is slightly lower than all the CP models, this was not programmed in the model but a result of its conservative strategy, of rarely investing fully in the assets. Although the relatively good performance of the RL-NN model above the CP models its timing is far from optimal, compared to the average terminal wealth of the perfect foresight model of 3.77, but this is nearly impossible out-of-sample.

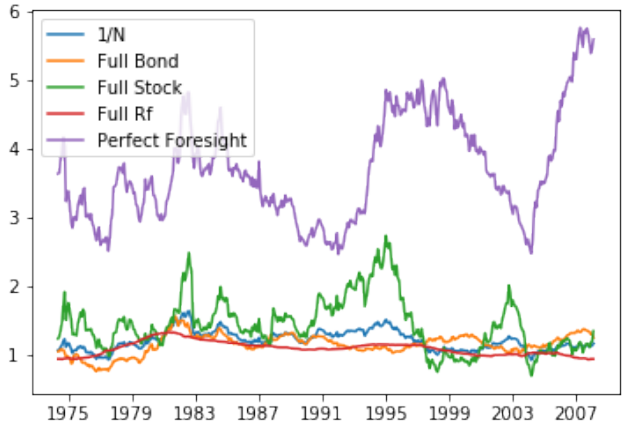



**Figure 6** – Time series of the terminal wealth for the four benchmark strategies and the maximum strategy=perfect foresight.

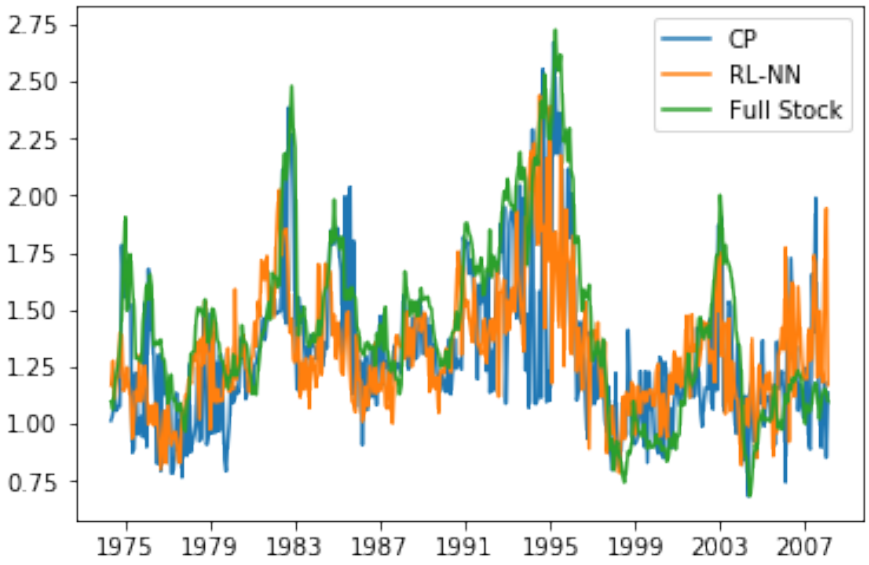
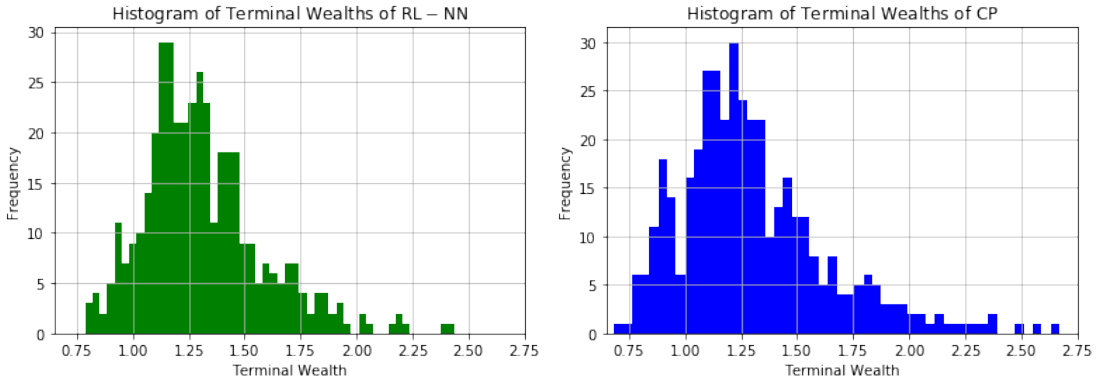



**Figure 7** – Time series plot of the terminal wealth of three methods (RL with NN, full investment in the stock, and classical portfolio management with $\gamma = 2$).

The time series of the terminal wealth for selected methods is shown in Figure 7. Two things can be concluded from this figure; one, the RL-NN method is less volatile compared to the CP method and two, only at the end of the dataset the methods are able to outperform the Full Stocks strategy by effectively timing the portfolio weights. This could suggest that both models need much data to train on or that both methods are simply lucky in this period, due to the low returns in the stocks.

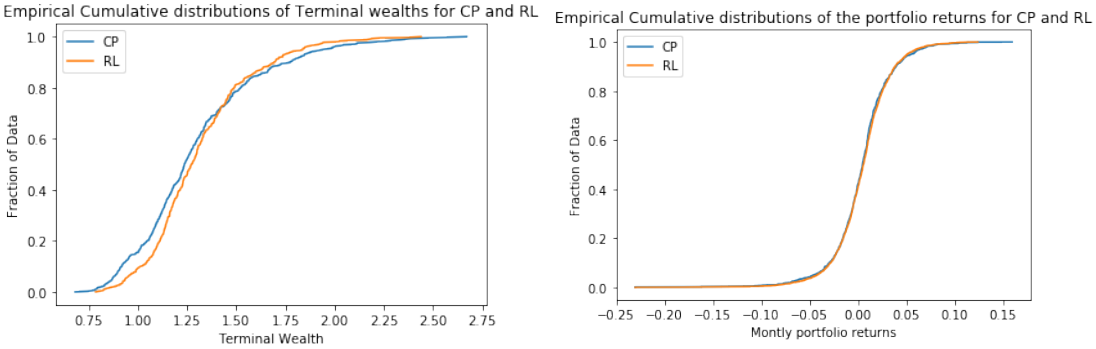



**(a)** Histogram of the terminal wealths of the RL method.

**(b)** Histogram of the terminal wealths of the CP method.

**Figure 8** – Histograms of terminal wealths of the long term portfolio techniques.

The Empirical Cumulative distribution is constructed for the CP and RL method from Figure 8 are graphically presented in Figure 9. You can not detect stochastic dominance, when a certain CDF is smaller than or equal to other CDF for all fractions, with the different models in subfigure 9a. Although it does not differ much, the loss of the CP model, with a higher density for the region with low terminal wealth, is compromised by several high terminal wealth values. When, for ,example, RL would have a smaller CDF as a whole and therefore stochastically dominate, it would always have an higher expected terminal wealth. But this is only the case for 72,4 % of the dataset for terminal wealth values under the threshold of 1.43. Therefore, RL second-order stochastically dominates for a threshold of Terminal wealth equal to 1.43. This remarkable result, however, cannot be detected by the portfolio returns in subfigure 9b.




**(a)** Empirical CDF for terminal wealths of the CP with $\gamma = 2$ and RL methods.

**(b)** Empirical CDF for portfolio returns of the CP with $\gamma = 2$ and RL methods.

**Figure 9** – Empirical Cumulative Distributions Functions of the terminal wealth values of the Long Term portfolio techniques.

When you further zoom in into the average weights which construct the terminal wealth values of Figure 7, you can clearly see the stated conservativity of the RL-NN method compared to the CP. The CP method regularly takes a full stance in one of the assets for the full period of 5 years, This is not something a practitioner likes and would do. Another observation is that the RL method grows in standard deviation of the average weights over time, this could be explained by the fact that the method has seen more from the past than in the beginning and is better to time the market, while it also is a period with a low volatility compared to the other periods, which is a more appropriate conclusion.

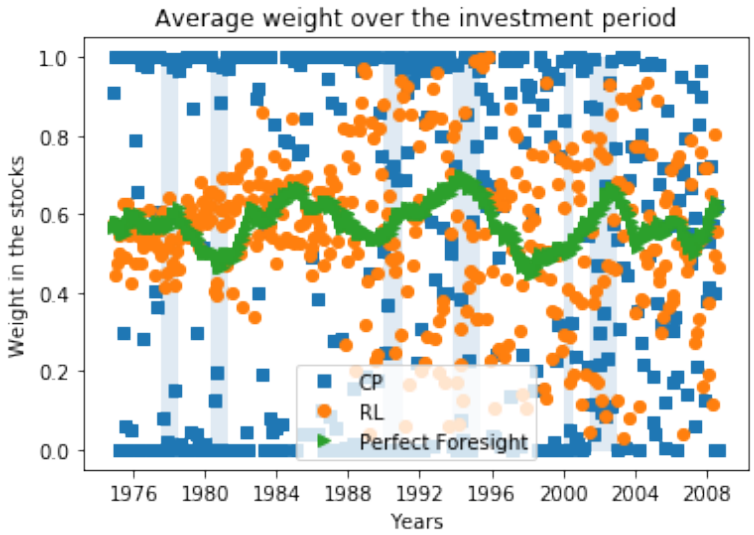



**Figure 10** – Average weight for the investment period of 60 months corresponding to the time series of Terminal Wealths shown in Figure 7, for the following methods: Perfect foresigth, CP with $\gamma = 2$, and RL-NN.

# 5 Conclusion and Discussion

Compared to the Classical portfolio management models the Reinforcement learning model shows great resemblance with a slightly risk-averse investor (with a $\gamma$ of 2) in terms of average terminal wealth while having a lower standard deviation and lower turnover.

The RL model rarely takes a position fully in the stocks or in the bonds, and therefore is able to lose less compared to the CP model when a drop in the stock prices happens. Also, the RL model is much faster in responding to certain events while the CP model stays a period with high weights in the stocks while the downfall of the stock index is started. However, when the RL takes a full stance in either sides, one could argue that the method knows completely sure that the market will go strongly up or down.

In the simulated environment of the CER, the RL model switches between all period the weight of 1 or all the periods the weight of 0.78. This is probably due to the 5 percent of the dataset in which the returns of the bonds are higher than the returns on the stock. I think there is one explanation for it in my opinion, the randomizing of input had such a large influence that the cases where the weights are all 0.78 one of the recent input was one of these special cases. This makes the method more fault proof due to the randomizing of the input, this could be adjusted by giving the model the input chronologically with an experience replay, this experience replay returns a random input from the past, this would improve on the accuracy of the model.

Neural networks should not be too complex, in this case study, otherwise it does not generalize anything, it will only replicate previously seen state action and reward combinations. In the context of Finance this is a bad practice, because it rarely happens that the past is a good predicament for the future.

More epochs improves the results, while there is a tipping point, when the network is overfitting the data. Ideally in the case of Finance one would like to have no epochs, because all returns and their historical returns are unique and non repetitive. This could be achieve for assets which have a higher frequency, for example, digital currencies or individual stocks traded through the stock exchange. It would be very informative to further investigate in assets with various frequencies and also more assets than the three stated in this research.

Mainly because of the time series of Figure 7, I strongly think there is not much more information in the data set at hand due to the state of the art Reinforcement learning methods performing almost equally well in comparison to the Classical Portfolio Management method and while being unable to beat the full stock model-free method. This should not come as a big surprise, because the data used is limited. However, this is a conscious choice because there would be an unlimited amount of other factors influencing the problem statement when comparing these methods. I would suggest other kinds of data than simply stock returns. One needs to search for data which is not already incorporated in the expected returns thanks to the Efficient Market Hypothesis. For example, sentiment data for the specific asset scraped from Twitter or sentiment analysis bureaus.

Machine learning methods work well in the case where much data is available and therefore also much information is available in the data. When the data is scarce it is not possible to get informative data. So implementing a naive machine learning method when only considering simple time series would fall short. Together with the extremely tedious job of the fine tuning of the many hyperparameters with the machine learning methods, it introduces new challenges. For the hyperparameters selection a genetic algorithm could be used with the inverse of the loss after training as fitness function.

Instead of using an expanding window which proved to be more useful than a moving window by Diris et al. [2015], it would also nice to look at the moving window cases for the reinforcement learning methods. This, however, has the disadvantage of lesser data which is really necessary for this kind of machine learning methods.

# References

B. Diris, F. Palm, and P. Schotman. Long-term strategic asset allocation: An out-of-sample evaluation. *Management Science*, 61(9):2185–2202, 2015. doi: http://dx.doi.org/10.1287/mnsc.2014.1924.

E.F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25, 1970. doi: 10.1111/j.1540-6261.1970.tb00518.x.

N. Gârleanu and L.H. Pedersen. Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, 67, 2013. doi: 10.1111/jofi.12080.

H. Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver. Learning values across many orders of magnitude. *Google Deepmind*, 2016.

T. Hens and P. Woehrmann. Strategic asset allocation and market timing: A reinforcement learning approach. *Computational Economics*, 29(3):369–381, 2007. doi: 10.1007/s10614-006-9064-0.

T. Jaakkola, M.I. Jordan, and S.P. Singh. Convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 1994.

Z. Jiang and J. Liang. Cryptocurrency portfolio management with deep reinforcement learning. 2017. URL `http://arxiv.org/abs/1612.01277v5`.

O. Jin and H. El-Saawy. Portfolio management using reinforcement learning. 2017.

F.S. Melo. Convergence of q-learning: a simple proof. 2001.

C. Metz. The rise of the artificially intelligent hedge fund. URL `https://www.wired.com/2016/01/the-rise-of-the-artificially-intelligent-hedge-fund/`. Accessed: 2017-21-10.

R.S. Sutton and A.G. Barto. *Introduction to Reinforcement learning*. MIT press, 1st edition, 1998.

# Appendix

First the exact hyperparameters for the models are stated and their respective descriptions. Also the functioning of the algorithms are further investigated. Also several non-main results are shown here to gain extra insight into the functioning of the models.

**Table 8** – Hyperparameters for the different models stated in this research paper.

| **Hyperparameters** | **Value** | **Description** |
|---|---|---|
| Classical Portfolio Methods | | |
| Periods ($K$) | 60 | The amount of months to take into account in the future. |
| Window | Expanding | The model should have a certain data points as history to estimate its Vector Autoregressive(VAR) Model on, due to the proven dominance over an expanding window over a moving window is chosen [Diris et al., 2015]. |
| Simulations | 400 | From the VAR model simulated paths of future returns are simulated in order to determine the path with the best utility. |
| Reinforcement learning Methods | | |
| Number of lags | 10 | As input for the function approximation the last 10 lags are given to estimate the next state action value one period ahead in time. |
| Number of assets | 3 | For each asset(riskfree, bond, and stock) the number of lags are provided for the function approximation. |
| Number of actions | 10 | The amount of actions the agent could take is 10, this is a grid of possible weights in the stocks from ranging from 0 to 1. |
| Structure Neural Network | (20,45,10) | The Neural Network used as function approximation is structured with one input layer, one hidden layer, and one output layer. The amount of nodes per layer are stated in the brackets. |
| Activation function | softmax | At the end of the neural network, in th output layer the values are transformed with a softmax activation function in order to convert the values of the network into probabilities for the best state. |
| Learning rate (Neural Network) | 0.1 | The learning rate for the Gradient Descent Optimizer which updates the neural network. |
| Learning rate (Q value) | 0.01 | The learning rate to update the new Q value to, to train the function approximation on. |
| Epochs | 20 | The amount of times the dataset is given (randomized) to the Reinforcement Learning method in order to train on the past, this is kept low because normally in Finance the same past returns and future returns are rarely the same. |
| Epsilon ($\varepsilon$) | 0.1 | A random real number i chosen between 0 and 1, if it is smaller than 0.1 the action to take by the reinforcement learning method is also randomized to improve exploration otherwise the maximum of the output of the function approximation is taken. |