Imperial College
London

Reinforcement Learning for Portfolio Management

MEng Dissertation: Interim Report

Angelos Filos

January 29, 2018

Advisors: Prof. Danilo Mandic, Mr. Bruno Scalzo Dees

Department of Electrical and Electronic Engineering, Imperial College London

Contents

I Project Specification **1**

1 Introduction **2**
1.1 Problem Definition . . . 3
1.2 Applications . . . 3

2 Related Work **4**
2.1 Traditional Portfolio Management . . . 4
2.2 Reinforcement Learning and Algorithmic Trading . . . 4
2.3 Reinforcement Learning for Asset Allocation . . . 5
2.4 Our Contribution . . . 5

II Background **6**

3 Financial Signal Processing **7**
3.1 Terms & Concepts . . . 7
3.1.1 Asset . . . 7
3.1.2 Portfolio . . . 7
3.1.3 Short Sales . . . 8
3.2 Prices . . . 9
3.3 Returns . . . 9
3.3.1 Simple Return . . . 9
3.3.2 Log Return . . . 11

4 Portfolio Optimization **13**
4.1 Metrics . . . 14
4.1.1 Portfolio Mean & Variance . . . 14
4.1.2 Sharpe Ratio . . . 19
4.2 Markowitz Model . . . 19
4.2.1 Mean-Variance Optimization . . . 19

4.2.2 Quadratic Programming . . . 21
4.3 Transaction Costs . . . 21
4.3.1 Multi-Stage Decision Problem . . . 21

5 Reinforcement Learning 22
5.1 Basic Concepts . . . 23
5.1.1 Agent & Environment . . . 23
5.1.2 Action . . . 23
5.1.3 Reward . . . 23
5.1.4 State . . . 24
5.2 Major Components of Reinforcement Learning . . . 25
5.2.1 Return . . . 25
5.2.2 Policy . . . 25
5.2.3 Value Function . . . 25
5.2.4 Model . . . 26
5.3 Markov Decision Process . . . 26
5.3.1 Markov Property . . . 26
5.3.2 Definition . . . 26
5.3.3 Optimality . . . 27
5.3.4 Bellman Equation . . . 27
5.3.5 Extensions . . . 28
5.4 Reinforcement Learning Algorithms . . . 29
5.4.1 Q-Learning . . . 30

III Research 32

6 Portfolio Management as IPOMDP 33
6.1 Assumptions . . . 33
6.1.1 Zero Slippage . . . 33
6.1.2 Zero Market Impact . . . 34
6.2 State Space & Observation Space . . . 34
6.3 Action Space . . . 34
6.4 Reward Signal . . . 35

7 Algorithmic Portfolio Manager 36
7.1 DQRN: Deep Q-Recurrent Network . . . 36
7.1.1 Architecture . . . 36
7.1.2 Affine Layer . . . 37
7.1.3 Recurrent Gated Rectified Unit Layer . . . 37
7.1.4 Pseudo-Softmax Layer . . . 38
7.1.5 Gaussian Policy . . . 38
7.2 Simulations & Results . . . 39
7.2.1 Synthetic Market . . . 39

7.2.2 NASDAQ 40

IV Future Work **42**

8 Market Simulation **43**
8.1 Problem Definition 43
8.2 Implementation Plan 43

9 Actor Critic Agent **44**
9.1 Implementation Plan 44
9.2 Evaluation Plan 44

Bibliography **45**

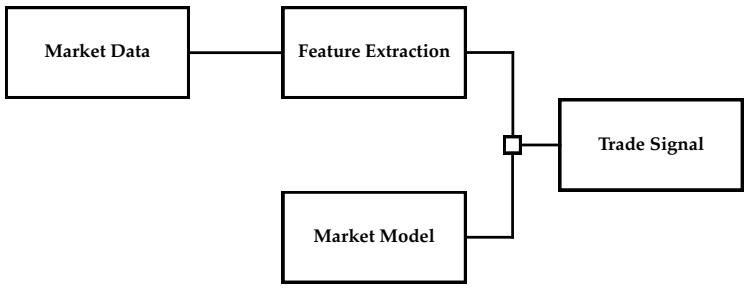
Part I

Project Specification

Chapter 1

Introduction

Reinforcement Learning has contributed to the automation of various tasks (Silver, 2015a), including electronic algorithmic trading. The main benefit of Reinforcement Learning in financial applications is the relaxation of assumptions on market models and hand-picked features, allowing a data-driven, automated process (I. Goodfellow et al., 2016). In figure 1.1 the advantage of Reinforcement Learning agents compared to traditional feature extractors and human-tailored models is illustrated.



(a) Traditional Algorithmic Trading

(b) Reinforcement Learning Trading Agent

Figure 1.1: Trading pipelines for a traditional algorithmic trading system and a Reinforcement Learning agent based system

1.1 Problem Definition

The effectiveness of Reinforcement Learning agents on asset allocation[1] is investigated. A finite universe of financial instruments, assets, such as stocks, is selected and the trained agent is constructing an internal representation (model) of the market, allowing it to determine how to optimally allocate funds of a finite budget to those assets. The agent is trained on both synthetic market data and on actual market prices. Then its performance is compared with standard portfolio management algorithms on an out-of-sample dataset; data that the agent has not been trained on (test set).

Agents have been developed to address similar problems in other areas, such as Beamforming (Almeida et al., 2015) in Adaptive Signal Processing and Wireless Communications or Medication Dosing (Nemati et al., 2016) in Medicine.

1.2 Applications

A successful agent can be used as a consulting software for portfolio manager or it can be used for low-frequency algorithmic trading. Moreover, it can allow us to identify the missing pieces of the existing models and suggest directions to improve them.

[1]The terms *Asset Allocation* and *Portfolio Management* are used interchangeably throughout the report.

Chapter 2

Related Work

Portfolio Management has been approached by various angles, such as Econometrics (Luenberger, 1998), Financial Signal Processing (Zhang and Wang, 2017), Supervised Machine Learning (I. Goodfellow et al., 2016) and Reinforcement Learning (Neuneier, 1997). A review of the most relevant and influential work on the field is provided in the following sections.

2.1 Traditional Portfolio Management

Portfolio Management was mathematically formalized by Markowitz (1952) and is known as **Modern Portfolio Theory** (**MPT**). MPT and Markowitz model relies on quadratic programming solutions and make assumptions about the distribution of expected returns and volatility of the assets, often using linear estimators (Kennedy, 2016). Then the **Capital Asset Pricing Model** (**CAPM**) was introduced by Markowitz (1961) and Sharpe (1964), independently, which extends the scope of Markowitz Model. Various nonlinear regression models (Wilmott, 2007; Luenberger, 1998), inspired by Signal Processing (Danilo P. Mandic and Chambers, 2001) and Machine Learning (Graves, 2012) have been used since then in order to solve the asset allocation problem.

2.2 Reinforcement Learning and Algorithmic Trading

In the late 1990's Reinforcement Learning started being used in finance for algorithmic trading (John Moody and Matthew Saffell, 1997; Neuneier, 1997). Because of the processing power available at the time and the availability of data, neural network architectures were shallow and highly unstable. As a result, the scope of the reinforcement agents was very narrow, addressing only very small scale problems, such as binary traders[1] (John Moody and

[1]One risky asset and one risk-free asset (i.e T-Bill bonds.

Matthew Saffell, 1997). Reinforcement Learning was preferred over Supervised Learning (Mitchell, 1997) mainly because of the multi-stage decision making nature of the asset allocation problem when transaction costs are taken into account (Necchi, 2016).

2.3 Reinforcement Learning for Asset Allocation

After the successful implementation of a Deep Q-Network (**DQN**) by Google DeepMind (Mnih, Kavukcuoglu, et al., 2013) there has been an explosive adoption of Reinforcement Learning by many fields, including the financial sector. Jiang et al. (2017) have successfully developed a deep reinforcement agent, which uses 1D convolution layers (I. Goodfellow et al., 2016) to extract feature maps from the market time series.

2.4 Our Contribution

Despite the numerous attempts that have been made to combine Reinforcement Learning with Portfolio Management, the main unaddressed points are

- Deep Recurrent Neural Network Architectures
- Actor Critic Agents for Continuous Action Spaces
- Market Simulations

Part II

Background

Chapter 3

Financial Signal Processing

Time Series is a sequence of observations indexed by time order. Signal Processing (**SP**) focuses[1] on the analysis and manipulation of signals, usually time-indexed, providing a rich toolbox for systematic time series analysis, modelling and forecasting (Danilo P. Mandic and Chambers, 2001).

Financial applications usually involve stochastic modelling of dynamic systems, combining principles and methods developed in the context of Signal Processing and Systems Theory (Feng and Palomar, 2016). In this chapter, an attempt to bridge the gap between Finance and Signal Processing is made, focusing on the similarities and connections between concepts in the two areas as well as framing financial terms in an SP-friendly way.

3.1 Terms & Concepts

3.1.1 Asset

Asset is an item of economic value. Examples of assets are cash (in hand or in a bank), stocks, loans and advances, accrued incomes etc. Our main focus on this report is on cash and stocks, but general principles apply to all kinds of assets.

3.1.2 Portfolio

Portfolio is a collection of multiple financial assets is called. A portfolio is characterized by its:

- **constituents**: M assets of which it consists

[1] this is **not** an exhaustive list of the scope of SP

- **portfolio vector**, $\boldsymbol{w} \in [0,1]^M$: its i^{th} component represents the ratio of the total budget invested to the i^{th} asset, such that:

$$\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_M \end{bmatrix}^T \quad \text{and} \quad \sum_{i=1}^{M} w_i = 1 \tag{3.1}$$

For a fixed portfolio vector $\boldsymbol{w}$ and fixed constituents, we can treat a portfolio as a single complex[2] asset. Therefore, the analysis of single simple assets can be applied to portfolios when the portfolio vector and the corresponding constituents are determined.

Portfolios are more powerful, general representation of financial assets since the single asset case can be represented by a portfolio, whose portfolio vector is a one-hot encoded vector with the j^{th} term equal to one and the rest zero.

Portfolios are also preferred over single assets in order to minimize risk, a proof is provided in subsection 4.1.1.

3.1.3 Short Sales

Sometimes is it possible to sell an asset that we do not own. This process is called **short selling** or **shorting** (Luenberger, 1998). The exact shorting mechanism varies between markets, but it can be generally summarized as

1. **borrowing** an asset from someone who owns it at time t
2. **selling** it immediately to someone else at price p_t
3. **buying back** the asset at time $t+k$, where $k > 0$, at price p_{t+k}
4. **returning** the asset to the lender

Therefore, the **overall gross return** is $p_t - p_{t+k}$ [3] and as a result short selling is profitable only if the asset price declines between time t and $t+k$ or $p_{t+k} < p_t$. Nonetheless, we note that the potential loss of short selling is unbounded, since asset prices are not bounded from above. If short selling is allowed then the portfolio vector satisfies

$$\boldsymbol{w} \in [-1,1]^M \quad \text{and} \quad \sum_{i=1}^{M} |w_i| = 1 \tag{3.2}$$

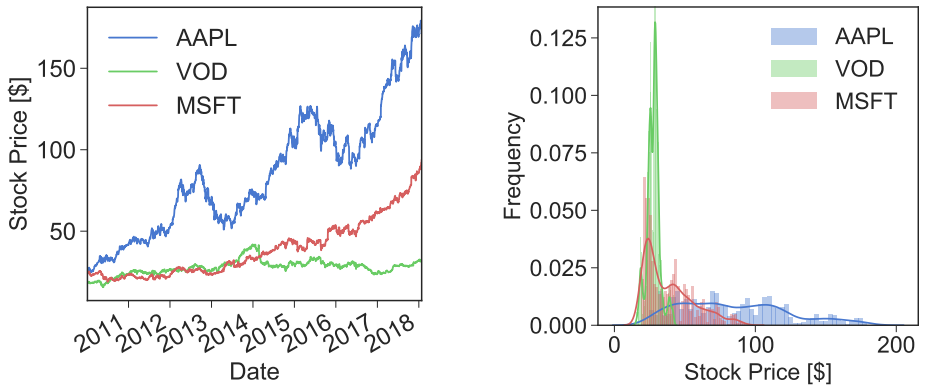
Usually the terms **long** and **short** position to an asset are used to refer to investments where we hold or short sell the asset, respectively.

[2] *complex* is used here to reflect the fact that many assets contribute to the portfolio value, it does not refer the complex numbers $\mathbb{C}$.

[3] assume that only one unit is shorted

3.2 Prices

Let p_t be the **price** or value of an asset at discrete time index t (Feng and Palomar, 2016), then the sequence $p_1, p_2, \ldots, p_t$ is a time series. Examples of asset prices time series are provided in figure 3.1a.



(a) Time Series (b) Histogram

Figure 3.1: Single Assets Prices

3.3 Returns

Because of the uncertainty of the financial markets (Kennedy, 2016) future asset prices are often treated as random variables. As a result, useful information can be obtained by studying the statistical properties of them. The histograms in figure 3.1b of the example asset prices suggest that the prices of the different assets are unnormalized and therefore difficult to compare. This encourages measuring all variables in a comparable metric, thus enabling the evaluation of analytic relationships amongst variables despite originating from asset price series of unequal values. Various **returns** metrics are presented below that overcome this problem (Zhang and Wang, 2017).

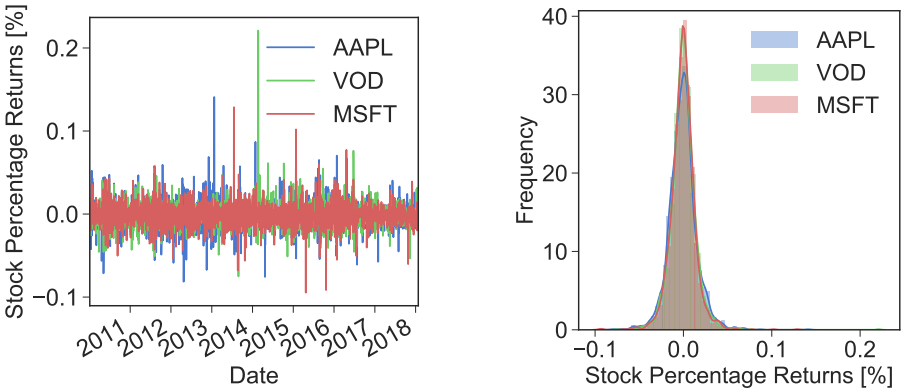
3.3.1 Simple Return

Single Asset

Simple return R_t of an asset with price p_t at time index t is

$$R_t \triangleq \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1 \tag{3.3}$$

We note that the return of an asset is also a time series, representing the percentage change of the asset's price over one period. The simple returns of the example assets are illustrated in figure 3.2a, which confirms our selection of simple returns as a metric for comparing assets, since the resulting distributions at figure 3.2b can be visually approximated by normal distributions.



(a) Time Series (b) Histogram

Figure 3.2: Single Assets Simple Returns

Additionally, **gross return** of an asset at time t usually refers to the term $1 + R_t$ representing the scaling factor of an investment at time $t - 1$ at the asset (Feng and Palomar, 2016). For example, a B dollars investment at an asset at time $t - 1$ will worth $B(1 + R_t)$ dollars at time t.

Holding the asset for k periods between time indexes $t - k$ and t gives a ***k*-period simple return**

$$\begin{aligned} R_t[k] \triangleq \frac{p_t - p_{t-k}}{p_{t-k}} &= \frac{p_t}{p_{t-k}} - 1 \\ &= -1 + \frac{p_t}{p_{t-1}} \frac{p_{t-1}}{p_{t-2}} \cdots \frac{p_{t-k+1}}{p_{t-k}} \\ &\overset{(3.3)}{=} -1 + (1 + R_t)(1 + R_{t-1}) \cdots (1 + R_{t-k+1}) \\ &= -1 + \prod_{i=0}^{k-1} (1 + R_{t-i}) \end{aligned} \tag{3.4}$$

Therefore the k-period simple return involves the product of the one-period gross returns.

Portfolio

Let the **returns vector** $\boldsymbol{R} = [R_t^{(1)} \cdots R_t^{(M)}]^T \in \mathbb{R}^M$, where $R_t^{(i)}$ the simple return of the i^{th} asset at time index t, then the **portfolio simple return** is defined as the weighted sum of the returns of each constituents

$$\begin{aligned} R_t^{(p)} \triangleq \frac{p_t^{(p)} - p_{t-1}^{(p)}}{p_{t-1}^{(p)}} &= \frac{p_t^{(p)}}{p_{t-1}^{(p)}} - 1 \\ &= w_1 R_t^{(1)} + w_2 R_t^{(2)} + \cdots + w_M R_t^{(M)} \\ &= \sum_{i=1}^{M} w_i R_t^{(i)} = \boldsymbol{w}^T \boldsymbol{R}_t \end{aligned} \tag{3.5}$$

Example: Portfolio Time Series

Let a portfolio with M assets, whose prices for T periods are given in a matrix $\boldsymbol{P} \in \mathbb{R}^{T \times M}$ and the portfolio vector $\boldsymbol{m} \in \mathbb{R}^M$, then the simple returns matrix $\boldsymbol{R} \in \mathbb{R}^{(T-1) \times M}$ is calculated by application of equation (3.3) column-wise, such that $R_t^{(i)} = \frac{R_t^{(i)}}{R_{t-1}^{(i)}} - 1$

$$\boldsymbol{P} = \begin{bmatrix} p_1^{(1)} & p_1^{(2)} & \cdots & p_1^{(M)} \\ p_2^{(1)} & p_2^{(2)} & \cdots & p_2^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ p_T^{(1)} & p_T^{(2)} & \cdots & p_T^{(M)} \end{bmatrix} \xrightarrow{(3.3)} \boldsymbol{R} = \begin{bmatrix} R_1^{(1)} & R_1^{(2)} & \cdots & R_1^{(M)} \\ R_2^{(1)} & R_2^{(2)} & \cdots & R_2^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ R_{T-1}^{(1)} & R_{T-1}^{(2)} & \cdots & R_{T-1}^{(M)} \end{bmatrix} \tag{3.6}$$

Then the **portfolio returns time series** $\boldsymbol{R}^{(p)} \in \mathbb{R}^{T-1}$, is the dot product of $\boldsymbol{R}$ and $\boldsymbol{m}$

$$\boldsymbol{R}^{(p)} = \begin{bmatrix} R_1^{(p)} \\ R_2^{(p)} \\ \vdots \\ R_{T-1}^{(p)} \end{bmatrix} = \boldsymbol{R}\boldsymbol{w} = \begin{bmatrix} R_1^{(1)} & R_1^{(2)} & \cdots & R_1^{(M)} \\ R_2^{(1)} & R_2^{(2)} & \cdots & R_2^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ R_{T-1}^{(1)} & R_{T-1}^{(2)} & \cdots & R_{T-1}^{(M)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \tag{3.7}$$

3.3.2 Log Return

As mentioned in section 3.3, simple returns were introduced as a comparable metric between asset price changes. Nonetheless, table 3.1 suggests that simple returns are asymmetric and therefore practitioners use log returns instead (Kennedy, 2016), which eliminate this bias.

time t	simple return	price (\$)	log return
0	-	100	-
1	+0.15	110	+0.13
2	-0.15	99	-0.16
3	+0.01	100	+0.01
4	-0.14	86	-0.15
5	+0.16	100	+0.15

Table 3.1: Simple Return Asymmetry & Log Return Symmetry

Single Asset

Let the **log return** r_t at time t be

$$r_t \triangleq ln(p_t) - ln(p_{t-1}) = ln(\frac{p_t}{p_{t-1}}) \overset{(3.3)}{=} ln(1 + R_t) \tag{3.8}$$

Considering the **multi-period log returns** by taking the natural logarithm of simple gross return $1 + R_t[k]$

$$\begin{aligned} r_t[k] \triangleq ln(1 + R_t[k]) &\overset{(3.4)}{=} ln \prod_{i=0}^{k-1}(1 + R_{t-i}) \\ &= \sum_{i=0}^{k-1} ln(1 + R_{t-i}) \overset{(3.8)}{=} \sum_{i=0}^{k-1} r_{t-i} \end{aligned} \tag{3.9}$$

Thus, the multi-period log (gross) return of an asset is simply the sum of one period log returns.

Portfolio

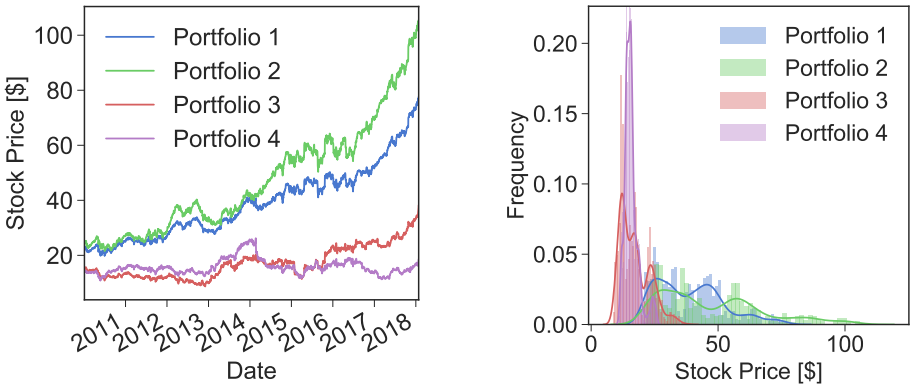
The **portfolio log return** $r_t^{(p)}$ is defined as

$$r_t^{(p)} \triangleq ln(p_t^{(p)}) - ln(p_{t-1}^{(p)}) = ln(\frac{p_t^{(p)}}{p_{t-1}^{(p)}}) \overset{(3.5)}{=} ln(1 + \boldsymbol{w}^T \boldsymbol{R}_t) \tag{3.10}$$

Chapter 4

Portfolio Optimization

In subsection 3.1.2 the trivial portfolio case was introduced where all the budget is allocated to a single asset. In any other case, the budget is spread across multiple assets and there infinitely many valid allocations, satisfying portfolio vector definitions (3.1) and (3.2). Examples of randomly allocated portfolios are illustrated in figure 4.1. We note that different linear combinations, portfolio vectors, lead to completely different portfolios with varying value. **Portfolio Optimization** aims to target the allocation problem in a systematic way, where an objective function reflecting the investor's preferences is constructed and optimized with respect to the portfolio vector.



(a) Time Series (b) Histogram

Figure 4.1: Prices of Randomly Allocated Portfolios

Various metrics are introduced in section 4.1, which are then combined in order to construct objective and utility functions that are then used to build the Markowitz Model, in section 4.2.

w	AAPL	VOD	MSFT
Portfolio 1	0.097	0.388	0.515
Portfolio 2	0.205	0.071	0.724
Portfolio 3	-0.152	0.247	0.601
Portfolio 4	-0.120	0.685	-0.435

Table 4.1: Random Portfolio Vectors of figure 4.1

4.1 Metrics

In section 3.3 we mentioned the random, stochastic nature of financial markets, encouraging the use of probabilistic models in order to capture their behaviour and statistical properties so as to analyze them. We will focus on the statistical properties of returns, simple and log, because of their more favorable statistical properties they have compared to raw prices, as figures 3.1b, 3.2b and section 3.3 suggest.

Initially, we will assume that future return at time index $t+1$ is a non-linear autoregressive process (Danilo P. Mandic, 2018a), which means that future values are conditioned, dependent on the present and past values of the time-series, such that

$$R_{t+1} = \mathbf{\Phi}(R_t, R_{t-1}, \ldots, R_0) + g(\text{noise}) \tag{4.1}$$

where $\mathbf{\Phi} : \mathbb{R}^t \to \mathbb{R}$ is the non-linear kernel function applied to present and past returns (deterministic part) and $g : \mathbb{R}^k \to \mathbb{R}$ is the noise transformation (stochastic part). Although this is not strictly true, if we relax the conditions on the nature of $\mathbf{\Phi}$ and g then the resulting family of models has general validity (Tsay, 2003; Danilo P. Mandic and Chambers, 2001).

4.1.1 Portfolio Mean & Variance

Expected Return

The **expected return** of an asset at $t+1$ time index is then given by

$$\mathbb{E}[R_{t+1}|R_t, R_{t-1}, \cdots, R_0] \overset{\text{in short}}{\longrightarrow} \mathbb{E}[R_{t+1}] = \mu_{t+1} \tag{4.2}$$

We will be using $\mathbb{E}[R_{t+1}]$ and μ_{t+1} notation interchangeably to refer to the expected return, without mentioning the conditioning on the past and present values, but we will assume that those values are given.

Because of the linearity property that the expected value operator ($\mathbb{E}$) satisfies, the expected return of a portfolio is the linear combination of the expected returns of its constituents, such that

$$\begin{aligned}\mathbb{E}[R_{t+1}^{(p)}] = \mu_{t+1}^{(p)} &\overset{(3.5)}{=} \mathbb{E}[w_1 R_{t+1}^{(1)} + w_2 R_{t+1}^{(2)} + \cdots + w_M R_{t+1}^{(M)}] \\ &= w_1 \mathbb{E}[R_{t+1}^{(1)}] + w_2 \mathbb{E}[R_{t+1}^{(2)}] + \cdots + w_M \mathbb{E}[R_{t+1}^{(M)}] \\ &\overset{(4.2)}{=} w_1 \mu_{t+1}^{(1)} + w_2 \mu_{t+1}^{(2)} + \cdots + w_M \mu_{t+1}^{(M)}] \\ &= \boldsymbol{w}^T \left[\mu_{t+1}^{(1)} \ \mu_{t+1}^{(2)}\right] \ \ldots \ \mu_{t+1}^{(M)}\right]^T = \boldsymbol{w}^T \boldsymbol{\mu_{t+1}} \end{aligned} \tag{4.3}$$

where $\boldsymbol{\mu}_{t+1} \in \mathbb{R}^M$ the **expected-returns vector** or **mean vector**.

Investor Advice: Maximum Returns (Greedy Criterion) (Wilmott, 2007)

> For the same level of risk, choose the portfolio that maximizes the expected returns.

All the expected values are calculated using the sample mean (Danilo P. Mandic, 2018b)

$$\mathbb{E}[R_t] = \frac{1}{T} \sum_{\tau=1}^{T} R_\tau \tag{4.4}$$

Volatility

The risk of an asset is represented by the variance $\sigma_{t+1}^2 = \text{Var}[R_{t+1}]$ or the square root of variance σ_{t+1}, also called **volatility** in finance.

The **portfolio variance** involves covariance terms between the assets such that

$$\begin{aligned}\text{Var}[R_{t+1}^{(p)}] = (\sigma_{t+1}^{(p)})^2 &\overset{(3.5)}{=} \text{Var}[w_1 R_{t+1}^{(1)} + w_2 R_{t+1}^{(2)} + \cdots + w_M R_{t+1}^{(M)}] \\ &= w_1^2 \text{Var}[R_{t+1}^{(1)}] + w_2^2 \text{Var}[R_{t+1}^{(2)}] + \cdots + w_M^2 \text{Var}[R_{t+1}^{(M)}] \\ &\quad + 2w_1 w_2 \text{Cov}[R_{t+1}^{(1)}, R_{t+1}^{(2)}] + 2w_1 w_3 \text{Cov}[R_{t+1}^{(1)}, R_{t+1}^{(3)}] \\ &\quad + \cdots \\ &\quad + 2w_M w_{M-2} \text{Cov}[R_{t+1}^{(M)}, R_{t+1}^{(M-2)}] \\ &\quad + 2w_M w_{M-1} \text{Cov}[R_{t+1}^{(M)}, R_{t+1}^{(M-1)}] \\ &= \sum_{i=1}^{M} \sum_{j=1}^{M} w_i \boldsymbol{\Sigma}_{t+1} w_j = \boldsymbol{w}^T \boldsymbol{\Sigma}_{t+1} \boldsymbol{w} \end{aligned} \tag{4.5}$$

where $\mathbf{\Sigma}_{t+1} \in \mathbb{R}^{M \times M}$ is the **covariance matrix** of the returns at time t for the returns of the M assets.

Investor Advice: Minimum Volatility (Risk-Aversion Criterion) (Wilmott, 2007)

> For the same expected returns, choose the portfolio that minimizes the volatility.

Variances are calculated using the Bessel's correction formula (Tsay, 2003)

$$\text{Var}[R_t] = \frac{1}{T-1} \sum_{\tau=1}^{T} (R_\tau - \bar{R})^2 \tag{4.6}$$

Covariances are also calculated, using the unbiased estimator (Wilmott, 2007)

$$\text{Cov}[R_t^{(i)}, R_t^{(j)}] = \frac{1}{T-1} \sum_{\tau=1}^{T} (R_\tau^{(i)} - \bar{R}^{(i)})(R_\tau^{(j)} - \bar{R}^{(j)}) \tag{4.7}$$

Example: Single Asset vs. Portfolio

Since portfolios are equivalent to complex assets, what makes them superior to simple single assets? Portfolios minimize risk by allowing diversification (Luenberger, 1998).

Let's consider a market of M stocks (assets), which satisfy

- same expected returns: $\mathbb{E}[R_{t+1}^{(i)}] = \mu,\ i \in \{1, 2, \ldots, M\}$
- same risk (variance, volatility): $\text{Var}[r_{t+1}^{(i)}] = \sigma^2,\ i \in \{1, 2, \ldots, M\}$
- mutually uncorrelated: $\text{Cov}[r_{t+1}^{(i)}, r_{t+1}^{(j)}] = 0,\ \forall i \neq j$

The investors are then presented with two options:

1. **All in One**
   Invest all their budget on a single asset, without loss of generality, we choose asset $i = 1$ then
   - Portfolio Vector: $\boldsymbol{w} = [1\ 0 \cdots\ 0]^T \in \mathbb{R}^M$
   - Expected Portfolio Return:

$$\mathbb{E}[R_{t+1}^{(p)}] = \mu_{t+1}^{(p)} \overset{(4.3)}{=} \boldsymbol{w}^T \boldsymbol{\mu_{t+1}} = [1\ 0 \cdots\ 0][\mu\ \mu \cdots\ \mu]^T = \mu$$

- Portfolio Variance

$$\begin{aligned}\mathrm{Var}[R_{t+1}^{(p)}] = (\sigma_{t+1}^{(p)})^2 &\overset{(4.5)}{=} \boldsymbol{w}^T \boldsymbol{\Sigma}_{t+1} \boldsymbol{w} \\ &= [1\ 0 \cdots\ 0] \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sigma^2\end{aligned}$$

2. **Uniform Portfolio**
   Split their budget uniformly to the M assets:

   - Portfolio Vector: $\boldsymbol{w} = [\frac{1}{M}\ \frac{1}{M} \cdots\ \frac{1}{M}]^T \in \mathbb{R}^M$

   - Expected Portfolio Return:

$$\begin{aligned}\mathbb{E}[R_{t+1}^{(p)}] = \mu t + 1^{(p)} &\overset{(4.3)}{=} \boldsymbol{w}^T \boldsymbol{\mu_{t+1}} \\ &= [\frac{1}{M}\ \frac{1}{M} \cdots\ \frac{1}{M}][\mu\ \mu \cdots\ \mu]^T = \frac{1}{M} M \mu = \mu\end{aligned}$$

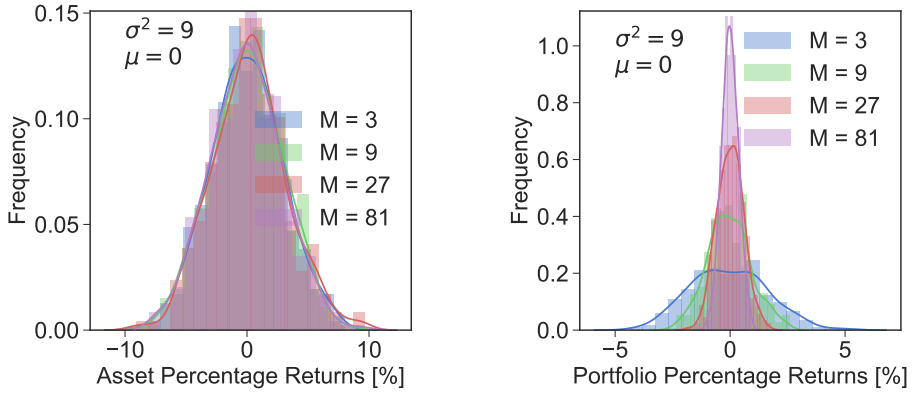
   - Portfolio Variance

$$\begin{aligned}\mathrm{Var}[R_{t+1}^{(p)}] = (\sigma_{t+1}^{(p)})^2 &\overset{(4.5)}{=} \boldsymbol{w}^T \boldsymbol{\Sigma}_{t+1} \boldsymbol{w} \\ &= [\frac{1}{M}\ \frac{1}{M} \cdots\ \frac{1}{M}] \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} \begin{bmatrix} \frac{1}{M} \\ \frac{1}{M} \\ \vdots \\ \frac{1}{M} \end{bmatrix} \\ &= \frac{1}{M^2} M \sigma^2 = \frac{\sigma^2}{M} \qquad (4.8)\end{aligned}$$

Both options have the same expected return $\mu_{t+1}^{(p)} = \mu$, but the variance of the second option is compressed by a factor of M. According to the investor advice "Minimize Volatility" in section 4.1.1, the second option "Uniform Portfolio" is selected by the investors. This is an example where a portfolio with the same expected return with a single asset is less risky, less volatile than the asset.

Examples of such a simulated market are provided at figure 4.2, where returns are modeled as normally distributed or $R_{t+1}^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$. We note that

in the "All in One" strategy the volatility of the investment is independent of the number of assets in the market M, while in the "Uniform Portfolio" case, the risk of the investment is inversely proportional to $\sqrt{M}$ [1].



(a) All in One Strategy (b) Uniform Portfolio Strategy

Figure 4.2: Histograms of uncorrelated market for single asset and uniform portfolio as a function of the total number of assets M.

Example: Fully Correlated Assets

In the previous example, it was assumed that the assets were mutually uncorrelated. Now we examine the case when the assets are fully correlated, then the non-diagonal elements of the covariance matrix $\mathbf{\Sigma}$ are non-zero and equal to the variance σ^2. The expected portfolio return (first-order moment) is not affected by the covariances of the assets, but the variance (second-order moment) is affected and the equation (4.8) is modified, such that

$$\begin{aligned}
\mathrm{Var}[R_{t+1}^{(p)}] = (\sigma_{t+1}^{(p)})^2 &\overset{(4.5)}{=} \boldsymbol{w}^T \mathbf{\Sigma}_{t+1} \boldsymbol{w} \\
&= \begin{bmatrix} \frac{1}{M} & \frac{1}{M} & \cdots & \frac{1}{M} \end{bmatrix} \begin{bmatrix} \sigma^2 & \sigma^2 & \cdots & \sigma^2 \\ \sigma^2 & \sigma^2 & \cdots & \sigma^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^2 & \sigma^2 & \cdots & \sigma^2 \end{bmatrix} \begin{bmatrix} \frac{1}{M} \\ \frac{1}{M} \\ \vdots \\ \frac{1}{M} \end{bmatrix} \\
&= \frac{1}{M^2} M M \sigma^2 = \sigma^2
\end{aligned} \tag{4.9}$$

Comparing equations (1), (4.8) and (4.9) we reach the conclusion that in

[1] Note that the variance $(\sigma^{(p)})^2 \propto \frac{1}{M}$, while the volatility $\sigma^{(p)} \propto \frac{1}{\sqrt{M}}$.

case of fully correlated assets any portfolio is equivalent to the trivial single asset portfolio, therefore the risk cannot be minimized by distribution of the budget over multiple assets (Luenberger, 1998).

4.1.2 Sharpe Ratio

A popular metric that adjusts expected returns with risk is the **Sharpe ratio** (Sharpe, 1970), defined as

$$\mathbf{SR} \triangleq \frac{\mathbb{E}[R]}{\sqrt{\mathrm{Var}[R]}} \overset{(4.3,\,4.5)}{=} \frac{\boldsymbol{w}^T\boldsymbol{\mu}}{\sqrt{\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}}} \tag{4.10}$$

Sharpe ratio is usually defined over a specified period (monthly, quarterly, annually) so that the sample estimates of the mean and the standard-deviation are reliable. The Sharpe ratio can be considered as the **Signal-to-Noise Ratio** (SNR) equivalent for finance (Zhang and Wang, 2017; Feng and Palomar, 2016).

4.2 Markowitz Model

Overall the two criteria that investors satisfy are:

1. **Risk-Aversion Criterion**: Given 2 portfolios with the same mean return, a risk-averse investor will prefer the one with the smaller risk (variance)
2. **Greedy Criterion**: Given 2 portfolios with the same risk (variance), a greedy investor will prefer the one with the higher mean return

Markowitz model (Kroll et al., 1984; Markowitz, 1952) mathematically formulates the portfolio allocation problem [2], under the assumption that the two criteria above are satisfied, constrained on the definitions (3.1), (3.2).

4.2.1 Mean-Variance Optimization

Assume there are M assets with

- expected returns $\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(M)}$
- covariances $\sigma^{(i,j)}, \forall i, j \in 1, 2, \ldots, M$

For given target expected return $\bar{\mu}_{target}$, determine the portfolio vector $\boldsymbol{w} \in \mathbb{R}^M$ such that

[2] finding portfolio vector $\boldsymbol{w}$ in a universe of M assets

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & w^T \Sigma w && (4.11) \\ \text{subject to} \quad & w^T \mu = \bar{\mu}_{target} && (4.12) \\ \text{and} \quad & \|w\|_1 = 1 && (4.13) \end{aligned}$$

For λ, κ the Lagrangian multipliers, we form the Lagrangian function $\mathcal{L}$ (Papadimitriou and Steiglitz, 1982) such that

$$\mathcal{L}(w, \lambda, \kappa) = \frac{1}{2} w^T \Sigma w - \lambda (w^T \mu - \bar{\mu}_{target}) - \kappa (|w\|_1 - 1) \tag{4.14}$$

We differentiating the Lagrangian function $\mathcal{L}$ [3] and setting it to zero to find its minimum

$$\frac{\partial \mathcal{L}}{\partial w} = \Sigma w - \lambda \mu - \kappa \mathbf{1}^4 = 0 \tag{4.15}$$

Collecting optimality condition equations (4.12), (4.13) and (4.15) in matrix form

$$\begin{bmatrix} \Sigma & \mu & \mathbf{1} \\ \mu^T & 0 & 0 \\ \mathbf{1}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ -\lambda \\ -\kappa \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{\mu}_{target} \\ \mathbf{1} \end{bmatrix} \tag{4.16}$$

Under the assumption that Σ is full rank and μ is not a multiple of $\mathbf{1}$, then equation (4.16) is solvable by matrix inversion (Boyd and Vandenberghe, 2004). The resulting portfolio vector w_{MVP} defines the **mean-variance optimal portfolio**.

$$\begin{bmatrix} w_{MVP} \\ -\lambda \\ -\kappa \end{bmatrix} = \begin{bmatrix} \Sigma & \mu & \mathbf{1} \\ \mu^T & 0 & 0 \\ \mathbf{1}^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \bar{\mu}_{target} \\ \mathbf{1} \end{bmatrix} \tag{4.17}$$

Note that mean-variance optimization is also used in signal processing and wireless communications in order to determine the optimal beamformer (Almeida et al., 2015), using, for example, Minimum Variance Distortion Response (MVDR) filters (D. P. Mandic and Xia, 2013).

[3] the covariance matrix Σ is by definition symmetric so $\frac{\partial (w^T \Sigma w)}{\partial w} = \Sigma w$

[4] $\mathbf{1} = [1\ 1\ \cdots\ 1] \in \mathbb{R}^M$

4.2.2 Quadratic Programming

According to constraint (4.12), the portfolio weights w_i can get negative values (short sales), as long as the L_1-norm of the portfolio vector is equal to unity. If short sales are prohibited, then an additional constraint should be added, and in this case, the optimization problem becomes

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} \quad & \boldsymbol{w}^T \boldsymbol{\mu} = \bar{\mu}_{target} \\ \text{and} \quad & \|\boldsymbol{w}\|_1 = 1 \\ \text{and} \quad & w_i \geq 0 \quad \forall i \in \{1, 2, \ldots, M\} \end{aligned} \tag{4.18}$$

This problem cannot be reduced to the solution of a set of linear equations. It is termed a **quadratic program** and it is solved numerically using gradient-based algorithms (Ricketts, 1982).

4.3 Transaction Costs

In real stock exchanges, such as NYSE (Wikipedia, 2018c), NASDAQ (Wikipedia, 2018b) and LSE (Wikipedia, 2018a), trading activities (buying or selling) are accompanied with expenses, including brokers' commissions and spreads (Investopedia, 2018; Quantopian, 2017), usually referring to them as **transaction costs**. Therefore every time a new portfolio vector is determined (portfolio re-balancing), the corresponding transaction costs should be subtracted from the budget.

In order to simplify the analysis around transaction costs we will use the rule of thumb (Quantopian, 2017), charging 0.2% for every activity. For example, if three stocks A are bought with price 100$ each, then the transaction costs will be 0.6$. Let the price of the stock A raising at price 107$, when we decide to sell all three stocks, then the transaction costs will be 0.642$.

4.3.1 Multi-Stage Decision Problem

Transaction costs make Portfolio Management a **Multi-Stage Decision Problem** (Neuneier, 1997), which in simple terms means that two sequences of states with the same start and end state will have different value and Reinforcement Learning can be used to identify the sequence that maximizes our target.

Chapter 5

Reinforcement Learning

In this chapter, the fundamental principles and algorithms of Reinforcement learning are introduced. In chapter 6 portfolio management and the asset allocation problem are linked to Reinforcement Learning. In chapter 7 various architectures of agents are suggested that address the optimal asset allocation problem.

Reinforcement learning (**RL**) refers to both a learning problem and a subfield of machine learning (I. Goodfellow et al., 2016). As a learning problem (Szepesvari, 2009), it refers to learning to control a system (*environment*) so as to maximize some numerical value, which represents a long-term objective (*discounted cumulative reward signal*). A typical setting where RL operates is shown in figure 5.1: A controller (*agent*) receives the controlled system's *state* and a *reward* associated with the last *state transition*. It then calculates an action which is sent back to the system. In response, the system makes a transition to a *new state* and the cycle is repeated. The problem is to learn a way of controlling the system (*policy*) so as to maximize the total reward.

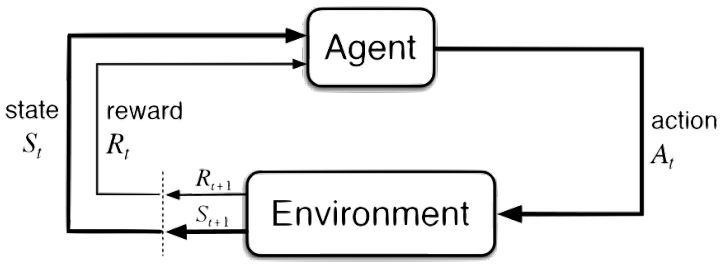


Figure 5.1: Reinforcement Learning Setting (Sutton and Barto, 1998)

5.1 Basic Concepts

5.1.1 Agent & Environment

The term **agent** is used to refer to the controller, while **environment** is used interchangeably with the term system. The goal of a reinforcement learning problem is the development (training) of an agent capable of successfully interacting with the environment, such that it maximizes some scalar objective over time.

5.1.2 Action

Action A_t is the control signal that the agent sends back to the system at time step t. It is the only way that the agent can influence the environment (system) state and as a result, lead to different reward signal sequences. The term **action space** $\mathbb{A}$ refers to the set of actions that the agent is allowed to make and it can be

- discrete: $\mathbb{A} \in \{\texttt{LONG}, \texttt{SHORT}, \texttt{HOLD}\} \rightarrow$ position
- continuous: $\mathbb{A} \in [-1, 1]^M \rightarrow$ allocation percentage

5.1.3 Reward

A **reward** R_t is a scalar feedback signal, $R_t \in \mathbb{B} \subseteq \mathbb{R}$, which indicates how well the agent is doing at step t. The agent's job is to maximize cumulative reward, over a sequence of steps.

Sequential Decision Making

Reinforcement learning addresses sequential decision-making problems (Silver, 2015a), by training agents that appreciate delayed rewards and can evaluate the long-term consequences of their actions, being able to sacrifice immediate reward to gain more long-term reward. This special property of reinforcement learning agents is very attractive to financial applications, where investment horizons range from few days and weeks to years or decades. In the latter cases, myopic agents can perform very poorly since evaluation of long-term rewards is essential in order to succeed.

Reward Hypothesis

However, reinforcement learning vitally depends on the **reward hypothesis**

Hypothesis 5.1 *All goals can be described by the maximization of expected cumulative reward*

Consequently, the selection of the appropriate reward signal for each application is very crucial. It influences the agent learned strategies since it

reflects its goals. In section 6.4 a justification for the selected reward signal is provided, along with an empirical comparison between other metrics mentioned in section 4.1.

5.1.4 State

The state is also a fundamental element of reinforcement learning, but it is usually used to refer to both the environment state and the agent state. We are focusing on the distinction between them because it will allow us to reason about the selection of the architectures in section 7.

Environment State

The **environment state** S_t^e is the system's private representation, used in order to determine the next observation O_{t+1} and reward R_{t+1}. The environment state is not usually visible to the agent and even if it visible, it may contain irrelevant information (Sutton and Barto, 1998).

Agent State

The **history** H_t at time t is the sequence of observations, actions and rewards up to that time step

$$H_t = O_1, A_1, R_1, O_2, A_2, R_2, \ldots, O_t, A_t, R_t \tag{5.1}$$

The **agent state** (a.k.a **state**) S_t^a is the agent's internal representation of the environment, used in order to select the next action A_{t+1} and it can be any function of the history

$$S_t^a = f(H_t) \tag{5.2}$$

The term **state space** $\mathbb{S}$ is used to refer to the set of possible states the agents can observe. Similar to the action space, it can be

- discrete: $\mathbb{S} \in \{\texttt{UP}, \texttt{DOWN}\} \rightarrow$ price movement (Neuneier, 1997)
- continuous: $\mathbb{S} \in \mathbb{R} \rightarrow$ percentage change (Jiang et al., 2017)

Observability

Fully observable environments allow the agent to directly observe the environment state, such that

$$O_t = S_t^e = S_t^a \tag{5.3}$$

Partially observable environments offer indirect access to the environment state, therefore the agent has to construct its own state representation S_t^a

- complete history $S_t^a = H_t$
- recurrent neural network $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$ (Graves, 2012)

5.2 Major Components of Reinforcement Learning

Reinforcement Learning agents may include one or more of these components (Silver, 2015a)

- **policy**: agent's behavior function
- **value function**: how good is each state and/or action
- **model**: agent's representation of the environment

5.2.1 Return

Let γ be the **discount factor** of future rewards, where $\gamma \in [0, 1]$, then the **return** G_t (a.k.a **future discounted reward**) is

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.4}$$

5.2.2 Policy

Policy π is the agent's behavior. It is a mapping function from state to action (Witten, 1977), such that

$$\pi : \mathbb{S} \to \mathbb{A} \tag{5.5}$$

where $\mathbb{S}, \mathbb{A}$ the state space and the action space, respectively. A policy function can be

- deterministic: $A_{t+1} = \pi(S_t)$
- stochastic: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

5.2.3 Value Function

State-value function v_π is the expected return G_t starting from state s, and then following policy π (Szepesvari, 2009)

$$v_\pi : \mathbb{S} \to \mathbb{B}, \quad v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{5.6}$$

where $\mathbb{S}, \mathbb{B}$ the state space and the rewards set ($\mathbb{B} \subseteq \mathbb{R}$), respectively.

Action-value function q_π is the expected return G_t starting from state s, taking action a, and then following policy π (Silver, 2015a)

$$q_\pi : \mathbb{S} \times \mathbb{A} \to \mathbb{B}, \quad q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{5.7}$$

where $\mathbb{S}, \mathbb{A}, \mathbb{B}$ the state space, the action space and and the reward set, respectively.

5.2.4 Model

A **model** predicts the next state of the environment S_{t+1}, given the current state S_t and the action taken A_t at time step t. It can be represented by a **state transition probability matrix** $\mathcal{P}$, such that

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \tag{5.8}$$

and a **reward function** $\mathcal{R}$, such that

$$\mathcal{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{B}, \quad \mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{5.9}$$

where $\mathbb{S}, \mathbb{A}, \mathbb{B}$ the state space, the action space and and the reward set, respectively.

5.3 Markov Decision Process

5.3.1 Markov Property

A state S_t (Silver, 2015b) satisfy **Markov property** if and only if

$$\mathbb{P}[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_t, S_{t-1}, \ldots, S_1] \tag{5.10}$$

This implies that the state S_t is a sufficient statistic of the future, therefore the history H_t can be dismissed.

5.3.2 Definition

A fully observable environment, which satisfies equation (5.3), can be modeled as a **Markov Decision Process** (**MDP**). A Markov Decision Process (Poole and Mackworth, 2010) is a tuple $\langle \mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathbb{S}$ is a *finite* set of states (state space), such that they satisfy the Markov property, as in definition (5.10)

- $\mathbb{A}$ is a *finite* set of actions (action space)
- $\mathcal{P}$ is a state transition probability matrix, as in definition (5.8)
- $\mathcal{R}$ is a reward function, as in definition (5.9)
- γ is a discount factor, $\gamma \in [0, 1]$

5.3.3 Optimality

Value Function

The **optimal state value function** v_* is the maximum state value function over all policies

$$v_*(s) = \max_{\pi} \; v_\pi(s), \quad \forall s \in \mathbb{S} \tag{5.11}$$

The **optimal action value function** q_* is the maximum action value function over all policies

$$q_*(s, a) = \max_{\pi} \; q_\pi(s, a), \quad \forall s \in \mathbb{S}, a \in \mathbb{A} \tag{5.12}$$

Policy

Define a partial ordering over policies (Silver, 2015b)

$$\pi \leq \pi' \quad \Leftarrow \quad v_\pi(s) \leq v'_\pi(s), \; \forall s \in \mathbb{S} \tag{5.13}$$

For an MDP the following theorems are true [1]

Theorem 5.2 *There exists an optimal policy π_* that is better than or equal to all other policies, such that $\pi_* \geq \pi, \; \forall \pi$*

Theorem 5.3 *All optimal policies achieve the optimal state value function, such that $v_{\pi_*}(s) = v_*(s), \; \forall s \in \mathbb{S}$*

Theorem 5.4 *All optimal policies achieve the optimal action value function, such that $q_{\pi_*}(s, a) = q_*(s, a), \; \forall s \in \mathbb{S}, a \in \mathbb{A}$*

5.3.4 Bellman Equation

Given a Markov Decision Process $\langle \mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ because of the Markov property (5.10) that states in $\mathbb{S}$ satisfy

[1] The Proofs are based on the contraction property of Bellman operator (Poole and Mackworth, 2010).

- the policy π is a distribution over actions given states

$$\pi(s|a) = \mathbb{P}[A_t = a|S_t = s] \tag{5.14}$$

Without loss of generality we assume that the policy π is stochastic because of the state transition probability matrix $\mathcal{P}$ (Sutton and Barto, 1998). Thanks to the Markov property, MDP policies depend only on the current state and are time-independent, stationary (Silver, 2015b), such that

$$A_t \sim \pi(\cdot|S_t), \quad \forall t > 0 \tag{5.15}$$

- the state value function v_π can be decomposed into two parts: the immediate reward and the discounted reward of successor state γR_{t+1}

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &\overset{(5.4)}{=} \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\ &\overset{(5.4)}{=} \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &\overset{(5.10)}{=} \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned} \tag{5.16}$$

- the action value function q_π can be similarly decomposed to

$$q_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \tag{5.17}$$

Equations (5.16) and (5.17) are the **Bellman Expectation Equations** for Markov Decision Processes formulated by Bellman (1957).

5.3.5 Extensions

Markov Decisions Processes can be exploited by reinforcement learning agents, who can optimally solve them (Sutton and Barto, 1998; Szepesvari, 2009). Nonetheless, most real-life applications are not satisfying one or more of the conditions stated in subsection 5.3.2. As a consequence, modifications of them lead to other types of processes, such as Infinite MDP and Partially Observable MDP.

Infinite Markov Decision Process

In the case of either the state space $\mathbb{S}$, or the action space $\mathbb{A}$, or both being infinite [2] then the environment can be modelled as an **Infinite Markov Decision Process**. Therefore, in order to implement the policy π or/and the action value function q_π in a computer, a differentiable function approximation method must be used (Sutton, McAllester, et al., 1999), such as a least squares function approximation or a neural network (Mitchell, 1997).

Partially Observable Markov Decision Process

If $S_t^e \neq S_t^a$ then the environment is partially observable and it can be modeled as a **Partially Observable Markov Decision Process** (**POMDP**). POMDP is a tuple $\langle \mathbb{S}, \mathbb{A}, \mathbb{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$ where

- $\mathbb{O}$ is a *finite* set of observations (observation space)
- $\mathcal{Z}$ is an observation function, $\mathcal{Z}^a_{s'o} = \mathbb{P}[O_{t+1}|S_{t+1} = s', A_t = a]$

Financial markets are treated as POMDPs in section 6, where assumptions are made about the link between sets $\mathbb{O}$ and $\mathbb{S}$.

5.4 Reinforcement Learning Algorithms

The reinforcement learning algorithms can be categorized by their components (Silver, 2015a)

	Value Based	Policy Based	Actor Critic
Policy	implicit	yes	yes
Value Function	yes	no	yes

Table 5.1: Reinforcement Learning Algorithms Categories: Policy & Value

	Model Free	Model Based
Policy	policy or/and value function	
Value Function		
Model	no	yes

Table 5.2: Reinforcement Learning Algorithms Categories: Model

RL agents usually suffer from local maxima because of exploitation. At the early stages of their training, they keep visiting the same states that locally maximize their reward, without exploring the rest of the state space. This is

[2] countably infinite (discrete) or continuous

the **exploration-exploitation** trade-off (Szepesvari, 2009) that many policies try to balance.

5.4.1 Q-Learning

Q-Learning is a value-based model-free reinforcement learning algorithm (Watkins and Hellaby, 1989). It works by successively improving its evaluations of the action value function q and thus the name. Let Q, be the estimate of the true action value function, then Q is updated online (every time step) according to

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_{a' \in \mathbb{A}} Q(S_{t+1}, a') - Q(S_t, A_t) \right] \tag{5.18}$$

where $\alpha \geq 0$ the learning rate and $\gamma \in [0, 1]$ the discount factor.

Theorem 5.5 *For a Markov Decision Process, Q-learning converges to the optimum action values with probability 1, so long as all actions are repeatedly sampled in all states and the action values are represented discretely*

The proof of theorem 5.5 is provided by Watkins and Hellaby (1989) and relies on the contraction property of the Bellman Operator (Sutton and Barto, 1998), showing that

$$Q(s, a) \rightarrow q_*(s, a) \tag{5.19}$$

ε-greedy Policy

From table 5.1 value based reinforcement learning algorithm, such as Q-learning, have an implicit policy. This means that their policy is directly related to their learned value function. A **greedy policy** improvement π_{greedy} over the action value function q is model free, such that

$$\pi^q_{\text{greedy}}(s) = \underset{a \in \mathbb{A}}{\text{argmax}} \; q(s, a) \tag{5.20}$$

Greedy policy improvement is a valid implicit policy that can be used for Q-learning, however it focuses only on exploitation, section 5.4, running the risk of converging to local maxima, losing the global one(s). An alternative policy improvement is the **ε-greedy policy** improvement $\pi_{\varepsilon\text{-greedy}}$ that ensures continual exploration (Silver, 2015c). It is a stochastic policy that with probability $1 - \varepsilon$ chooses the greedy policy (5.20) and with probability

ε chooses an action at random from $\mathbb{A}$.

$$\pi^{q}_{\varepsilon\text{-greedy}}(a|s) = \begin{cases} \frac{\varepsilon}{m} + 1 - \varepsilon & \text{if } a^* = \text{argmax}_{a \in \mathbb{A}}\, Q(s,a) \\ \frac{\varepsilon}{m} & \text{otherwise} \end{cases} \tag{5.21}$$

where $m = |\mathbb{A}|$ the number of possible actions (cardinality of action set).

Q-learning with ε-greedy policy training algorithm is shown in algorithm 1.

Algorithm 1: Q-Learning with ε-greedy Policy

input: learning rate α, finite state space $\mathbb{S}$,
finite action space $\mathbb{A}$,
number of episodes `num_episodes`,
terminal state $S_{terminal}$

$Q(s,a) \leftarrow 0,\ \forall s \in \mathbb{S}, a \in \mathbb{A}$
for $i \leftarrow 1$ **to** `num_episodes` **do**
  initialize state S
  repeat
    $A = \pi^{Q}_{\varepsilon\text{-greedy}}(S)$
    $R, S' = \texttt{take\_action}(A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_{a \in \mathbb{A}}\ Q(S',A) - Q(S,A)\big]$
    $S \leftarrow S'$
  until $S = S_{terminal}$
end

Part III

Research

Chapter 6

Portfolio Management as IPOMDP

In chapters 4 and 5 the asset allocation problem and the reinforcement learning were presented, respectively. In subsection 5.3.5 the concepts of Partially Observable Markov Decision Processes (POMDP) and Infinite Markov Decision Processes (IMDP) were formalized. In this chapter, we show how Portfolio Management problem can be treated as an Infinite POMDP or IPOMDP.

The goal is the development of a reinforcement learning **trading agent**, capable of solving the asset allocation problem, after being trained on the historical market data.

6.1 Assumptions

Since back-test tradings are only considered, where the trading agent pretends to be back in time at a point in the market history, not knowing any "future" market information, and does paper trading from then onward (Jiang et al., 2017). As a requirement for the back-test experiments, the following two assumptions must apply: zero slippage and zero market impact, both of which are realistic if the traded assets' volume in a market is high enough (Wilmott, 2007).

6.1.1 Zero Slippage

Slippage refers to the difference between the expected price of a trade and the price at which the trade is actually executed (Investopedia, 2017).

Zero Slippage: The liquidity of all market assets is high enough that, each trade can be carried out immediately at the last price when an order is placed.

6.1.2 Zero Market Impact

Asset prices are determined by the **Law of Supply and Demand** (Investopedia, 2016), therefore any trade affects the price of asset being traded. Nonetheless, we assume that the capital invested by the trading agent is so insignificant that is has no influence on the market, or **Zero Market Impact**.

6.2 State Space & Observation Space

Financial markets are not observable (Silver, 2015a), and therefore do not satisfy equation (5.3), because at any time t only the asset prices are available, which do not provide enough information for determining future prices[1]. As consequence, we should construct the agent's state space $\mathbb{S}$ from the received observations $\boldsymbol{o}_t \in \mathbb{O}$ (daily asset prices).

As covered in subsection 3.3.2, log returns' statistical properties are preferred over asset prices and simple returns, thus the (agent) state $\boldsymbol{S}_t$ at time step t is the **cumulative log returns** of window T of the M assets of the portfolio, tupled with the previous time step portfolio vector $\boldsymbol{w}_{t-1}$. Since the state space is continuous, the environment should be modelled as an Infinite Partially Observable MDP and function approximators (i.e neural networks) should be used in order to generalize state to action mapping. The selection of the state was made in order to preserve the sequential dependence of the observations, which will be exploited by the use of recurrent neural network architectures in chapter 7.

$$\boldsymbol{S}_t \triangleq \langle \boldsymbol{w}_{t-1}, \begin{bmatrix} r_t^{(1)}[1] & r_t^{(2)}[1] & \cdots & r_t^{(M)}[1] \\ r_t^{(1)}[2] & r_t^{(2)}[2] & \cdots & r_t^{(M)}[2] \\ \vdots & \vdots & \vdots & \vdots \\ r_t^{(1)}[T] & r_t^{(2)}[T] & \cdots & r_t^{(M)}[T] \end{bmatrix} \rangle \tag{6.1}$$

where $r_\tau^{(i)}[k] \in \mathbb{R}$ is the k-periods log return of the i^{th} asset at time step τ, calculated according to equation (3.9) and the state space $\mathbb{S} \subseteq \left(\mathbb{R}^{T \times M} \times [-1, 1]^M\right)$.

6.3 Action Space

The trading agent in order to solve the asset allocation problem, it should be able to determine the portfolio vector $\boldsymbol{w}_t$ at time every time step t, therefore

[1] If prices were an AR(1) process (Danilo P. Mandic, 2018a), then financial markets could be treated as pure MDPs.

the action $\boldsymbol{A}_t$ at time t is the portfolio vector $\boldsymbol{w}_{t+1}$ at time $t+1$

$$\boldsymbol{A}_t = \boldsymbol{w}_{t+1} \begin{bmatrix} w_{t+1}^{(1)} & w_{t+1}^{(2)} & \cdots & w_{t+1}^{(M)} \end{bmatrix}, \quad \boldsymbol{A}_t \in \mathbb{A} \subseteq [0,1]^M \tag{6.2}$$

Note that in case of short selling, $\mathbb{A} \subseteq [-1,1]^M$. The action space is also continuous and as a result reinforcement learning algorithms (Liang et al., 2015) should be used that fit this requirement as well.

6.4 Reward Signal

In section 4.1 various evaluation metrics were introduced for accessing the performance of a portfolio, such as daily (log) returns, volatility and Sharpe Ratio. All three metrics were used in three different experiments, suggesting that daily log returns is the best reward signal since Sharpe Ratio is slowly changes not providing large gradients for the network to be trained (I. Goodfellow et al., 2016) and (negative) volatility is highly unstable. The comparison matrices between the different reward signals are provided in figure 7.3. Overall, the reward signal R_t at time step t is the portfolio log return, given by equation (3.10)

$$R_t = r_t^{(p)} = ln(1 + \boldsymbol{w}_t^T \boldsymbol{R}_t) \tag{6.3}$$

where $\boldsymbol{R}_t \in \mathbb{R}^M$ the returns vector for the M assets at time step t, derived from the observations $\boldsymbol{o}_{t-1}$ and $\boldsymbol{o}_t$.

Chapter 7

Algorithmic Portfolio Manager

In this chapter, we present the algorithmic portfolio manager, `qtrader`, based on model-free value based Reinforcement Learning algorithm, implemented by a Deep Q-Recurrent Neural Network

7.1 DQRN: Deep Q-Recurrent Network

The basic Q-Learning algorithm, introduced in subsection 5.4.1, operates on finite state spaces and finite action spaces, reassuring convergence to the optimal action-value function q_* for pure Markov Decision Processes. Nonetheless, we showed in chapter 6 that portfolio management is an Infinite Partially Observable Markov Decision Process, thus architectural modifications should be made to the algorithm 1 in order to be compatible with continuous state and action spaces $\mathbb{S}$ and $\mathbb{A}$, respectively.

Firstly, a function approximator, neural network, will be used to deal with the infinite, continuous spaces $\mathbb{S}$ and $\mathbb{A}$. Additionally, recurrent layers will be used in the network to accommodate the partial observability of the environment, as justified in subsection 5.3.5.

7.1.1 Architecture

The `qtrader` agent consists of four layers: two Recurrent Gated Rectified Unit (GRU) layers (Cho et al., 2014), followed by an Affine (Fully-Connected) layer (Graves, 2012) estimating action value function Q and a Softmax layer (I. Goodfellow et al., 2016) that converts Q values to portfolio weights, estimated action $\hat{A}$. Then a Gaussian Policy π_{gaussian} is followed to determine action A.

Note that despite the inspiration from the original DQN (Mnih, Kavukcuoglu, et al., 2013), experience replay is *not* used, since it is breaking the sequential dependency of the time-series (asset log returns).

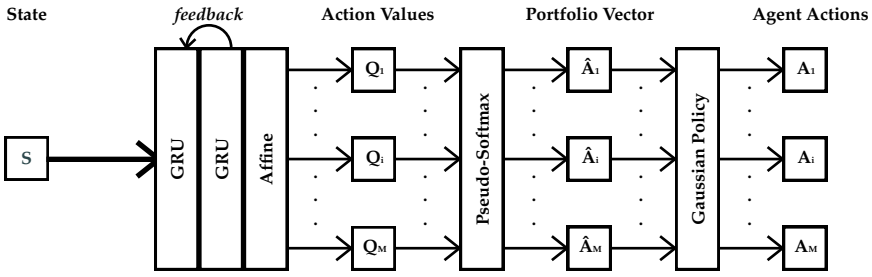


Figure 7.1: `qtrader` Architecture

7.1.2 Affine Layer

In Deep Learning literature (I. Goodfellow et al., 2016), the term **affine** refers to a neural network layer with parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ that performs a mapping from an input matrix $\boldsymbol{X}$ to a output vector $\boldsymbol{y}$ according to

$$f_{\text{affine}}(\boldsymbol{X}; \boldsymbol{W}, \boldsymbol{b}) = \hat{\boldsymbol{y}} \triangleq \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b} \tag{7.1}$$

7.1.3 Recurrent Gated Rectified Unit Layer

The sequential nature of the data (time-series) and the assumption (4.1) that returns follow a non-linear autoregressive process, encourages the use of Recurrent Neural Networks (RNN) (Danilo P. Mandic and Chambers, 2001). RNNs have feedback loops, compared to Feedforward Neural Networks (a.k.a Multi-Layer Perceptrons) (I. Goodfellow et al., 2016) that do not. Feedforward networks can be also used with sequential data when memory is brute-forced, leading to very large and deep architectures in order to achieve similar results with smaller RNNs (Graves, 2012). Vanilla RNN layers are not used because of the vanishing gradient (Pascanu et al., 2012), but instead **Gated Rectified Unit**s (GRU) are preferred because of their low computational complexity[1] and simple architecture.

The main difference of GRU-RNN compared to simple Feedforward network is the existence of an internal state $\boldsymbol{h}_{t-1}$ that contributes to the output $\boldsymbol{o}_t$, such that

$$\boldsymbol{o}_t \triangleq \boldsymbol{h}_t \triangleq f_{\text{RNN}}(g_{\text{GRU}}(\boldsymbol{h}_{t-1}), \boldsymbol{x}_t; \boldsymbol{\theta}) \tag{7.2}$$

where $\boldsymbol{\theta}$ the parameters of the model and g_{GRU} the GRU function.

The goal of training is finding the right parameters $\boldsymbol{\theta}$ that minimize an ob-

[1] compared to LSTM (Ortiz-Fuentes and Forcada, 1997).

jective function J, which in that case is given by the equation 5.18

$$J(\boldsymbol{\theta}) = R_t + \gamma \max_{a \in \mathbb{A}} Q(S_{t+1}, a; \boldsymbol{\theta}) - Q(S_t, A_t; \boldsymbol{\theta}) \tag{7.3}$$

The gradient $\nabla_{\boldsymbol{\theta}} J$ is calculated at every time step t with **Backpropagation Through Time** (**BPTT**) (Werbos, 1990) algorithm, and the weights are updated using **Stochastic Gradient Descent** (**SGD**) (Sirignano and Spiliopoulos, 2017) optimization

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J \tag{7.4}$$

where $\eta > 0$ is the learning rate hyperparameter (I. Goodfellow et al., 2016).

7.1.4 Pseudo-Softmax Layer

Softmax is a exponential normalization function which transforms a vector $\boldsymbol{x} \in \mathbb{R}^M$ to a normalized vector $\boldsymbol{h} \in [0,1]^M$, such that $\sum_{i=1}^{M} h_i = 1$, given by

$$\sigma_{\text{softmax}}(x_i) = h_i \triangleq \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}, \quad \forall i \in \{1, 2, \ldots, M\} \tag{7.5}$$

Note that the output vector $\boldsymbol{h}$ satisfy both portfolio vector conditions, as defined in subsection 3.1.2. However, when short sales are allowed, the portfolio weights can have negative values, which is not possible using the softmax function as defined in (7.5). Therefore, we suggest a new **Pseudo-Softmax** layer that applies softmax function to the absolute values of the x_i elements and keeps the sign of them after the normalization, such that

$$\varphi_{\text{pseudo-softmax}}(x_i) = w_i \triangleq \texttt{sign}(x_i) \frac{e^{|x_i|}}{\sum_{j=1}^{M} e^{|x_j|}}, \quad \forall i \in \{1, 2, \ldots, M\} \tag{7.6}$$

7.1.5 Gaussian Policy

In subsection 5.4.1 the implicit ε-greedy policy for Q-Learning was introduced, which trades exploitation for exploration by occasionally taking random actions. However, in the context of trading, random selection of assets can be very negative for performance, therefore a less noisy policy is used, **Gaussian Policy** π_{Gaussian}. Gaussian policy (Silver, 2015d) adds Gaussian noise to the output of the DQRN, with zero mean and variance the sample variance of each output individually. As a consequence, assets that have good returns for a long time will have a small variance and the Gaussian

Policy will not modify them, while the opposite applies to noise assets.

$$\pi^{q}_{\text{Gaussian}}(\boldsymbol{A}) \triangleq \frac{(\hat{\boldsymbol{A}} - \boldsymbol{\mu})}{\boldsymbol{\sigma}} \tag{7.7}$$

Algorithm 2: Deep Q-Recurrent Network with Gaussian Policy

input: learning rate α,
number of episodes `num_episodes`,
number of steps per episode `num_steps`

randomly initialize Q-network parameters $\boldsymbol{\theta}$
for *episode* $\leftarrow 1$ **to** `num_episodes` **do**
  initialize state S
  for *step* $\leftarrow 1$ **to** `num_steps` **do**
    $\boldsymbol{Q} = \texttt{DQRN\_forward}(S)$
    $\hat{\boldsymbol{A}} = \texttt{pseudo\_softmax}(S)$
    $\boldsymbol{A} = \pi^{Q}_{\text{Gaussian}}(\hat{\boldsymbol{A}})$
    $R, S' = \texttt{take\_action}(A)$
    $J = R + \gamma \max_{a \in \mathbb{A}} \; Q(S', a) - Q(S, A)$
    gradients $= \texttt{bptt}(J)$
    $\texttt{sgd}(\text{gradients})$
  end
end

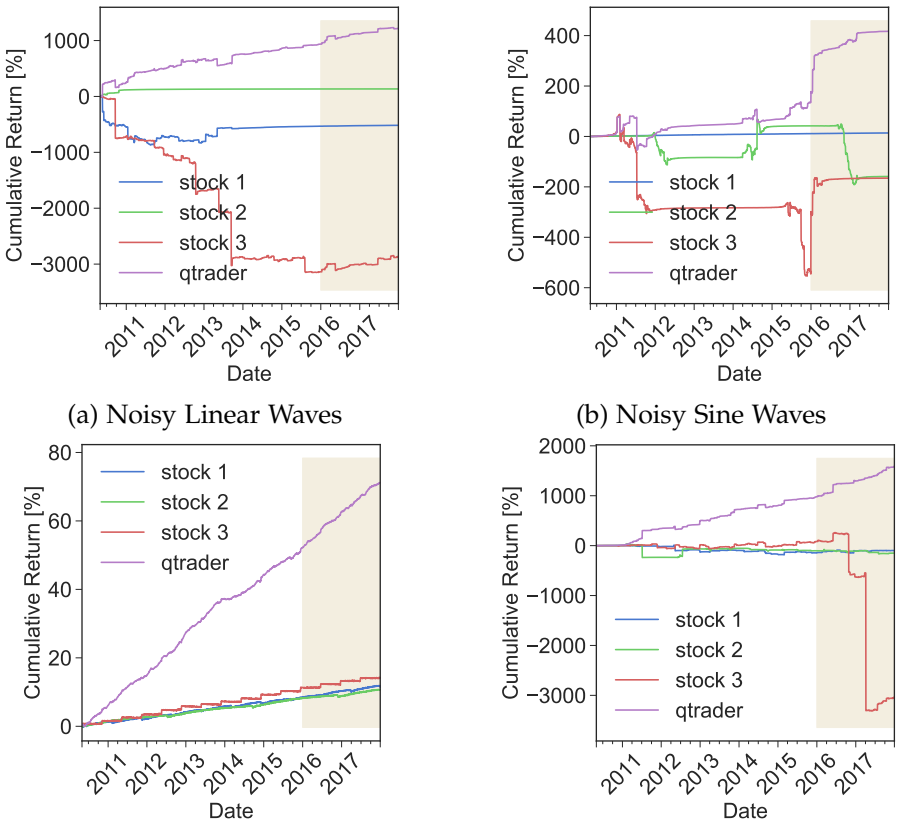
7.2 Simulations & Results

Experiments and simulations have been carried out on both synthetic data in order to verify the validity of the architecture and on NASDAQ (Wikipedia, 2018b) market, trading assets with large market capital so that the assumptions in section 6.1 are almost[2] satisfied.

7.2.1 Synthetic Market

The synthetic market is constructed using combinations of deterministic signals, such as sine waves, sawtooth waves, chirp waves and then using the suggested architecture from subsection 7.1.1. The cumulative reward is provided in figure 7.2 for the different deterministic signals.

[2]It still depends on the traded volumes of our agent, but it is assumed that they are negligible compared to the market capital of the traded assets



(a) Noisy Linear Waves

(b) Noisy Sine Waves

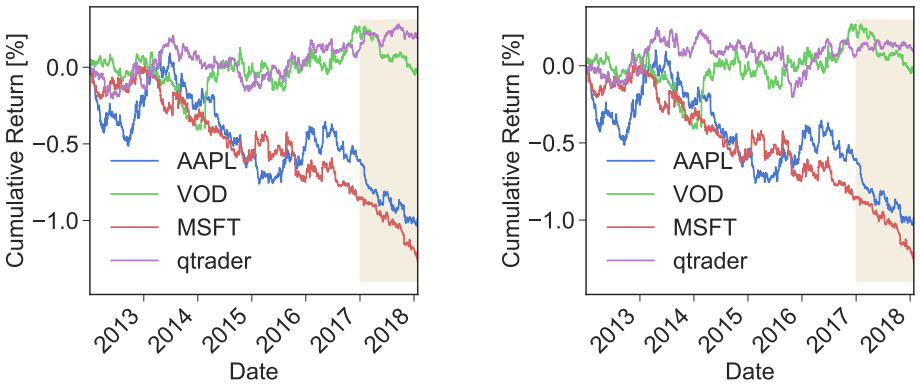
(c) Noisy Sawtooth Waves

(d) Noisy Chirp Waves

Figure 7.2: Cumulative Reward of `qtrader` agent on Synthetic Market of noisy deterministic signals. Yellow regions are the out-of-sample, test periods. In all situations the agent outperforms the individual assets, by significant orders of magnitude.

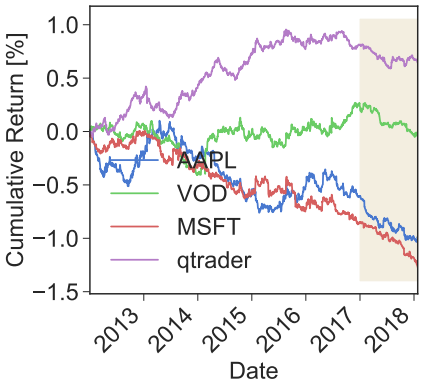
7.2.2 NASDAQ

The Deep Q-Recurrent Network with Gaussian Policy is tuned and trained on three NASDAQ assets, `AAPL`, `VOD`, `MSFT`. Different reward functions are used and their performance on the cumulative percentage reward is compared, proving empirically that daily returns is the better reward function out of expected returns, volatility and Sharpe Ratio. In all experiments transaction costs are taken into consideration with rate 0.2%, as explained in section 4.3.

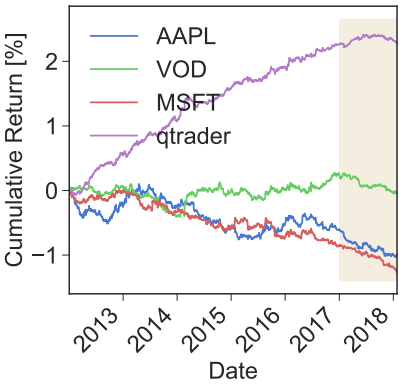


(a) Expected Returns

(b) Sharpe Ratio



(c) Volatility



(d) Daily Returns

Figure 7.3: Cumulative Reward of `qtrader` agent on NASDAQ, for different reward functions. Daily returns is the best out of expected returns, volatility and Sharpe Ratio. Yellow regions are the out-of-sample, test periods.

Part IV

Future Work

Chapter 8

Market Simulation

One of the main difficulties experienced when working with market data is the availability of data. The non-stationary nature[1] of financial time-series do not allow us to use very long periods of data since a good trading agent trained on 1990's data will be most likely unreliable today. Thus we are short of data when deep neural network architectures require large amounts of data to be trained. Therefore, we plan to develop a market simulator whose aim is to capture the dynamics of a market and generate signals that preserve the statistical properties of the provided historical data.

8.1 Problem Definition

Market simulation can be framed as a probability density estimation problem if the prices of the M assets are treated as a M-dimensional vector which is sampled at each time step. The goal is to approximate the generating process that produces the M-dimensional vector samples, such that their statistical properties[2] are respected. Then arbitrarily long sequences will be generated and used for training the agent, which will be finally trained on real data, using **Transfer Learning** (Wei et al., 2017).

8.2 Implementation Plan

Initially traditional methods such Copula Analysis (Luenberger, 1998) and Singular Value Decomposition (SVD) (I. Goodfellow et al., 2016) will be implemented and used as baseline models. Then more elaborate methods, such as Kernel Density Estimation (KDE) (Chen, 2017) will be tried, as well as non-parametric methods, such as Generative Adversarial Networks (I. J. Goodfellow et al., 2014).

[1] time-variant first and second statistical moments.

[2] ideally up to fourth order statistical moments.

Chapter 9

Actor Critic Agent

Deep Q-Learning has been proven very effective in applications with infinite state space, but finite actions set (Mnih, Kavukcuoglu, et al., 2013), while in our case the action space is also continuous. The Pseudo-Softmax layer along with the structure of portfolio vectors allow us to extend Deep Q-Learning to infinite action space in $[-1, 1]$, but it this case the policy is implicitly defined and is not trained. **Policy Gradient** are policy-based algorithms that extend to continuous, state and action, spaces directly from their definition and provide an end-to-end trainable pipeline (Sutton, McAllester, et al., 1999; Liang et al., 2015). Additionally, **Actor Critic** algorithms combine the benefits of both value-based algorithms, such as Q-Learning, and policy-based algorithms, such as Policy Gradient.

9.1 Implementation Plan

First, try to implement the vanilla policy gradient algorithms suggested by Necchi (2016) and reproduce his results on the synthetic market data. Then try to adapt the state-of-the-art **Asynchronous Advantage Actor Critic** (Mnih, Badia, et al., 2016) algorithm and leverage its architectural advantage in multi-core CPU training.

9.2 Evaluation Plan

Compare performance and training time of the DQRN and the Actor Critic agents on the synthetic market data, NASDAQ and the simulated market data.

Bibliography

Markowitz, Harry (1952). "Portfolio Selection". In: *The Journal of Finance* 7.1, pp. 77–91. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2975974.

Bellman, Richard Ernest (1957). *Dynamic Programming*.

Markowitz, Harry (1961). "Market Value, Time, and Risk". In:

Sharpe, William F. (1964). "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk". In: *The Journal of Finance* 19.3, pp. 425–442. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2977928.

— (1970). *Portfolio theory and capital markets*. New York, NY, USA: McGraw-Hill.

Witten, Ian H. (1977). *An adaptive optimal controller for Discrete-Time Markov Environments*. [Online]. Available: https://www.cs.waikato.ac.nz/~ihw/papers/77-IHW-AdaptiveController.pdf. [Accessed: 25- Jan- 2018].

Papadimitriou, Christos H. and Kenneth Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13-152462-3.

Ricketts, R. E. (1982). "Practical Optimization". In: *International Journal for Numerical Methods in Engineering* 18.6, pp. 954–954. ISSN: 1097-0207. DOI: 10.1002/nme.1620180612. URL: http://dx.doi.org/10.1002/nme.1620180612.

Kroll, Yoram, Haim Levy, and Harry M. Markowitz (1984). "Mean-Variance Versus Direct Utility Maximization". In: *The Journal of Finance* 39.1, pp. 47–61. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2327667.

Watkins, Christopher and John Cornish Hellaby (1989). "Learning from Delayed Rewards". PhD thesis. Cambridge, UK: King's College. URL: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Werbos, P. J. (1990). "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10, pp. 1550–1560. ISSN: 0018-9219. DOI: 10.1109/5.58337.

John Moody and Matthew Saffell (1997). *Reinforcement Learning for Trading Systems and Portfolios*. [Online]. Available: https://pdfs.semanticscholar.org/10f3/4407d0f7766cfb887334de4ce105d5aa8aae.pdf. [Accessed: 15- Jan- 2018].

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill. ISBN: 0070428077.

Neuneier, Ralph (1997). *Optimal Asset Allocation Using Adaptive Dynamic Programming*. [Online]. Available: http://papers.nips.cc/paper/1121-optimal-asset-allocation-using-adaptive-dynamic-programming.pdf. [Accessed: 15- Jan- 2018].

Ortiz-Fuentes, J. D. and M. L. Forcada (1997). "A comparison between recurrent neural network architectures for digital equalization". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4, 3281–3284 vol.4. DOI: 10.1109/ICASSP.1997.595494.

Luenberger, David B. (1998). *Investment Science*. New York: Oxford University Press.

Sutton, Richard S. and Andrew G. Barto (1998). *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press. ISBN: 0262193981.

Sutton, Richard S., David McAllester, et al. (1999). "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Denver, CO: MIT Press, pp. 1057–1063. URL: http://dl.acm.org/citation.cfm?id=3009657.3009806.

Mandic, Danilo P. and Jonathon A. Chambers (2001). *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures, and Stability*. John Wiley & Sons.

Tsay, Ruey S. (2003). *Analysis of Financial Time Series*. John Wiley & Sons. ISBN: 9780471264101. DOI: 10.1002/0471264105.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. New York, NY, USA: Cambridge University Press. ISBN: 0521833787.

Wilmott, Paul (2007). *Paul Wilmott Introduces Quantitative Finance*. 2nd ed. New York, NY, USA: Wiley-Interscience. ISBN: 0470319585, 9780470319581.

Szepesvari, Csaba (2009). *Algorithms for Reinforcement Learning*. https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf. Morgan & Claypool Publishers.

Poole, David L. and Alan K. Mackworth (2010). *Artificial Intelligence: Foundations of Computational Agents*. New York, NY, USA: Cambridge University Press. ISBN: 0521519004, 9780521519007.

Graves, Alex (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Vol. 385. Studies in Computational Intelligence. Springer. ISBN: 978-3-642-24796-5. DOI: 10.1007/978-3-642-24797-2. URL: https://doi.org/10.1007/978-3-642-24797-2.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Mandic, D. P. and Y. Xia (2013). "Augmented MVDR Spectrum-Based Frequency Estimation for Unbalanced Power Systems". In: *IEEE Transactions on Instrumentation and Measurement* 62.7, pp. 1917–1926. ISSN: 0018-9456. DOI: 10.1109/TIM.2013.2253160.

Mnih, Volodymyr, Koray Kavukcuoglu, et al. (2013). "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Goodfellow, I. J. et al. (2014). "Generative Adversarial Networks". In: *ArXiv e-prints*. arXiv: 1406.2661 [stat.ML].

Almeida, Náthalee C., Marcelo A.C. Fernandes, and Adrião D.D. Neto (2015). *Beamforming and Power Control in Sensor Arrays Using Reinforcement Learning*. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4435209/pdf/sensors-15-06668.pdf. [Accessed: 20- Jan- 2018].

Liang, Yitao et al. (2015). "State of the Art Control of Atari Games Using Shallow Reinforcement Learning". In: *CoRR* abs/1512.01563. arXiv: 1512.01563. URL: http://arxiv.org/abs/1512.01563.

Silver, David (2015a). *Introduction to Reinforcement Learning*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf. [Accessed: 25- Jan- 2018].

— (2015b). *Markov Decision Processes*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MDP.pdf. [Accessed: 25- Jan- 2018].

— (2015c). *Model-Free Control*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/control.pdf. [Accessed: 25- Jan- 2018].

— (2015d). *Policy Gradient*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/pg.pdf. [Accessed: 25- Jan- 2018].

Feng, Yiyong and Daniel P. Palomar (2016). *A Signal Processing Perspective on Financial Engineering*. [Online]. Available: http://www.ece.ust.hk/~palomar/Publications_files/2016/Feng&Palomar-FnT2016.pdf. [Accessed: 20- Jan- 2018].

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Investopedia (2016). *Law of Supply and Demand*. [Online]. Available: https://www.investopedia.com/terms/l/law-of-supply-demand.asp. [Accessed: 15- Jan- 2018].

Kennedy, Douglas (2016). *Stochastic Financial Models*. CRC Press.

Mnih, Volodymyr, Adrià Puigdomènech Badia, et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *CoRR* abs/1602.01783. arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

Necchi, Pierpaolo (2016). *Policy Gradient Algorithms for Asset Allocation Problem*. [Online]. Available: https://github.com/pnecchi/Thesis/blob/master/MS_Thesis_Pierpaolo_Necchi.pdf. [Accessed: 15- Jan- 2018].

Nemati, S., M. M. Ghassemi, and G. D. Clifford (2016). "Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach". In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2978–2981. DOI: 10.1109/EMBC.2016.7591355.

Chen, Y.-C. (2017). "A Tutorial on Kernel Density Estimation and Recent Advances". In: *ArXiv e-prints*. arXiv: 1704.03924 [stat.ME].

Investopedia (2017). *Slippage*. [Online]. Available: https://www.investopedia.com/terms/s/slippage.asp. [Accessed: 15- Jan- 2018].

Jiang, Zhengyao, Dixing Xu, and Jinjun Liang (2017). *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*. [Online]. Available: https://arxiv.org/pdf/1706.10059.pdf. [Accessed: 15- Jan- 2018].

Quantopian (2017). *Commission Models*. [Online]. Available: https://www.quantopian.com/help. [Accessed: 03- Dec- 2017].

Sirignano, J. and K. Spiliopoulos (2017). "Stochastic Gradient Descent in Continuous Time: A Central Limit Theorem". In: *ArXiv e-prints*. arXiv: 1710.04273 [math.PR].

Wei, Ying, Yu Zhang, and Qiang Yang (2017). "Learning to Transfer". In: *CoRR* abs/1708.05629. arXiv: 1708.05629. URL: http://arxiv.org/abs/1708.05629.

Zhang, X. P. S. and F. Wang (2017). "Signal Processing for Finance, Economics, and Marketing: Concepts, framework, and big data applications". In: *IEEE Signal Processing Magazine* 34.3, pp. 14–35. ISSN: 1053-5888. DOI: 10.1109/MSP.2017.2663138.

Investopedia (2018). *Transaction Costs*. [Online]. Available: https://www.investopedia.com/terms/t/transactioncosts.asp. [Accessed: 15- Jan- 2018].

Mandic, Danilo P. (2018a). *Advanced Signal Processing: Linear Stochastic Processes*. [Online]. Available: http://www.commsp.ee.ic.ac.uk/~mandic/ASP_Slides/ASP_Lecture_2_ARMA_Modelling_2018.pdf. [Accessed: 25- Jan- 2018].

— (2018b). *Introduction to Estimation Theory*. [Online]. Available: http://www.commsp.ee.ic.ac.uk/~mandic/ASP_Slides/AASP_Lecture_3_Estimation_Intro_2018.pdf. [Accessed: 25- Jan- 2018].

Wikipedia (2018a). *London Stock Exchange*. [Online]. Available: https://en.wikipedia.org/wiki/London_Stock_Exchange. [Accessed: 15- Jan- 2018].

Wikipedia (2018b). *NASDAQ*. [Online]. Available: https://en.wikipedia.org/wiki/NASDAQ. [Accessed: 15- Jan- 2018].

— (2018c). *New York Stock Exchange*. [Online]. Available: https://en.wikipedia.org/wiki/New_York_Stock_Exchange. [Accessed: 15- Jan- 2018].