UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science: Agents, Interaction & Complexity
and
Faculty of Social Sciences
Southampton Business School: Centre for Digital Finance

Machine Learning in Fixed Income Markets: Forecasting and Portfolio Management

by

Manuel Clemente Mendonça Nunes

BSc, MSc, MBA, PhD

ORCiD: 0000-0002-7116-5502

A thesis for the degree of
Doctor of Philosophy

January 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science: Agents, Interaction & Complexity
and
Faculty of Social Sciences
Southampton Business School: Centre for Digital Finance

Doctor of Philosophy

Machine Learning in Fixed Income Markets:
Forecasting and Portfolio Management

by Manuel Clemente Mendonça Nunes

The fixed income market (i.e. bonds) is a massive asset class with an overall size of USD 100 trillion that remains relatively under-investigated using machine learning. The yield curve is its centrepiece for investors, regulators and the overall economy. In this thesis we apply machine learning to both bond forecasting and portfolio management. More specifically, we consider three different topics. The first two topics focus on machine learning models for forecasting, using multilayer perceptrons (MLPs) and long short-term memory (LSTM) networks, respectively. The third and final topic is on using reinforcement learning (RL) for portfolio management. These topics address specific gaps in the literature. In particular, existing literature lacks direct solutions for modelling the yield curve as a whole using machine learning. In addition, there is lack of work analysing and drawing interpretations from internal signals of black-box type of models like MLPs and LSTMs. Finally, there is no work that establishes RL as a framework for bond portfolio management.

In more detail, for the first topic, two models are used for forecasting the European yield curve: multivariate linear regression and MLP, at five forecasting horizons. Five variants of MLPs are analysed, using different sets of features and, in some cases, including artificially-generated data from the linear regression model. We introduce a methodology relying on a rigorous feature selection process to identify the most relevant features, which we found to be different for each target and each forecasting horizon studied, reinforcing the importance of custom-built models. Considering all forecasting horizons, the results show that the MLP using the most relevant features achieve the best results and the addition of artificially-generated data tends to improve accuracy. Overall, the results demonstrate the superiority of MLPs to forecast the yield curve, when compared to benchmarks and other studies in the literature.

For the second topic, we conduct a study of 10-year bond yield forecasting using dynamic LSTMs. In doing so, we study multiple temporal horizons, and compare the results to static MLPs using different covariates. Results show that the LSTM is capable of achieving lower prediction errors with higher confidence. Using a novel method we refer to as *LSTM-LagLasso*, we then go on to study the internal gating signal of a trained LSTM, and explain their dynamics using exogenous variables that can potentially influence bond price formation. By considering these variables at various lags and the Lasso method to select features, we show how different hidden units of the LSTM dynamically switch to make predictions during different temporal regimes and how their evolution is influenced by different external variables.

Finally, for the third topic we develop an autonomous RL system which includes a purpose-built environment for fixed income portfolio management. It interacts with an agent with a state-of-the-art algorithm, the deep deterministic policy gradient (DDPG). We successfully test this system with environment inputs from four bond exchange traded funds (ETFs), using a novel methodology which involves a number of modifications in relation to the literature. The RL algorithm showed some signs of instability requiring further research. Despite this, we demonstrate how to extract the best agents from the system, during training, using principled selection criteria. With this selection, we are able to find agents capable of outperforming both the static buy and hold strategy and the best asset in the portfolio. Overall the results confirm the potential of RL for this application.

In conclusion, this research covers three topics using machine learning in fixed income markets. In doing so, we have extended the state-of-the-art literature. The main overall objective is to provide financial practitioners with additional machine learning tools.

Contents

List of Figures

List of Tables

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published or presented in conferences as:
   Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting. *Expert Systems with Applications*, 119:362–375, 2019a. https://doi.org/10.1016/j.eswa.2018.11.012
   Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. The memory advantage of long short-term memory networks for bond yield forecasting. *International Conference on Forecasting Financial Markets, FFM*, 2019b. https://doi.org/10.2139/ssrn.3415219
   Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. LSTM-LagLasso for bond yield forecasting: Peeping into the long short-term memory networks' black box. *Workshop Advancing Machine Learning in Finance, Insurance and Economics*, 2020. https://arxiv.org/abs/2005.02217

Signed: Date:

Acknowledgements

The author wishes to express his sincere gratitude to his supervisory team, Professor Enrico Gerding, Professor Frank McGroarty, and Professor Mahesan Niranjan, for their guidance and expert advice throughout this research and for the opportunity given to conduct this study. Apart from embracing this research from the beginning, their friendship and unreserved support was permanently present, something I will remember forever with gratitude.

The author would like to extend this acknowledgement to Professor Adam Prügel-Bennett , and Professor Georgios Sermpinis, for having accepted the invitation to be my examiners, and to all my colleagues and friends in the department.

Acknowledgement is also due to the UK Engineering and Physical Sciences Research Council (EPSRC Award 1921702), Doctoral Training Partnership with the University of Southampton, for sponsoring this research programme.

In addition, the author would like to express his profound indebtedness to Professor Peter Johnson and all his team at the University Hospital Southampton.

Finally, I am deeply grateful to all my family members, in particular my wife and sons, for their unconditional love and support.

To my family for their love and support

Nomenclature

$\boldsymbol{a}$ linear regression: $(p+1) \times 1$ vector of weights. 32

E linear regression: error function. 32

$\boldsymbol{\epsilon}$ linear regression: error. 32

$\boldsymbol{f}$ linear regression: $N \times 1$ vector of outputs. 32

$f(x)$ linear regression: target or dependent variable. 32

γ linear regression: regularisation parameter. 33, 34, 75, 81, 82, 91, 98, 99

$\boldsymbol{I}$ identity matrix. 33

$||.||_1$ L_1-norm. 34, 98

$||.||_2$ L_2-norm. 34, 98

λ MTL: parameter that controls how sensitive learning is to the extra tasks. 40, 76

N linear regression: number of observations. 32

p linear regression: number of features. 32

w_j, w_0 linear regression: weights (unknown parameters). 32

x_j linear regression: inputs, independent variables or features. 32

$\boldsymbol{Y}$ linear regression: $N \times (p+1)$ matrix. 32

Double Machine Learning

D double machine learning: variable of interest, treatment, policy or intervention variable. 28

$\mathbb{E}[.]$ double machine learning: denotes the expected value of a random variable. 28

g_0, m_0 double machine learning: generic functions. 28

θ_0 double machine learning: parameter of interest or causal parameter. 28

U, V double machine learning: disturbance or error terms. 28

X double machine learning: set of control variables. 28

Y double machine learning: outcome variable. 28

Long Short-Term Memory

$\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_g, \boldsymbol{b}_o$ bias vectors. 48

$\boldsymbol{c}^{(t)}$ cell state at time step t. 48

$\boldsymbol{c}^{(t-1)}$ cell state at time step t-1. 48

$\boldsymbol{f}^{(t)}$ function for the forget gate. 48

$\boldsymbol{g}^{(t)}$ function for the input node. 48

$\boldsymbol{h}^{(t)}$ hidden state at time step t. 48

$\boldsymbol{h}^{(t-1)}$ hidden state at time step t-1. 48

$\boldsymbol{i}^{(t)}$ function for the input gate. 48

$\boldsymbol{s}_{lstm}$ LSTM-LagLasso: LSTM cell and hidden state signals (target vectors). 98

$\boldsymbol{w}$ LSTM-LagLasso: vector of unknown parameters. 98

$\boldsymbol{X}$ LSTM-LagLasso: matrix of features and respective lags. 98

$\boldsymbol{o}^{(t)}$ function for the output gate. 48

$\otimes$ Hadamard product (element-wise multiplication). 48

σ logistic sigmoid activation function. 48

$tanh$ hyperbolic tangent activation function. 48

$\boldsymbol{W}$ weight matrices. 48

$\boldsymbol{W}^{fx}$ weight matrix for the connection input-to-forget gate. 48

$\boldsymbol{x}^{(t)}$ input vector at time step t. 48

Recurrent Neural Network

$\boldsymbol{b}_h$ bias vector at the hidden level. 44

$\boldsymbol{b}_y$ bias vector at the output level. 44

$\boldsymbol{h}^{(t)}$ hidden state at time step t. 44

$\boldsymbol{h}^{(t-1)}$ hidden state at time step t-1. 44

σ_h activation function at the hidden level. 44

σ_y activation function at the output level. 44

$\boldsymbol{W}$ weight matrices. 44

$\boldsymbol{W}^{hh}$ recurrent weight matrix for hidden-to-hidden connections. 44

$\boldsymbol{W}^{hx}$ weight matrix for input-to-hidden connections. 44

$\boldsymbol{W}^{yh}$ weight matrix for hidden-to-output connections. 44

$\boldsymbol{x}^{(t)}$ input vector at time step t. 44

$\hat{\boldsymbol{y}}^{(t)}$ output at time step t. 44

Reinforcement Learning

$\mathbf{1}_m$ m-dimensional column vector of ones. 57, 113

$\{1 + R_{P,t}\}$ reward received by the agent at time step t. 114, 130, 131

$\mathcal{A}$ action space. 116

$\boldsymbol{a_t}$ agent actions at time step t. 111, 113

$\boldsymbol{a_{t-1}}$ agent actions at time step $t-1$. 113

β behaviour policy. 116

E environment. 111

$\mathbb{E}\,[.]$ denotes the expected value of a random variable. 114

$f(.)$ unscaled output, before the softmax function is applied. 115

γ reward discounting factor. 111, 114

J objective function. 114

m number of assets. 57, 113

μ deterministic policy. 116

n number of previous time steps of historical prices. 113

∇_a gradients with regards to the action a. 125

∇_{θ^μ} gradients with regards to the parameters of the behaviour policy network θ^μ. 125

$\mathcal{N}$ stochastic noise sampled from a noise process. 125

P_0 initial value of the portfolio. 114

$P_{A,t-1,close}$ closing prices of the assets at time step $t-1$. 114

$P_{A,t,close}$ closing prices of the assets at time step t. 114

π policy. 114, 116

$\mu(s|\theta^{\mu})$ behaviour policy network. 124

$\mu'(s|\theta^{\mu'})$ target policy network. 124

$\mathcal{P}(\mathcal{A})$ probability distribution over the actions. 116

P_t total portfolio value. 113, 114

Q^{μ} action-value of the state-action pair (deterministic policy). 116

$Q(s,a|\theta^{Q})$ Q network. 124

$Q'(s,a|\theta^{Q'})$ target Q network. 124

$Q^{\pi}(s_t,a_t)$ action-value of the state-action pair. 115

R_1 state return from the initial state. 114

$\boldsymbol{r_A}$ vector of expected returns of the assets in the portfolio. 57

$\boldsymbol{r_{A,t}}$ return of the assets at time step t. 114

ρ^{μ} discounted state visitation distribution for a policy μ. 124

θ^{Q} parameters of the Q network. 124

$\theta^{Q'}$ parameters of the target Q network. 124

$R_{P,t}$ return of the portfolio at time step t. 114

$R_{P,target}$ target expected portfolio return. 57

$r(s_i,a_i)$ reward received following state s_i and action a_i. 114

$r(s_t,a_t)$ immediate reward at time step t. 116

R_t state return at time step t. 114

r_t reward at time step t. 111

$\boldsymbol{\Sigma}$ covariance matrix. 57

$\boldsymbol{s_t}$ state at time step t. 111, 113

$\mathcal{S}$ state space. 116

$\tau \ll 1$ target smooth factor used for the updates of the target networks. 125

θ^{μ} parameters of the behaviour policy network. 125

$\theta^{\mu'}$ parameters of the target policy network. 124

$\boldsymbol{\theta_t}$ function approximator parameters at time step t. 115

$\boldsymbol{w}$ weights of the assets in the portfolio. 57

$\boldsymbol{w_t}$ weights of the assets in the portfolio at time step t. 113

$\boldsymbol{w_{t-1}}$ weights of the assets in the portfolio at time step $t-1$. 113

(s_t, a_t) state-action pair at time step t. 115

$\boldsymbol{x_t}$ historical prices of the assets. 113

y_t target action-value. 124

Acronyms

A | **B** | **C** | **D** | **E** | **F** | **G** | **L** | **M** | **N** | **O** | **P** | **R** | **S** | **T** | **V** | **X**

A

ANN Artificial Neural Network. 5, 6, 16, 19, 20, 22, 30, 34, 35, 69

AR AutoRegressive. 12, 14, 68

ARCH AutoRegressive with Conditional Heteroskedasticity. 13, 68

ARIMA AutoRegressive Integrated Moving Average. 12, 14, 16, 29, 68

ARMA AutoRegressive Moving Average. 12, 14, 17, 30, 68

B

bp basis point. xi, 86, 117, 130

BPTT Backpropagation Through Time. 45, 46

C

CAPM Capital Asset Pricing Model. 25, 26

CDS Credit Default Swap. 3

CEC Constant Error Carousel (LSTM networks). 51

COPDAC Compatible Off-Policy Deterministic Actor-Critic. 123

CPI Consumer Price Index. 13, 16, 41, 105, 145

D

DAX German Stock Market Index. 21, 62

DDPG Deep Deterministic Policy Gradient. iv, xi, xxviii, 9, 64, 65, 68, 121–126, 129–132, 135, 140, 142

DNS Dynamic Nelson–Siegel model. 13, 14, 17, 37, 68, 95

Double ML Double Machine Learning. 27, 28

DPG Deterministic Policy Gradient. xi, 64, 122–124

DQN Deep Q Network. xi, xxviii, 62, 63, 122–125

E

ETF Exchange Traded Fund. iv, xi, 63, 66, 116, 117, 120, 128, 130, 140

EUR-USD Currency pair euro / United States dollar. 18, 108, 145

F

forex Foreign Exchange. 5, 7, 15, 18, 19, 24, 53, 63, 66, 68, 90, 108, 116, 137

G

GARCH Generalised AutoRegressive with Conditional Heteroskedasticity. 13, 21, 68

GDP Gross Domestic Product. 23, 24, 145

GRU Gated Recurrent Unit. 53

L

LagLasso Variant of Lasso, which includes the selection of features and lags. 9, 34, 94, 98, 105, 140

Lasso Least Absolute Shrinkage and Selection Operator. iv, xxvi, 9, 31, 33, 34, 75, 76, 94, 97–99, 104, 139, 140

Libor London InterBank Offered Rate. 16

Logit Logistic Model. 17, 24

LSTM Long Short-Term Memory. iii, iv, x, xi, xx, xxv, 2, 6, 9, 10, 30, 42, 47, 48, 51–55, 69, 93–102, 104, 105, 107–110, 117, 139, 140, 142

LSTMVis Long Short-Term Memory visualisation tool. 93

M

MA Moving Average. x, xi, 12, 106, 107, 146, 149

MACD Moving Average Convergence Divergence. 17

MAE Mean Absolute Error. 80, 86

MAPE Mean Absolute Percentage Error. xi, 56, 80, 84, 86

MDP Markov Decision Process. 111, 112

MLP MultiLayer Perceptron. iii, iv, ix–xi, 5, 8–10, 16, 18, 24, 30, 35, 36, 43–45, 52, 69, 71, 74–77, 84–88, 90–92, 95–97, 99–102, 109, 117, 138, 139

MSE Mean Squared Error. 80, 86, 87, 91, 100, 108

MTL Multitask Learning. xi, xix, 11, 38–41, 54, 55, 69, 71, 75, 76, 82, 84, 89, 92

N

NS Nelson–Siegel model. 13, 37, 68

O

OECD Organisation for Economic Co-operation and Development. 108

P

PMI Purchasing Managers Index. 16, 145

PPO Proximal Policy Optimization. xi, 64, 65, 122, 142

Probit Probit Model. 24

R

R&D Research & Development. 28

RBF Radial Basis Functions. 35, 69

ReLU Rectified Linear Unit (activation function). 36, 126, 127

RF Random Forests. 21, 22, 69

RL Reinforcement Learning. iii, iv, ix, xi, 6, 7, 9–11, 55, 56, 58–67, 70, 111, 112, 114–123, 125, 127–132, 134, 135, 138, 140, 141, 151

RMSE Root Mean Squared Error. 80

RNN Recurrent Neural Network. 5, 6, 9, 24, 25, 30, 35, 42–47, 51–54, 69, 93, 94

S

S&P 500 Standard & Poor's US Stock Market Index. 21, 22, 63, 64, 66, 94

SAC Soft Actor-Critic. xi, 64, 122, 142

SARSA State-Action-Reward-State-Action. xi, 122

SPG Stochastic Policy Gradient. 64

SVM Support Vector Machines. 20–25, 69

SVR Support Vector Regression. 17, 18

T

TD3 Twin Delayed DDPG. xi, 64, 122, 123, 142

TDQN Trading DQN. 63

TRPO Trust Region Policy Optimisation. 64

V

VIX Volatility Index (US). 16, 145

X

X-Rate Exchange Rate. 145

Chapter 1

Introduction

In this introductory chapter, we will present an overview of the motivations to conduct this research, we detail the research objectives, hypotheses and contributions that were made. We finalise with the structure of the thesis.

1.1 Motivation and challenges

In this section, we detail the motivations for embracing this research incorporating four key themes that make the title of this thesis, namely: fixed income markets, machine learning, forecasting and portfolio management. We outline the motivations and challenges of each area.

1.1.1 Fixed income markets

Fixed income is one of the segments of financial markets, where debt instruments are traded among market participants. Bonds represent the most important debt instrument and is the focus of this research, but this market is much wider and encompasses an extensive number of assets such as treasury bills, commercial paper, certificate of deposits, preferred stocks and structured products. In turn, bonds are financial assets representing loans between an investor and a borrower, for a specified period of time and interest rate, which can be traded in the fixed income marketplace.

In this type of instrument, investors have a “fixed” income in the sense that the obligations from the borrower are specified upfront in the conditions of the instrument, specifically: the interest rate (which may be fixed, floating or variable); the timing of payments; and the return of capital invested. In other words, the cash flow from this financial instrument is fixed a priori. This contrasts with “variable” income associated to equity instruments, where there is no obligation to pay a certain dividend or assure any fixed or otherwise return to investors.

Importance of fixed income markets

The bond market, in particular the government sector, plays a fundamental role in the overall functioning of the economy and is of paramount importance for financial markets. This is the case as an asset class by itself, with an overall size of USD 102.0 trillion, as of 31-Dec-2016 (Bloomberg, 2017a), which compares to a global equity market of USD 66.3 trillion. But it is also because the valuation methods of other asset classes often depend on bond yields as input information, especially for equities and corporate bond yields, as well as in pricing derivatives. In addition, bonds being fixed income securities form significant components of the portfolios held by pension funds and insurance companies (OECD, 2015a) whose investment interests tend to be long term.

Despite the importance of this asset class in financial markets, empirical work on bond yields is significantly lower than the corresponding work on equity and index prices (Booth et al., 2014a; Ballings et al., 2015; Fischer and Krauss, 2018; Kraus and Feuerriegel, 2017; Qin et al., 2017; Sermpinis et al., 2019) or foreign exchange rates (Gradojevic and Yang, 2006; Choudhry et al., 2012; Sermpinis et al., 2013; Hassanniakalager et al., 2021). When we move to more complex deep learning models, almost no scientific publications have been found covering yield forecasting with the long short-term memory networks – LSTMs (for transparency, we found parallel and independent work conducted by Ying et al. (2019)).

To conclude, the societal value of this research is directly related to the importance of bonds in pension funds and insurance portfolio holdings. However, several sectors of society and industry could benefit directly from this research, in particular: central banks, the asset and risk management industries, and academia.

Present market conditions

In recent years and especially after the last global financial crisis, financial markets have been under a specially dangerous combination of three factors. First, on the financial markets side, we have witnessed declining yields in fixed income markets for four decades. As a result, yield levels are extremely low, even with incursions into negative yields, which by its nature should tend to be limited both in magnitude and in time. Consequently, there is, potentially, a very high market risk for investors in the case of an inversion of the cycle.

Second, since the last global financial crisis of 2008-2009 (the so-called Great Recession), central banks around the world are taking unprecedented actions. This comprises strong conventional and also unconventional monetary policy. Central banks and the economy are clearly in uncharted territory and, as a result, it is no surprise that conventional thinking and models have been challenged by the financial crisis. Several studies are referred as indicating that while standard models are still leading to reasonable predictions, in some cases their forecasting ability has been diminished in the recent past (Gogas et al., 2015; Morell, 2018).

Third, given the new market conditions, in particular the very low yield environment, a generalised increase in risk exposure in investment portfolios has been observed as a result of a tendency to "search for yield" from investors (Mello, 2015; Kräussl et al., 2017). This has been observed both in pension funds' portfolios (OECD, 2015a,b), and in insurance companies (Becker and Ivashina, 2015), which represent the most important investors in fixed income markets. Insurance companies have capital requirements dependent on the credit ratings of their investment, discouraging excessive risk taking. However, conditional on ratings, the study showed that their portfolios are systematically biased towards bonds with higher yield and higher credit default swap (CDS) spreads, and therefore to higher levels of risk. Overall, theses two sectors have to meet returns promised when financial markets were yielding higher returns. The resulting increased exposure to risk is a dangerous path that may lead to higher levels of insolvencies which is a global concern (OECD, 2015a).

In summary, fixed income markets are presently operating under very special circumstances in historical terms: higher market risk (potential inversion of cycle); higher levels of risk in investment portfolios; higher levels of uncertainty and lower prediction ability of conventional models and tools used for policy making and asset management. Consequently, the study of this asset class gains special relevance in the present moment for all those intervening in the financial industry and respective regulatory bodies.

Focus of research and challenges

In our research we focus on forecasting, individual yields and the yield curve, and also on bond portfolio management. Taking government bonds as an example, the yield curve represents the annualised interest rates (or "yield") that a particular government has to pay to borrow funds from investors, as a function of the length of time in which the borrowing occurs (or "time to maturity"). It is also known as the term structure of interest rates. In fact, the yield curve is the centrepiece in bond markets and the overall economy. It is used by borrowers, investors and central banks. In fact, central banks use short-term rates' fixing, among other instruments at their disposal, to fulfil its mission of maintaining global monetary and financial stability.

Forecasting yields is a particularly challenging subject due to the complex and dynamic nature of financial markets, and the number of factors that may influence the pricing of securities. Both the market dynamics and participants reflect nonlinear behaviour and evolve over time.

1.1.2 Machine learning

The developments in the last thirty years in the computer industry and the widespread availability of powerful computers and networks have contributed to a parallel evolution in sophisticated modelling techniques that have been introduced successfully in several scientific fields. Among those techniques, machine learning has evolved as a subfield of computer science and artificial intelligence and has been applied in biology, medicine, engineering and other scientific domains. Its use is widespread nowadays in web search engines, online shopping, social and professional networking, text and speech recognition, and high profile projects such as self-driving cars and the understanding of the human genome. Machine learning tools are also readily available in free and open-source software such as Python and *R*. More importantly, machine learning models have become more efficient and powerful following far-reaching new developments in deep learning and the diverse number of architectures under this umbrella. Hence, these models seem more adequate and capable of modelling fixed income markets now than in the past and for this reason they are the main focus of this research.

Additionally, one has to take into account the interest from the industry. It is evident that machine learning has been used in finance by companies such as Bloomberg, the financial software, data and media company (Bloomberg, 2017b) and other financial institutions such as banks, asset management companies and hedge funds (Kolanovic and Krishnamachari, 2017; Petropoulos et al., 2022). In this industry, however, systems and applications tend to be proprietary and do not result in publications available to the scientific community. The development of such systems are time-consuming, resources and financially demanding and can be an important source of revenue for the company. Therefore, they do not want to make them available to third parties in this competitive world.

The range of applications of machine learning to finance is wide and include (Hambly et al., 2021): optimal execution, portfolio optimisation, option pricing and hedging, market making, robo-advising, and smart order routing. Regarding one of those applications, the introduction of robo-advisors (platforms using automated algorithms for financial advice to investors), has been considered one of the important contributions of the fourth industrial revolution (Tao et al., 2021). The authors found that on average, robo-advisors demonstrate superior risk-adjusted performance in comparison to conventional mutual funds.

In sharp contrast to equities and foreign exchange (forex) rates, it is clear that fixed income markets are in the early stages of both quantitative investing and electronic market-making (Wigglesworth and Fletcher, 2019; Gutscher et al., 2019). And the fast developments on both sides may result in what some characterise as a bond market revolution (Wigglesworth and Fletcher, 2021).

Moreover, there are several examples of very successful hedge funds working with black-box type of systems. One of the most famous that could be mentioned for its long standing success is Medallion, the flagship fund managed by Renaissance Technologies, founded in 1982 by American mathematician James Harris Simons. The company was an early adopter of systematic, quantitative trading strategies in futures, currencies and equities. At the core of Medallion was a proprietary high frequency futures trading system developed in the late 1980s (FRM, 2002). In the early 2000s, before closing the fund to Renaissance employees and their families only, the fund was charging astonishing fixed management fees of 5% plus a performance fee of 45% on new profits. Despite this high level of fees, the fund went on to become one of the most successful in history (Burton, 2016; Roux and Burton, 2017).

1.1.3 Forecasting

Time series forecasting has attracted much interest in the literature on empirical finance. Financial time series arise from a complex system in which the interactions between the underlying economic conditions, flow of information (Atkins et al., 2018) and investor behaviour determine the prices of various financial instruments. Such complex settings lead to nonlinear relationships between covariates and non-stationary variations of such prices over time. Hence several tools of statistical signal analysis and time series modelling have been developed around the challenging forecasting problems.

Artificial neural networks (ANNs) are a particular form of nonlinear function approximation methods that have been applied to financial time series analysis, among a wide range of other applications. While there is some motivation behind neural networks that arise from a biological analogy, the capability to model nonlinear relationships, cast in a probabilistic framework along with rich optimisation algorithms, make this class of models appealing tools. The most popular ANN is the multilayer perceptron (MLP), also known as a feed forward network, which implements a static mapping between input covariates and a target to predict. An alternate architecture with more favourable capabilities for modelling in dynamic environments is the recurrent neural network (RNN) which includes feedback of its internal states over time, *i.e.* among its inputs we include time delayed versions of its internal states. Such architectures have been used with great success in various applications in finance (Hutchinson et al., 1994; Niranjan, 1996). However, there are situations in which the use of a basic RNN is problematic, because its forgetting behaviour is exponential in time. This is when one needs to strike a differential balance on how far back in time one needs to remember in modelling and forecasting. This long-short memory compromise led to the LSTM architecture, currently very popular in sequence modelling tasks such as signal processing and language modelling, and forms the focus of attention in our forecasting study.

Despite successful empirical performance shown in a range of tasks, ANN or RNN models are often criticized for being black-box models with no explanatory power. After all, the purpose of modelling is not merely to make predictions, but to be able to make some sensible statement about the underlying physical system from which data being modelled arose. For financial data, decoupling the effects of various forces that determine traded prices would be that ultimate explanation. Our work is a step in that direction. Overall, the extraction of information through internal signals is a challenging task, without obvious positive results in the literature (Giles et al., 2001; Persio and Honchar, 2017).

1.1.4 Portfolio Management

Dynamic portfolio management is a sequential-decision making process in which an investor or portfolio manager allocates the available capital to invest, over time, among a universe of possible financial assets. This allocation is optimised to meet the investor's objective for a specified period. Transitioning to RL, this involves the agent acting upon the knowledge of the environment, specifying the allocation that optimises a specific objective function, over time. Portfolio optimisation is one of the most studied problems in finance (Haugh and Lo, 2001), due to the challenges involved which are truly motivating for researchers, and because it is a fundamental activity within asset management.

Some of the first problems in portfolio management with RL is that the environment is only partially observed, it is non-stationary and regime dependent, and is influenced by a multiplicity of factors. For this reason, feeding the agent with the best possible representation of the environment is a difficult task. Being a multi-asset type of problem, it is more demanding computationally than, for example, trading of an individual asset. In fact, with one asset it is easier to discretise the actions of the agent, although this is a limiting solution when compared to the continuous alternative (we will discuss this in more detail in Section 5.3.1.1). In addition, there are real-world constraints that add complexity to the problem. For example, the consideration of transaction costs, investor taxes, allocation limits and investor preferences.

On the agent side, different types of algorithms have been developed recently (Mnih et al., 2015; Lillicrap et al., 2015). Although they have achieved in some cases outstanding success in applications such as games, their use in portfolio management has been less clear. In fact, some of the algorithms suffer from known problems of instability and divergence (Sutton and Barto (2020), and Section 2.7.5 for more detail). Our work aims to contribute to a better understanding of these systems.

Overall, this is the most challenging topic of this thesis, an area that has seen important recent developments due to parallel breakthroughs in deep learning, which have been subsequently incorporated in the latest RL algorithms. In other RL tasks, where the environment follows defined rules, e.g. in games, or are governed by physics laws, e.g. CartPole or MountainCar, the chances of success are higher. Despite the above, RL remains one of the most promising research areas for financial markets.

1.2 Objectives

While considerable attention is devoted to the use and development of techniques especially for equity markets, and also for forex markets, there is a significant gap in both the academic literature and the finance industry when it comes to the application of machine learning in fixed income markets. The aim of this research is to fill this gap.

The overall objective of this research is to demonstrate that machine learning can represent an edge in present financial markets and contribute to the development of appropriate models/methodologies that could be used in the industry and by policy makers.

Given the importance of the yield curve to this market and to the economy, this research will focus on ways to improve the body of knowledge in this particular topic. Specifically, the objectives of this thesis are:

1. To consider a wide range of macroeconomic and financial time series and determine the most important explanatory variables through feature selection.
2. To assess a range of existing machine learning techniques, including standard and deep learning models, and evaluate their adequacy for fixed income forecasting, in particular for yield forecasting.
3. To determine the impact of additional financial markets and macroeconomic information on forecasting results.
4. To evaluate the benefits of multitask learning when compared to the transformation of the problem into independent single-output problems (single task learning).
5. To develop and test methodologies that could result in improved forecasting, such as combination of models, or adaptation to a hybrid type of model with inputs from other models.

6. To develop and test an autonomous system, using online learning, capable of performing fixed income portfolio management. In particular, the system should be able to allocate capital among a selected number of fixed income assets.

1.3 Hypotheses

The hypotheses to be tested are:

1. The use of machine learning to forecast the yield curve, combined with appropriate feature selection based on both macroeconomic and market variables, leads to better forecasting results when compared to the baseline linear regression model.
2. The use of multitask learning to forecast the yield curve, help hidden layers to learn better models and improve forecasting, when compared to single task learning.
3. It is possible to improve forecasting performance of machine learning models by combination of models, or adaptation to a hybrid type of model.
4. It is possible to build autonomous systems capable of performing fixed income portfolio management tasks, using online learning.

1.4 Contributions

The main innovative contribution of this research has been the extension of existing machine learning models to the fixed income asset class that has not been comprehensively and extensively covered using these techniques, as it is the case with other financial asset classes. Specifically, the contributions to current state of the art are described below, covering Objectives 1–6.

In our work in Chapter 3, we conduct the first comprehensive study on yield curve forecasting using MLPs and multitask learning, showing that MLPs can be used to forecast the yield curve (Objectives 2 and 4). We introduce a methodology which includes the following characteristics. We use a rigorous feature selection process, instead of relying on a pre-selection of individual features. This is a vital part of the methodology given that the most relevant features depend on both target yield to predict and forecasting horizon. In addition, we use a moving window of training data to incorporate the most recent information as it becomes available in financial markets, and the retraining of models at every time step for enhanced results. This methodology makes models more flexible to changing market conditions. Finally, our dynamic cross-validation technique presents a number of advantages and innovative aspects in comparison to some of the most popular and widely used techniques. First, it introduces diversity in the training dataset through the bootstrapped samples. Second, it ensures a fair comparison of models, by forecasting yields at the same dates. Third, it guarantees forward-looking forecasting (using only past data to predict future data, unknown to the model). Fourth, it avoids any type of overlapping between training data and data used for calculating the forecasting errors, by using testing data to determine the metrics. Fifth and last, it enables a direct calculation of out-of-sample forecasting errors (Objectives 1 and 3). We expand existing methodologies by including synthetic data generated by the linear regression model in our neural network models, and show that they outperform models where this data is not included. This is relevant for the possibility of using hybrid models incorporating data generated by industry-established models (Objective 5).

Using the application of LSTM neural networks to bond yield prediction (Chapter 4), our work makes two important contributions. First, from the point of view of finance, our study is among the first to explore RNNs in bond yield forecasting, and, by doing a comparison with MLPs with various inputs and prediction horizons on the same tasks, we are able to identify regions in which the advantages of recurrent modelling is helpful (Objective 2). Second, from the neural network point of view, we contribute to a particular analysis of explanations of hidden unit representations, by extracting the internal signals and explaining them by regressing on covariates of putative relevance. For this purpose, we develop a new methodology which we refer to as *LSTM-LagLasso*, which is based on both Lasso (Tibshirani, 1996) and LagLasso (Mahler, 2009). Applied to our setting, this sparsity-inducing regression approach is able to identify economic variables that are most relevant for the individual internal signals of the LSTM. Such "peeping into" what the model learns is bound to have broader use in nonlinear and non-stationary signal processing than the particular financial application in which we develop it (Objectives 1 and 2).

Finally, in the RL work (Chapter 5) we develop an autonomous RL system using an environment created specifically for fixed income portfolio management (Objective 6). The methodology had to adopt a number of modifications in relation to studies in the literature. This included the scaling of rewards which was identified in the literature but not given proper attention in subsequent research. Our work is among a reduced number of studies testing the state-of-the-art deep deterministic policy gradient (DDPG) algorithm in portfolio management. With our work, we confirm the potential of RL for fixed income portfolio management. In fact, we show that even when the algorithm is not stable, it is possible to select agents during the learning process, capable of outperforming both static buy and hold strategy and the best asset in the portfolio.

1.5 Structure of report

This thesis is structured in six chapters, the first being the present introduction. In Chapter 2, the relevant literature review is presented, with special focus on studies using machine learning both for financial applications in general and specifically for fixed income markets. Then, the theory behind the selected machine learning models used in our research and the multitask learning methodologies are described. In Chapter 3, the empirical research carried out with MLPs and multitask learning for yield curve forecasting is presented. It includes the collection of macroeconomic and market variables data, with full description of the dataset used and pre-modelling operations, together with the results from the linear regression and MLP models (in single task and multitask learning mode). In Chapter 4, we present the study conducted with the LSTM networks, which has three main components: bond yield forecasting using LSTMs and comparison to the previous work with MLPs (Chapter 3); extraction of internal signals within the LSTM cell; and explanation of the states' signals using exogenous information. Finally, in Chapter 5, we present the work carried out on RL. It comprises the mathematical formulation of the portfolio management problem, the data used, all aspects of the methodology developed bearing in mind our target asset class (bonds), together with the presentation and discussion of results. Finally, in Chapter 6, a summary of the main conclusions is outlined, as well as potential themes for future work.

Chapter 2

Literature Review

The literature review will cover a number of topics. First, classical financial models for time series and yield curve models are briefly presented (Section 2.1), with relevance for Objectives 1 and 5 (Section 1.2). Second, a review of the literature using machine learning specifically for fixed income markets is carried out (Section 2.2.1). Given that this literature is limited, the review was then extended to adjacent areas in financial markets (Sections 2.2.2 to 2.2.5), with this part of the literature review organised essentially by asset class. This review is particularly important for Objectives 1–3. For clarity, at the end of each section a brief link between the studies covered and the main objectives of our research is outlined. Third, some methodologies for combination of models are very briefly presented in Section 2.3. This was not a specific topic of this review, but a result of the broader literature review. This subject is presented due to its potential interest for Objectives 2, 5 and 6. Fourth, the standard and deep learning models used in this research and the multitask learning (MTL) methodology will be covered (Sections 2.4, 2.5 and 2.6). These three topics are linked to Objectives 2 and 4. Finally, in Section 2.7 we introduce reinforcement learning (RL) to address Objective 6. We describe the concept and components of an RL system, contrasting with supervised learning and traditional industry methods for portfolio management.

2.1 Classical financial modelling

In this section, classical financial models are presented, including all models that may still be considered within statistics, leaving those on the machine learning field to be covered separately in this chapter. Although these models are clearly not the main focus of our research, they are important for background, because they are still relevant and used in the industry, and some of those models use additional macroeconomic variables that are relevant information for our research.

2.1.1 Time series models

Time series modelling has been a well established subject for many years and there are many comprehensive textbooks available (Hamilton, 1994; Enders, 2014; Box et al., 2015). Since these models will not be the principal focus of our research a very brief listing of the main methods available is presented below.

The simplest model for stationary time series is the autoregressive (AR) model, where the predicted values correspond to weighted combinations of past values plus a noise component. In other words, the model regresses on itself, on the same variable, although at different time steps.

Also for stationary time series, there is the moving average (MA) model in which the future value depends on the error values from prior periods plus a noise component. This model combines the moving average concept and the random walk, in which the best prediction is the last available value plus a random error term. Nonetheless, this model is not commonly used.

The combining of these two processes (AR and MA), led to one of the most popular models for time series analysis and forecasting, the autoregressive moving average (ARMA) model, which uses pure statistical methods for modelling and forecasting future values (Box and Jenkins, 1968). In this case, the available data for the model are also current and past observations of the variable (univariate model). The model assumes persistence or autocorrelation in the movement of the time series, with the autocorrelation referring to the correlation between current and lagged value. In financial time series it is a positive autocorrelation, i.e. when prices (assuming for example a prices series) have been moving up, they will tend to continue moving up and vice-versa. Further, the model comprises an AR component and a MA component, reason why it is usually referred to as ARMA(p, q) model, where p is the order of the AR part and q is the order of the MA part. The order refers to the number of prior values used for the autoregression and for the moving average.

Additionally, an autoregressive integrated moving average (ARIMA) model is a generalization of an ARMA model and it may be applied in cases where data show evidence of homogeneous non-stationarity (Box and Jenkins, 1968). In fact, the first stage with this model is to transform the time series by differencing in relation to previous values, until it becomes stationary, thus removing trend and seasonal components. Non-seasonal ARIMA models are generally referred to as ARIMA(p, d, q) model, where p is the order of the AR component, d is the degree of differencing needed to achieve stationarity, and q is the order of the moving average component. In practice, when a time series is stationary the ARIMA model becomes ARMA model.

When heteroskedasticity is present, the variance of the errors is not constant in time, a necessary assumption for linear regression. This is common in financial time series. In these cases, the variance of the error terms is modelled using the autoregressive with conditional heteroskedasticity (ARCH) model (Engle, 1982) or the generalised ARCH (GARCH) model (Bollerslev, 1986). These models are very useful to analyse and forecast volatility.

More complex econometric modelling and forecasting seek to introduce additional independent variables to the problem that may explain the dependent variable. Those variables should be a result of the overall understanding of economic theory and factors that may influence the result of the dependent variable. In other words, there should be economic reasoning behind the selection of model structure and independent variables to be included.

2.1.2 Yield curve models

In the previous section, several models for generic time series were covered. In this section, models for the complete yield curve are outlined, for which a different approach is needed. There are two main groups of models. The first, "yields-only models" use only yield data to estimate the complete yield curve. Two of the most widely used models within this category are polynomial functions and Nelson–Siegel (NS) functions (Nelson and Siegel, 1987). The second group, "yields-macro models" predict specified macroeconomic variables using the yield curve or vice-versa, i.e. predict the shape of the yield curve using macroeconomic variables.

Most of these yields-macro models assume that the influence happens only in one-direction, i.e. macroeconomic variables affecting the yield curve or vice-versa, but without feedback effects. However, other models have been developed to study the possible feedback effects (Diebold and Li, 2006; Diebold et al., 2006; Diebold and Rudebusch, 2013). Departing from the Nelson and Siegel curve, they developed a yield curve model incorporating both intrinsic yield factors (level, slope, and curvature) and macroeconomic factors (manufacturing capacity utilization, change of consumer price index (CPI) over the past 12 months or annual price inflation and federal funds rate). This model is generally known as the Dynamic Nelson–Siegel model (DNS).

On these two types of model, the time series estimates of level, slope, and curvature factors obtained with the yields-macro model were found to be very similar to the ones obtained with the yields-only model, fitting well US Treasuries data over the period January 1972 to December 2000 (Diebold et al., 2006). Regarding the dynamic interactions between macroeconomics and the yield curve, the authors found strong evidence of macroeconomic effects on the future yield curve, but, on the other hand, weaker evidence of yield curve effects on future macroeconomic developments.

Other studies extended the geographical application of this yields-macro model to estimate government bond yield curves of the US, Japan and Germany (Tam and Yu, 2008). The results confirm the dynamic interaction between yield curve latent factors and the macroeconomic variables for two of the three bond markets analysed, US and Germany. For the Japanese bond markets the insignificant interaction between yield curve factors and macro-variables is justified by the authors with the ineffectiveness of monetary policy in a period of prolonged depression.

An alternative for yield curve modelling can be found in an emerging area in statistics called functional data analysis. This is a nonparametric statistical technique dealing with infinite-dimensional data as in the case of functions, curves, surfaces and images. This type of models has been applied to forecast the US yield curve, where the yield curve is considered the functional variable (curve) that links maturities to yields (Caldeira and Torrent, 2017). The dataset used comprised end-of-month yield curves for the US zero-coupon bond. The benchmarks for comparison are well-known parametric estimators, most of them already covered in the present and previous sections (Sections 2.1.1 and 2.1.2): linear models (random walk, autoregressive models, vector autoregressions) and dynamic factor models (DNS model). The findings of this research produced mixed results, not demonstrating a systematic superiority of functional data analysis, although that was the case for forecasting short-maturity interest rates. Besides, the study did not include any macro or financial data apart from the yield curve itself, which could also be a limitation.

2.1.3 Limitations of classical financial modelling

Financial modelling and forecasting of financial assets is a non-trivial exercise considering the markets' complexity, its dynamics, the number of factors affecting valuations with varying degrees of importance over time, and the behaviour of market participants. For this purpose, there is some evidence in the literature that classical models considering exclusively past values of the variable to predict have limitations. To start with, autocorrelation is the basic assumption in some of the previously presented models, in particular the AR, ARMA and ARIMA models. However, in a comprehensive literature review of the characterisation of financial time series (Sewell, 2011), the author clearly concludes that the autocorrelation of price changes, or returns, is largely insignificant, presenting results based on twenty financial time series.

Furthermore, excluding other potentially strong explanatory variables that could also be considered into a model may represent a limitation of the model by itself. This is also emphasised by the fact that several studies in the literature refer that additional domain-specific features could improve forecasting ability (Dunis and Morrison, 2007; Mettenheim and Breitner, 2010, 2011). In fact, recent studies also reinforce that in the literature approaching the yield curve from an empirical perspective. One of the prevalent lines of research latterly has been through data-driven approaches using expert systems and knowledge-discovery techniques. This tendency is motivated by the need to address the limitations of previous econometric models (Diaz et al., 2016). Overall, machine learning models make fewer assumptions about the mapping function, being more flexible and capable of learning the complex nonlinear relationships necessary to model financial markets.

2.2 Machine learning models in financial applications

In this section, a review of the literature using machine learning for financial markets is presented, with particular relevance for Objectives 1–3 and 5 (Section 1.2) covering the type of features, their impact in model predictions, type of models and methodologies to improve forecasting. The section starts by reporting on studies focusing specifically on fixed income markets (Section 2.2.1). Then, the review is extended to other areas of research in financial markets, namely: forex, equities, options and other applications that could not fit directly into the previous asset classes (Sections 2.2.2 to 2.2.5). Finally, we briefly present the topics of asset pricing with factor models and double machine learning in separate sections given their importance and weight in the literature as applications of machine learning in finance.

In these areas adjacent to fixed income only a small number of studies is selected from each asset class. This is because first, these classes are not the main focus of our research, second, different assets classes have their own specific issues and are affected by different factors, and third, they have their own pricing methodologies not transferable to other asset classes. As a result, particular attention is given to the following aspects: methodologies and models that have demonstrated promising results in previous research works; use of ensembles; type of variables and general problems encountered in previous research, such as overfitting of models; and to areas that could be exploited in our research.

2.2.1 Fixed income

Studies applied specifically to fixed income markets are much less common in the literature. In this market, we can model individual assets, such as bonds, bond indices, bond funds or bond futures. In this case the datasets for this purpose are time series and this is a single target regression problem. However, the main focus of our research is the yield curve. This is a more complex issue because we have a modelling target which is a curve and not a single value. In fact, this is not the one-dimension type of problem we had before, but a two-dimensional one: time and maturity. With interest rates data in a two-dimensional space, modelling and forecasting is additionally challenging. One of the objectives of this literature review is to find a solution for this.

Along this line, a study was conducted (Kanevski et al., 2008; Kanevski and Timonin, 2010) using spacial statistics, to map the yield curves into a two-dimensional space (maturity and time). Then via interpolation, using geostatistical or machine learning models, the authors reconstruct full yield curves from a specified number of points considered in the data as inputs. In this case, the points of the curve selected included 13 maturities: Libor rates - 1 week, 1, 2, 3, 6 and 9 months; and swap rates - 1, 2, 3, 4, 5, 7 and 10 years. At this stage, after reconstruction of the yield curves, a full representation of the yield "surface" against maturity and time is available. Additionally, by using extrapolation techniques this methodology permits forecasting of the yield curve into the future. Promising results were obtained with this methodology using ANNs, although additional simulations are necessary under different market conditions and time horizons, to validate this methodology. In this study, two areas for further improvement were identified, which are relevant to our research. First, the development and retraining of models performed within a flexible moving window. The advantage is that this moving window will take into account the evolution of the market (different market conditions). Second, the assessment of hybrid approaches combining machine learning and geostatistical models that have been applied successfully with spatial environmental data.

A different approach is needed when modelling individual assets, since we do not have the additional maturity dimension. An example of this type of modelling was carried out to evaluate the performance of "artificial intelligence" and classical algorithms for forecasting one individual asset, in this case the monthly yield of the US 10-year Treasury bond (Castellani and Santos, 2006). The "artificial intelligence" approaches considered were: manually built fuzzy logic model, machine learned fuzzy logic model, a self-organising map model and neural networks (an MLP). For comparison purposes two models were taken into account: the statistical ARIMA model and an econometric error correction model. The data used was a complete series of monthly US 10-year Treasury bond yields ranging from January-1986 to December-2004 (dependent variable) and four economic indicators (independent variables), namely: purchasing managers index (PMI), CPI, Libor and the volatility index (VIX). The results of this study were not very encouraging. First, the best results were obtained by the ARIMA model, the econometric model and the neural networks, with the econometric model slightly better than the other two models. Second, in terms of the prediction accuracy the best models are only marginally better than a basic one-step lag predictor, which predicts the yield of the US 10-year Treasury bond from the figure of the previous month. The study concludes pointing out the difficulty in building reliable predictors for financial markets in general. However, it also identifies the sparseness of the data as a possible problem, due to the fact that they used a monthly series of the 10-year US Treasuries, with a total of only 216 data points available for modelling. Finally, and with potential relevance for Objective 5 of our research, on other type of models, Castellani and Santos (2006) suggest a combination of statistical or machine learning models with economic theory and expert knowledge of the financial markets to improve forecasting results; a second alternative suggested is the combination of different predictors into a final forecast.

With different type of results when compared to the previous study, another single-asset research was conducted by Dunis and Morrison (2007). By using state space modelling with a Kalman filter and neural network regression, they aimed to forecast the 10-year yield government bond of three countries: United Kingdom, United States and Germany. The performance of these models was compared with more traditional methods to serve as benchmarks, specifically: moving average convergence divergence (MACD), ARMA and a logistic binary estimation model (Logit). There are two interesting aspects in this study. First, they included a wide range of additional financial variables from main European countries, the United States and Japan, to work as features: bond yields, short-term interest rates, index stock prices, exchange rates and commodities. Second, the performance evaluation of the models was based on measures of accuracy, and also on results from a simulated trading strategy, with proper consideration of trading costs. For the period considered in this research, April 2001 to June 2003, the neural network regression showed the best results and a more consistent outperformance of the benchmarks in both performance measures. For the Kalman filter model the results were mixed, with outperformance using accuracy but not in the trading strategy performance. The authors concluded that neural network regression models represent a promising alternative to more traditional techniques currently used in the industry.

From the studies covering directly fixed income, it was not possible to find a direct solution for modelling the yield curve using machine learning. A notable exception is the work carried out by Sambasivan and Das (2017) proposing a dynamic Gaussian process for modelling the yield curve. In this work, the authors compare the results of this machine learning model with multivariate time series forecasting (vector autoregressive model) and the DNS model. The results show that multivariate time series method performed best for yields with maturities up to 1 year, while the dynamic Gaussian process model was superior for the longer maturities (2 to 30 years). These results will be mentioned again in Section 3.3.2.4 for comparison purposes.

Other publications cover different targets within fixed income, such as corporate bonds or bond spreads. Although they fall outside the objectives of our study of forecasting government yields and government yield curves, those studies represent an important part of the research using machine learning in fixed income. In this vein, Fernandes et al. (2019) conduct a study to forecast daily and weekly long-term government bond spreads from five European countries, using support vector regression (SVR) models. For their parameterisation process, the authors explore heuristic and metaheuristic algorithms. Apart from that, they use several linear and nonlinear models to create a large pool of individual predictors, which are then subject to dimensionality reduction techniques (in this case, principal component analysis), to reduce the number of inputs. The resulting best predictors are then combined with the aim of achieving superior out-of-sample statistical performance. Overall, the results indicate that this technique produces strong performance for this type of forecasting problem, and the metaheuristic calibration of the SVR parameters lead to better forecasting performance. In addition, the combination of predictors from individual models improves accuracy. This aspect will be further covered in Section 2.3.

To summarize, the details described in these studies are important for Objectives 1–3 and 5 of our research (Section 1.2). In fact, the studies were particularly fertile regarding ideas for the latter objective and for potentially better models: from the use of ensembles to different types of hybrid models (Castellani and Santos, 2006; Kanevski et al., 2008; Kanevski and Timonin, 2010), with inclusion of broad information from several sources (Dunis and Morrison, 2007). Among those they should incorporate macroeconomic, financial and, whenever possible, practitioner type of information. Regarding Objective 2, mixed information and results were seen (Castellani and Santos, 2006; Dunis and Morrison, 2007) while using the same type of models, in this particular case, neural networks (MLP). From the study with negative prediction results, another important conclusion is that insufficient amounts of data may lead to poor performance of models and to minimum differentiation among them.

Overall, the studies covered did not provide a solution for modelling the yield curve using machine learning, with the exception of the work mentioned above (Sambasivan and Das, 2017). Notwithstanding the potential of spatial / geostatistical models and possibly of hybrid approaches combining them with machine learning models, they base the whole mapping and forecasting processes in intra and extrapolation using historical yield curve data only. For this reason, we perceive machine learning models with greater potential and flexibility. Furthermore, our research has the objective of going beyond the use of historical data from the variable to predict. In particular, Objectives 1 and 3 have in mind the assessment of a wide range of potential explanatory variables from a variety of fields. This is a gap in terms of academic research and we will proceed the literature review in adjacent asset classes pursuing a possible solution for the yield curve modelling.

2.2.2 Foreign exchange

In a recent study (Fletcher, 2012; Fletcher and Shawe-Taylor, 2013), machine learning techniques incorporating exogenous financial data from forex were investigated. This work concludes that machine learning models can be theoretically used to make effective predictions in currencies. In particular, they can be used for forecasting the direction of the movement of the currency pair euro / US dollar (EUR-USD) exchange rate, for periods of 5 to 200 seconds into the future, with accuracies ranging from 90%

to 53%, respectively. The research also demonstrated the possibility of forecasting turning points in the prices of a basket of currencies. In addition, the incorporation of domain knowledge from forex markets, for example market microstructure, to improve predictive ability was also covered. Market microstructure focuses on the real trading process, analysing how the different mechanisms affect prices, volume, trading costs and trading behaviour. The following two main conclusions were reached. First, the predictive accuracy improved significantly when models incorporated features based on commonly used trading rules. And second, the author points out that given the results obtained with very simple features, a better performance could be achieved by increasing the number of features translating common trading concepts.

Choudhry et al. (2012) also conducted a study to forecast forex rates using ANNs, based on high frequency market microstructure variables. The high frequency forex data on global inter-dealer electronic transactions covered three main exchange rates: JPY-USD (yen-US dollar), DEM-USD (German mark-US dollar) and USD-EUR (US dollar-euro). On the one hand, the dependent variable modelled was return, defined as the difference between consecutive log prices. On the other hand, the independent market microstructure variables considered were: previous return values; the last available value of high, low and average price; last available order flow and trading volume (here cumulative values are used, with positive or negative signs for buying or selling, respectively); and finally, last available bid and ask prices. The authors concluded that ANNs can be used to forecast exchange rates and lead to profitable strategies, with proper consideration of transaction costs. In fact, the results show a very significant outperformance versus the buy and hold benchmark in all cases. Further, apart from transaction prices, the most important microstructure variables identified were the immediately preceding bid and ask prices, which were shown to contain separate and additional information for the model.

Other studies have been carried out with different nuances using ANNs and market microstructure variables (Gradojevic and Yang, 2006; Huang et al., 2007). These are not detailed here, since this is not the main focus of our research.

To sum up, the aspects described in these studies are important for Objectives 1–3 of our research (Section 1.2). Within machine learning models neural networks have provided good results in forex markets and could be of interest also for fixed income. Although the data used in forex markets is in general high frequency data, it may be possible to use data with lower frequencies envisaged for our research, namely daily data. Besides, the inclusion of additional domain-specific features, in this case market microstructure variables, seem to be positive for performance. Furthermore, there are two important aspects implicit in the objectives of our research that are not answered by these studies. First, these models do not use additional macroeconomic variables, one of the objectives of our research. This is due to the fact that, for forex forecasting high frequency data is generally used in order to enable intraday predictions. In turn, macroeconomic data is released at a completely different and much lower frequency, not compatible with high frequency models. Second, this type of model does not cover multitarget modelling, necessary for the main focus of our research, the yield curve.

2.2.3 Equities

Studies covering equities are the most common applications of machine learning in financial markets. In this section only a small number of those studies will be briefly presented, starting with a couple of published state-of-the-art reviews in this area, for their broad scope. As mentioned before, at the end of the section, the link to our research objectives will be emphasised.

In the first review, Agrawal et al. (2013) present state-of-the-art techniques for stock prediction. It starts with a brief incursion into basic ratios for fundamental analysis and also technical analysis indicators. Then, and more relevant to our research, it continues with a summary from a collection of papers using a variety of quantitative methods for stock prediction: multinomial logistic regression, time series and chaotic time series forecasting, neural networks, including neural and neuro-fuzzy models, support vector machine (SVM) and a hybrid machine learning system based on both genetic algorithm and SVMs. The authors then conclude that *"On the basis of published and available literature, it can be safely concluded that the existing techniques are not suitable for prediction of stock market trends as well as price of different stocks"*. Nevertheless, they point out two methods for potentially improving results: the inclusion of political and economic factors that affect the stock markets, as input variables, in addition to the technical indicators most frequently used; the incorporation of market-specific domain knowledge into the system.

The second review covers the application of ANNs again in the field of stock market prediction (Vui et al., 2013). The study shows encouraging results with the use of this technique and points out two themes researchers are embracing in a quest to improve the accuracy of stock market predictions: the use of hybrid methods and the incorporation of additional external factors. These two reviews agree on the importance of additional information for better models.

Moving onto individual research studies, Arrieta-Ibarra and Lobato (2015) pointed out the inconclusiveness of most of the results presented in the literature, because they are mostly concerned with the comparison between different techniques without presenting statistical inference to assess the improvement in predictive ability. These researchers conducted a study using several machine learning models to forecast stock market daily returns and squared returns, considering three forecasting horizons: 1, 5 and 20 days. The data used was comprised of: series of past values of daily returns of the

S&P 500 index and the combination of those series plus additional economic/financial variables, specifically: interest rates (returns using the US 3-month treasury bill); commodities prices (gold price in dollar per troy ounce, London Bullion Market and crude oil prices in dollar per barrel, West Texas Intermediate); and exchange rates (yen, pound and Swiss franc against the dollar). The machine learning models tested were: regression trees, conditional regression trees, random forests (RF), neural networks and SVMs. In order to rank the out-of-sample forecasting performance of these methods and test their predictive ability, they used as benchmark the GARCH(1, 1). The results were not fully conclusive, but there are some positive indications for our research. First, for predicting returns, the results did not show any evidence that machine learning improves the predictive ability of the simple historical sample mean. Second, for predicting squared returns, neural networks and SVMs showed real potential for improving forecasting ability, with the latter improving on the results of the benchmark by about 10%. To conclude, this study encountered some problems that are worth mentioning, due to its relevance to our research: tendency to overfit in all machine learning procedures tested; necessity to assess cautiously the number of predictors to use, since the use of too many variables lead to overfitting and to larger standard errors; while the study was conducted with standard computing packages (R packages) using its default settings, they concluded that some improvements are still necessary before a strong case for the use of these techniques can be made.

Since forecasting is frequently connected to the trading activity that could directly benefit from superior sources of information, some research studies aim to integrate the forecasting models within a trading platform. In one of those studies, an automated trading system based on ensembles of random forests was developed (Booth et al., 2014b, 2015; Booth, 2016), predicting price returns of equities from the German stock market index (DAX) and integrating those forecasts in a three-layer trading system (random forest predictions, expert weighting and risk management). The study analysed specifically the possibility of taking advantage of three seasonality events described in the literature, namely: downward bias over weekends; upward biases over exchange holidays and at the turn-of-the-month. Some results of this research are most relevant for our research, in particular the methodologies adopted. First, to capture the state of the market, the system used a range of features: open, high, low and close values of the day before the seasonality events; a number of technical analysis indicators; the DAX index closing price; and a proprietary risk-based indicator based on the concept of value-at-risk. Features were then ranked and only the most important used. Second, all continuous input data was taken as logs and normalised using the closing price of the stock. Third, random forests were trained based on moving windows of training data and added to the ensemble in an online fashion. This has the advantage of adapting dynamically to changes in market conditions, being more appropriate for non-stationary financial time series. Additionally, the use of random forests has the advantage of not having the tendency to overfitting. Finally, regarding the ensembles used, the recency biased performance-weighted ensemble (exponential moving average of returns) was shown to produce the best results both in terms of profitability and prediction accuracy. Hence, the results achieved a 10% improvement in risk adjusted return, decreased drawdowns and prediction error. In addition, the ensemble results are significantly better than simple averaging (70% higher returns).

Other studies have also emphasised the benefits of using ensembles for forecasting stock price direction with classifiers (Ballings et al., 2015) and for sentiment analysis in social applications (Araque et al., 2017).

In some of these studies, the challenges and difficulties of forecasting stock markets is emphasised (Agrawal et al., 2013; Arrieta-Ibarra and Lobato, 2015). This is a concern that is also applicable to fixed income markets. Nevertheless, it should be stressed that suitable techniques for market forecasting may be developed. In fact, there is evidence that they have been used in the industry, as referred in Section 1.1.2.

Overall these studies are important for Objectives 1–3 (Section 1.2). In fact, although they are applied to another asset class, with different characteristics, important information can be taken from them regarding methodologies, type of features and models. On features, the benefits of including additional information from different sources as independent variables have been reported, including political, economic and financial factors. On the type of models, results achieved are sometimes mixed, even with the same type of machine learning model, and are not totally conclusive in their superiority versus common benchmarks. Nonetheless, ANN, SVM and RF have demonstrated potential in previous studies for forecasting equities. This could be a positive indication also for forecasting fixed income. Again, as in the previous section, possible solutions for modelling the yield curve are not evident in these studies.

2.2.4 Equity options

Other financial applications have been the object of study through the use of machine learning. This is the case in equity and future options, in the derivatives asset class (Hutchinson et al., 1994; Niranjan, 1996; Montesdeoca and Niranjan, 2016).

For pricing options, the traditional method and one of the most commonly used in the industry is the parametric Black-Scholes model (Black and Scholes, 1973). Hutchinson et al. (1994) demonstrated that this formula can be learnt by a radial basis function network when trained with data in which the call option prices (target variable) were calculated using the Black-Scholes formula. The technique was also applied to real data, daily call options on the S&P 500 futures, and the results compared with the Black-Scholes formula. The authors found that the radial basis function network outperforms the parametric model in many cases. The main advantages of this method are: the extra flexibility for cases when the Black-Scholes formula is not applicable given its assumptions; and the model is more robust being able to adapt to structural changes, something that a parametric model is not able to do.

Recent developments have been introduced into this model by including additional input variables capable of better explaining the option price (Montesdeoca and Niranjan, 2016). Hence, apart from the normalised asset price (stock price / strike price) and time to maturity, the additional inputs introduced were: volume traded, historical volatility, risk-free interest rates and combinations of these variables. The authors demonstrate that model accuracy is improved with the new features.

In summary, the aspects described in these studies are linked to Objectives 1–3 of our research work (Section 1.2). A further objective in this section was to briefly present another financial area where machine learning has been used. This is a very specific field within derivatives, but one where the type of features used and models may have interest for fixed income. From the studies covered, it emerges that radial basis functions demonstrated the capacity to model the complex relationship between option price and the underlying stock price. Additionally, the inclusion of financial information leads to better forecasting performance. This is a common theme in the literature for other asset classes as well. In our research the extra features included is broader, both financial and macroeconomic variables, bearing in mind the number of factors that could influence fixed income prices and yields. The type of model used for option pricing is also single target, not ideal for yield curve modelling.

2.2.5 Other financial applications

In the financial applications included in this section it was not possible to group them by asset class, the general rule followed so far. Two different research studies are presented. The first covers the prediction not of a financial asset, but rather the prediction of a macroeconomic variable. In the second, a different architecture of neural networks is used in three different financial applications, and for this reason not fitting directly in one of the asset classes considered in previous sections.

In the first study, machine learning frameworks were used to predict recessions in the United States, in what the authors claim to be first application of SVM using the information from the yield curve to forecast the gross domestic product (GDP) cycle (Gogas et al., 2015). In this study, short and long-term interest rates were used as independent variables (US Treasuries: 3 and 6-month bills and 2, 3, 5, 7 and 10-year bonds) and real GDP data for the dependent variable (real seasonally-adjusted US GDP, transformed into natural logarithms). The GDP series was decomposed into the cyclical and trend components using the Hodrick - Prescott filter (Hodrick and Prescott, 1997) and the cyclical component transformed into a binary variable (equal to +1 for cyclical component higher than zero and equal to -1 otherwise). Gogas et al. (2015) used an SVM model (with linear and radial basis function kernels), for classification of future economic activity into two outcomes of the dependent variable: recession or output gap, defined as GDP below the long-term trend of real GDP (class "-1"); and inflationary gap, defined as GDP above the long-term trend (class "+1"). The forecasting of the cyclical components was performed for 1, 2 and 3 quarters in the future. To avoid overfitting, the authors used in this case the k-fold cross-validation. The results were promising but not completely satisfactory, with the out-of-sample overall accuracy of 66.7%, predicting correctly all recession periods one quarter ahead, but at the same time predicting as recession 60% of the growth periods, clearly undesirable. The results from the SVM model was compared to econometric methods used in binary forecasting / classification, namely the Logit and Probit models. In both cases the overall accuracy was 50%, comparing unfavourably with the performance of the SVM model. The authors emphasised the importance of correctly predicting recession periods with this type of model for policy makers, governments and central banks, to adjust in advance fiscal and monetary policies to the GDP cycle. From the results it is clear that some further work will be needed.

In the second study, an RNN topology called shared layer perceptron was developed (Mettenheim, 2010; Mettenheim and Breitner, 2010, 2011). Despite its name, it is not a standard MLP and the network allows multi-asset and multi-step forecast. The authors have strong claims on the new architecture: consistent and robust performance over a period of eight years without the need for retraining, demonstrating superiority in relation to all benchmarks considered. The neural network considers not only the observable states (inputs) but additional hidden states, which enables the model to build memory. Further, by entering extra hidden states in the model, it is implicitly assuming that the input variables do not represent a complete set of explanatory variables. That is, it allows for uncertainty by incorporating these hidden states, which will also influence the model. At each time step the network produces as output all inputs necessary for the next time step forecast. On the one hand, this structure enables the forecasting of all inputs simultaneously; on the other hand, using the outputs it can follow on forecasting the next step ahead. The data collected was for a period of ten years, from 1999 to 2009, and used 25 time series, including equity indices, interest rates, forex rates and commodities. In the case of interest rates, a proxy was used corresponding to the slope of the yield curve between the maturities of 3 months and 10 years. Three applications were used for testing the new model.

First, for forecasting market value at risk, over the next 10 days, benchmarked against historical simulation. Second, for forecasting the economic indicator Baltic Dry Index, over the next 20 days, to identify a low entry point. This is benchmarked against the investment on a given fixed day over the 20-day time frame. Finally, for forecasting the sign of next day return of portfolios, the benchmarks being a simple naive and a moving average strategies. In the latter application, the results are reported to be particularly satisfactory for equities and currencies. As to the first two applications, the results of the shared layer perceptron were consistently better than those of the benchmarks.

To summarize, the aspects described in this section are linked to Objectives 1–3 of our research (Section 1.2). However, given the results described, the main link is to Objective 2, covering the assessment of machine learning models.

In the first study, the SVM model did not deliver satisfactory recession prediction results, even though they were better than those of the benchmarks used. There are two main fundamental differences in relation to our research for forecasting yield curves. First, the research focused on the prediction of recessions using exclusively several points of the yield curve, not including any other information. In our research a more comprehensive set of variables will be considered. Second, the forecasting problem was also converted into a classification problem, while in our research, for forecasting yield curves, we have essentially a regression problem.

In the second study, the benchmarks used in the three applications described above are simplistic, taking into account that the objective is to assess sophisticated neural network models. Nevertheless, the selected benchmarks may be acceptable from a passive investment point of view. The characteristics of the RNN/shared layer perceptron appear to indicate high potential for yield curve forecasting, given the multi-asset and multi-step characteristics. This directed our interest to the RNN family of models, which will be the focus of Section 2.6.

2.2.6 Asset pricing and factor models

The asset pricing theory started with Sharpe (1964), and the capital asset pricing model (CAPM). This is the first single-factor model developed. Early work by Fama and French (1989) on multifactor asset pricing point out to mounting evidence that stock and bond returns are predictable and show that the expected asset returns are directly dependent on economic conditions. Subsequent work by the same team, has produced seminal papers leading to the well-established three- and five-factor Fama and French (1993, 2015) models. There are two main approaches to factor modelling considering the type of regression used for estimating and testing the models: time-series or cross-sectional regression. However, since the pioneer work by Fama and French (1993), their proposed time-series methodology became dominant and generally used by academics and researchers (Jacobs and Levy, 2021).

Asset pricing has been one of the most studied fields in finance over the years, but recently with the incorporation of machine learning models they have gained a new dimension and it is another fundamental example of applications of machine learning to finance. A thorough survey of recent contributions in asset pricing using factor models is the work by Giglio et al. (2021), where they emphasise the importance of machine learning models for future empirical asset pricing research, appropriately combined with economic theory.

Reinforcing the importance of machine learning in asset pricing, Gu et al. (2020) conduct a broad study involving all stocks listed in three US stock exchanges, comparing a wide range of machine learning methods. In more detail, they include the following models: linear regression, generalized linear models with penalisation, regression trees (boosted regression trees and random forests), and neural networks. The latter is responsible for the best performance, with shallow networks outperforming deep networks, contrary to what happens in other scientific fields. Considering their results, they demonstrate large economic gains from using machine learning, in some cases doubling the performance of standard strategies, such as the Fama-French portfolios. Finally, regarding the most relevant factors, the models used point out to variables associated with price trends (return reversal and momentum), stock liquidity, stock volatility, and valuation ratios.

In this field of research but applied to fixed income, Bali et al. (2022) conduct a comprehensive study, focusing on cross-sectional techniques to forecast future corporate bond returns (one month ahead), using as inputs corporate bond and stock characteristics, also known as factors. Most of the studies for this purpose use linear methods. However, machine learning is a better alternative given the nonlinear payoffs of bonds in general and the high correlation between some of those bond and stock characteristics. Inspired by the Merton (1974) model, the authors explicitly linked the functional forms of corporate bond and stock expected returns. They conclude that using stock characteristics in addition to bond characteristics is beneficial for bond return forecasting. Moreover, imposing the Merton model dependence between expected bond and stock returns leads to better bond returns predictions, when compared to the models where that dependence is not taken into account (which the authors call the reduced-form approach). In order to determine the most relevant bond characteristics for each machine learning model, the authors follow the ranking methodology proposed in Kelly et al. (2019) and Gu et al. (2020). From their results, the most influential variables for the forecasting performance can be grouped as follows (Bali et al., 2022): interest rate risk (e.g. duration, time-to-maturity); downside risk (proxied for example by value-at-risk, expected shortfall, total return volatility); systematic risk (proxied by bond market beta, default beta and term beta); liquidity risk (e.g. bid-ask spread); and historical return characteristics such as bond momentum, short- and long-term reversals.

Since the CAPM model was created, over 300 different risk factors have been identified by researchers, in some cases resulting in new multifactor asset pricing models being developed (Maiti, 2021). To navigate the potential high dimensionality of factors that could be relevant to the forecasting process, Feng et al. (2020) proposed a model selection methodology to evaluate the contribution of each additional new factor we want to study, in relation to a pre-existing set of factors. Hence, new factors are considered only when they result in increased explanatory power for asset pricing. The authors concluded that if their methodology was applied recursively over time, only a small number of the factors identified in the literature would have been considered as significant.

Another example of recent research using machine learning in this area is the work of Zhao et al. (2019). The authors use the five-factor Fama and French (2015) model and explore their optimal portfolio allocation for factor investing. They propose a novel methodology comprising three main stages: First, obtain a large number of individual forecasts of the five Fama-French factors using linear and non-linear forecasting models. Second, combine the individual forecasts using Bayesian dynamic model averaging, which results in more robust and improved forecasting performance. Finally, determine the optimal portfolio allocations for each factor using a dynamic asymmetric copula model. Overall, the results from this study reveal factor investing as a robust asset allocation technique.

2.2.7 Double machine learning

Machine learning in finance is mainly devoted to forecasting tasks, while econometrics is more interested in causation and decision making. The study of causal relationships has become a hot research topic in artificial intelligence and it is believed that it may be an essential tool to address some limitations of correlation-based machine learning models (Nogueira et al., 2022). Double, or debiased, machine learning (Double ML), an evolution in relation to the double selection technique (Belloni et al., 2012), is a framework developed by Chernozhukov et al. (2018), applying machine learning to financial forecasting, planning and analysis. It enables statistical inference, thus going beyond forecasting to infer causal relationships. For this reason, it is at the intersection of machine learning and econometrics, taking advantage of the best of both worlds. More specifically, the main objective of Double ML is to produce valid inferential statements about the effect of a specific intervention (also called variable of interest, "policy" or "treatment", the latter imported from the use of these methods in medicine), on an outcome variable we are analysing, in the presence of a potentially high-dimensional set of control variables or confounders. These confounders are factors that influence both treatment and outcome variable.

Let us consider the simplest example of a partially linear regression model (Robinson, 1988; Chernozhukov et al., 2018):

$$Y = D\theta_0 + g_0(X) + U, \qquad \mathbb{E}[U \mid X, D] = 0 \tag{2.1}$$

$$D = m_0(X) + V, \qquad \mathbb{E}[V \mid X] = 0 \tag{2.2}$$

where Y is the outcome variable; D is the variable of interest or treatment; θ_0 is the parameter of interest or causal parameter, i.e. the main regression coefficient that we want to infer; g_0, m_0 are generic functions, which may be nonlinear; X is a set of control variables; U, V are disturbances or error terms; and $\mathbb{E}[.]$ denotes the expected value of a random variable.

Naive attempts to use machine learning for directly determining the parameter of interest (θ_0) result in biased estimations. There are two biases involved, namely: the regularization bias and the overfitting bias. Chernozhukov et al. (2018) propose two ingenious solutions to produce debiased estimations: using Neyman-orthogonal moments or scores to address the regularization bias; and data-splitting followed by cross-fitting to tackle the overfitting bias.

Briefly, the method can be described as follows. First, we regress the outcome variable on the control variables. Second, we regress the treatment variable on the control variables. Then we compute the residuals for both, outcome and treatment. Finally, to estimate the parameter of interest (θ_0) we regress the residuals of the outcome on the residuals of treatment. To address the overfitting bias, this process is conducted on subsamples of the original data and averaging the results of the estimated parameter of interest. This is the final estimate of the causal effect of the treatment variable on the outcome. In addition, Chernozhukov et al. (2018) verify that the resulting solution achieves root-N-consistency (Robinson, 1988), with N being the sample size, with values approximately unbiased and closely following a normal distribution. Consequently, it is possible to determine valid confidence intervals for the treatment effect estimates.

The application of Double ML in finance is still at very early stages. Nevertheless, there are some good examples of empirical applications to financial planning (Wasserbacher and Spindler, 2022), specifically to assess the effectiveness of marketing activities (treatment or intervention) in generating sales (outcome). The authors conclude that using the naive approach, instead of Double ML, would greatly overestimate the causal effect of the marketing activities on sales. This would represent a mismanagement of resources. We can also find applications of Double ML in public policy recommendations and regulatory oversight (Varaku, 2021), with multiple objectives: to study the effects of public subsidies on innovation, i.e. on companies' research & development (R&D) input and output; or the effects of public subsidies incentivising, or not, collaborative agreements among different organisations, on R&D output; the effect on the performance of microfinance institutions (outcome), of having a product range including only core microfinance products (loans and savings) or a more extended range including non-financial services (treatment); and the impacts of regulation on microfinance institutions' performance. Finally, we refer the work of Farbmacher et al. (2022), where the authors investigate the causal effect of health insurance coverage (treatment) on general health of selected individuals (outcome).

2.3 Combining predictors

Another approach to forecasting is the combination of various predictors, which tends to improve accuracy, reducing the bias and variance components of the prediction error. The traditional combination is carried out with predictions from different algorithms using model ensembling. Different techniques have been employed in the literature and several variants of the same technique can also be found (Ballings et al., 2015; Booth et al., 2015; Araque et al., 2017).

An alternative to the combination of different algorithms is to perform repeated subsampling of the dataset and run a single model on each subsample. This technique is commonly applied for predictive classification and creates diversity in the data rather than in the models used (Barrow and Crone, 2016). The most common methods used for combining similar algorithms are: bootstrap aggregating (or bagging), boosting and random forests.

Yet another approach to combine predictors is the use of cross-validation techniques. Along this line, Donate et al. (2013) built ensembles of neural networks using weighted k-fold cross-validation for the prediction of six real-world time series. The resulting accuracy was enhanced for short and medium time series. In a different study, Sorić and Lolić (2015) employed leave-h-out cross-validation in a study to forecast euro area inflation. The results from this study have shown that forecasting accuracy was improved when using this methodology, especially for longer forecasting horizons. For short horizons the improvement was not so evident.

A new type of ensembling method has been proposed in a study for time series forecasting by Oliveira and Torgo (2014), which departs from standard bagging. In this technique (bagging), diversity in the predictors is a result of the different random bootstrap samples. Considering the diversity of members as one of the most important drivers of the success of ensembles, and also aiming to deal with the diverse dynamic regimes and non-stationarities of time series, the authors introduce different forms of diversity related with properties of the time series. Specifically they used: different embed sizes (embed is the set of recent past values of a time series used in the model); additional variables related to the past data and that may explain the recent dynamics of the time series (in this particular case, they used the mean and variance); and additional models with combinations of the previous points. For the implementation bagged regression trees were used on fourteen different real-world time series, treated independently. The final prediction of each ensemble was calculated using the standard bagging approach of averaging the predictions of all models. The standard ARIMA model was also used for comparison purposes, but all performances were evaluated in relation to the standard bagging approach.

Results from this research showed that the models where additional diversity was introduced showed better prediction accuracy. In fact, the model that introduces more diversity is the best performing, always beating the standard bagging in terms of prediction accuracy, although not always with statistical significance (Oliveira and Torgo, 2014). One additional aspect from this research that may be useful for our work is that, instead of using the absolute values of the time series, they used the differences between consecutive values to avoid trend effects.

In financial applications, the combination of predictors through ensembles is frequently referred in the literature as a methodology that could result in improved forecasting and potentially better models (Booth et al., 2014a; Ballings et al., 2015; Booth et al., 2015; Caldeira and Torrent, 2017).

2.4 Machine learning models

The standard machine learning models presented in this section are the ones that will be used in Chapters 3 and 4 for implementation.

2.4.1 Selection of models

From the literature review, ANNs, in particular MLPs, stand out as a model with potential to be used as a forecasting tool in fixed income markets. This type of model possesses the necessary flexibility, taking into account the fact that we aim to incorporate a wide range of features. In addition, we identified the RNN family of models as having high potential (Section 2.2.5) Thus, in our empirical work we consider several MLP models, with different sets of features (Section 2.4.3 and Chapter 3), and also a variety of LSTM models (Section 2.6.2 and Chapter 4). Both MLPs and LSTMs are compared to each other and benchmarked against the multivariate linear regression model, covering a wide range in terms of model complexity.

The research presented in this thesis focuses only on machine learning models, and excludes industry models. The main reasons for not considering one of those as benchmark are twofold: previous work already covers the comparison of machine learning versus that type of model; and due to the limitations of the most popular industry models used in fixed income markets. Indeed, there is already abundant literature on these subjects. On the first reason, the following work can be referred. First, Sambasivan and Das (2017) compared the Nelson–Siegel model for yield curve forecasting to the multivariate time series method and to the dynamic Gaussian Processes, obtaining the best results with the latter model and poor results with the first. Second, Dunis and Morrison (2007) compared the ARMA to neural network models, concluding that the latter offer a promising alternative to the traditional models. A comparison of the results from our research with those of the studies mentioned above is included in Section 3.3.2.4.

Regarding the limitations of the most popular industry models used in fixed income markets, the following studies are some possible examples. First, classical financial models for government bonds, a "risk-free" type of asset trading in large and very liquid markets, tend to follow the theoretical restrictions that prevent arbitrage opportunities. However, Björk and Christensen (1999) demonstrated that the Nelson–Siegel model does not ensure the arbitrage-free condition. Second, Pooter (2007) studied several extensions of the Nelson–Siegel model and found that with more flexible models predictability improves, both in-sample and out-of-sample. Third, Tam and Yu (2008) also point out their limitations. Hence, yields-only models are simpler for yield forecasting but they have lower explanatory power, while yields-macro models are more explanatory due to the link to economic variables, but they are less capable of describing several shapes of the yield curve. Fourth, Ullah et al. (2015) faced difficulties fitting particular shapes of the yield curve. In fact, they reported that the Nelson–Siegel model, and even the Svensson generalisation of this model, do not fit well the Japanese yield curve. Finally, Caldeira and Torrent (2017) emphasize the need for alternatives to the available estimators. In the case of factor models, the authors point out that they fit the yield curve assuming a known function of maturities with unknown parameters. They show that better out-of-sample performance can be achieved with a more flexible non-parametric model.

2.4.2 Linear regression

In this section, the multivariate linear regression model is presented, with the main objective of describing the feature selection approach we use in this research called Lasso.

2.4.2.1 Linear regression models

Linear regression models are still very popular nowadays, despite the advances in computer science: they are simple; for a large number of applications they provide adequate models; the interpretability of those models is much higher; and the optimisation process is easier and faster. Unless more complex nonlinear models offer a clear improvement versus the linear solution, one should favour the linear models. It is also a good model to take as baseline to compare more complex solutions.

The linear regression models can be expressed in the following form (Bishop, 2006; Hastie et al., 2013):

$$f(x) = \sum_{j=1}^{p} x_j \, w_j + w_0 + \epsilon \tag{2.3}$$

where $f(x)$ is the target or dependent variable; x_j represent the inputs, independent variables or features; w_j, w_0 are the weights (unknown parameters); ϵ is the error; and p is the total number of features.

In vector notation, it can be written as follows. Note that bold lowercase will be used for vectors, bold capital letter will be used for matrices.

$$f = \boldsymbol{w}^T \, \boldsymbol{x} + w_0 + \epsilon \tag{2.4}$$

In order to avoid treating the constant w_0 independently, one can work with an additional dimension (p+1), with the following changes:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \qquad \boldsymbol{a} = \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix} \tag{2.5}$$

$$f = \boldsymbol{y}^T \, \boldsymbol{a} + \epsilon \tag{2.6}$$

As a result, the general equation for linear regression models, considering all observations of the input dataset can be written as:

$$\boldsymbol{f} = \boldsymbol{Y} \, \boldsymbol{a} + \boldsymbol{\epsilon} \tag{2.7}$$

where $\boldsymbol{f}$ is the $N \times 1$ vector of outputs; $\boldsymbol{Y}$ is an $N \times (p+1)$ matrix, with n^{th} row corresponding to $\boldsymbol{y}_n^T$, N being the number of observations and p the number of features; $\boldsymbol{a}$ is the $(p+1) \times 1$ vector of unknown parameters; and $\boldsymbol{\epsilon}$ is the $N \times 1$ vector of errors.

For the calculation of the unknown parameters a training dataset is used and the most common method for this purpose is the least squares. The objective in this case is to minimise the residual sum of squares, that is, to minimise the error function E:

$$E = \|\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f}\|^2 \tag{2.8}$$

The gradient can then be written as follows:

$$\Delta_a E = 2 \, \boldsymbol{Y}^T \, (\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f}) \tag{2.9}$$

Finally, the solution for the linear regression model can be obtained by equating the gradient to zero, thus obtaining the unknown parameters:

$$\boldsymbol{a} = (\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\,\boldsymbol{Y}^T\boldsymbol{f} \tag{2.10}$$

This is called the pseudo inverse solution and $(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\,\boldsymbol{Y}^T$ the pseudo inverse. Alternatively, the solution can also be obtained using a gradient descent algorithm to minimise Equation 2.8.

Whenever we have a matrix $\boldsymbol{Y}^T\boldsymbol{Y}$ close to being singular, i.e. it is ill-conditioned, one possible approach to make it better conditioned is to use regularisation as shown below:

$$\boldsymbol{a} = (\boldsymbol{Y}^T\boldsymbol{Y} + \gamma\boldsymbol{I})^{-1}\,\boldsymbol{Y}^T\boldsymbol{f} \tag{2.11}$$

where $\boldsymbol{I}$ is the identity matrix and γ the regularisation parameter.

This is equivalent to minimising Equation 2.8, imposing a quadratic penalty term. The resulting equation is presented below. A small variant of this will be covered in the next section and it is important for feature selection.

$$E = \|\boldsymbol{Ya} - \boldsymbol{f}\|^2 + \gamma\|\boldsymbol{a}\|^2 \tag{2.12}$$

2.4.2.2 Feature selection using Lasso

It is well known that the ordinary least squares method as a regression tool has some shortcomings in terms of prediction accuracy and also in terms of interpretation (Tibshirani, 1996; Hastie et al., 2013). In fact, least squares predictions tend to suffer from low bias but high variance and the interpretability of models with a large number of features is challenging.

Several methods may be used to overcome this issue and improve out-of-sample prediction accuracy, such as: selection of a subset of features, the well-known shrinkage methods, including ridge and Lasso regression, and other strategies to reduce the dimensionality of the problem. In more detail, the ridge regression is attributed to Hoerl and Kennard (1970), while the Lasso (standing for Least Absolute Shrinkage and Selection Operator) regression was proposed by Tibshirani (1996). The error function to minimise may be expressed in the following forms:

Ridge regression

$$E = \|\boldsymbol{Ya} - \boldsymbol{f}\|_2^2 + \gamma\|\boldsymbol{a}\|_2^2 \tag{2.13}$$

Lasso regression

$$E = \|\boldsymbol{Ya} - \boldsymbol{f}\|_2^2 + \gamma\|\boldsymbol{a}\|_1 \tag{2.14}$$

where $\|.\|_1$ denotes the L_1-norm; $\|.\|_2$ the L_2-norm; and γ the regularisation parameter.

As can be seen from these equations, the L_2-norm in the ridge regression penalty is replaced by the L_1-norm in the Lasso regression. Hence, the Lasso regression determines the parameters of the model by minimising the sum of squared residuals, using an L_1-norm penalty for the weights. Due to the type of constraint, it tends to lead to sparse solutions, i.e. some coefficients are exactly zero and as a result the corresponding features are discarded. This is particularly important since it enables a continuous type of feature selection through the tuning of the regularisation parameter γ, and the identification of the most relevant features for the model.

A variant of this method, called LagLasso, has been used by Mahler (2009), which includes the selection of features but also of lags, to take into account that the influence of those features may be lagged. In this work, three different maximum number of lags were considered: 1, meaning that only the last value was included in the simulation, 6 and 12 lags. A small improvement in the results can be identified from 1 lag to 6 lags, but not from 6 to 12. Yet, overall the results for the different lags considered were mixed and very close, not highlighting its importance to improve predictions.

The Lasso procedure for feature selection is directly applicable to linear regression models. Although this can be used as a first approach to selecting features for other techniques, different methodologies exist for nonlinear methods. For neural networks, a Bayesian regularisation procedure has been proposed using a Laplace rather than a Gaussian prior (Williams, 1995).

Alternatively, Takeda et al. (2013) perform model selection using a greedy forward selection algorithm in the context of index tracking. They show that the inclusion of an L_2 regulariser enhances out-of-sample performance. However, greedy feature selection is usually more complex than Lasso, which gives a relaxed convex problem to solve.

To conclude, the Lasso regression was used in this study for the feature selection, which seems to combine the benefits of subset selection and ridge regression (Tibshirani, 1996). The reduction in features improves interpretability of models, helping also in cases of low bias / high variance, thus improving generalisation.

2.4.3 Artificial neural networks

The first study on ANNs is associated to the work of neurophysiologist Warren McCulloch and mathematician Walter Pitts, to simulate the activity of the nervous system as a net of neurons using electrical circuits (McCulloch and Pitts, 1943). And that is where the term "neural" comes from. Although we are still far from completely understanding the functioning of the brain, the neural network unit would represent a neuron and the connections would represent synapses.

Recently, this field of computer science has been experiencing a very rapid development and there are now many different types of neural networks. In this section, only the feed-forward neural network will be covered, i.e. the MLP. The RNN will be presented in Section 2.6.

A feed-forward neural network is the simplest type of ANN, where the information flows from the inputs, to the hidden layers, to the outputs, only in one direction. That is, the information is "fed forward", as shown in Figure 2.1. There are no connections

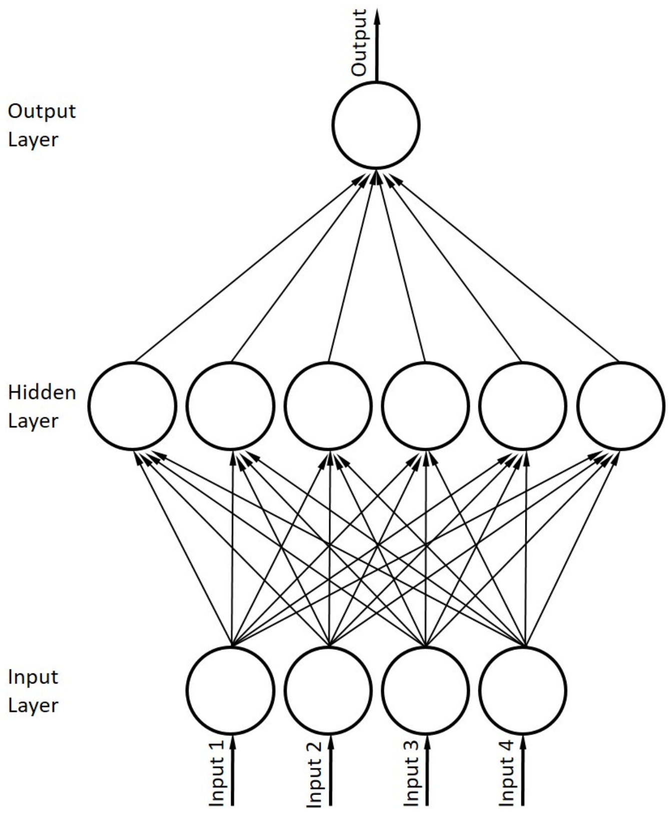


FIGURE 2.1: Feed-forward neural network example.

between units in the same layer, and also no connections from the hidden layers to the outside environment, only through the input and output layers. The most important types of feed-forward neural networks are: radial basis functions (RBF) and MLP. We briefly describe the latter, since it will be used in Chapters 3 and 4.

Multilayer perceptron

The structure of an MLP is similar to the one presented previously in Figure 2.1. It includes an input layer, one or more hidden layers and one output layer. In terms of terminology, the inputs may or may not be considered a layer in the literature. For this type of neural network, the functioning of a unit (also called node and neuron), is schematically presented in Figure 2.2.

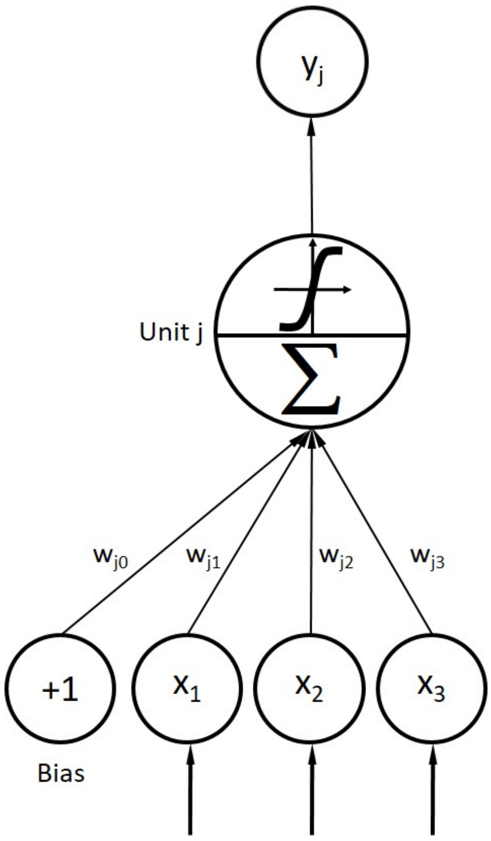


FIGURE 2.2: Neuron of a multilayer perceptron.

Therefore, an MLP neuron comprises a linear function, and a nonlinear function, called activation function. The linear component corresponds to the weighted sum of the inputs plus an independent term. The result of the linear function is then squashed through a differentiable, nonlinear activation function.

The most common nonlinear functions used are the logistic sigmoid function and the hyperbolic tangent function ($tanh$). More recently, another activation function used is the rectified linear unit (ReLU). This activation function has gained importance in deep learning research, where it has been shown to improve training of deeper networks (Glorot et al., 2011).

At the output nodes' level the activation function depends on the type of problem. For classification problems sigmoid or softmax functions may be used, for binary classification and classification tasks with more than two classes, respectively. For regression type of problems, in general a linear output is chosen (Graves, 2012; Lipton et al., 2015).

The training of an MLP is usually performed using stochastic gradient descent and the backpropagation algorithm, first described in 1970 (Linnainmaa, 1970; Schmidhuber, 2015). However, it was only several years later that this technique gained popularity in its application to neural networks due to the work of Rumelhart et al. (1987), demonstrating the development of internal representations on the hidden units. The main objective of the training process is to minimise the error at each neuron output and, as a consequence, for the overall neural network. For this purpose, the gradients of the network error function with respect to the weights are determined, by applying the chain rule of differentiation and new weights calculated. Then all weights are updated simultaneously and used for a new prediction. And the process is repeated until convergence.

2.5 Multitask learning

The interest rate curve has intrinsic properties where the yields for the various maturities are in higher or lower degrees correlated to each other. Those yields also tend to move together forming continuous and coherent lines at any point in time. Consequently, the simultaneous modelling of all benchmarks of the yield curve should be beneficial for the model learning process. Additionally, the modelling of the whole yield curve has potentially higher attractiveness for central banks and national government agencies, but also for the financial industry in general.

However, this modelling is clearly more complex than that of individual assets, since it introduces an additional dimension to the problem, to model both yield and maturity, necessary for dealing with the complete term structure of interest rates. Techniques to deal with the two dimensions using spatial statistics and its modelling tools have been used and were mentioned in Section 2.2.1 (Kanevski et al., 2008; Kanevski and Timonin, 2010).

Recall from Section 2.1.2, that among the most important and widely used models for yield curve modelling and forecasting are the NS (Nelson and Siegel, 1987) and the DNS (Diebold et al., 2006) models. In the latter, the model incorporates both intrinsic or latent factors of the yield curve (level, slope and curvature) and a restricted number of macroeconomic factors (manufacturing capacity utilization, annual price inflation and federal funds rate). Notwithstanding the real and incontestable value of these models in practice and for the industry, a comprehensive research on modelling and forecasting the yield curve using machine learning techniques should start without these factor boundaries already established for the NS and DNS family of models. In fact, we may find other variables that are as important or even more than the ones considered in those models. For this reason, leaving feature selection for the models to decide, through a well designed methodology, seems a better approach.

Taking all this into consideration, an alternative to modelling the curve itself could be the modelling of several points of the yield curve, identifying those points as the most relevant benchmarks in the money and bond markets. For example, the benchmarks for the following maturities: 3 and 12 months (money market), 2, 5, 7, 10 and 30 years (bond market). Focusing only on the bond market, those benchmarks could be reduced to 2, 5, 10 and 30 years, representing four variables to be predicted. Now we will explain how to consider this extended number of targets in a model.

In the machine learning domain, the standard methodology for regression problems is the modelling of one target variable (single task learning), using several inputs. For the yield curve, if we consider a reduced number of benchmarks, for example four, it would represent four different models to forecast the target bond yields. This is represented in Figure 2.3 using four neural networks. This method does not take into

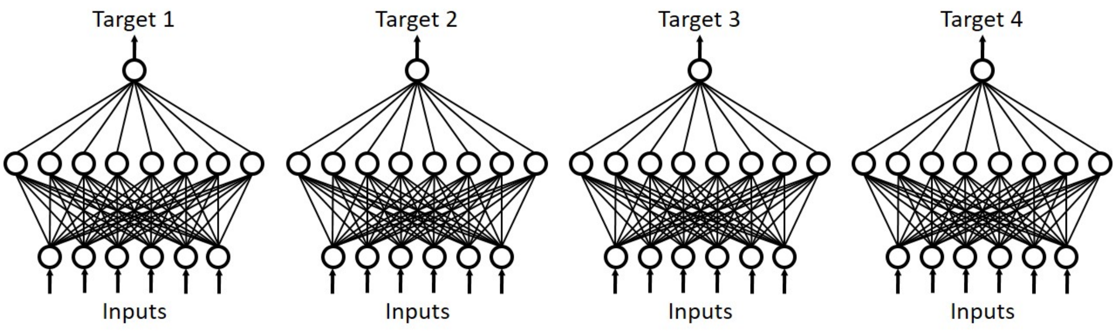


FIGURE 2.3: Single task learning for four different targets.

account the functional form of the yield curve. In other words, it does not consider that those interest rates in the yield curve tend to move together having some functional relationship, which could be beneficial for the model.

In contrast to single task learning, MTL, also known as multi-target regression, multi-output or multi-response (the terminology task, target and output will be used with the same meaning henceforth), enables the learning of several targets simultaneously. This is represented in Figure 2.4 and this methodology could be used to model the yield curve, through the modelling of its most relevant benchmarks.

From the literature and the graphical representation of MTL (Figure 2.4) result some of the characteristics of multitask learning, which include the following (Caruana, 1993, 1997; Cai et al., 2014; Borchani et al., 2015; Ruder, 2017): the hidden layer of the neural network is shared by all targets; the learning process occurs in parallel, simultaneously for all targets; some hidden units may specialise in specific targets, which can be useful for yield curve modelling where some features may be more important for short-term bonds and others for long-term bonds; the use of the domain specific information from additional targets functions as constraints to the overall model, improving generalisation accuracy.

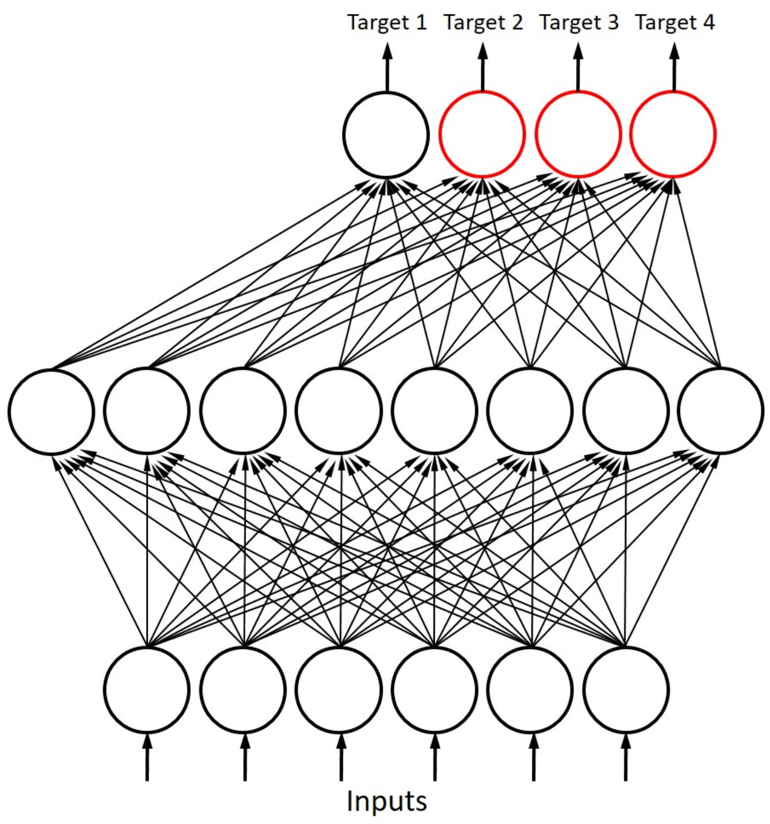


FIGURE 2.4: Multitask learning for four different targets.

In a comprehensive study on MTL (Caruana, 1997), a review of prior work was carried out and new evidence on how MTL functions was also presented. This study emphasizes the importance of the information contained in the extra targets in helping the hidden layer to learn better representations. This research was also careful in ruling out some alternative explanations for the improvement observed. On the one hand, by showing that breaking the relationship between main and additional targets, performance was negatively affected. On the other hand, by excluding the hypothesis that better results were due to restricting net capacity with the introduction of additional targets.

On the architecture of backpropagation neural networks for MTL, the above mentioned study concluded that the number of units in the hidden layer used for MTL should not be much lower than the sum of hidden units leading to good performance when using single target learning on each task individually. The objective is to allow targets to learn individual models, overlapping only when there is common hidden structure (Caruana, 1997).

The type of applications described in the same study, seems to give good indication regarding its potential for applications in the financial domain. One such example, is its use with the data from a robot that captures sonar readings and camera images while moving. MTL is then used to predict what the robot will sense at several distances in the future (1, 2, 4 and 8 meters). One interesting result from this particular study, worth emphasising here, is that the error increases with the distance (as one would expect), but the use of MTL is more beneficial for the most challenging tasks. In other words, MTL leads to a higher decrease in error for the longer distances ahead (2, 4 and 8 meters). On the contrary, it leads to an increase in error for the shorter distance (1 meter). In summary, MTL seems to be most useful for the most difficult tasks.

The main objective with MTL is to predict the main target, with the additional targets being used to help in that purpose. But, in principle, it may be necessary to train and run additional MTL structures for each main target to be forecast (Caruana, 1997). This may be a limitation of this methodology. Nevertheless, for backpropagation neural networks, with enough hidden units, it is also referred that some of these units may specialise in different targets, so that one neural network model may be enough for the complete set of targets.

The differentiation between main target and additional targets is in general natural and domain specific, as will become clear from the pneumonia prediction example, described below. When that is not the case, the original study (Caruana, 1997) suggests that the differentiation may also be imposed by considering different relative weights for each target, as presented below (Equation 2.15). The study also notes that this differentiation may not be necessary in the case of backpropagation neural networks.

$$EvaluationMetric = PerformanceMainTask + \sum_{i=1}^{NoTasks} \lambda_i \, PerformanceExtraTask_i \tag{2.15}$$

where λ is the parameter that controls how sensitive learning is to the extra tasks. When $\lambda = 0$ the extra task is ignored, while $\lambda = 1$ gives the main and extra task the same weight on the performance metric.

Another important application of MTL described by Caruana (1997) is on pneumonia prediction and the filtering of low risk patients. For these, an admission to the hospital can be avoided since they could be better treated at home. However, the most important laboratory tests for that screening are performed after they are hospitalised. The decision on whether or not to hospitalise a patient has to be taken before those tests are carried out, based on the results of a preliminary evaluation of the patient. A solution for this dilemma was found with the use of MTL. With this methodology, the inputs of the model are features available before admission to hospital, and the future laboratory results are used as additional targets in the neural network structure. The model is trained using a database of patients for which both results are available (preliminary and future labs). In this example, the differentiation between main task (pneumonia mortality ranking) and additional tasks (future lab results) is evident and domain specific.

This solution is also very meaningful for financial data, given that many potential features for financial data may not be available in advance to the data to be forecast. This is also the case when the data is published on a less frequent basis, for example quarterly, and advanced interpolation is not possible. In this case, rather than just excluding the data, which may be very relevant for the problem, it is possible to include it as additional targets. One such example is inflation in the United States, or the CPI, which is published quarterly.

Furthermore, the data to be considered for additional targets could also result from the use of synthetic data generated from existing domain-specific models. Obviously, it is necessary to bear in mind that if the model is poor, they will not become a good constraint for the overall model (Caruana, 1997).

According to a recent survey on multitask regression (Borchani et al., 2015), covering a varied number of applications of this methodology in different scientific fields, existing MTL methods can be categorised into two groups: problem transformation methods, in which the problem is transformed into independent single target problems; and algorithm adaptation methods, implying the modification of single output methods in order to handle multiple targets.

Several advanced multi-output regression methods are covered in the above mentioned survey, namely (Borchani et al., 2015): statistical methods, multioutput support vector regression, kernel methods, multi-target regression trees and rule methods. Even though neural networks were not covered as a model in the survey, MTL using neural networks is not a new theme, as mentioned previously in this chapter (Caruana, 1997; Ghosn and Bengio, 1997).

Taking into account the applications of this methodology, several uses strike as possible for financial time series prediction in the bond market, where targets would represent: different points in the yield curve at the same time t, representing a multi-asset process and enabling the forecast of the overall curve; the same yield of a particular bond, at different times, thus enabling the estimation of several time steps ahead in the future; a combination of previous items, enabling a multi-asset and multi-step forecast; additionally, when data is not available on time to be used as feature, it can still be used as target variable if it is relevant for the model.

In summary, MTL can be performed using two different methodologies: transforming the problem into multiple single target and using simultaneous modelling of all targets (multitask learning). Both techniques were used in our research.

2.6 Deep learning models

The literature review presented in Section 2.2.5 revealed the importance of RNNs for financial applications. In this section, an overview of this type of model is provided, including the LSTM used for the comparative study presented in Chapter 4. Advantages and limitations of this type of model are listed, as well as the most common type of practical applications. Finally, the LSTM networks' potential for yield and yield curve forecasting is discussed, mainly with regards to Objective 2 (Section 1.2) of the present work, but also with potential implications for Objective 5.

To introduce the concept of deep learning, it is part of a broader type of machine learning models, capable of learning high-level representations in data. They have a deep type of structure and at each level, using simple but nonlinear modules, the representation is transformed successively into higher and more abstract levels (LeCun et al., 2015). At the end of such structure, very complex nonlinear representations can be learnt directly from the data. In feedforward neural networks, the "deep" in deep learning implies architectures with many hidden layers, in contrast with "shallow" neural networks. But there are other types of deep learning model such as deep belief networks and RNNs.

2.6.1 Standard recurrent neural networks

In this section, we will be presenting the standard RNN, including an overview and the limitations of feed-forward neural networks versus this type of model. In addition, some specific technical aspects will be covered, namely, backpropagation through time and the vanishing or exploding gradients. These led to the so-called fundamental deep learning problem, which for several years was an obstacle to the progress of deep learning.

2.6.1.1 Overview

Neural networks in general are inspired in the functioning of the brain as mentioned in Section 2.4.3, based on the early work of McCulloch and Pitts (1943). One of the characteristic of brain functioning and of our thought process is that we do not start from zero at every moment in time. Our thoughts and knowledge persist in time and we can use what we have learnt in the past at every moment. In other words, we have memory. Additionally, there is also an implicit recurrent process in the sense that, at any point in time, we are receiving different inputs (e.g. environmental factors), that are processed by our brain producing results (e.g. knowledge). In fact, there is a new line of research in neuroscience using this type of model, and Güçlü and van Gerven

(2017) have shown that RNNs can be used to predict how the human brain processes sensory information.

The initial studies on RNNs date back to the 1980s with the work of Hopfield (1982), Jordan (1986) and Elman (1990). The new architecture of RNNs includes a clever feedback loop mechanism delayed in time that enables the model to "remember" past information. This loop system is shown in Figure 2.5 for a particular time step *t*.

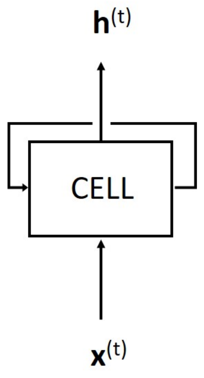


FIGURE 2.5: Recurrent neural network for a particular time step.

Consequently, the output at each time step depends on both the inputs at that time step and the feedback from the previous hidden layers, which form the hidden or internal state responsible for making the connection across time. Furthermore, we can unroll the RNN in time to have a full perspective of the network. This is shown in Figure 2.6.

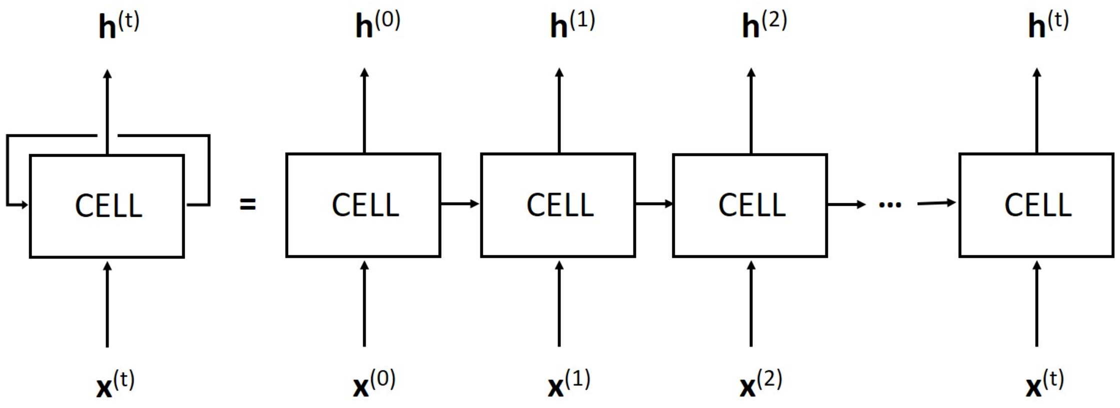


FIGURE 2.6: Recurrent neural network unrolled in time.

Considering the unrolled architecture, each cell in the RNN is, in fact, a standard neural network and the cells are connected by the hidden state. This state can be interpreted as the "memory" of the network. The implications for sequence learning are immense. While the MLP maps inputs to outputs in a rigid manner, theoretically the RNN can map the entire history of inputs to each output (Graves, 2012). We will see that in practice this is not the case and the "memory" of the recurrent network fades as the time steps become distant from the present time step (Section 2.6.1.3). Nevertheless, the previous inputs persist in time, affecting outputs, and this is a characteristic that completely separates feed-forward neural networks from RNNs.

Continuing the analysis of the unrolled architecture, the RNN can also be interpreted as a deep feedforward learning network, with each time step as an additional hidden layer of the network (LeCun et al., 2015). However, in deep learning the weights of the network differ at each layer. On the contrary, RNNs share the same weights across time steps, with multiple repetitions of the same model across time steps. This reduces significantly the number of parameters the network needs to learn during training.

More formally, the equations for the forward pass of a vanilla RNN are as follows (Graves, 2012; Lipton et al., 2015; Goodfellow et al., 2016):

$$\boldsymbol{h}^{(t)} = \sigma_h \left(\boldsymbol{W}^{hx} \, \boldsymbol{x}^{(t)} + \boldsymbol{W}^{hh} \, \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h \right) \tag{2.16}$$

$$\hat{\boldsymbol{y}}^{(t)} = \sigma_y \left(\boldsymbol{W}^{yh} \, \boldsymbol{h}^{(t)} + \boldsymbol{b}_y \right) \tag{2.17}$$

where $\boldsymbol{x}^{(t)}$ is the input vector at time step t; $\boldsymbol{h}^{(t)}$ and $\boldsymbol{h}^{(t-1)}$ are the hidden states at time step t and t-1; $\hat{\boldsymbol{y}}^{(t)}$ is the output at time step t; $\boldsymbol{W}$ are the weight matrices; $\boldsymbol{W}^{hx}$ and $\boldsymbol{W}^{yh}$ represent the conventional weight matrices for input-to-hidden and hidden-to-output connections, respectively; and $\boldsymbol{W}^{hh}$ representing the recurrent weight matrix for hidden-to-hidden connections at adjacent time steps; $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are the bias vectors; and finally σ_h and σ_y the activation functions at the hidden and output nodes' level, respectively.

2.6.1.2 Limitations of feed-forward neural networks

After describing the architecture of the RNNs and its capabilities we now identify the limitations of the traditional feed-forward neural networks, in particular the MLP covered in Section 2.4.3, that RNNs can overcome (Prügel-Bennett, 2017; Brownlee, 2018). First, the memory capability is one of the characteristics that clearly separates both types of models. Given the stateless condition of MLP models they only learn fixed function approximations. Second, in sequence prediction problems, the sequence imposes an order on the data that must be respected when training and forecasting, i.e. the order of the observations is important for the modelling process. This is the case with financial time series. However, in feed-forward neural networks / MLPs, the modelling of time series' temporal structure is only done indirectly through the consideration of multiple time steps as different input features. Although using this method previous values are included in the regression problem, the natural "sequence" or structure of the time series is not really present in the modelling process and the model does not have any knowledge of it. Third, and also related with the previous limitation, when we are dealing with problems involving multiple input time series the ensuing number of input features grow rapidly, resulting in a complex and computationally demanding scaling up of this type of model. Furthermore, there is not a clear separation between time series, all considered as additional input features. The final disadvantage of feed-forward NNs is that both inputs and outputs need to have fixed size. Thus, this type of model has serious limitations for some applications involving variable length input and/or output sequences.

Despite its limitations, feed-forward neural networks / MLPs are always a good starting point for modelling financial time series. This is especially the case given the dominant characteristics of the most recent historical values of the target variable (as will be shown by the results presented in Section 3.3.1).

2.6.1.3 Backpropagation through time

The training of RNNs is done using the same technique used for feedforward neural networks, namely the backpropagation algorithm (Section 2.4.3). However, given the dependence of previous states, the back propagation of error must be carried out on the unrolled RNN (Figure 2.6), considering not only the present time step corresponding to the error being backpropagated but also previous time steps up to the first of the sequence. This is why in the case of RNNs, the technique is called backpropagation through time, also known as BPTT. The corresponding equations were first introduced by Werbos (1990). Given this architecture, the model may become computationally expensive for networks with a large number of time steps. Although truncated back-propagation through time may be a solution, it comes at the cost of not being able to learn long-term dependencies (Williams and Peng, 1990).

In the late 1980s and early 1990s, it became more evident that, although RNNs were theoretically capable of modelling those long-term dependencies, they appear to work only for shallow problems, i.e. considering shorter sequences (Schmidhuber, 2015). This problem was also being observed with feedforward neural networks, where the benefits of going deeper with additional hidden layers did not appear to produce empirical benefits. This is now known as the fundamental deep learning problem of gradient descent and for several years it was a huge obstacle for training deep neural networks.

2.6.1.4 Vanishing or exploding gradients

The first milestone for identifying the now well known problem of vanishing or exploding gradients was the work of Hochreiter (1991) in a diploma thesis (Hochreiter et al., 2001; Schmidhuber, 2015).

In parallel and independent work, the research by Bengio et al. (1993, 1994) contributed to overcome the impasse that this problem has then caused to the progress of deep learning. Hence, these authors published theoretical and experimental evidence that the training with gradient descent and BPTT could be inadequate, in a paper appropriately titled "Learning long-term dependencies with gradient descent is difficult" (Bengio et al., 1994). More recent research by Sutskever (2013) presents a methodology to enable the training of RNNs with long-term dependencies using gradient descent with momentum, which involves random initialisation of parameters. The author also observes that this work directly contradicts previous findings and suggests that the lack of satisfactory results with RNNs having long-term dependencies could be due to deficiencies in the random initialization.

The vanishing or exploding gradients problem was presented in an elegant and simplified way by Goodfellow et al. (2016). RNNs share the same weights across time steps, implying the application of the same function at each time step (Section 2.6.1.1), in what compares with matrix multiplication. To explain, let us consider the simplest form of an RNN without inputs, which corresponds to isolating the recurrence term in Equation 2.16 for calculating the hidden state at time step t. Assuming for simplicity a linear activation function and zero bias, we obtain the following equation:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^T \, \boldsymbol{h}^{(t-1)} \tag{2.18}$$

This recurrence term essentially describes the power method, so we can rewrite it as:

$$\boldsymbol{h}^{(t)} = (\boldsymbol{W}^t)^T \, \boldsymbol{h}^{(0)} \tag{2.19}$$

Let us assume that $\boldsymbol{W}$ can be eigendecomposed in the following form:

$$\boldsymbol{W} = \boldsymbol{Q} \, \boldsymbol{\Lambda} \, \boldsymbol{Q}^T \tag{2.20}$$

Then, the recurrence term may be written as:

$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}^T \, \boldsymbol{\Lambda}^{\boldsymbol{t}} \, \boldsymbol{Q} \;\, \boldsymbol{h}^{(0)} \tag{2.21}$$

From this equation where the eigenvalues are raised to the power of t, we can easily conclude that any eigenvalue lower than 1 will tend to decay exponentially ("vanish") and any eigenvalue higher than 1 will tend to increase exponentially ("explode").

The vanishing or exploding gradients problem refers to the fact that during backpropagation the gradients will suffer the same effect of vanishing or exploding as described above. In practice, vanishing gradients will cause difficulties in knowing which direction the parameter should move to minimise the cost function, while exploding gradients will cause the learning process to be unstable. This is a specific problem of RNNs given that they share the same weights across time steps, i.e. they use the same matrix $\boldsymbol{W}$ at each time step.

Subsequently, extensive research has been carried out to find a solution to the problem just described. Some methods proposed include (Hochreiter et al., 2001; Schmidhuber, 2015): unsupervised pre-training for a hierarchy of RNNs; Hessian-free optimization, in the case of feedforward neural networks; and searching the space of weight matrices using alternative methods to error gradients. But the most important of them was the development of a new sub-type of RNNs, whose architecture is unaffected by the fundamental deep learning problem. This model is the LSTM and will be presented in Section 2.6.2.

2.6.2 Long short-term memory networks

The LSTM architecture was first introduced by Hochreiter and Schmidhuber (1997), and subsequently adapted by other researchers such as Gers et al. (1999), Gers and Schmidhuber (2000), and Graves and Schmidhuber (2005).

The LSTM model is a type of RNN and consequently the diagrams presented in Section 2.6.1.1 are also valid for LSTMs, i.e. they also have a structure that can be unrolled in time. In the case of RNNs the repeating cells (Figure 2.6) have a single neural network layer. In fact, at each time step the structure of an RNN is a standard feedforward neural network. In contrast, the LSTM architecture is substantially more complex, incorporating four complete neural network structures in each of those cells, in this model also called memory cells. In Figure 2.7 a simplified diagram of the unrolled chain-type structure is presented, identifying the main components of a memory cell. There are some variants to this standard LSTM model, but they do not differ significantly from this structure (Prügel-Bennett, 2017).

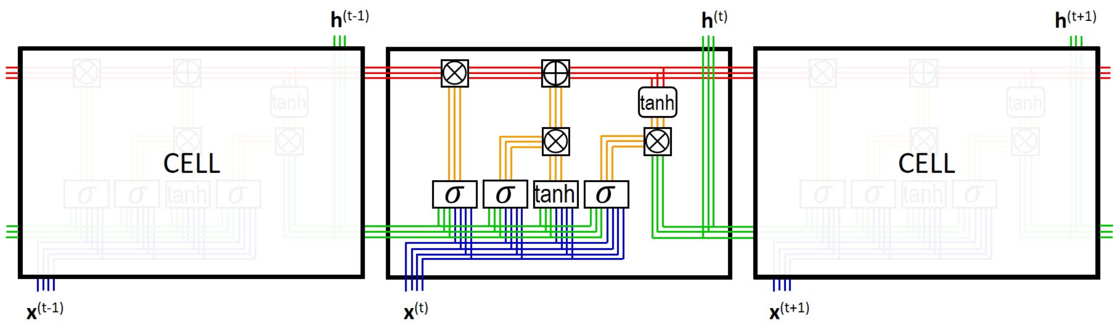


FIGURE 2.7: Long short-term memory cells (based on Olah (2015), with modifications made by the author).

A more detailed representation of an LSTM cell is presented in Figure 2.8. The main components of the memory cell will be explained in the following text.

The corresponding equations that govern the modern LSTM model can be expressed in the following form (Hochreiter and Schmidhuber, 1997; Lipton et al., 2015; Goodfellow et al., 2016):

$$\boldsymbol{f}^{(t)} = \sigma\,(\boldsymbol{W}^{fx}\,\boldsymbol{x}^{(t)} + \boldsymbol{W}^{fh}\,\boldsymbol{h}^{(t-1)} + b_f) \tag{2.22}$$
$$\boldsymbol{i}^{(t)} = \sigma\,(\boldsymbol{W}^{ix}\,\boldsymbol{x}^{(t)} + \boldsymbol{W}^{ih}\,\boldsymbol{h}^{(t-1)} + b_i) \tag{2.23}$$
$$\boldsymbol{g}^{(t)} = tanh\,(\boldsymbol{W}^{gx}\,\boldsymbol{x}^{(t)} + \boldsymbol{W}^{gh}\,\boldsymbol{h}^{(t-1)} + b_g) \tag{2.24}$$
$$\boldsymbol{o}^{(t)} = \sigma\,(\boldsymbol{W}^{ox}\,\boldsymbol{x}^{(t)} + \boldsymbol{W}^{oh}\,\boldsymbol{h}^{(t-1)} + b_o) \tag{2.25}$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \otimes\ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \otimes\ \boldsymbol{g}^{(t)} \tag{2.26}$$
$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \otimes\ tanh(\boldsymbol{c}^{(t)}) \tag{2.27}$$

where $\boldsymbol{f}^{(t)}$ is the function for the forget gate; $\boldsymbol{i}^{(t)}$ and $\boldsymbol{g}^{(t)}$ are the functions for the input gate and for the input node, respectively; $\boldsymbol{o}^{(t)}$ is the function for the output gate; $\boldsymbol{c}^{(t)}$ and $\boldsymbol{c}^{(t-1)}$ are the cell states (also known as internal state) at time step t and t-1; $\boldsymbol{h}^{(t)}$ and $\boldsymbol{h}^{(t-1)}$ are the hidden states at time step t and t-1; $\boldsymbol{x}^{(t)}$ is the input vector at time step t; $\boldsymbol{W}$ are the weight matrices, with $\boldsymbol{W}^{fx}$ as an example, representing the weight matrix for the connection input-to-forget gate (indices indicating 'to-from' connections); $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_g, \boldsymbol{b}_o$ are the bias vectors; σ is the logistic sigmoid activation function; *tanh* the hyperbolic tangent activation function; and $\otimes$ represents the Hadamard product (element-wise multiplication).

Figure 2.8 show where these operations (Equations 2.22–2.27) occur in the LSTM cell. It is possible to simplify Equations 2.22 to 2.25, by concatenating the inputs to the cell and the previous hidden state as follows:

$$\boldsymbol{x}_{\boldsymbol{c}}^{(t)} = [\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}], \tag{2.28}$$

resulting in the equations presented below:

$$\boldsymbol{f}^{(t)} = \sigma\,(\boldsymbol{W}^{f}\,\boldsymbol{x}_{\boldsymbol{c}}^{(t)} + b_f) \tag{2.29}$$
$$\boldsymbol{i}^{(t)} = \sigma\,(\boldsymbol{W}^{i}\,\boldsymbol{x}_{\boldsymbol{c}}^{(t)} + b_i) \tag{2.30}$$
$$\boldsymbol{g}^{(t)} = tanh\,(\boldsymbol{W}^{g}\,\boldsymbol{x}_{\boldsymbol{c}}^{(t)} + b_g) \tag{2.31}$$
$$\boldsymbol{o}^{(t)} = \sigma\,(\boldsymbol{W}^{o}\,\boldsymbol{x}_{\boldsymbol{c}}^{(t)} + b_o) \tag{2.32}$$

We will now explain in detail how an LSTM works, the main components of the cell (Figure 2.8) and the complete algorithm (Equations 2.22 to 2.27). In each LSTM cell the

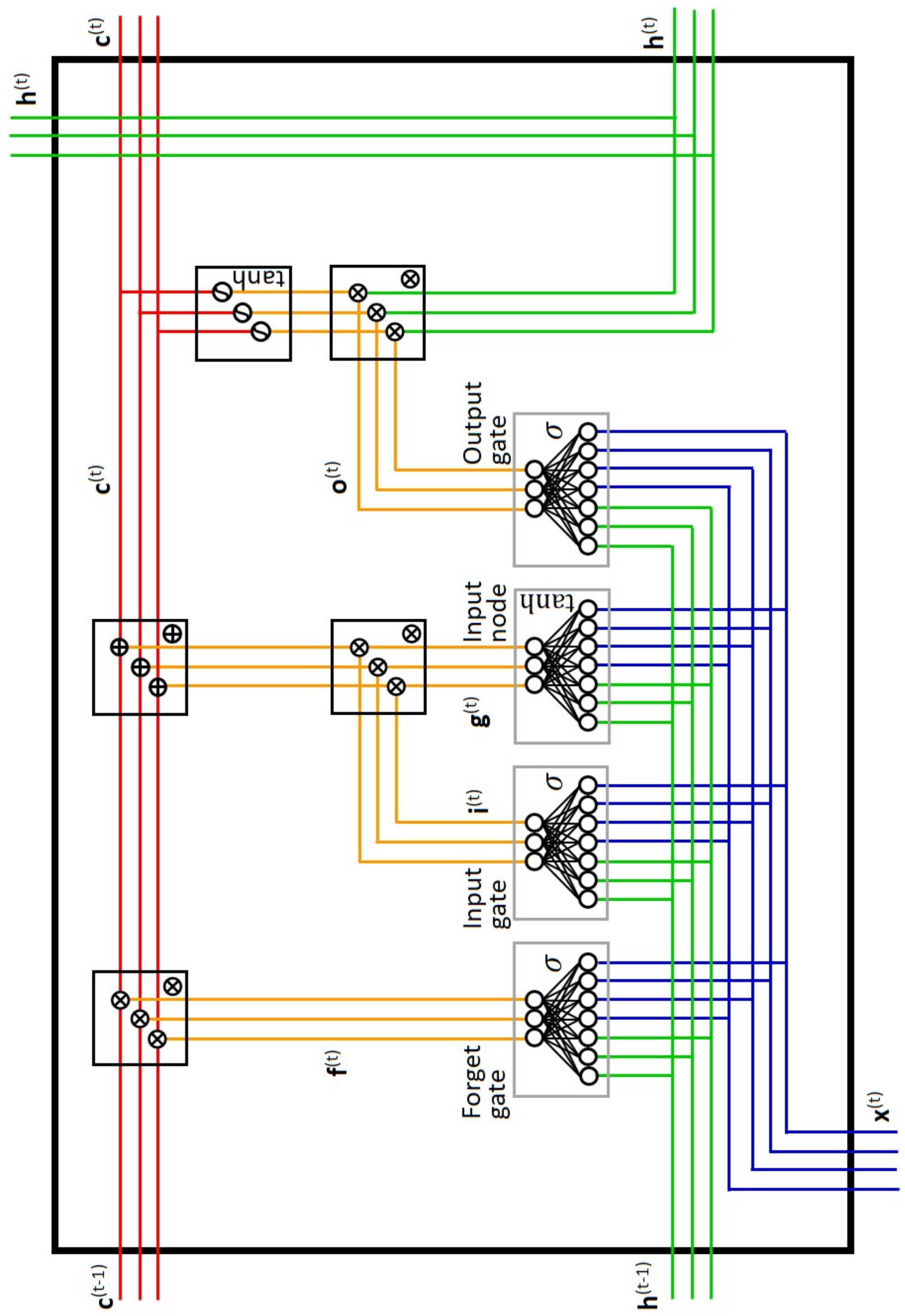
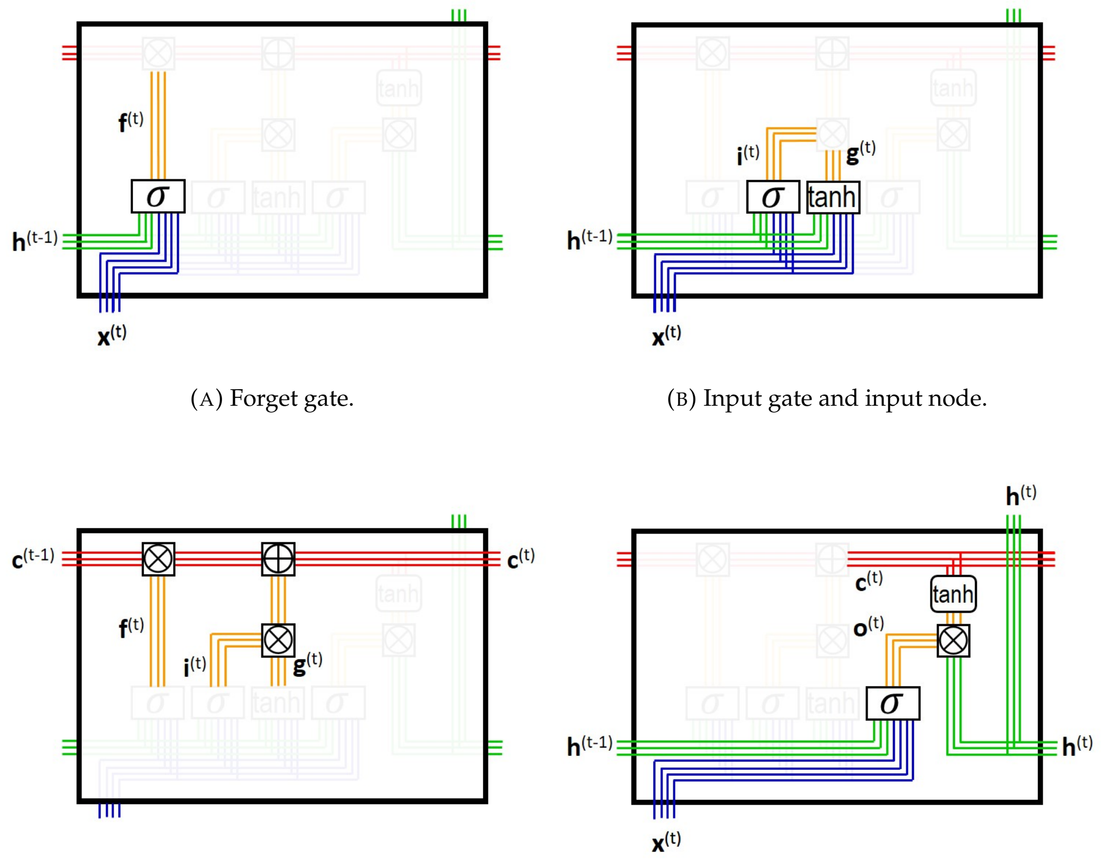


FIGURE 2.8: Long short-term memory detailed cell diagram (based on Prügel-Bennett (2017), with modifications made by the author).

flow of information is controlled by tree gates, namely: forget, input and output gates. The operations performed at cell level are schematically presented in Figure 2.9.



(A) Forget gate.

(B) Input gate and input node.

(C) Cell state update.

(D) Output gate.

FIGURE 2.9: Long short-term memory cell gates.

All calculations at each gate depend on the current inputs at the same time step and the previous hidden state (at time step t-1). The output from the cell depends on those two variables plus the cell state at time step t. Additionally, the cell or internal state represents the long-term memory of the model and the hidden state corresponds to the short-term memory.

Forget gate

The forget gate defines which information to remove or "forget" from the cell state. For this purpose, the forget gate has a neural network with a logistic sigmoid activation function ranging from 0 to 1 (Figure 2.9a). The extremes of that interval correspond to: keep this information (1) or completely remove this information (0). The maths operations performed at this gate are represented by Equation 2.22.

The forget gate was not present in the original formulation of this model (Hochreiter and Schmidhuber, 1997). It was introduced at a later stage by Gers et al. (1999). This gate is especially important for very long or continuous time series, for which the authors found that the network needed resets to release internal resources and remove irrelevant information.

Input gate and input node

The input gate specifies which information to add to the previous cell state. This part of the cell comprises two elements as shown in Figure 2.9b. The first component is the input gate, where the inputs (hidden state at time step t-1 and inputs at time step t) go through a neural network with a logistic sigmoid activation function (corresponds to node i and function $\boldsymbol{i}^{(t)}$). The second component is the input node (to differentiate from "gate") and represents the new "candidates" that could be added to the cell state. These are generated through a neural network with a hyperbolic tangent activation function (corresponds to node g and function $\boldsymbol{g}^{(t)}$). The corresponding operations carried out in this section of the cell are represented by Equations 2.23 and 2.24.

In the original structure proposed for this model (Hochreiter and Schmidhuber, 1997), the activation function in node g was also a logistic sigmoid function as in the other gates. However, the standard LSTM nowadays uses a hyperbolic tangent function (Lipton et al., 2015).

Cell state update

The cell state update is performed using the results of both the forget and input gates. The operations implemented for this purpose are presented schematically in Figure 2.9c and in mathematical terms by Equation 2.26.

This cell or internal state is the recurrent connection across time steps, having unit weights. It is the basis of the so-called "constant error carousel" (CEC) described in the original LSTM work by Hochreiter and Schmidhuber (1997). This together with the gated type of structure is the reason why, during the backwards pass, gradients can propagate in the unrolled-in-time network without vanishing or exploding, thus keeping the network stable. The main implication is that it enables the LSTM to learn the long term dependencies that were difficult for the standard RNNs.

Output gate

The output gate defines the information from the cell state that will be used as output of the memory cell for the present time step. This gate has the fourth neural network of the LSTM cell, with a logistic sigmoid activation function (Figure 2.9d). The operations applied in this gate are represented by Equation 2.25. Finally, the actual output from the cell is computed using Equation 2.27, taking into account the results of the output gate and the present cell state, pushed through a hyperbolic tangent function.

An overall observation that can be made from the previous description is that the training of this type of model is usually slow, considering that the LSTM gates comprise four neural networks inside each memory cell as mentioned before.

2.6.3 RNN-LSTM advantages and limitations

The main advantage of the LSTM model is related to the reason why it was developed in the first place. The RNNs could not capture the long-term dependencies due to the vanishing or exploding gradients problem already described in Section 2.6.1.4. Another important characteristic of this type of model is that it can input and output sequences time step by time step, enabling variable length inputs and/or outputs. With this property, they overcome one of the main limitations of standard feedforward neural networks.

However, some time series forecasting problems are technically simpler, not requiring the characteristics of a recurrent type of model. This is the case in particular when the most relevant data for making the prediction is within a small window of recent historical values. Here, the capability to deal with long-term dependencies and the model "memory" is clearly not necessary. In this type of situation, MLPs and even linear models may outperform the LSTM pure-autoregressive univariate model, with lower complexity (Gers et al., 2002; Brownlee, 2018). Overall, given the additional complexity of the model, it should be used only when the type of problem we have is better modelled by this type of neural network architecture. And this is reflected in two main conditions: sequential data and when the long-term dependencies may help the forecasting process.

Several other issues have been pointed out to RNN/LSTMs. Culurciello (2018) emphasises the limitations of the recurrent family of models, highlighting that some of the largest tech companies such as Google, Meta and Salesforce are moving away from these models. Although LSTMs in particular can learn long-term information, that becomes more difficult for long sequences.

The sequential nature of recurrent models, requiring sequential computation, represent a fundamental limitation (Vaswani et al., 2017). This makes this type of model hardware unfriendly, inhibiting parallelisation within training examples, requiring more resources and time to train and run than, for example, attention-based models. As a result, RNN/LSTM networks are inefficient and not scalable (Culurciello, 2018).

For clarity, the initial implementations of attention mechanisms were proposed with architectures based on recurrent or convolutional neural networks that include an encoder and a decoder (Bahdanau et al., 2015; Luong et al., 2015). Even though this type of model is not covered in this thesis, we refer the Transformer as a high profile and popular example. The Transformer, initially proposed by Vaswani et al. (2017), significantly changed the use of attention in machine learning, totally dispensing recurrence and convolutions, and enabling more parallelisation. Its self-attention mechanism is what permits the Transformer to map global dependencies between inputs and outputs and deal with long sequences, by selectively focusing on the relevant parts of the input data. Attention-based models were developed with neural machine translation tasks in mind. However, they have been successfully implemented for time series prediction in finance (Qin et al., 2017).

2.6.4 RNN-LSTM applications in finance and other fields

This family of model, especially the LSTM networks, has shown to significantly outperform. In fact, in one of the most recent and comprehensive books on deep learning the authors categorically affirm that gated RNNs are the most effective sequence models used in practical applications, i.e. LSTMs and gated recurrent unit (GRU) based networks (Goodfellow et al., 2016).

The applications are almost endless, and span a wide variety of activities and scientific fields, such as (Lipton et al., 2015): handwriting recognition, text generation, natural language processing (recognition, understanding and generation), time series prediction, video analysis, musical information retrieval, image captioning, music generation, and in interactive type of problems such as controlling a robot. For natural language processing in particular, LSTMs are among the most widely used deep learning models to date.

Most of these activities have in common the fact that they have sequential data. And this is what RNNs and LSTMs do best. They can process sequences as input, as output or in the most general case on both sides (Karpathy, 2015). Furthermore, the LSTMs are used to take advantage of their capability to learn long-term dependencies.

In the financial domain, the situation is even more extreme than the one found for the previously presented machine learning models (Section 2.4). In fact, no publication could be found in the current literature using this type of model in fixed income markets. A discussion on its potential in this area will be the focus of Section 2.6.5.

Indeed, applications of RNN and LSTM models were found first and foremost for equities. See, for example, the works by Xiong et al. (2016), Persio and Honchar (2016a, 2017), Fischer and Krauss (2018), Munkhdalai et al. (2017) and Qin et al. (2017). In addition, substantial work can be found on forex markets (Giles et al., 2001; Maknickienė and Maknickas, 2012; Persio and Honchar, 2016b). Other applications can be detected for financial crises prediction (Gilardoni, 2017) and for credit risk evaluation in P2P platforms, or peer-to-peer lending (Zhang et al., 2017).

2.6.5 LSTM networks' potential for yield and yield curve forecasting

Following the presentation of the deep learning models, in particular the RNN and the LSTM networks, we will discuss the potential of this type of model for yield and yield curve forecasting. This is important to fulfil Objective 2 (Section 1.2), covering the assessment of existing machine learning techniques. Besides, it is also relevant given the lack of published work in this area, which may reflect lack of adequacy of the model when applied to the bond market or an opportunity worth exploiting in our current research.

In Section 2.2.5, we have mentioned an RNN topology called shared layer perceptron, and argued that its characteristics appear to indicate high potential for yield curve forecasting. Now that the models were described in greater detail, it is also possible to give more concrete justifications.

First, with sequences we can model directly the time structure of time series, which is not possible with feedforward neural networks. Recall from Section 2.6.1.2 that, in the latter type of network, the temporal structure was considered indirectly through different input features representing a chosen number of time steps. The LSTMs are the most effective models for sequence learning, modelling these time sequences naturally. Effectively, the consideration of a sequence in the model is adding a new dimension to the mapping function being learnt by the model. Instead of a simpler mapping function input-to-output, the network learns a more complex function of the type "input over time"-to-output.

Second, it seems probable that long-term dependencies are also important for modelling financial time series. Even though the last available value of the series is the one collecting all the available information in the market up to the present moment, inversion points tend to follow certain patterns, frequently exploited by technical analysts looking essentially at chart data. A model with memory and capable of learning long-term dependencies may be beneficial for this reason. The LSTMs were specifically designed to deal with the long-term dependencies' problem of RNNs, so they should also fulfil this requisite.

Third, for modelling the yield curve we have some additional requirements from the model to be used. We should be able to perform multi-asset forecasting, for the prediction of several points of the yield curve to define the whole curve as accurately as possible or with more detail in the areas we would like to cover. The LSTM is capable of dealing with multivariate type of problem. Consequently, the MTL required for yield curve forecasting can be carried out with LSTMs in both forms, MTL or by transformation into multiple single task problems.

Finally, it would be desirable to perform multi-step forecasting, to consider several forecasting horizons into the future. This can be accomplished naturally with LSTMs by using sequence-to-sequence architectures (a sequence as input and as output, also known in short as "seq2seq"). In this configuration, the output sequence contains predictions of the same time series at different time steps in the future.

Overall, LSTM networks' potential for yield and yield curve forecasting seems evident. Despite being predominantly used for non-financial applications, their characteristics make them suitable for financial time series predictions.

Notwithstanding, in some cases the LSTM complexity is not really required and a simpler model can result in better results (Section 2.6.3 and Gers et al. (2002)). The main conclusion is that simpler models should be tested first during the experimental part of the research. Then, more sophisticated models can be employed and compared to the previous solutions, which may work as benchmarks.

2.7 Reinforcement learning

In this section, we introduce the topic of RL, starting by outlining the problems and limitations of supervised learning methods when applied to common activities such as portfolio management and trading. We then present the base concept of an RL system, its components, the main types of algorithms that are available, and a review of recent studies. We conclude with a discussion of its potential, specifically for fixed income portfolio management.

2.7.1 Limitations of supervised learning for portfolio management

In the work conducted so far, we have covered direct forecasting of assets, in this case to predict yields. This is undoubtedly very important for practitioners and regulators across financial markets. But the ultimate purpose in this industry is for the forecasting to be used both in the portfolio management and trading activities. These are the areas where better forecasting tools can produce profits for the institution and their clients. However, this is a different type of problem when compared to the predictions of a random-type variable, which could be price or yield.

In fact, when using supervised machine learning for portfolio management or trading, it is common practice to predict asset values or even their returns first, and then follow a strategy that takes into account that forecast (Freitas et al., 2009; Abe and Nakayama, 2018). Nevertheless, although these methods may improve forecasts when compared to traditional methods, in general, the prediction accuracy in financial markets tends to be lower than in other scientific fields. And this is both a problem and a limitation of supervised learning approaches.

Another important aspect, it that in portfolio management and trading, the aim is to optimise a certain strategy metric, which could be simply the portfolio return, or any other type of risk-adjusted metric, such as Sharpe ratio. Any of these metrics are a function of the variable (price or yield) and, as a result, getting correct predictions on the variable does not necessarily translate into profitable strategies. To take this into account, the evaluation of models in some studies covered in the literature is carried out considering not only standard measures for forecasting accuracy, such as RMSE and MAPE, but also the results from simulated trading strategies (Dunis and Morrison, 2007; Mettenheim, 2014; Caldeira and Torrent, 2017). But they still have to deal with the problem of low asset-value accuracy in financial markets.

In summary, for those working in asset management, forecasting asset values is an intermediate problem (with colossal challenges), while total investment returns is the target problem. Focusing directly on the target problem is what RL enables.

2.7.2 Traditional portfolio management methods

A portfolio is formed by a group of financial assets. The main purpose of investing in portfolios instead of in one individual asset, is diversification and, through it, risk reduction per unit of risk in the investment (Fabozzi et al., 2021). It is also easier to combine assets and adjust the composition of a portfolio to the requirements and risk characteristics of an investor, than to find only one asset that could satisfy the same conditions.

Markowitz is the pioneer of what is known as modern portfolio theory, with the publication of a seminal paper entitled "Portfolio selection" (Markowitz, 1952). Markowitz's mathematical formulation determines the weights of the assets to form portfolios that fall in the *efficient frontier*. They are called efficient portfolios in the sense that they correspond to the best mean-variance (return-risk) combinations, from the universe of possible portfolios built with the available assets. In other words, they correspond to portfolios that minimise the risk for a given level of expected returns, (investor property called risk aversion), or maximise the expected returns for a given level of risk (investor property called nonsatiation). Considering the first of those two styles, for a portfolio of m assets, a specific target expected portfolio return, and for the case where the asset weights can be negative (short selling of assets permitted), we obtain the minimum-variance portfolio as follows (Markowitz, 1952; Kroll et al., 1984):

$$\underset{\boldsymbol{w}}{\text{minimise}} \quad \frac{1}{2}\boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w} \tag{2.33}$$

$$\text{subject to} \quad \boldsymbol{w}^T \boldsymbol{r_A} = R_{P,target} \tag{2.34}$$

$$\text{and} \quad \mathbf{1}_m^T \boldsymbol{w} = 1 \tag{2.35}$$

where $\boldsymbol{w}$ is the vector of weights of the assets in the portfolio; $\boldsymbol{\Sigma}$ denotes the covariance matrix; $\boldsymbol{r_A}$ is the vector of expected returns of the assets in the portfolio; $R_{P,target}$ is the target expected portfolio return; and $\mathbf{1}_m$ represents an m-dimensional column vector of ones. Both the expected returns of the assets and their covariances need to rely on forecasts for the future period in analysis, which would involve additional models for this purpose. In practice, they may simply be estimated using historical data, with all the shortcomings of this solution.

To minimise Equation 2.33, the problem is solved using Lagrange multipliers, differentiating the Lagrangian with respect to each weight of the assets and setting the partial derivatives equal to zero. Hence, for a portfolio of m assets, we will have $m + 2$ linear equations (two for the constraints), to determine m weights and the 2 Lagrange multipliers.

When short selling is not permitted, i.e. the portfolio can only contain long positions, we need an extra constraint as follows:

$$\underset{\boldsymbol{w}}{\text{minimise}} \quad \frac{1}{2}\boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w} \tag{2.36}$$

$$\text{subject to} \quad \boldsymbol{w}^T \boldsymbol{r_A} = R_{P,target} \tag{2.37}$$

$$\text{and} \quad \mathbf{1}_m^T \boldsymbol{w} = 1 \tag{2.38}$$

$$\text{and} \quad \boldsymbol{w} \succeq 0 \tag{2.39}$$

With these three constraints, the optimisation problem becomes a quadratic programming problem, with a quadratic objective function and linear constraints.

The Markowitz mean-variance portfolio is widely used in the industry, it is relatively simple to implement and the idea behind it makes intuitive sense. However, there are a number of limitations pointed out in this model (Board et al., 2009; Zhang et al., 2018; Hambly et al., 2021). Some of those limitations are related to specific practical problems that cannot be addressed with this formulation. Some examples are: the inclusion of constraints related to trading rules, asset classes, type of security, transaction costs, and others to enable risk diversification (by sources of risk).

Another common criticism is the single-period framework, i.e. it assumes that the investor decides on allocation at the beginning of the period, freezing the exposure for the whole period. Indeed, it does not consider the benefits and costs of rebalancing the portfolios, usually necessary as market conditions change. In some way connected to the previous point, additional criticism is related to time inconsistencies, in the sense that the optimal portfolio selected at time t, is no longer optimal in subsequent time steps.

In addition, several characteristics of the inputs (expected return means, variances and covariances) have also been criticised, namely: the fact that they are static over the period in analysis; the estimation of the inputs in the future may have questionable accuracy and the optimal portfolios are sensitive to those errors; it assumes that returns have a Gaussian distribution, which is known not to be the case, so, the first two moments of the distribution (mean and variance) are a poor representation of the risk-return profile; and finally, the representation of risk by variance, which considers equally positive and negative returns.

To contrast more directly this model with an approach using RL, we discuss below a number of points. First, in this formulation, the agent (the portfolio manager, the investor, or even the final client), needs to provide in advance a specific desired target return or target risk. Similar to what we have seen in Section 2.7.1, in this type of solution the focus is not in the ultimate objective (to maximise return or maximise risk-adjusted return), but to minimise risk for a desired, predetermined return (or maximise return for a desired level of risk).

Second, the mean-variance model is also not compatible with the concept of outperforming the assets we have in the portfolio. In fact, we are inputting a target return, based on the expected returns of the assets, and we cannot set a target above the best performing asset in the portfolio, given the nature of Markowitz's efficient frontier. The exception occurs if we allow short positions (Luenberger, 1998), in other words, if we leverage the portfolio. In RL we do not have this limitation and the agent will focus only on maximising its objective function.

Third and last, the model takes as input the expected mean returns and covariances, resulting from forecast or simply calculated from historical prices only. It cannot, for example, use additional information from other variables, or benefit from the information contained in the sequence and shape of the historical values used. In essence it is a relatively rigid model, in contrast to the model-free type of architectures in RL, that have all the flexibility required.

Finally, we should refer that departing from Markowitz's work, an extensive number of adaptations have been made in order to overcome some of its drawbacks Zhang et al. (2018); Hambly et al. (2021). However, they fall outside the scope of this review.

2.7.3 Reinforcement learning concept

RL is the third area of machine learning together with supervised and unsupervised learning. It can be used whenever we have sequential decision-making problems, as in the case of trading and portfolio management, and we want to find an optimal strategy to achieve a certain objective.

In more detail, the objective of the agent (Figure 2.10) is to learn what action to take, by trial-and-error, given the conditions or state of the environment. Hence, at each

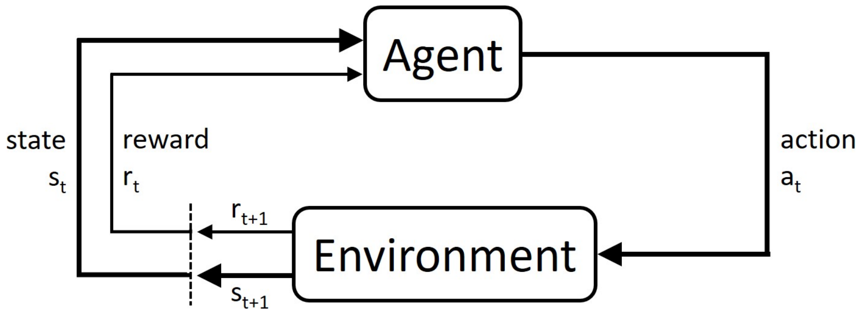


FIGURE 2.10: Simplified agent-environment interaction in RL (Sutton and Barto, 2020).

time step in the learning process, the agent receives (or observes) the state of the environment and also receives a delayed feedback reward corresponding to the last state transition. In fact, the two most important characteristics of RL are precisely the trial-and-error search and the delayed reward (Sutton and Barto, 2020).

With knowledge of the environment state and the reward received, the agent can calculate the action that will be sent to the environment. As a result, the system will transition to the next state, and the whole cycle is repeated. The final objective of the agent is to learn how to control this dynamic system (the policy), in order to maximise the cumulative rewards. Indeed, the agent is the RL system component responsible for updating and improving the policy. For this purpose, the agent comprises an algorithm which is the "brain" of the overall system. The main types of algorithms available will be presented in Section 2.7.4.

For a top-level comparison, in supervised learning we map features to targets, whereas in RL the mapping occurs from state to action, or to a probability distribution over actions, maximising reward. Thus, RL is fundamentally different from forecasting asset prices or yields. Above all, there is a perfect alignment of the objectives of the RL system user, and the agent controlling that system. This is something that was missing in previous frameworks (Sections 2.7.1 and 2.7.2). Apart from the ultimate objective alignment, we have the flexibility of a framework that does not require models of the environment in which it operates. Thus, the need to build forecasting models is eliminated. This is achieved with model-free type of algorithms (more in Section 2.7.4) and the introduction of function approximators, both being components of the agent.

From a historic perspective, RL has been present since the early 1980s (Sutton and Barto, 2020). But recently this area has managed to progress at a faster pace. Reasons for this may be found in the advances in computational resources, together with the developments in machine learning, and in particular deep learning (Goodfellow et al., 2016), feeding through RL via the introduction of deep neural networks as function approximators. And it was only recently, that it has been possible to stabilise the learning with deep neural networks through the work of Mnih et al. (2013, 2015). We will elaborate on the techniques used in Section 5.3.2.2.

These developments have translated in practice into brilliant results achieved in some areas. Some of the most emblematic were achieved in games, such as the game of Go (Silver et al., 2016), and Atari (Mnih et al., 2015), where the RL agent was able to reach professional players' performance and, in some cases, outperform them (beating the European Go champion). But impressive results have also been achieved in robotics (see for example Andrychowicz et al. (2020)).

2.7.4 Agent taxonomy

The first distinction of the type of RL agent, and consequently, of the type of algorithm, is between model-based and model-free. Model-based RL agents are used when we have a model that totally represents the workings of the environment, or it will be an additional task of the agent to learn. This model needs to be capable of predicting both state transitions and rewards. It has been shown that, for systems with relatively simple dynamics, this type of RL is appropriate, being more data efficient, finding better policies and handling changing objectives better than the model-free counterparts. However, they are subject to errors in case of inaccurate model representations (Ray and Tadepalli, 2010). For financial markets, the model-based approach is usually not considered, although there are exceptions, such as the work of Yu et al. (2019).

Regarding model-free agents, we have three main types of architectures that can be used, as the diagrams in Figure 2.11 indicate.

In the first group of agents, they use in their algorithms a value function. In this case, the agent is said to be value-based and this type of architecture correspond to a *critic-only*. In this group, the value function is used to determine the value of each possible action from a given state, so that the action with the highest value is then selected (policy). For this reason, it is used for discrete action spaces, and it becomes computationally expensive when the number of discrete actions is high, due to the "curse of

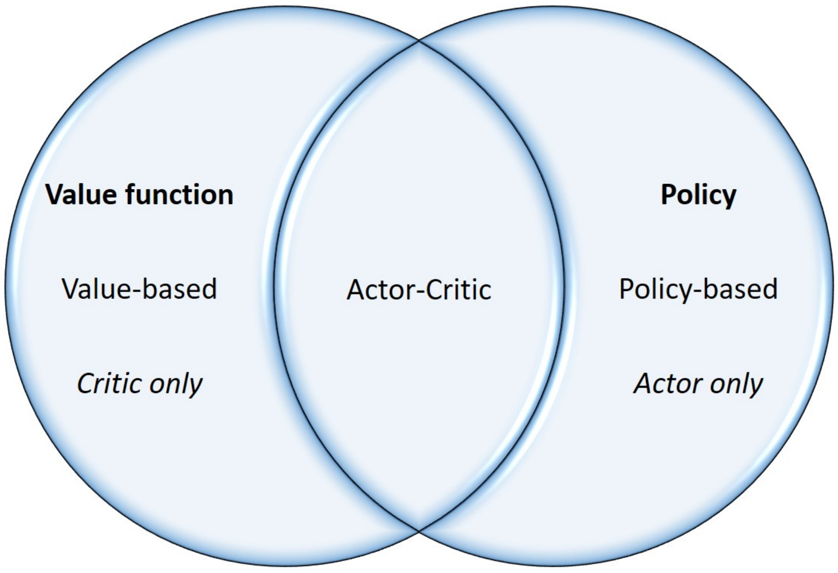


FIGURE 2.11: Types of model-free RL agent.

dimensionality" (we will explain this concept in more detail in Section 5.3.1.1). The critic-only type is the most commonly used in the literature (Fischer, 2018).

The second group uses a function to map state to actions, i.e. the agents focus directly on the policy. These agents are known as policy-based and this type of architecture corresponds to an *actor-only*. Policy improvement and function parameters' updates are obtained by maximisation of the objective function, using gradient ascent. Contrary to the previous group, this type of agent can deal with both discrete and continuous action spaces. This is also a common type of agent and some of the early work was produced by Moody and Wu (1997).

The last group is known as the *actor-critic*, and is a combination of the previous models. Thus, they incorporate both an actor and a critic, with the corresponding value and policy functions. The overall idea is to have both the actor and the critic in the same agent, interconnected and working with the same overall objective. In this way, the actor focuses on the policy and determines the agents' actions, but taking into account the work of the critic. In turn, the critic judges the actions taken by the actor, by estimating the value of the state-action pair, while trying to improve its estimations. Contrary to the critic-only case, the value function is used to estimate the value of the single action taken by the actor, and not of all possible actions. As a result, most algorithms in this class can be used for both discrete and continuous action spaces.

The actor-critic class is the latest type of state-of-the-art algorithms, where less published work is available. In fact, a recent survey of RL in financial markets (Fischer, 2018), confirms that this is the least researched type of algorithm in financial applications. This may be related with the higher complexity of the corresponding algorithms and the fact that most of them were developed recently. Nevertheless, it is somewhat surprising given their potential advantages.

Additional information about the policy, the value function and algorithms will be presented in Chapter 5, targeting the specific algorithms that are relevant for our empirical work.

2.7.5 RL applications in portfolio management and trading

In this section, we present some examples of recent studies where RL is used for portfolio management and trading, sub-dividing the review by type of agent and algorithm used and the target asset class, and followed by a brief discussion.

2.7.5.1 Critic-only

In the critic-only arena, Q-learning dominates the early works in this area. The model was developed by Watkins (1989) and Watkins and Dayan (1992). Recognising the early interest from investors in computerised portfolio management, Neuneier (1996) adapted Q-learning to this problem and introduced a function approximator to represent the action-value function. This initial work was subsequently developed by the same author (Neuneier, 1998), simplifying the action-value representation and allowing model-free policy iteration. The application was specifically for asset allocation, using real market data from the Genman DAX index (equities). The results showed a 18.5% outperformance at the end of the testing phase (of almost three years). It should be noted that this asset allocation problem includes only one asset (DAX index), which is not the typical multi-asset portfolio management problem. In fact, it is closer to the trading activity than to portfolio management. In addition, the action state is totally discrete, with only “invest” and “not-invest” options. The discrete action space is a common characteristic of all studies using the critic-only type of algorithm, due to the need to perform a greedy maximisation of Q at every time step. Nevertheless, the studies may include different levels of discretisation (finer or coarser).

This type of RL architecture has been used since then and the initial Q-learning has evolved into more sophisticated algorithms. Pigorsch and Schäfer (2021) used the Deep Q-learning (DQN) algorithm developed by Mnih et al. (2013, 2015). Its application was on 48 US equity portfolios, including between 10 and 500 stocks each. The input data comprises the stock historical prices and fundamental indicators extracted from the companies’ quarterly statements. Regarding the agent’s actions, they are also discrete, of the type invest ($a_t = 1$) and not-invest ($a_t = 0$). The algorithm results outperform the benchmarks considered (equal-weight buy and hold, and simplistic momentum and reversion strategies), in 75% of the portfolios. However, the results also show that the agent could not avoid the big market correction (in February–March/2020) included in the data, underperforming during this period.

In the same line of model, Théate and Ernst (2021) adapted the DQN algorithm, calling the modified version the trading DQN (TDQN). The application carried out is in algorithmic trading, applied individually to several equity indices and individual stocks, covering several sectors and world regions. As inputs to the system they used historical data for the periods 2012–2017, training data, and 2018–2019, testing data. With respect to the agent actions enabled in this model, the authors considered three discrete actions, where the agent can buy, sell or hold. The results of this study were promising, but mixed, showing outperformance in some stocks but not in others. Overall, the TDQN algorithm only barely surpasses the passive buy and hold strategy.

Finally, we also present the recent study from Park et al. (2020), which overall produces slightly better results. This study applies the DQN algorithm to management of two equity portfolios composed by three assets each: US equities' ETF funds (first portfolio) and Korean stock market indices, assumed as tradeable (second portfolio). Adopting discrete action space, they demonstrate that the model outperforms their benchmarks, specifically: buy and hold strategy; randomly selection; momentum strategy, looking at the previous period return only; and finally, reversion strategy, the opposite of the momentum strategy. These benchmarks are relatively simplistic and no information is provided regarding the comparison of their results versus the performance of the best asset in the portfolio as a reference. Nevertheless, they present good results, being a promising outcome for this type of application.

2.7.5.2 Actor-only

Contemporary to the work of Neuneier reported earlier, Moody and Wu (1997) have pioneered the approach to portfolio management optimising directly an objective function that measures investment performance. This approach is also called direct RL, in contrast to indirect RL, where the optimal policy is obtained "indirectly" by solving the Bellman equation (Bellman, 1957; Guan et al., 2021). The authors also developed the concept of differential Sharpe ratio as new reward signal, which enables this type of agent to learn efficiently online at each time step, rather than having to wait until the end of the episode. This addresses one of the problems pointed out to the actor-only architecture.

The corresponding experimental work, targeted first a portfolio of the S&P 500 index (assumed as tradeable) and US 3-month Treasury Bills, considering monthly data over 45 years (25 years for testing), and enabling leverage of 2:1 in the portfolio (Moody et al., 1998). This was benchmarked against Q-learning in Moody and Saffell (1999) and the buy and hold strategy. In subsequent studies, they also targeted forex, specifically the currency pair US dollar / British pound, using 30-minute data, for a period of 8 months during 1996 (Moody and Saffell, 2001). Overall, the results demonstrate outperformance in relation to the benchmarks, including the Q-learning algorithm. In addition, in a parallel simulation with artificially generated data, the authors also demonstrated superiority versus a supervised method, minimising the forecasting error.

It is worth referring that, although the the actor-only architecture is fundamentally different from supervised learning (as RL in general, see Section 2.7.3), this is the simplest RL problem representation, and conceptually closer to the supervised learning approach. In fact, recent studies inspired by the work of Moody and Wu (1997), have followed exclusively deep learning techniques to optimise directly the weights of the assets in the portfolio and bypassing the need to forecast expected returns, the traditional method in supervised learning (Zhang et al., 2020, 2021). For this purpose they consider directly objective functions such as Sharpe ratio, portfolio variance or a general expression translating the mean-variance problem. This is an interesting example where the frontiers between deep learning and RL with deep learning function approximators are somewhat diluted.

2.7.5.3 Actor-critic

In this class of agents, one of the most comprehensive studies of state-of-the-art algorithms applied to portfolio management is the work of Aboussalah et al. (2021). The authors started by considering a very wide range of algorithms suitable for continuous action spaces, namely: Deterministic Policy Gradient (DPG); Stochastic Policy Gradient (SPG); Deep DPG (DDPG); Trust Region Policy Optimization (TRPO); Proximal Policy Optimization (PPO); Twin Delayed DDPG (TD3); Soft Actor-Critic (SAC); and Evolution Strategies. However, initial results were unsatisfactory, with most of the algorithms demonstrating difficulties converging during the training phase. As a result, the authors decided to streamline the initial list, selecting four algorithms that had the most promising performance for further studies, specifically: DPG, SPG, DDPG, and PPO. The portfolio included ten stocks from the US S&P 500 index and the data was historical open, low, high, close prices, and volume, for the years 2013 and 2014. Despite the initial filtering of algorithms to the most promising ones, the final results were disappointing considering that most of the actor-critic models could not outperform the benchmarks, namely buy and hold strategy and dynamical multi-period mean-variance portfolio. Moreover, the DPG model was the one obtaining the best performance.

Snow (2020a,b) also presented an implementation of the DDPG algorithm to portfolio management of a group of fifteen stocks from the US stock market. The analysis covers the first ten months of 2018 using historical minute data, and the agent is enabled to trade every seven minutes. In terms of results, the agent underperforms the market for most of the months in analysis, in some cases significantly.

The DDPG algorithm has been used in at least two additional studies, with contrasting results. The first from Xiong et al. (2018) is an application to stock trading, using as input data the daily prices of the thirty stocks composing the US Dow Jones index. They cover the period from 2009 to 2018, with the testing phase between 2016 and 2018. Despite using an algorithm that permits continuous actions, the authors considered a discrete action state of the type buy, hold and sell. The results in this case demonstrated a significant outperformance of the mean-variance and buy and hold benchmarks, with final annualised returns of approximately 26% versus 16% for the benchmarks.

The second study on DDPG used for portfolio management by Liang et al. (2018) also included two additional algorithms: PPO and policy gradient. The target asset class was equities and they used market data from five stocks from the Chinese stock market. As far as results are concerned, only the policy gradient agent outperformed their benchmark. Notably, with the DDPG and the PPO models the authors reported that the algorithms could not converge to an optimal policy during the training phase.

In strong contrast with the latest example, Betancourt and Chen (2021) adapted PPO to manage a portfolio of cryptocurrencies. Considering that their approach can deal with portfolio management of a dynamic number of assets, they named their method DNA. Their study included a variable number of cryptocurrencies, from 3 to 85, covering a period between 2017 and 2019. The inputs to the model are historical prices (open, high, low and close) and two types of volume, with different frequencies: 30 min, six hours and one day. The authors benchmarked their results against three previous studies selected from the literature (Jiang and Liang, 2017; Bu and Cho, 2018; Pendharkar and Cusatis, 2018), obtained by other researchers using the same asset class (cryptocurrencies). Overall, the results obtained were impressive, with significant outperformance of the benchmarks.

2.7.5.4 Discussion and potential advantages of actor-critic

Target asset class

In what concerns the target asset class considered in RL studies, we have seen in previous sections (2.7.5.1– 2.7.5.3) that the vast majority is applied to equities or equity instruments (Neuneier, 1996; Moody and Wu, 1997; Liang et al., 2018; Xiong et al., 2018; Park et al., 2020; Snow, 2020b; Théate and Ernst, 2021; Aboussalah et al., 2021; Pigorsch and Schäfer, 2021). Lately, an increasing number of studies is also dedicated to cryptocurrencies as it has become a popular asset class and research topic (Jiang et al., 2017;

Bu and Cho, 2018; Betancourt and Chen, 2021). We can also find published work targeting forex, although on a less frequent basis (Moody and Saffell, 2001; Dempster and Leemans, 2006).

Other researchers target simplified mixed portfolios of equities and bonds. One such example is the work of Pendharkar and Cusatis (2018) investigating the use of RL in personal retirement portfolio management. The authors consider a portfolio with only two assets, one from the equities class and the other from bonds, specifically: the S&P 500 ETF and either the US Aggregate Bond ETF or the 10-year US Treasury note. In a different study, Almahdi and Yang (2017) propose an RL method to produce buy and sell signals and asset allocation weights, using a portfolio made of five assets represented by five of the most commonly traded ETFs, of which four are equity ETFs and one from the bond asset class.

Finally, to put our own work in context, it is worth emphasising that no study was found where the target asset class is specifically bonds or other financial securities in which the composition is exclusively bonds.

Quality of results

As far as quality of results is concerned, the literature is mixed on this type of RL application, showing that algorithms do not always work as expected, and sometimes the results are suboptimal. While some studies showcase good results (Neuneier, 1996; Moody and Wu, 1997; Xiong et al., 2018; Betancourt and Chen, 2021; Pigorsch and Schäfer, 2021), others are not able to outperform their benchmarks, reporting convergence, stability and overfitting problems (Liang et al., 2018; Snow, 2020b; Aboussalah et al., 2021). This overall picture clearly demonstrates the need for additional research in this field.

Type of agent

The actor-only type of agent directly optimises the objective function the investor is interested in. This fact makes it conceptually stronger and thus, it tends to be more stable and reliable (OpenAI, 2022b). In fact, it is the simplest RL problem representation, avoiding the "curse of dimensionality" (we will discuss this concept in more detail in Section 5.3.1.1).

In contrast, critic-only methods optimise the performance of the agent "indirectly" by learning a value function and solving the Bellman equation. As a result, this approach tends to be less stable and many possible problems have been pointed out. Hence, Sutton and Barto (2020) refer the "deadly triad" of function approximation, bootstrapping, and off-policy data, which cause instability in value-learning algorithms when they coexist. If only two of those are present instability can be avoided. The same potential problem has been reported by Tsitsiklis and Van Roy (1997), when using function approximators and temporal difference learning. To clarify, *function approximators* refer to linear or nonlinear functions to estimate the action value; *off-policy* is when the agent is learning the value of a policy and not using it for control, i.e. when the target policy is different from the behaviour policy; and the concept of *bootstrapping* in RL refers to the case when estimates are updated on the basis of other estimates without waiting until the end of the episode, as in temporal difference learning.

Despite these potential problems, the critic-only benefits from higher sample efficiency, since data can be reused more effectively than with actor-only methods (Schulman et al., 2017a). The authors also demonstrated that under specific circumstances (considering entropy-regularized Q-learning), both methods are equivalent.

The actor-critic type of agent, combines the previous solutions. As a result, it should be able to trade-off between the strengths and weaknesses of each, since they are integrated and use the information from one to improve the task of the other. However, there is much less work we can refer to, so the challenges are higher in this approach. This is compounded by the reports in the literature of difficulties encountered when using this architecture.

2.7.6 RL potential for portfolio management

In this section, we present the main points justifying our selection of RL to address Objective 6 (Section 1.2). We start by pointing out that the literature is very supportive for research in this area. Indeed, there is a considerable amount of research in applications to portfolio management and trading, and the field is still at a very early stage of development. While we feel some reassurance from the success that RL algorithms have been achieving in other areas (Silver et al., 2016; Mnih et al., 2015; Andrychowicz et al., 2020), published results related to our application are also encouraging, making it a challenging option.

In addition, it has been demonstrated that in comparison to alternative methods using supervised learning, RL is more "elegant and effective" for systems where trading frictions are taken into account. Besides, the need to build forecasting models is eliminated, and better performance has been obtained (Moody and Saffell, 1999, 2001).

Furthermore, as a proof of concept in financial applications, it has also been demonstrated that RL is able to find and exploit arbitrage opportunities. Hence, in simulated market conditions, where the existence of arbitrage opportunities was known in advance, Ritter (2017) developed a system where the agent was able to find them in a successful trading strategy, with proper consideration of transaction costs. The author used the Q-learning technique for this purpose.

Finally, in our work we will focus on a specific type of agent (actor-critic), using a specific algorithm (DDPG), and applied to a specific asset class (bonds). All of these correspond to niches where less or no research has been produced.

2.8 Summary and conclusions

The principal focus of our research is on machine learning for forecasting and portfolio management in fixed income markets. We also selected as a target the yield curve, taking into account its importance for bond markets and for the overall economy.

The most common models for time series are the AR, ARMA, ARIMA, ARCH and GARCH type of models. These are based on the assumption of autocorrelation in the movement of the time series, that is correlation between current and lagged values. However, there is some evidence from the literature, in particular for time series of returns, that the autocorrelation is largely insignificant.

On yield curve models, the most important and widely used models are the NS and the DNS models. The latter is a development of the original model and incorporates both intrinsic factors of the yield curve, namely level, slope and curvature, and a restricted number of macroeconomic factors, which are manufacturing capacity utilization, annual price inflation and federal funds rate. These models are undoubtedly valuable and well established in the industry, but for a new research using machine learning techniques, the limitation by those factors is undesirable, mainly the macroeconomic factors.

Overall, the literature review revealed that most of the research carried out on the use of machine learning for financial applications focuses mainly on equities, and also on forex. The coverage of fixed income markets using machine learning is much less significant. This is an opportunity (and a gap) to explore this field with additional research.

Asset classes review

From the studies covered in the asset classes review it was not possible to find a direct solution for modelling the yield curve using machine learning (Objective 2, Section 1.2), with one exception referred in Section 2.2.1. This is a gap in terms of academic research and the aim of our research is to fill this gap. Notwithstanding the potential of spacial / geostatistical models, these models base the whole mapping and forecasting processes in intra and extrapolation using historic yield curve data only. For this reason, we perceive machine learning models with greater potential and flexibility.

Another interesting model covered in this literature review (Objective 2), was an RNN topology called shared layer perceptron. Although the research programme did not include any application directly in the fixed income area, the characteristics of the model appear to indicate high potential for yield curve forecasting, given the multi-asset and multi-step characteristics.

The review was particularly interesting regarding ideas for Objective 5 and for potentially better models: from the use of ensembles to different types of hybrid models, with inclusion of broad information from several sources. Among those they should incorporate macroeconomic, financial and, whenever possible, practitioner type of information.

On single target problems, although the results achieved were sometimes mixed, even with the same type of machine learning model, the following models were reported with positive results, in diverse markets: RBF, MLP, SVM, RF, and RNN.

Regarding features (Objectives 1 and 3), our research has the objective of going beyond the exclusive use of historic data from the variable to predict. Specifically, in the objectives we include the assessment of a wide range of potential explanatory variables from a variety of fields. This is a common theme in the literature, reporting the benefits of including in the models additional information from different sources as independent variables. These may include political, economic, financial and domain-specific factors.

Standard and deep learning models

The machine learning models that will be used in Chapters 3 and 4 for implementation were presented (Objective 5), namely: linear regression, MLP and the LSTM. They cover a range of models in terms of complexity, from the simpler linear model to the more complex, nonlinear standard and recurrent type of neural networks.

Multitask learning

MTL enables the learning of several targets simultaneously and there are reasons to believe that this technique could be used to model and forecast the yield curve (Objectives 2 and 4). Furthermore, the type of applications described in the literature seems to give a good indication regarding its potential use with this objective. It is definitely a research path worth pursuing. To the best of the author's knowledge this is a novel application of ANNs and MTL for yield curve forecasting.

In summary, MTL can be used for yield curve forecasting in various modes. In fact, MTL targets can represent: different points in the yield curve at the same point in time; the yield of a particular bond, at different times; a combination of previous items, enabling multi-asset and multi-step forecast; and additional variables to include in the structure of the network as targets.

Reinforcement learning

Finally, the limitations of supervised learning and of the traditional methods for portfolio management used in the industry led to the selection of RL to fulfill Objective 6 of this work. This selection is also supported by the need for additional research using RL for this type of application, especially in specific niches within the agent taxonomy and having bonds as the target asset. For that purpose, the conditions in RL have developed favourably to support the execution of this work. This is due to the rapid development in the last few years of powerful new algorithms, in part as a result of recent developments in supervised machine learning, feeding through the field of RL. Although the progress has been rapid, not enough research has been done yet, so that these techniques could be used more widely in the industry. The results presented from the literature also ensure that this is a challenging topic, considering the high number of studies reporting unsatisfactory performances.

Chapter 3

Multilayer Perceptrons for Yield Curve Forecasting

This chapter describes the work conducted with MLPs and MTL for forecasting the yield curve. In more detail, we will be describing all phases of the research, covering Objectives 1–5 (Section 1.2), specifically: selection of fixed income asset, the collection and transformation of data, the identification of MLP models, the methodologies used, the testing and experimental results obtained. Finally, a comparison of the results obtained with those found in related works in the literature is also included.

3.1 Data

In this section we describe the dataset used for the empirical work, identifying features, targets and pre-modelling operations.

3.1.1 Targets

The focus of our work is the government bond asset class, which was selected for the following reasons. First, liquidity of this asset class is clearly higher than for the other bond classes. Second, the size of the market is also considerably higher. Third, this class encompasses a wide range of financial instruments available. Fourth and last, research on government bonds will attract the interest of entities such as national and supra-national institutions, in particular central banks, national government agencies managing the public debt, as well as asset management companies. Within this asset class, the Euro benchmark yield curve was selected and Figure 3.1 is its representation across time. Its modelling will be done through the modelling of its most relevant benchmarks, as described in Section 2.5. The benchmarks considered were: 3-month, 2,

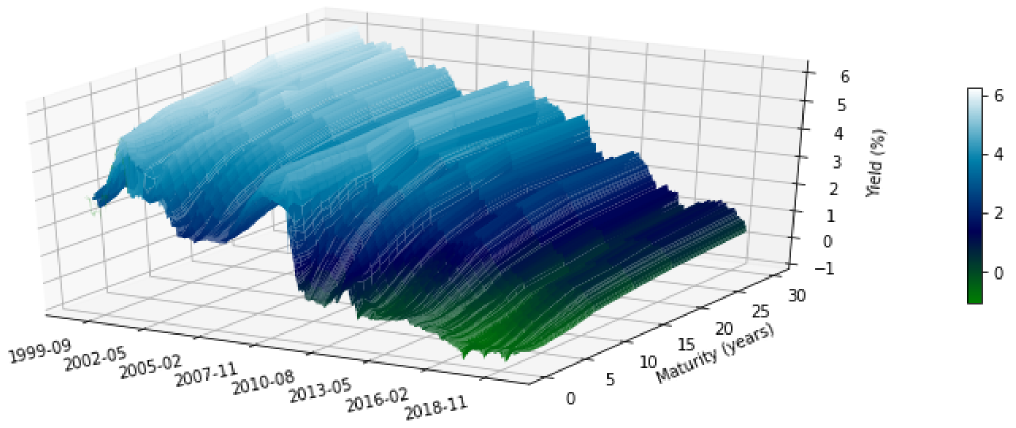


FIGURE 3.1: Representation of the yield curve across time.

5, 10 and 30-year bond yield, representing five targets to be predicted. The corresponding summary statistics is presented in Table 3.1.

TABLE 3.1: Descriptive statistics of the target variables.

Statistic	**Yield**				
	3M	2Y	5Y	10Y	30Y
Mean (%)	1.73	2.00	2.49	3.10	3.70
Standard deviation (%)	1.65	1.74	1.72	1.56	1.47
Kurtosis	1.65	1.65	1.76	2.06	2.24
Skewness	0.18	0.00	-0.29	-0.51	-0.55
Jarque-Bera p-value	0.00	0.00	0.00	0.00	0.00
Augmented Dickey-Fuller p-value	0.20	0.20	0.23	0.22	0.19

All the target time series exhibit kurtosis below the normal distribution (with kurtosis equal to 3), i.e. they present negative excess kurtosis. The skewness is mixed and close to zero. On the Jarque and Bera (1980) test, the results indicate that the time series does not present a normal distribution. In turn, the Augmented Dickey and Fuller (1979) test shows that they are non-stationary. This data is subsequently normalised as explained in Section 3.1.5, to have zero mean and unit standard deviation.

3.1.2 Features

Choosing relevant features is one of the most important factors to improve the performance of models. Given the interconnectedness and mutual influence of various asset classes in the markets, a large number of features from financial markets were considered. These were selected from government bond markets and from related classes and indicators: corporate bonds, equities, currencies, commodities and volatility. Additional features were added, directly calculated from the previous features, mainly bond spreads, slope of the yield curve and simple technical analysis indicators. Furthermore, economic variables are also very important, as clearly exemplified by the well established yields-macro models presented in Section 2.1.2. Hence, a vast range of economic indicators is also included, from different geographic locations. The complete list includes 159 features and, because it is so extensive, is presented in Table A.1 of Appendix A.

3.1.3 Datasets

The dataset was obtained from Bloomberg database and it covers the period from January 1999 to April 2017 (Bloomberg, 2017a). This is a longer period than covered in other studies (Dunis and Morrison, 2007; Sambasivan and Das, 2017). From the markets' point of view, this is an interesting period to study, spanning from the euro inception date on the 1st of January, 1999. This is also the starting date for most time series of the Euro benchmarks, in particular the yield curve data. Additionally, this period covers several temporary bull and bear markets and market moving events, such as: the dot-com bubble in 2000; the global financial crisis of 2008-2009, the Great Recession; the subsequent European debt crisis; the European recession in 2012-2013; and several phases of quantitative easing by the US Federal Reserve, European and UK central banks. Of note is the fact that, the principal overall trend in the bond market during this period has been of declining yields, although with significant and frequent temporary reversals.

Regarding data frequency, the selection was daily closing values, which are easily available for financial assets in general. For economic data, only final values were considered. Besides, in order to have an uniform dataset, needing less initial cleaning and analysis, the same daily imposition was used for all macroeconomic variables, carrying over previous values when their frequency is lower.

3.1.4 Generation of additional features

In financial time series there is a natural temporal order that cannot be disrupted during modelling, since that ordering has in itself relevant information. Taking this into account, it is worth incorporating into the models past values of the time series. Hence, new features are generated from the original ones, corresponding to the lagged values of the respective time series. In our research, six time steps were considered, corresponding to one working-days week plus one day from the previous week. A similar window size has been used in previous studies (Mahler, 2009; Brownlee, 2020). These generated features are subject to the feature selection process, described in detail in Sections 2.4.2.1 and 3.2.2.

3.1.5 Train-test split and normalisation

As is common, we divided the data into two groups, for training and testing the models. In this case, a 70% / 30% split was considered. The resulting groups are the initial "static" training and testing datasets. Preliminary simulations to determine some hyperparameters to be used in the final experimental work (for example the assessment of number of hidden units to consider in the MLP model), were done with this static training dataset to avoid any use of testing data for this purpose. However, the final training of models was always conducted with the use of dynamic moving windows of training data. These will be explained in depth in Section 3.2.6. Given the long period covered by the overall data, both training and testing datasets are significant in size and encompass several bull and bear cycles (3296 and 1413 observations, respectively).

Finally, all data was normalised by subtracting the mean and dividing by the standard deviation of the training dataset. This is also essential, given the wide range of features we are considering, which have very different scales in some cases.

3.2 Methodology

In this section, the details of the methodology adopted are presented, including various analyses carried out in advance to the modelling process to justify the parameters adopted (Section 3.2.3 for hyperparameter "number of hidden units"; and Section 3.2.6 for hyperparameter "moving window size"). Then, all models considered in this study are detailed. Finally, the concepts of moving window, retraining of models and cross-validation are described. A global view of the empirical work carried out is summarised in Table 3.2 and explained below.

3.2.1 Forecasting horizon

Given a specific training dataset, forecasting the next value in the time series should be less complex than forecasting further into the future, when the time distance to the known data increases. Taking this into consideration, a forecasting horizon parameter was introduced in this study, equal to the number of days, or time steps, from the next value of the time series. In practice, a forecasting horizon equal to zero corresponds to forecasting the next value, i.e. one time step ahead, while a forecasting horizon equal to 20 corresponds to predicting the next value plus 20 days ahead. Our research was conducted using a range from 0 to 20, with 5 days increment (Table 3.2). The next day plus 20-day range (working days) was considered as it corresponds to one month, approximately. These limits have also been used in other studies (Arrieta-Ibarra and Lobato, 2015).

TABLE 3.2: Summary of empirical work.

Parameters	
Original features	159
Targets	3M, 2Y, 5Y, 10Y, 30Y
Forecasting horizons	0 (next day), 5, 10, 15, 20 days
Analyses	
Regularisation param.	0 to 4, step 0.1
Selected	2 and 4
Moving window size	30, 100, 300, 500, 1000, 2000, 3000, 3290
Selected	3000
No. of hidden units	5, 10, 20, 50, 100, 150, 200
Selected	10
MTL mode	Yields as targets
	Forecasting horizon as targets
Models	
LR Linear Reg	Linear regression
1. NN GenFeat	MLP with all generated features
2. NN RelFeat	MLP with relevant features
3. NN TgtOnly	MLP with target data only
4. NN RelFeat+LRdata	NN RelFeat with synthetic data from Linear Regression model
5. NN TgtOnly+LRdata	NN TgtOnly with synthetic data from Linear Regression model

3.2.2 Feature selection

It should be emphasised that the most relevant features for each target yield are not known in advance and this is why this study included a wide range of original features (Table 3.2) to be submitted to feature selection. Linear regression using the Lasso method was performed to select the most relevant features (Equation 2.14). A range from 0 to 4 was considered for the regularisation parameter γ, with values 2 and 4 being selected. This selection will be explained in detail when discussing the results in Section 3.3.1. Furthermore, as the impact on relevant features can change for different forecasting periods, we determine the relevant features separately per target and per forecasting horizon, resulting in a total of 25 combinations.

3.2.3 Number of hidden units

An additional analysis was conducted to define the number of hidden units to use in the neural network model. Based on the previous literature (Section 2.2), the specific range of hidden units shown in Table 3.2 is explored. This analysis was conducted for each target and for both single task and multitask learning. For this selection, the static training dataset was divided into a training and validation datasets, again using the traditional 70%/30% split. The training dataset was used to train the models and the errors calculated on the validation dataset were used as the selection criteria for the final value of the number of hidden units. This procedure was followed to avoid a common mistake of using the final testing dataset for choosing parameters, which may give an unfair advantage to the model. No data from the static testing dataset was used to select the number of hidden units. The main conclusion from the results is that 10 hidden units is a good compromise for the subsequent studies, with significant overfitting observed for neural networks with more than 100 units.

3.2.4 Single task and multitask learning

The modelling was carried out using the concepts of MTL described in Section 2.5, both in multitask learning mode, that is, considering simultaneously all targets in the same model, and through problem transformation into five single task learning models. For the multitask learning mode, two analyses were considered: with yields as targets (multi-asset forecasting) and forecasting horizon as targets (multi-step forecasting).

An additional issue in MTL, is that a solution needs to be found for the relative importance among target variables. This can be done through the consideration of different weights, using Equation 2.15 (Caruana, 1997). In that study, a range of weights (from 0 to 2) for the extra tasks was tested and the best results were obtained for values of the parameter λ around 1, meaning all tasks having equal weight and importance. Bearing in mind these results, and the fact that this differentiation between main and extra tasks may not be necessary in the case of backpropagation neural networks, all results presented subsequently are considering all task with the same importance (λ parameter equal to 1).

The implementation of the various techniques of MTL is particularly important in the modelling of the yield curve, because it takes into consideration the functional form of the curve and the high levels of correlation between adjacent yields in the curve.

3.2.5 Models

All models studied in our research are listed in Table 3.2. The multivariate linear regression model is used as the baseline for comparison with the MLPs. This model uses the Lasso regression described in Section 2.4.2.2 to perform feature selection and identify the most relevant features as explanatory variables, which are then used in the final multivariate linear regression model. For this reason, it represents a stronger baseline to beat.

The following three main models considered use the MLP architecture. Model 1 uses the complete list of generated features. Model 2 uses the relevant features determined during the feature selection process (features selected per target and per forecasting horizon). Finally, in Model 3 only past values of the target(s) to predict are taken into account, i.e. a univariate type of model.

In addition, we observed that results obtained with linear regression performed surprisingly well in some cases (further details in Section 3.3.2). For this reason, it was decided to test the performance of hybrid models, incorporating the alternatives referred above with better performance (Models 2 and 3) and synthetic data generated by the linear regression model, used as additional feature(s). In more detail, Model 4 (NN RelFeat+LRdata) and Model 5 (NN TgtOnly+LRdata), are constructed using models 2 and 3 as base, respectively. But they differ from those by the fact that they extend the set of features used in the base model, with additional feature(s) artificially generated by another model, in this case the linear regression model. Hence, they incorporate one additional feature for each target (when in single task learning), or five additional features for all targets (when in multitask learning). New MLP models are then run for the new set of data.

When forecasting beyond one step ahead, for longer forecasting horizons, there are two methods that can be used: direct or iterative forecasting. On the one hand, in the direct forecast only current and past data is used to forecast directly the time step required, using a horizon-specific model. On the other hand, in the iterative forecast a one step ahead model is iterated forward until the target forecasting horizon is reached. In this case, recent predictions are included as input to predict the desired target time step.

All our models use direct forecasting of targets. Iterative predictions are known to work best when the time series is generated by a nonlinear dynamical system which can be written as a mathematical formula, as shown in very early work by Wan (1993). However, this is not the case with financial time series and prediction errors tend to propagate fast if we were to do iterative predictions. As a result, new neural networks are built for each forecasting horizon, both in single task and multitask learning, when using yields as targets.

3.2.6 Moving window and retraining of models

One of the characteristics of financial time series is that they become available at a specified frequency, in this case on a daily basis. As the new information becomes available, it can be incorporated in the models which are then retrained using the new training dataset that results from eliminating the oldest values and including the newly available values. Hence, the training dataset is a moving window of historical data up to the time step being considered, corresponding to the last known data. The retraining of models using a moving window is feasible in real time and the technique was used to take full advantage of the models.

Furthermore, due to the large size of the testing dataset and the computing time necessary to retrain the models and forecast all points, only fifty random points were selected from the unseen testing dataset for forecasting error calculations (out of sample error). Figure 3.2 is a graphical demonstration of the moving windows.

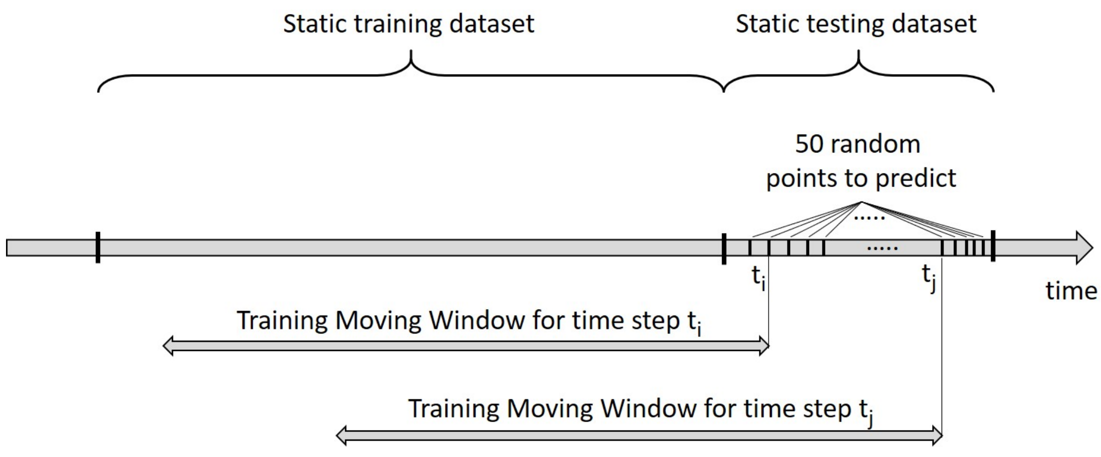

Static training dataset
Static testing dataset
50 random points to predict
t_i
t_j
time
Training Moving Window for time step t_i
Training Moving Window for time step t_j

FIGURE 3.2: Moving window methodology. Note that at any time step t (present time for the correspondent time step), all data up to this point is historical data and is incorporated in the training moving window for better results.

The moving window size is another parameter that needs to be set and we performed a sensitivity analysis to study its impact on forecasting errors, using the range shown in Table 3.2. The same procedure described in Section 3.2.3 was followed to avoid using data from the static testing dataset (Figure 3.2) to select the moving window size. In short, the static training dataset was divided into a training and validation dataset, and the errors calculated on the validation dataset were used as the selection criteria for the final value of this parameter. The results showed that better predictions were obtained with larger windows, with a significant improvement until it reached 2000 observations and then the benefits were much smaller. A final moving window size of 3000 observations was used.

3.2.7 Cross-validation

In classic regression problems, cross-validation is often done using k-fold (usually 10 folds), which randomly splits the data into training and validation sets and uses these partitions to run the model. The process is repeated for all k folds. However, in time series data this approach is not correct because we have to respect the order of the time series. In particular, it makes no sense to take the data and randomize partitions into 10 folds and then train on 9 and validate in the 10th partition, because in some cases we will be forecasting backwards, using future data to predict past data (for example when we forecast fold number 1, using folds 2 to 10).

Another method frequently used is the repeated random subsampling validation, also known as Monte Carlo cross-validation (Picard and Cook, 1984). In this method, the training dataset is randomly split into training and validation datasets a selected number of times. Hence, in each iteration we randomly draw without replacement new training datasets with the remaining data considered as validation dataset. In this respect Monte Carlo cross-validation is close to the concept of bagging (Breiman, 1996), further described below, but the extraction of samples is performed without replacement. An estimation of out-of-sample forecasting errors is obtained by fitting the model in the new training datasets and determining the errors in the validation datasets.

When compared to k-fold cross-validation, the splitting into training and validation in the Monte Carlo cross-validation do not depend on the number of folds chosen, enabling additional flexibility in terms of size and number of training/validation datasets (Barrow and Crone, 2016). It has also been shown to be asymptotically consistent (i.e. probability of selecting the best model converges to 1 as the number of observations tends to infinity) and less susceptible to overfitting (Shao, 1993). However, this method does not solve the problem found in the k-fold cross-validation of including backwards forecasting. Additionally, and despite the fact that in each iteration both methods generate mutually exclusive training and validation datasets, in different iterations there is overlap between data being used in training the models and in validation.

The cross-validation methodology we use instead is based on the concepts of bootstrap (Efron, 1979; Efron and Tibshirani, 1993) and bagging (Breiman, 1996). The first concept, bootstrap, consists of extracting samples randomly from the original dataset, with replacement and of the same size of the original dataset. It is a powerful statistical tool that enables the quantification of uncertainty around the forecasting error using a particular model. The second concept, bagging, is directly related to bootstrap. It uses bootstrap to generate samples to train multiple predictors and then the results are combined by averaging or voting. This is a well-known and effective ensemble technique, where the diversity in the predictors is a result of the different random bootstrap samples. Recall from Section 2.3 that considering the diversity of ensemble members as one of the most important drivers of the success of ensembles, a recent study on fourteen different real-world time series (Oliveira and Torgo, 2014) concluded that the models where additional diversity was introduced showed better prediction accuracy.

Considering the concepts explained above, we do the cross-validation dynamically using a moving window that incorporates all data available up to the present moment for the corresponding time step, i.e. considering all historical data up to that time step. From the corresponding moving window we extract twenty different bootstrap samples, by sampling with replacement and of the same size of the original moving window, i.e. 3000 observations (Section 3.2.6). The selected number of twenty bootstrap samples seems appropriate taking into consideration previous research work, where a range of bootstrap samples was tested (Kohavi, 1995).

Then, the time series is reordered chronologically and we proceed with training the models for each bootstrap sample. The forecasting error calculations are always carried out for the same fifty random points of the testing dataset (Figure 3.2). This is an important point since we aim to evaluate all models when forecasting exactly the same out-of-sample points for a fair comparison of models, objective which by itself would exclude all the previous cross-validation methods mentioned above in this section, namely the k-fold and Monte Carlo cross-validations.

In summary, our cross-validation method quantifies uncertainty around the model metrics obtained, having the following advantages. First, it introduces diversity in the training dataset through the bootstrapped samples, which has been found to lead to better forecasting accuracies (Oliveira and Torgo, 2014). Second, it ensures a fair comparison of models, since they are forecasting yields for the same dates, the fifty fixed random points. Third, it also guarantees that we only use past data to predict future data, unknown to the model, avoiding one of the problems in the other methodologies referred previously. Fourth, by calculating the metrics directly on testing data, we also avoid any type of overlapping, in different iterations, between training data and data used for calculating the forecasting errors. Fifth and last, we calculate out-of-sample forecasting errors instead of an estimation of them obtained using the other methods described. Overall, cross-validation is especially important for the neural network model, but in order to have a fair comparison, the same methodology was followed for the linear regression model.

3.2.8 Model comparison metrics

The main metric used for presenting the results was the mean squared error (MSE), which is commonly used for this purpose. Nevertheless, other metrics were also calculated: mean absolute error (MAE), mean absolute percentage error (MAPE) and root mean squared error (RMSE). Although the latter evaluates the same information about model performance as the MSE, we include it for convenience as having the same unit as the targets may in some cases be helpful to understand more directly the magnitude of the error versus the real variable. Additionally, the statistical significance of differences was determined for all possible combinations.

These metrics were calculated in two different forms: normalised and non-normalised. As mentioned in Section 3.1.5 the data was previously normalised. In the normalised version of the metrics, it was calculated directly from normalised yields (real and predicted). In the non-normalised version, the yields were converted back to real yields and then the metric determined. The results are presented using the normalised metric since the non-normalised equivalents are scale dependent and, consequently, depend on the period we are analysing and the level of yield at that particular period. Hence, the normalised metric is used to facilitate the comparison of models in the literature.

3.3 Results and discussion

In this section the main results are presented and discussed, divided into two separate topics. First, we present the feature selection results to identify the most relevant features. Second, a thorough comparison of the models used and their variants is carried out, together with a comparison to results from the literature.

3.3.1 Feature selection

A typical example of the feature selection results obtained, in this case for the 30-year bond yield, is shown in Figure 3.3. As can be seen, it is not necessary to examine a larger range for the regularisation parameter γ, because it starts stabilising very quickly with a small number of features.

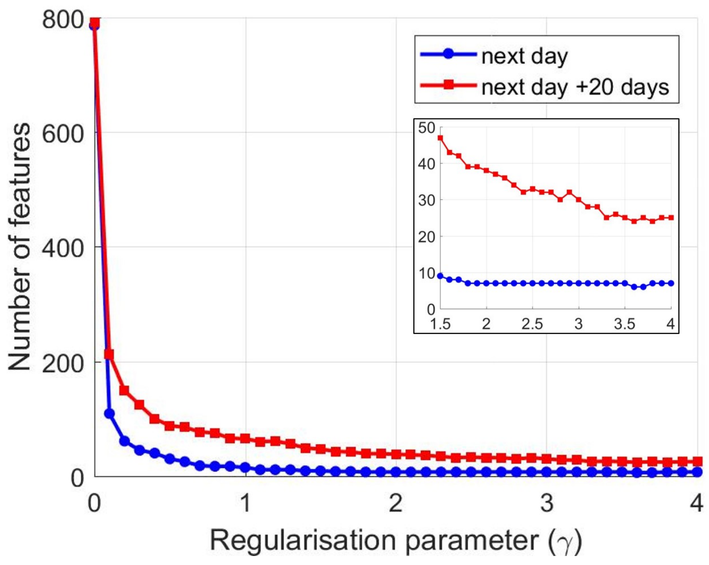


FIGURE 3.3: Change in the number of features as a function of the linear regression's regularisation parameter (γ), for target 30-year bond yield. Inside chart: zoom in range γ = 1.5 to 4.

However, since we are considering five different targets and most of the relevant features are not common to all targets, the total number of features to consider for the simultaneous modelling of all targets in MTL mode increases significantly, in relation to the number of features to consider in single task learning. For this reason, experiments with two selections of features were conducted: using linear regression with γ equal to 2 and 4. The latter value of the regularisation parameter further reduces the number of relevant features to consider in the models. In fact, despite the stabilisation trend shown in the plot (Figure 3.3), the number of relevant features continues to decrease with γ, in particular for predictions further away in the future. The results presented henceforth refer to the feature selection with $\gamma = 4$ (results with $\gamma = 2$ are not included in this work because they do not provide any additional information to the main findings). Comparatively to those obtained with $\gamma = 2$, it leads to better results, with lower spread, mainly for the longer maturities considered (5, 10 and 30 years). The much higher number of features using $\gamma = 2$ tend to result in some overfitting of the neural network model.

An analysis of the feature selection results reveals that the relevant features depend on both the target yield to predict and the forecasting horizon. Table 3.3 shows the top relevant features per target, selected by weight above 0.01 and when they remain relevant in at least 4 of the 5 forecasting horizons studied.

For all targets, there is a dominant feature which is the last value of the target to predict. This is an expected result, since the last value should reflect all information available to the markets. This strong dominance is clear for the one step ahead forecasting but rapidly diminishes as the forecasting horizon increases and additional features are included in the model. Apart from this dominant feature, additional relevant features tend to come from assets with the same or adjacent type of maturity.

Now we will analyse the relevant features across target yields and across forecasting horizons. On the one hand, for a specific target, the number of features increases with the forecasting horizon (see Table 3.4). Most of the relevant features for one step ahead predictions remain relevant for forecasting more distant future values, but additional features are required for those more distant and more complex predictions. On the other hand, considering a particular forecasting horizon, each target yield tends to have a specific set of features. Adjacent targets may in some cases have some equal relevant features, but rarely does a specific feature remain relevant across the yield curve for all targets. The last row of Table 3.4 shows that the number of relevant features necessary to model all targets simultaneously (MTL) increases continuously with the forecasting horizon, from 31 (0 days) to 90 (20 days).

Overall, the most relevant features for yield curve forecasting (dominant features) are the last available values of the targets. Table 3.3 also stresses the importance of assets with the same or adjacent types of maturity, in particular neighbouring yields of the

TABLE 3.3: Top relevant features per target, considering only those with weights above 0.01 and when they remain relevant in at least 4 of the 5 forecasting horizon studied. Dominant feature in bold.

ID	Feature Name	Ticker	Time step
3M			
4	Interest Rate Overnight	EUDR1T	t-1
5	Interest Rate Overnight	EUDR1T	t
772	Euro Generic Govt 3 Month Yield	GECU3M	t-3
773	Euro Generic Govt 3 Month Yield	GECU3M	t-2
775	**Euro Generic Govt 3 Month Yield**	**GECU3M**	**t**
780	Euro Generic Govt 2 Year Yield	GECU2YR	t
2Y			
45	Generic 2nd 3M Euribor Future	ER2	t
275	Equities Euro Stoxx 50 Index	SX5E	t
780	**Euro Generic Govt 2 Year Yield**	**GECU2YR**	**t**
5Y			
200	Bond Future Europe 2 Year Yield	DU1	t
291	Equities Tokyo Topix Index	TPX	t-4
785	**Euro Generic Govt 5 Year Yield**	**GECU5YR**	**t**
10Y			
210	Bond Future Europe 10 Year Yield	RX1	t
230	Swaps rate 10 Year	EUSA10	t
785	Euro Generic Govt 5 Year Yield	GECU5YR	t
790	**Euro Generic Govt 10 Year Yield**	**GECU10YR**	**t**
30Y			
210	Bond Future Europe 10 Year Yield	RX1	t
215	Bond Future Europe 30 Year Yield	UB1	t
235	Swaps rate 30 Year	EUSA30	t
356	Commodities Corn	C 1	t-4
795	**Euro Generic Govt 30 Year Yield**	**GECU30YR**	**t**

yield curve, to facilitate the forecasting process. Other macroeconomic indicators related to inflation and economic activity, commonly used for forecasting purposes and directly related to the components of a bond yield (Section 2.1), also contributed to some yields or to some forecasting horizons. Likewise, other features related to the European Central Bank balance sheet were also useful. These assume particular relevance especially after the 2008 recession and the introduction by central banks of unparalleled levels of non-conventional monetary policy.

As mentioned, the most relevant features were dependent on both target yield to predict and forecasting horizon. Consequently, the main lesson from the feature selection

TABLE 3.4: Number of relevant features per yield, per forecasting horizon and in MTL mode (simultaneous modelling of all yields).

Yield	**Forecasting horizon (days)**				
	0	5	10	15	20
3M	11	22	23	29	36
2Y	5	18	19	18	23
5Y	8	11	17	17	25
10Y	5	13	17	22	25
30Y	7	11	20	21	25
MTL	31	58	71	76	90

process is that the methodology plays an important role and is more desirable than having a small number of pre-determined individual features. This small number of features may limit the capacity of the model to predict with higher forecasting accuracy. To conclude, the results from this study indicate that it is preferable to have a more significant number of features and submit them to a rigorous selection method for the specific conditions of the regression problem (in our case the target yield and the forecasting horizon).

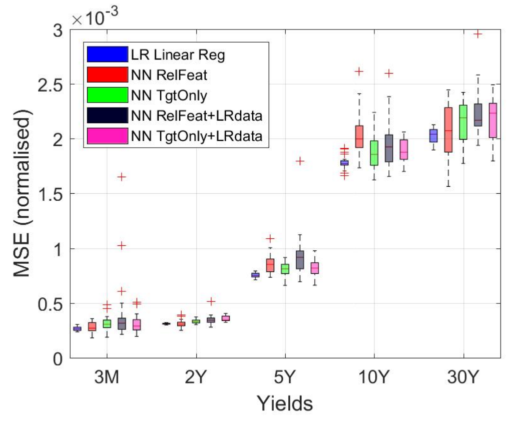
3.3.2 Comparison of models

In this section, modelling results are presented, discussed in depth and finally compared with other results in the literature.

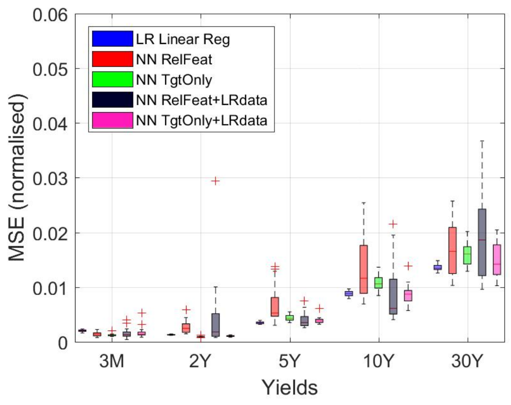
3.3.2.1 Introduction

Results from a direct comparison of the MLP model using all generated features (Model 1, Table 3.2) versus MLP with relevant features (Model 2), demonstrated the clear advantage of performing an initial feature selection. The main advantages are twofold: better forecasting (lower errors and lower spread) and lighter models with lower number of features meaning less computing time. For this reason, the results presented in Figure 3.4 exclude Model 1.

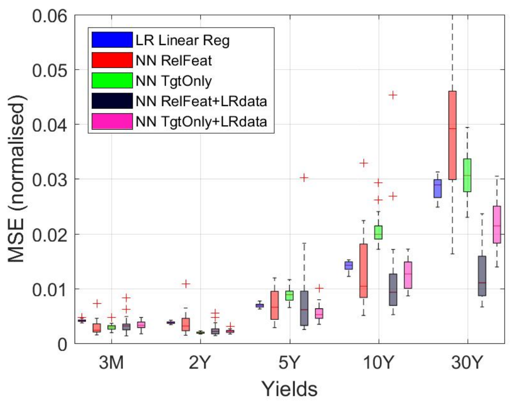
The results are presented using the normalised metric (Section 3.2.8). However, in order to enable a point of comparison between normalised and non-normalised results, an example is presented for the 10-year yield in Table 3.5. Given the MAPE calculation method, this metric tends to give very volatile results when the real yield is close to zero (denominator in the calculating equation). For this reason, the MAPE non-normalised metric was calculated excluding two data points with real yields equal to 0.0%, i.e. less



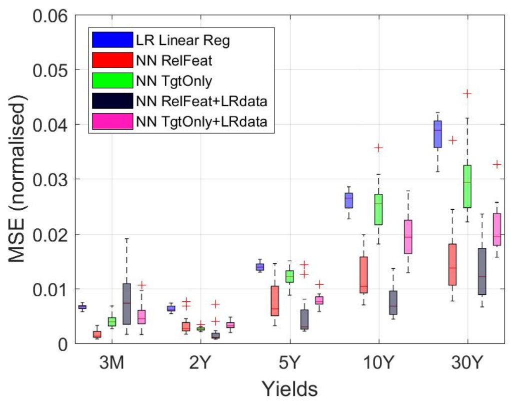
(A) Forecasting horizon = 0 days (next day).



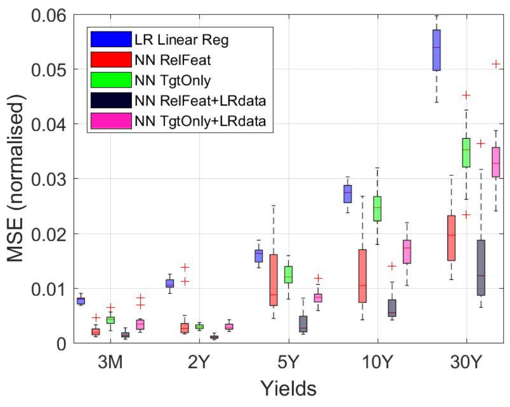
(B) Forecasting horizon = 5 days.



(C) Forecasting horizon = 10 days.



(D) Forecasting horizon = 15 days.



(E) Forecasting horizon = 20 days.

FIGURE 3.4: Comparison of models: linear regression (LR Linear Reg); MLP using relevant features per target and per forecasting horizon (NN RelFeat); MLP using only past values of the target(s) to predict (NN TgtOnly); and the last two models with synthetic data from the linear regression model as additional feature (NN RelFeat+LRdata and NN TgtOnly+LRdata, respectively). In all cases: neural network (NN) models with 10 hidden units and feature selection with regularisation parameter γ equal to 4.

than 5 basis points (bp), where this metric does not become appropriate. On the comparison between normalised and non-normalised metrics, as can be seen the difference between them is not substantial given the range and level of 10-year yields analysed.

TABLE 3.5: Forecasting errors for 10Y yield, for *Model* MLP using relevant features, and *Forecasting horizon* next day. The non-normalised MAPE excludes two data points with real yields equal to 0.0% (less than 5 bp), where this metric does not become appropriate.

Error	**Normalised**		**Non-normalised**	
	Mean	Std Dev	Mean	Std Dev
MAE	0.0319	0.00194	0.0334	0.00185
MAPE	1.61	0.09	6.02	0.34
MSE	0.00206	0.00021	0.00226	0.00018
RMSE	0.04529	0.00229	0.04754	0.00189

3.3.2.2 Multilayer perceptron models

The one step ahead forecasting, shown in Figure 3.4a, tends to produce results of the same magnitude for all models considered, with no significant difference between them. In fact, analysing the baseline linear regression model, the results were surprisingly good for next day forecasting. However, we need to stress that linear regression and neural network models followed exactly the same procedure in what concerns: feature selection per target and per forecasting horizon and retraining of models at every time step (as discussed in Sections 3.2.2 and 3.2.6). As a result, this model is a more difficult benchmark to beat when forecasting the next day.

However, when we move from forecasting one step ahead to forecasting further into the future, the superior performance of the MLP models compared to the benchmark linear regression becomes more prominent, especially above next day + 5 days (Figures 3.4c to 3.4e). Specifically, the best MLPs achieved reductions of MSE w.r.t. linear regression in the range of 46% to 81% (median values, for forecasting horizons of 15 and 20 days).

Additionally, the model using relevant features (Model 2, Table 3.2), starts outperforming the model using only past values of the target yield to predict (Model 3), especially for longer forecasting horizons and longer maturities (10 and 30-year bonds). The latter results demonstrate the importance of incorporating features from markets and economy in the models.

The analysis of the MLP models using synthetic data, reveals another interesting point. Despite the simplicity of linear regression, the neural network models including synthetic data generated by this model tend to improve results (Figures 3.4c to 3.4e). Once again, this effect is more pronounced for longer forecasting horizons and longer maturities (2 to 30 years), achieving MSE reductions compared to the model without synthetic data in the range of 11% to 70% (median values, for forecasting horizons of 15 and 20 days). This is a promising result, showing the potential for the development of hybrid models using synthetic data from other models.

Globally, considering all models and forecasting horizons studied, the results presented show that the MLP using relevant features achieves the best overall results for yield forecasting.

Also of note, and as one would expect, when we move from forecasting one step ahead to forecasting further into the future, the error increases, due to the more demanding forecast for longer horizons. This can be seen in Figures 3.4b to 3.4e, when compared to Figure 3.4a.

All results in Figure 3.4 are presented in terms of normalised MSE, as this was the main metric chosen for presenting the results (Section 3.2.8). To visualise what this represents in real yields and give the reader an idea of the models' forecasting capability, a scatter plot of actual versus predicted yield is shown in Figure 3.5. This is presented as an example of results for 10-year yield and next day forecasting horizon. The figure shows a very good fit for data unknown to the model, with only one point more distant to the equality line.

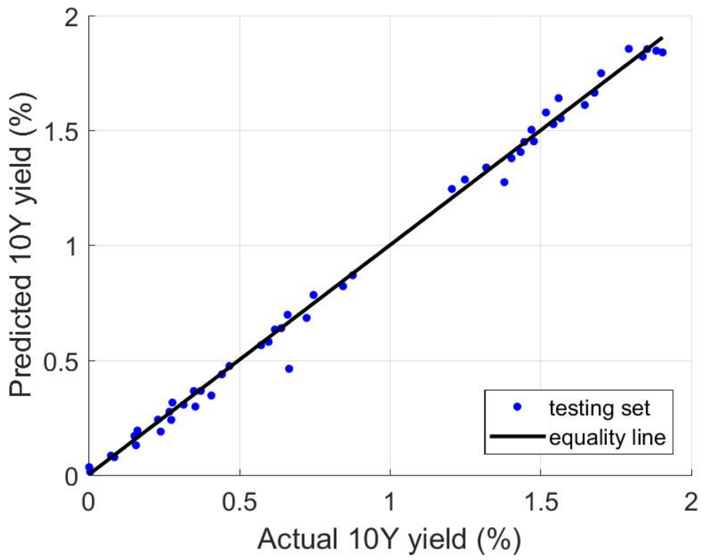
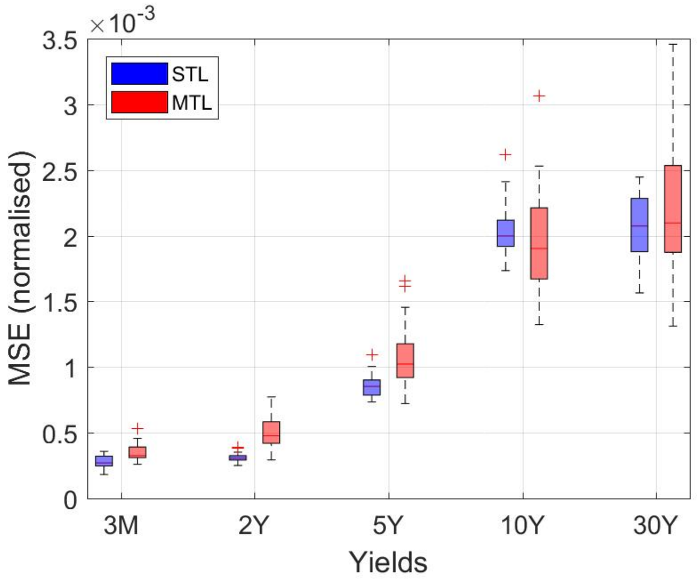


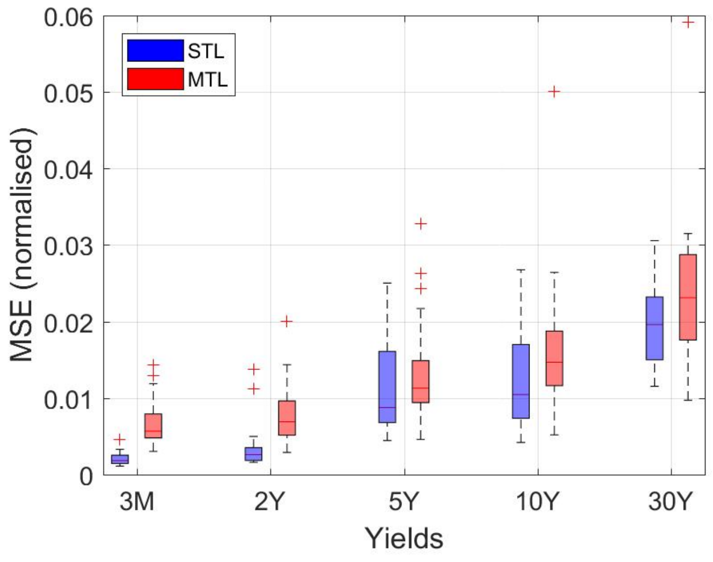
FIGURE 3.5: Forecasting results for 10Y yield (model: MLP using relevant features; forecasting horizon: next day).

3.3.2.3 Single task versus multitask learning

Regarding the comparison of single task with multitask learning, using both yields as targets (multi-assets analysis) and forecasting horizon as targets (multi-step analysis), no clear differentiation among those two techniques could be demonstrated. As a result, we present only a few examples, shown in Figure 3.6.



(A) Forecasting horizon = 0 days (next day).



(B) Forecasting horizon = 20 days.

FIGURE 3.6: Example of single task versus multitask learning for the MLP model using relevant features per target and per forecasting horizon. In both cases: neural network models with 10 hidden units and feature selection with regularisation parameter γ equal to 4.

Given that the literature highlighted numerous benefits of simultaneous modelling of targets in MTL mode on a wide range of applications, this lack of differentiation was somehow unexpected. In fact, in our research study we have also compared these two techniques when models are trained with a fixed training dataset and no retraining (see Section 3.2.6) is carried out. Some benefits of MTL were observed in this case, which could not be reproduced once retraining was used (Figure 3.6).

Thus, it is worth reflecting on possible reasons justifying the results obtained. In the neural network model with relevant features, going from a single task learning method to a multitask mode implies the incorporation of all relevant features for each target (in MTL with yields as targets) or all relevant features for each forecasting horizon (in MTL with forecasting horizon as targets). This corresponds to a very significant increase in the number of features the model has to deal with, which may not be best for generalisation, i.e. performance outside the known training data.

Other possible reasons for the lack of performance of MTL may be obtained from a recent study by Ciliberto et al. (2015). The authors reported the benefits of using MTL, but also concluded that its advantage decreases as the amount of training examples increases. From both mean and standard deviation of errors presented in the same research study, it can be concluded that the performance improvement from MTL is not substantial. Also, in the cases where the benefit of MTL is higher, the overall performance of models is comparatively poor, with the amount of data probably also having an important effect. The benefits of MTL with limited data have also been reported by Benton et al. (2017). In the research reported here, the amount of data collected was large considering both the overall period and the amount of observations available.

In summary, there are several factors that may have contributed to the lack of differentiation between single task and multitask learning: the large amount of data used for training the models; the optimisation of models by using the relevant features per target and per forecasting horizon together with full retraining of models at every time step; the large increase in the number of features as we consider MTL; and the fact that the relevant features tend to be different for all targets. There is an advantage with MTL, which is the running of only one model for all targets instead of five. However, this is not as relevant nowadays, given the modern computing capabilities of supercomputers and GPU computing.

3.3.2.4 Comparison with results in the literature

A comparison of results with the ones available in the literature is difficult given the scarce number of studies having some type of overlapping with this study. Notwithstanding, some common ground can be found in Castellani and Santos (2006), Dunis and Morrison (2007) and Sambasivan and Das (2017), covered in Section 2.2.1. A direct comparison of results can only be achieved by using exactly the same data for the models being compared, which falls outside the objectives of our research. Bearing in mind the limitations, having an indicative comparison of the magnitude of errors would be important in this type of empirical work.

In more detail, in Castellani and Santos (2006), monthly data was used for forecasting the US 10-year yield, and the best models achieved levels of accuracy only marginally better than forecasting using the last available value. The closest situation in this study would be the comparison with results obtained for a forecasting horizon of 20 (working) days, approximately a calendar month. However, given the different type of data frequency used (monthly versus daily), the comparison is done on a qualitative basis only, emphasising the fact that in this study all models led to results significantly better than using the last available value.

A closer comparison can be attempted with Dunis and Morrison (2007). The authors used daily data for next-day forecasting 10-year yields (German, UK and US), using a feedforward neural network with one hidden layer and five hidden units, among other models. This may be compared with our results shown in Table 3.5. The results presented in this work compare favourably in all cases, being of the same magnitude as the best results obtained in that study, achieved in the case of the UK yields. Main limitations of this comparison are due to the different dataset used, both in terms of period analysed and features considered.

Finally, a comparison of results with the work carried out by Sambasivan and Das (2017). The dataset used in our research (January 1999 to April 2017) fully includes the period considered in the above mentioned study (February 2006 to February 2017). In this case, we have to take into account that only one step ahead forecasting was implemented. Taking all this into consideration, the results presented in this work are significantly better for all target yields considered. In our research additional information from macroeconomic and market features was included, as well as a more extended period for the datasets, totalling over 18 years of data.

3.4 Summary and conclusions

In our research we apply machine learning to fixed income markets, an area which has received relatively little attention in the literature compared to other areas of finance, such as equities and forex markets. In particular, this is the first study which applies machine learning, specifically multilayer perceptrons, to model the yield curve as a whole. To this end, we apply a technique called multitask learning, which enables learning multiple targets simultaneously, and compare this to an approach having multiple single task learning models, i.e. one for each target. In addition to MLP models, we also compare the results to using linear regression. In our analysis we consider five different targets (i.e., 3 months, 2, 5, 10 and 30 years), and we consider both next-day forecasting, as well as forecasting further into the future. The latter has important applications in areas such as supervision and regulation, economic forecasting, financial strategy and portfolio management.

Our findings using data from European yield markets suggest that the methodology needs to be carefully chosen in order to achieve good prediction results. In particular, we noticed that, for the MLP, the feature selection process is vital (Objective 1, Section 1.2), and that having a pre-determined set of features results in poor performance. Indeed, our results clearly show that the relevant features depend on both the targets selected, as well as the forecasting horizon, demonstrating that a "one set of features fits all" methodology does not work for forecasting bond yields. It was also important to fine-tune the regularisation parameter γ, leading to better results with lower standard deviations, due to less overfitting. Furthermore, we used an innovative approach of combining the linear regression and MLP model, by using the predictions from the linear regression as input for the MLP. This approach with synthetic data resulted in superior performance (Objective 5). Finally, for both MLP and linear regression, we use a moving window of training data to incorporate the most recent information as it becomes available, and the retraining of models happens at every time step. This has the advantage of increased flexibility to changing market conditions.

In more detail, the MLP using only the most relevant features for each target and each forecasting horizon achieved overall higher levels of accuracy when compared to linear regression and to MLP using only past values of the target variable (Objectives 2–3). While the performance is comparable for next-day forecasting, the differences become significant for longer forecasting horizons. Specifically, the best MLPs achieved reductions of MSE w.r.t. linear regression in the range of 46% to 81% (median values, for forecasting horizons of 15 and 20 days). The results also compare positively with the limited existing results from the literature on yield forecasting.

Furthermore, the results obtained with the MLP models incorporating synthetic data generated by the linear regression model tend to improve forecasting accuracy (Objective 5). This effect is again more pronounced for longer forecasting horizons and longer maturities (2 to 30 years), achieving MSE reductions compared to the model without synthetic data in the range of 11% to 70% (median values, for forecasting horizons of 15 and 20 days). These results suggest that the use of hybrid models incorporating data generated by industry-established models as additional features can be beneficial.

On the comparison of the multi vs single task learning used for yield curve forecasting, no clear differentiation could be demonstrated (Objective 4). Several factors were pointed out that could justify these results, which are supported by previous studies found in the available literature, i.e.: large amount of training data; full retraining of models at every time step; large increase in the number of features as we consider MTL; and the fact that relevant features tend to be different for all targets.

In summary, we showed that MLPs can be successfully used to forecast the yield curve and we believe that the methodology and techniques described and proposed in this work will help practitioners to build better forecasting systems for bonds.

Chapter 4

Long Short-Term Memory Networks for Bond Yield Forecasting

In this Chapter, we use an LSTM architecture to build predictors of time series, extract the internal signals of the model and regress their temporal variation on underlying economic variables. This adds to the work reported in Chapter 3 towards addressing Objectives 1 and 2 (Section 1.2).

4.1 Specific background literature

With the objective of placing our work in context, in this section we give a succinct review of literature specific to this Chapter, including research that attempts to explain how black-box models work and tools available for analysing latent signals within models.

The importance of going beyond black-box modelling and attempting to interpret how a model makes predictions (or, broadly, decisions) when deployed in a specific application is widely accepted. The determination of Shapley values (Shapley, 1953) for each feature is a technique used for interpretability of models (Molnar, 2022). It evaluates the average marginal contribution of each feature, across all possible subsets of features, towards the model prediction. For this reason, it scales exponentially with the number of features. Consequently, given the complexity of our data and the high dimension of the inputs, we cannot explore this technique in our forecasting problem.

One way of understanding the internal workings of a model is visualization. A tool for such visualization of LSTM internal signals is LSTMVis, developed by Strobelt et al. (2018). The specific application of this tool is in natural language processing, extracting the relationship between input word sequences and semantic information such as sentiments. Using LSTMVis, Persio and Honchar (2017) analysed activations of RNNs when modelling time series, and could be seen as the first attempt at "opening the black-box", similar to ours, for time series analysis.

In an earlier study, Giles et al. (2001) refer to the possibility of extracting rules and knowledge from trained recurrent networks modelling noisy time series. To that end, they start by converting the input time series into a symbolic representation using self-organizing maps. Then the problem becomes one of grammatical inference and they use RNNs considering the sequence of symbols as inputs. More specifically, they use an Elman-type architecture for the RNN (Elman, 1990). In addition, the converted inputs facilitate the extraction of symbolic information from the trained RNNs in the form of deterministic finite state automata. The interpretation of that information resulted in the extraction of simple rules, such as trend following and mean reversion.

Fischer and Krauss (2018) have proposed a different approach with the same objective of interpreting models in the analysis of components of the S&P 500 index. LSTMs are used to forecast the next time step of assets and ranked according to the probability of outperforming the cross sectional median. Ranking these in order of performance and selecting subsets of top and bottom performers, the authors compute descriptive statistics in order to identify features that may explain such deviation from the expected performance of the index. Characteristics they identify include high volatility, below-mean momentum, and extreme returns in the last few days with a tendency to revert them in the short-term. This study is an example of using a model to separate the behaviour of the components of a basket of assets, and attempting to explain component behaviours that deviates from expected ones. Such analysis is done on the "outside" of the model, distinct from our objective of looking "inside" it.

In an elegant study that observes that asset prices are influenced not only by past trends, but also by explanatory exogenous variables, Mahler (2009) introduced the Kalman-LagLasso method that inspires the work in this Chapter. The method fits a linear autoregressive model to a given time series, computes the residual signal and then attempts to explain the residual by a regression taken over several explanatory variables. The essential idea behind the methodology is the argument that systematic effects of the exogenous variables can be seen in the residuals, and by inducing sparsity in the regression by means of a Lasso penalty (Tibshirani, 1996; Takeda et al., 2013), one can identify the most relevant variables. Mahler uses a range of macroeconomic variables, each set to have a specific lag in estimating their contributions. The autoregressive model of the time series is estimated recursively using a Kalman filter.

A recent adoption of Mahler (2009) is the work of Montesdeoca and Niranjan (2020). In this work the authors compare the explanatory variables of time series prediction residuals in stock indices (S&P 500 and Dow Jones Industrial Average) and cryptocurrencies (Bitcoin and Ethereum). The motivation of the authors is that whereas equity prices (and hence the index which is derived from them) are driven by an underlying economy, cryptocurrency values depend purely on trading. Thus, they argue and go on to demonstrate, that the explanatory value of macroeconomic variables is far greater on stock indices than on cryptocurrencies.

4.2 Data

We build on our previous work (Chapter 3) to implement and compare the relative performances of MLPs to the LSTMs from the current Chapter. Previous work did not include LSTMs and is used for benchmarking purposes only. As both studies are conducted on the same datasets, we benefit from some computationally intense tasks of model selection previously achieved.

The dataset is composed of daily closing values obtained from Bloomberg database (Bloomberg, 2017a) and covers the period from January 1999 to April 2017, giving 4779 time points of data. The former is the starting date for most time series of the Euro benchmarks and the interval considered covers several bull and bear markets. Following previous work, we take the 10-year Euro government bond yield (EUR 10Y) as the target time series, representing one of the most important maturities in the government yield curve and as a proof of concept. In addition, we consider a total of 159 features from the markets as covariates in various parts of the modelling (Section 3.1.2). These are selected from government bond markets and from related classes and indicators: credit (corporate bonds), equities, currencies, commodities and their volatility. From these, we compute various statistics such as bond spreads, slope of the yield curve and simple technical analysis indicators. Furthermore, as suggested in the well established yields-macro models such as the DNS model (Diebold and Li, 2006), we also include a range of economic indicators, from different geographic locations.

4.3 Methodology

In this section, we present implementation details of the modelling approach which consist of time series modelling with LSTMs, extraction of their gate signals and internal states (cell and hidden), and sparsity inducing regression of the latter on a set of underlying economic covariates.

4.3.1 LSTM networks for bond yield forecasting

With the purpose of assessing the capability of LSTMs to strike a balance between recent and distant pasts, we model univariate time series with LSTMs, i.e. using only past values of the target time series (bond yields) being predicted. A summary of the models used and additional model information is presented in Table 4.1.

TABLE 4.1: Summary of the models used.

Model information		
Original features		159
Target		10-year yield (EUR 10Y)
Forecasting horizons		0 (next day), 5, 10, 15, 20
Moving window size		3000 days
Hidden units MLP		10
Hidden units LSTM		100
Time steps MLP		6 days
Time steps LSTM		6, 21, 61 days
Model	**Short name**	**Description**
Direct comparison MLP vs. LSTM		
MLP	NN TgtOnly	MLP with target data only
LSTM	LSTM06	LSTM using input sequence of 6 time steps
LSTMs with different input sequences		
MLP	NN RelFeat	MLP with relevant features
	NN TgtOnly	MLP with target data only
LSTM	LSTM06	LSTM using input sequence of 6 time steps
	LSTM21	21 time steps
	LSTM61	61 time steps

In the labels used in Table 4.1, *NN TgtOnly* is an MLP whose inputs are identical to those of the LSTM (*LSTM06*), taking six past values as inputs, enabling direct comparison between the use of static and recurrent architectures. Additionally, the model *NN RelFeat*, is a static MLP model, where the inputs are the relevant features that resulted from the feature selection process (Sections 3.2.2 and 3.3.1). The number of features used in each MLP model depends on the forecasting horizon and is presented in Table 3.4, varying from 5 (next-day forecasting) to 25 (next-day forecasting plus 20 days), for the 10-year yield. We also explore a range of different LSTM architectures by varying the number of past values being used as inputs, as well as predicting different forecasting horizons.

For training the models, we use moving windows, updating their weights at every time step. After exploring several architectures, we converged on one with 100 hidden units for the LSTMs to compare their performance against that of MLPs. Increasing the model size results in higher computational cost and no additional benefit in learning the time series. In setting the number of hidden units for the MLP, 10 hidden units was found to be a good compromise, with significant overfitting observed for neural networks with more than 100 units, consistent with what we reported previously (Section 3.2.3). Overall, increasing the number of hidden units does not substantially change the results, having the advantage of using one MLP architecture instead of 25, for each yield and forecasting horizon studied in Chapter 3.

When comparing model sizes, the MLP with 159 input features had a total of $9,561$ free parameters and the LSTM had $40,901$. An LSTM with four times as many parameters producing similar results is intriguing, but is a consequence of regularization acting in different ways in these models. We trained the models by gradient descent, using the ADAM optimiser (Kingma and Ba, 2015) throughout, as it is the state-of-the-art technique for such problems (Brownlee, 2018).

4.3.2 Extraction of LSTM internal signals

In this section, we define the LSTM model and identify the hidden unit responses that are extracted for analysis with the *LSTM-LagLasso* framework (Section 4.3.3). The equations governing the LSTM model were presented in Section 2.6.2 (Equations 2.22–2.27). A diagrammatic representation of an LSTM cell was also presented in Figure 2.8, showing where the math operations occur in the LSTM cell.

The LSTM set of equations enable us to calculate the internal signals at various locations. We consider the following locations in this study: forget gate (Equation 2.22); product of the outputs from the input gate and input node (Equations 2.23 and 2.24); output gate (Equation 2.25); cell state (Equation 2.26); and hidden state (Equation 2.27). We consider the product of the outputs from input gate and input node to be a suitable signal because it is what gets added to update the previous cell state, thus having most relevant and interpretable information. It is worth emphasising that the hidden and cell states are the most important signals, since they summarise all the information going through all the other gates. We observed that the cell gates' signals are helpful to understand and confirm what happens at the cell states' level. However, they do not add relevant information, which is why they are not included in the results presented in Section 4.4.2. Details of the LSTM model used in this study are shown in Table 4.2.

4.3.3 Exogenous covariates explaining LSTM internal signals

To recap for emphasis, in pursuit of explaining the internal representations learned by the black-box LSTM model, we use the framework introduced by Mahler (2009). In this, the residual signals of a recursively computed autoregressive time series prediction model are regressed on a set of exogenous covariates with a sparsity inducing (Lasso) penalty. Should the internal states extract information that is meaningful in

TABLE 4.2: Summary of the LSTM model used for signal analysis.

LSTM architecture		
Features	set 1	10-year yield (EUR 10Y)
	set 2	10-year, momentum indicator
	set 3	10-year, 5-year, 30-year yield
Target		10-year yield
Forecasting horizon		next day + 5 days
Moving window size		3000 days
Hidden units		3
Sequence in		6 days
Sequence out		6 days

some way, we would expect it to be modelled by a small number of underlying economic variables, possibly different ones during distinct market regimes. Staying with Mahler (2009)'s nomenclature, our model is referred to as *LSTM-LagLasso*.

In adapting the Kalman-LagLasso framework, we differ in two ways. First, we do not denoise the covariates using a Kalman filter, and use them directly as inputs. This is because the Kalman filter, with its wide use in sensor data processing type applications (Park et al., 2019; Maybeck, 1979), has an additive measurement noise term implicit in its formulation. This instrument noise is usually assumed to be zero mean Gaussian, with its variance taken from sensor calibration. In the financial context, however, we need to understand noise in a very different way. Variability arising from complex dynamics and interactions is unlikely to manifest as additive Gaussian noise. Hence, using the original time series instead of denoising them, appears more appropriate.

The second main difference in our methodology is that we consider all lags selected as relevant in the *LSTM-LagLasso* algorithm and not only one for each feature, as the simplified setting used by Mahler (2009). We found no reason to limit the number of lags that may participate in the forecasting process to only one. The choice of relevant lag(s) can be left to the sparsity inducing algorithm (Tibshirani, 1996), and our results support this option (Section 4.4.3).

Our complete algorithm is presented in Algorithm 1. For completeness, the objective function minimized by the Lasso method is given by:

$$\underset{w}{min} \; \{ \; \|\boldsymbol{X} \, \boldsymbol{w} - \boldsymbol{s}_{lstm}\|_2^2 + \gamma \; \|\boldsymbol{w}\|_1 \; \}, \tag{4.1}$$

where $\boldsymbol{X}$ is the matrix of features and respective lags; $\boldsymbol{w}$ the vector of unknown parameters; $\boldsymbol{s}_{lstm}$ are the LSTM cell and hidden state signals (target vectors); γ is the regularisation parameter; $\|.\|_1$ denotes the L_1-norm; and $\|.\|_2$ the L_2-norm.

Algorithm 1: *LSTM-LagLasso* algorithm.

```
input: Macroeconomic and market features
target: LSTM cell and hidden state signals
1  Extract LSTM cell and hidden state signals and set them as targets in
     independent models (one per state).
2  Select the number of lags to consider $\{k \in [1, 6]\}$.
3  Build matrix $\boldsymbol{M}$ containing macroeconomic and market features.
4  Transform matrix $\boldsymbol{M}$ into matrix $\boldsymbol{X}$ by incorporating $k$ lags.
5  Standardize all variables (features and target) to have zero mean and unit
     standard deviation.
6  for $i \leftarrow \gamma_{min}$ to $\gamma_{max}$ do
7      Perform Lasso regression (Equation 4.1).
8      Identify non-zero weight values in vector $\boldsymbol{w}$.
9  end for
10 Select $\gamma$ (look for stabilising trend in the number of features against $\gamma$ and
     forecasting error).
11 Perform Lasso regression with selected $\gamma$.
12 Identify the final most relevant variables and lags (non-zero weight values in
     vector $\boldsymbol{w}$).
```

For interpretation of the LSTM internal states, we use a simpler model with only three hidden units. This is necessary because, with a 100-unit LSTM, visualization and interpretation becomes difficult. As exogenous variables to explain the LSTM internal states we used the same large set of features used as inputs to the MLP, setting the number of lags in each of them to six. We exclude from this list the variables considered in feature sets 1 to 3 because they are already known to the model (Table 4.2).

Finally, for tuning the regularisation parameter (γ), we considered both the trend in the number of features against γ and the forecasting error (Algorithm 1, Line 10). After an initial rapid drop in the number of features, the trend changes significantly, but a clear period of stabilisation cannot be observed. Additionally, for γ above 1.0, the error starts increasing more rapidly and the quality of fit deteriorating. For this reason we selected $\gamma = 1.0$ for the final Lasso run, as a good compromise between stabilisation and quality of fit.

4.4 Results and discussion

In this section, we present and discuss the results obtained in experimental work in three parts. First, we establish the relative merits of LSTM, a recurrent neural network containing feedback, over a static MLP, thereby identifying a useful operating regime to analyse the inner workings of an LSTM. Secondly, we analyse the internal signals of the LSTM to show how, in non-stationary regions of the financial time series, different hidden units capture information, achieving a regime-switching behaviour. And, finally, we model the extracted internal signals by a sparsity inducing regression on exogenous variables (with lags) to show how different variables influence different signals.

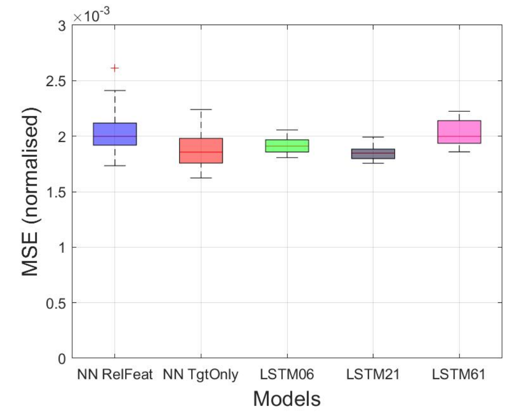
4.4.1 Dynamic versus static models: comparing LSTM and MLP

A summary of comparing prediction error distributions from LSTM and MLP models for different forecasting ranges is presented in Figure 4.1. In these results, the label *NN TgtOnly* denotes performance of a univariate MLP, which can be compared to the various LSTMs. *NN RelFeat* corresponds to an MLP which takes external inputs optimised for 10-year yield forecasting and for each forecasting horizon.

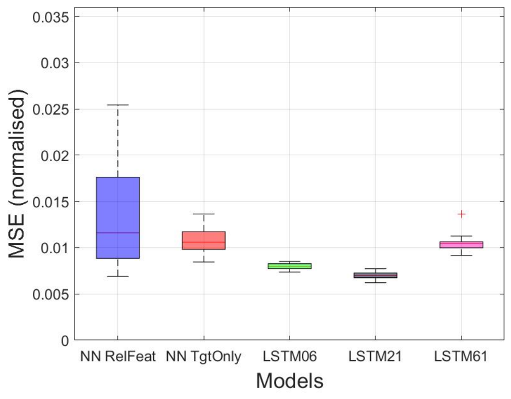
We make several observations from the comparisons shown. First, considering the direct comparison of models in a univariate setting, the LSTM models are systematically better for forecasting horizons of 5 and 10 days, achieving MSE reductions of 25% and 14%, respectively (median values), when compared to the univariate MLP. However, when the horizon is too large (20 days) the advantage of the LSTM is lost. In all other cases, the results are not significantly different. Additional results along the same lines are included in Figure 4.2. Second, the uncertainty in estimation (standard deviation of prediction errors) is lower for the LSTM for all forecasting horizons considered. This is an important point suggesting the dynamic nature of the LSTM results in more robust model estimations for this class of models over more traditional, and widely used, MLPs. Third, when we consider different LSTM input sequences and the forecasting horizon of one day ahead (Figure 4.1a), we do not observe any significant difference between the models, consistent with our previous study on one-day ahead forecasting with memory-free models (Section 3.3.2.2). However, the comparison shows that, at longer forecasting range in the region of 5 - 15 days ahead, the LSTM gives a modelling advantage over the static MLP, in terms of average prediction errors and/or its variance.

Finally, we observe that LSTMs with longer input sequences are able to reach similar levels of forecasting error to the MLP with the most relevant features (Model *NN RelFeat*), again with lower standard deviation. This is an important point in the sense that, over long windows, the memory in the LSTM is capable of capturing, at least in part, the additional information that the static model requires from market data exogenous to the time series.

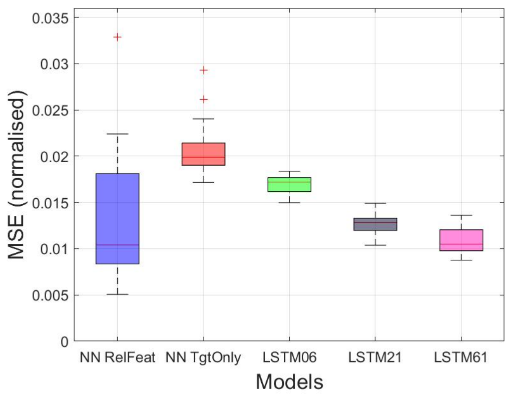
In summary, LSTM networks demonstrate better performance (lower error and/or standard deviation) for a range of forecasting horizons, in particular for next day plus 5 to 15 days. However, we also observe conditions in which its superiority is not evident. In this line, it is worth highlighting the good performance of the MLPs using the relevant features as inputs, specifically for forecasting horizons between 10 and 20 days



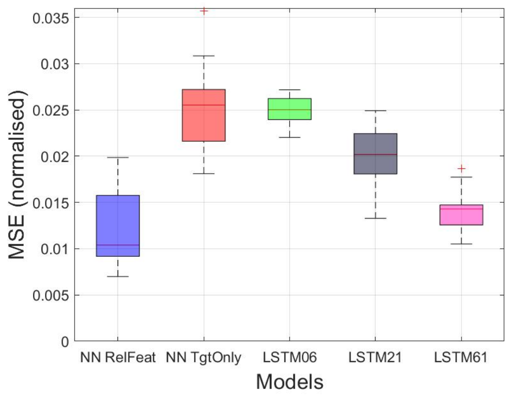
(A) Forecasting horizon = 0 days (next day).



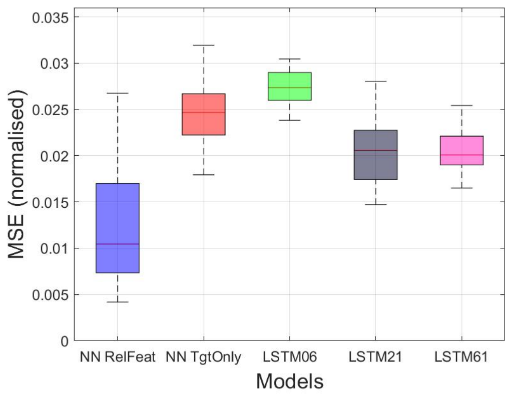
(B) Forecasting horizon = 5 days.



(C) Forecasting horizon = 10 days.



(D) Forecasting horizon = 15 days.



(E) Forecasting horizon = 20 days.

FIGURE 4.1: Comparison of models: two types of MLPs (*NN RelFeat* using the relevant features determined for the 10-year yield target and for each forecasting horizon and *NN TgtOnly* the univariate MLP) vs. LSTMs with input sequences of 6, 21 and 61 time steps (*LSTM06*, *LSTM21* and *LSTM61*, respectively).

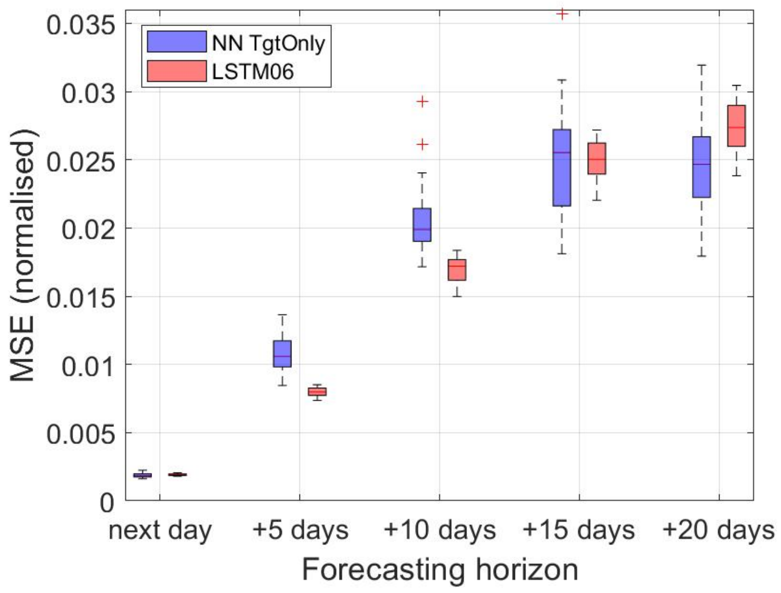


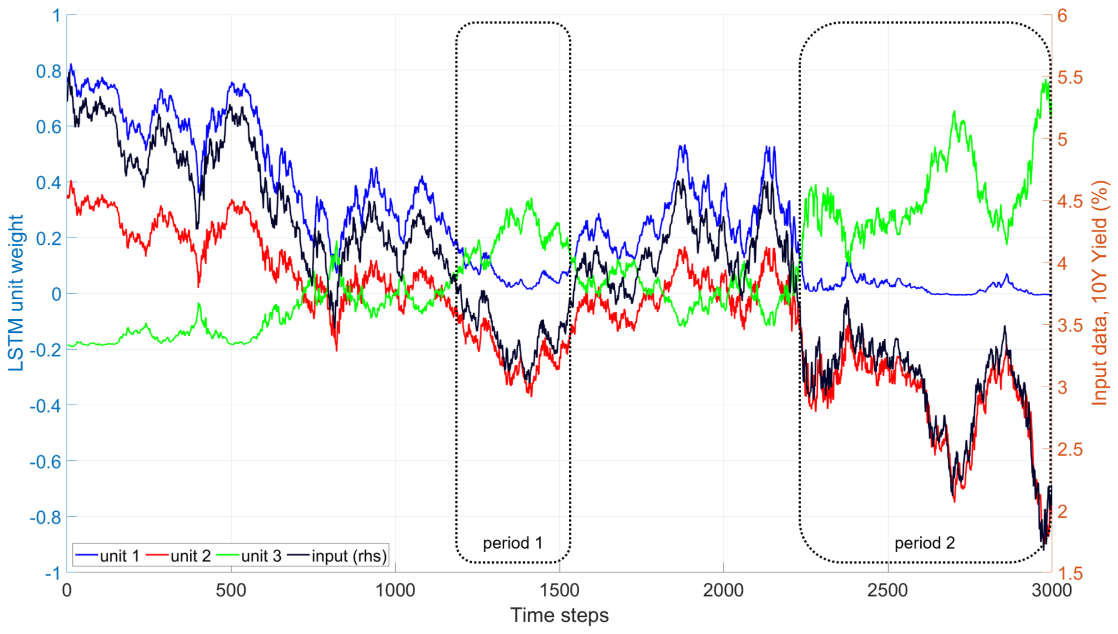
FIGURE 4.2: Direct comparison of models: univariate multilayer perceptron *NN TgtOnly* vs. long short-term memory networks for input sequence of 6 time steps *LSTM06*.

ahead. Hence, given the higher complexity of LSTMs, there are forecasting horizons for which other models should be considered for the forecasting task. Bearing in mind that this study only considers univariate LSTMs, a direct and fair comparison can only be made with regards to the univariate MLPs. Consequently, the obvious next research step should be to test how markets information can improve the LSTM results as it is evident in the case of the MLPs, and then compare these models, both with access to market information. This is something worth exploring in future work. As it stands with the present study, the LSTM demonstrates superiority for specific ranges, without having access to that additional information. This is an important outcome by itself.

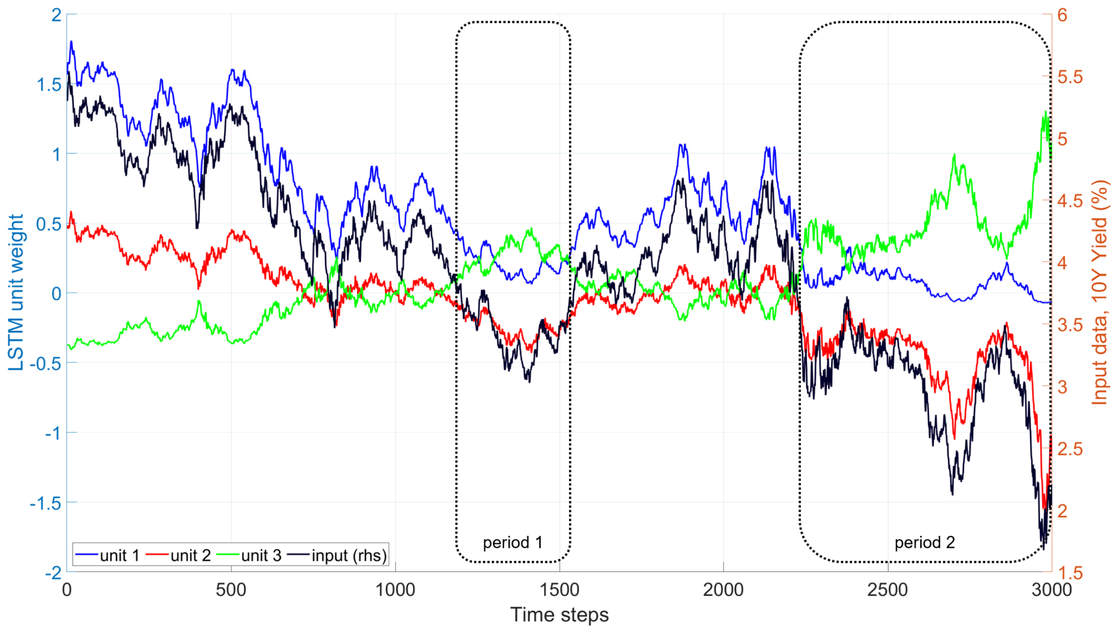
4.4.2 Analysis of LSTM internal signals

The internal states of LSTM models trained to forecast 10-year yield are presented in Figure 4.3, where we also plot the time series of the 10-year yield for reference.

The graphs show how the hidden and cell states have correlated behaviour with the target being modelled. Additionally, some level of similarity between hidden and cell states' behaviour is to be expected as, at time step t, they relate via Equation 2.27. More interestingly, we see switching behaviour of the individual hidden units. In fact, unit 1 becomes almost inactive, both in terms of volatility and weight, during two different periods identified as period 1 and 2 in Figure 4.3. We also observe that, during these intervals, while unit 1 contributes a zero weight, units 2 and 3 take over becoming more



(A) Hidden state signals.



(B) Cell state signals.

FIGURE 4.3: Long short-term memory network signals.

active and following more closely the visible temporal changes of the 10-year bond yield. Of note is the fact that the periods in which the units have switched roles are periods in which the 10-year yield assumes downward extreme values, more specifically yields of 3.6-3.7% and below. Hence, we could conclude that the predictive model is being built of a combination of hidden unit responses, with each unit specialising to capture some underlying trend. In this specific case, we might postulate that upward values are covered by unit 1 and downward extreme values controlled by units 2 and 3. Thus, the response of the hidden and cell states are helpful in understanding how the time series is being modelled within the LSTM.

In Appendix B (Figure B.1), we present similar illustrations of LSTMs trained with feature sets two and three. Again, similar activation / deactivation of the signals with regime switching can be observed. More specifically, for set 2 (Figure B.1a), despite the high volatility of the second feature (momentum indicator, Table 4.2), the same two periods can clearly be observed as described for feature set 1 (Figure 4.3a), with unit 3 behaving in a manner similar to that of unit 1 in the previous case. These labels are, however, arbitrary and interchangeable. Interestingly, we note that, given the high volatility of the momentum indicator, two of the units tend to "specialise" in this feature and only one unit in the 10-year yield.

For feature set 3 (Figure B.1b), the pattern of the signals in relation to the 10-year yield data is not so stark, though different units behave differently, showing a degree of specialisation. One reason for this may be the higher correlation among all yields considered as features (5, 10 and 30-year yield), smoothing out the temporal variation of unit 3.

Overall, we note that the activation of the hidden units of a trained LSTM carries information about how the target signal is decomposed internally. The dynamics contained in the internal signals appear to show regime switching behaviour, presumably because the units lock onto and track some latent factors over time.

4.4.3 Explaining internal states with exogenous variables

In Figure 4.4, we present results of modelling the signal captured in unit 1 of the hidden state, using *LSTM-LagLasso*. Additional results for unit 3 are given in Appendix B (Figure B.2). Sparsity inducing regression (Lasso) extracts the most significant explanatory variables. The absolute values of their weights are shown in the figures. Since we are using market variables to explain the LSTM signals and not a financial asset target directly, it is only the magnitude of the weights that matter to judge their relative relevance. For reasons of brevity, we only present the analysis of the hidden state and skip the cell state, but the main points we wish to make can be adequately illustrated using the hidden state. This is supported by the observation that 79.3% of the relevant features identified are common to both hidden and cell states. In contrast, the percentage of top relevant features common to all hidden units is much lower than between states, i.e. 24.7% of them are common to all three units in the hidden state and 21.4% in the cell state.

A summary of the most influential variables that model both states is presented in Table 4.3. For selecting the variables for this table, we set a threshold of 0.15 on the sum of the absolute weights (for the six time steps considered).

4.4.4 Discussion

We draw several conclusions from the results presented here (Figures 4.4, B.2 and Table 4.3). First, the internal units of an LSTM decompose the target into components that can be explained using exogenous information which was not used in training the model. Second, they capture dynamics in the data, each specialising to some underlying latent information as shown in the regime switching behaviour. While the LSTM internal representations may not be explainable by human intuition, the fact that different sets of underlying economic variables are dominant in explaining the variance in the internal signals is a significant observation (Table 4.3). Third, we observe that the influence of different exogenous variables feed through different time lags. While the lags of the two immediate past days are the most important, often the lag of one week is also significant for several of the variables, pointing to weekly seasonality of influence that is not immediately apparent simply when looking at the raw time series. Given the conclusion that lags are important, selecting only one lag per feature, as proposed in the Kalman-LagLasso method (Mahler, 2009), would eliminate this additional information, limiting the forecasting ability.

Finally, the features selected as relevant in explaining the hidden units are themselves of interest. Contrary to what one would often expect as relevant features, from their frequent use in past work on modelling asset prices (Nelson and Siegel, 1987; Dunis and Morrison, 2007; Arrieta-Ibarra and Lobato, 2015) several other features often carry higher weights. Established modelling research often use variables such as central bank reference rates (ECB refinancing rate), macroeconomic indicators of inflation, (US CPI less food and energy, Eurozone Core Monetary Union Index of Consumer Prices), economic growth / growth expectations (Institute for Supply Management Manufacturing, US Industrial Production, US Capacity Utilisation, ZEW Eurozone Expectation of Economic Growth) and labour market (Eurozone Unemployment).

But the explanatory variables our model focuses on go well beyond this group, with some recognisable as specific to bond markets, adding significant information to those in the above list. For example, the top relevant feature by weight is the long German government bond future (DEU Bond Fut 30Y). This is a leveraged instrument with

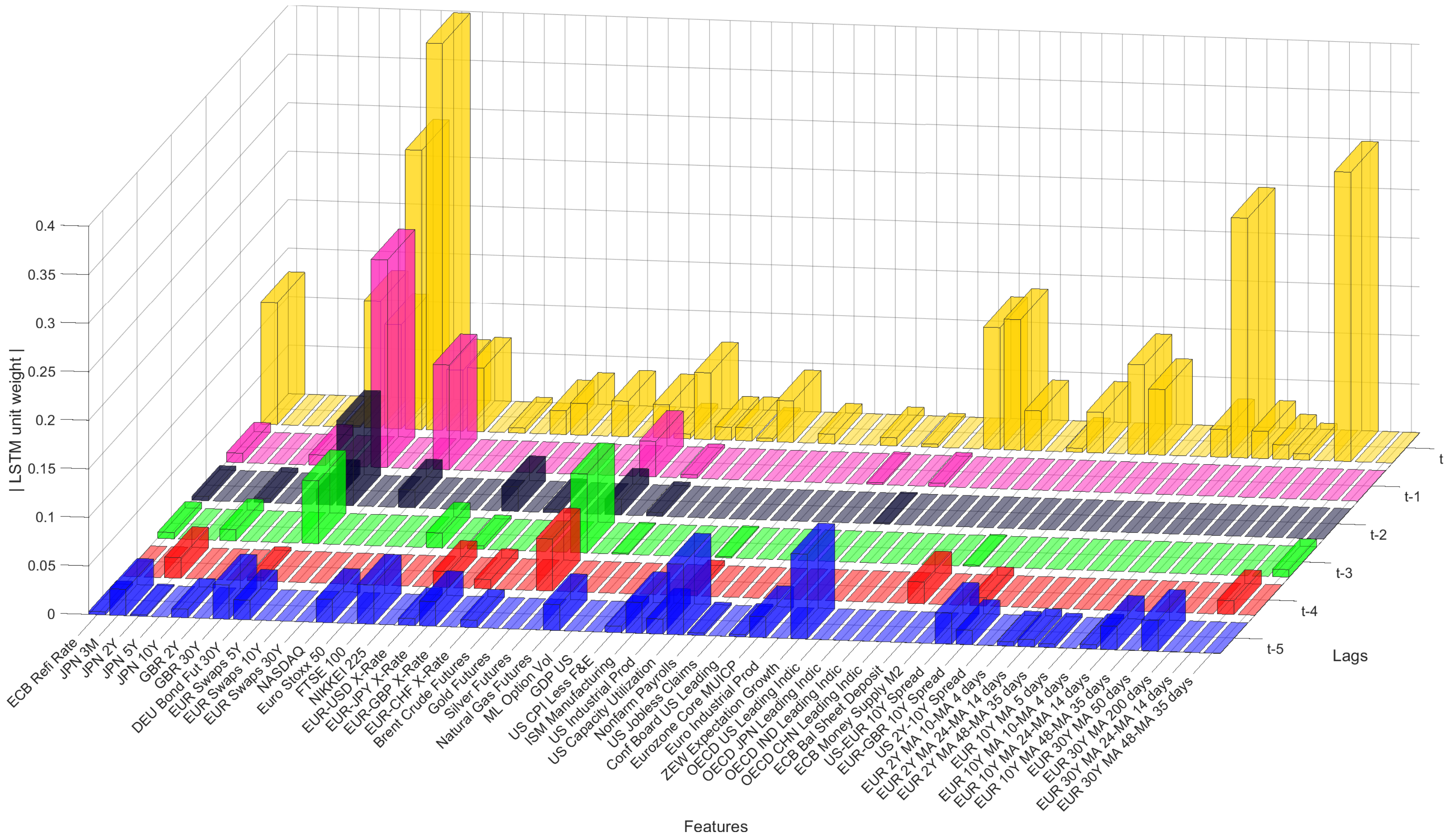


FIGURE 4.4: *LSTM-LagLasso* relevant features for the hidden state, unit 1, considering a regularisation parameter γ equal to 1.0. [Terminology: An example of technical analysis indicator used is “EUR 30Y MA 48-MA 35 days” corresponding to the 48-day Moving Average (MA) minus 35-day MA, of the 30-year Euro government bond yield (EUR 30Y)].

TABLE 4.3: Summary of most influential relevant features for both states. The listed features have a sum of absolute weights greater than or equal to 0.15 in at least one of the LSTM units. The cells in the table for which the weight is equal to zero are left empty. [Terminology: An example of technical analysis indicator used is “EUR 30Y MA 200 days” corresponding to the 200-day Moving Average (MA), of the 30-year Euro government bond yield (EUR 30Y)].

Feature name	**Weight**		
	Unit 1	**Unit 2**	**Unit 3**
Hidden state			
ECB Refi Rate	0.150		
3M Euribor Fut 4th		0.177	0.248
GBR 30Y	0.161		0.053
DEU Bond Fut 30Y	0.674	0.287	0.410
EUR Swaps 5Y	0.398		
EUR Swaps 30Y	0.192	0.199	0.315
FTSE 100	0.056	0.160	0.107
Gold Futures	0.193	0.012	0.004
US 2Y-10Y Spread	0.068	0.087	0.198
EUR 10Y-30Y Spread		0.264	0.214
EUR 10Y MA 5 days	0.247		0.082
EUR 30Y MA 200 days	0.298		
EUR 5Y		0.221	0.202
EUR 30Y		0.450	0.168
Cell state			
3M Euribor Fut 4th			0.212
90day Euro$ Fut 5th		0.203	
GBR 30Y	0.151	0.038	
DEU Bond Fut 30Y	0.723	0.921	0.533
EUR Swaps 5Y	0.258		
EUR Swaps 30Y	0.220		0.186
Gold Futures	0.168	0.109	0.051
EUR 10Y-30Y Spread		0.271	0.270
EUR 10Y MA 5 days	0.525		
EUR 30Y MA 200 days	0.273		
EUR 5Y		0.003	0.172
EUR 30Y			0.240

very long maturity and duration. Consequently, they are highly price-sensitive and react very quickly to market movements. This justifies its role in the model, with the sum of the weights reaching the value of 0.921 for unit 2 of the cell state (Table 4.3). As a second example, within the futures asset class, we find the 3M Euribor and 90d Euro$ Futures (4th and 5th contracts). Given these contracts have approximately a one year-ahead horizon, this could be seen as incorporating investors' expectations on the evolution of short-term rates.

Additionally, financial instruments with maturities adjacent to the one we want to predict are also included in this group of relevant features, namely: 5 and 30-year Euro government bond yield (these tend to lead flattening and steepening movements of the yield curve around the 10-year maturity); 2, 10 and 30-year UK government bond yield; and 5, 10 and 30-year EUR swap rates. Directly related with yields, we can also identify as relevant features intra-curve spreads such as US 2–10-year spread and EUR 10–30-year spread; as well as inter-curve spreads, specifically, EUR-GBR 10-year spread and EUR-JPN 10-year spread.

Furthermore, other relevant features include the following asset classes and macroeconomic variables: commodities (Gold Futures and Brent Crude Futures); equity indices (Euro Stoxx 50, FTSE100, and S&P500); forex rates (EUR-USD and EUR-JPY); the ECB Balance Sheet Long-Term Refinancing Operations (especially relevant due to large-scale central bank interventions); OECD Leading Indicators; and finally technical analysis indicators (5, 50 and 200-day moving averages of 5, 10 and 30-year yields).

In summary, peeping into the internal states of a trained LSTM gives the modeller a handle on variables that have an influence on the predictive ability of models, but their role becomes apparent at the level of the hidden units which operate in regime switching manner to control the overall predictive ability of the model.

A sanity check

We perform a simple sanity check to see if the regression of the LSTM states on the exogenous variables produces relative weights merely by chance. The question we want to answer is: if we were to regress the signals on a set of Gaussian random variables spanning the same dimensions as the economic variables, would the signals be modelled to the same accuracy? To answer this, we apply the *LSTM-LagLasso* method replacing the macroeconomic and market features by the same number of Gaussian random variables, running the simulation one hundred times. The corresponding distribution of errors is shown in the Figure 4.5. With the resulting MSE two orders of magnitude smaller for the market data than the average of the 100 simulations, this clearly shows the relevance of the market data in modelling over the set of orthogonal basis vectors that span the same space.

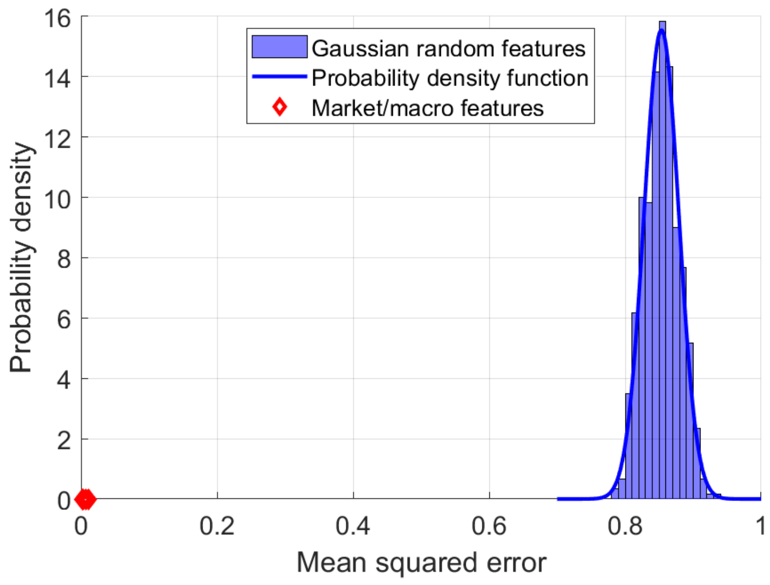


FIGURE 4.5: Forecasting error of *LSTM-LagLasso* using macroeconomic market features and using Gaussian random features.

4.5 Summary and conclusions

While black-box models of data modelling are powerful and widely used, extracting explanations from them is often very difficult. In this context, we have shown that useful information may be extracted by looking into the internal states of an LSTM model of financial time series forecasting. Focusing on bond yields as target time series, we build a predictor, extract its internal states, and regress them on exogenous variables that are likely to drive market behaviour. Starting with finding a regime in which an LSTM gives a modelling advantage over a static MLP, we show that the internal units learn specialised switching behaviour, tracking sharp non-stationarities in the target time series.

With respect to bond yields, our study is the first comparison between static (MLP) and dynamic (LSTM) models showing that the dynamics captured through network memory can compensate the use of additional market-related features useful in a static model. We carry out a systematic comparison using a range of features (past values and side information), forecasting over several horizons, and observe that the LSTM usually achieves lower prediction errors and more prediction confidence (lower variance of estimation).

Second, rather than stop at simple time series prediction, this work looks at the learned internal representations of the model. The purpose of capturing dynamics within the

LSTM architecture, striking a balance between influences at different time scales, manifests as differential contributions from the specialised units, exhibiting regime switching behaviour over time. Past research, reviewed in Section 4.1, makes very little attempt at such explanations.

In the final part of the study, we show how signals extracted from the internal units of the LSTM are influenced by exogenous information in the markets. By adopting a sparsity inducing linear regression to model the internal signals, we show that those variables found to be relevant are not only different from the most established among empirical finance modellers, but different combinations of such variables relate to the contributions of different units. We also observe the importance of lags in these variables.

Overall, the novel methodology we develop, which we refer to as *LSTM-LagLasso*, offers a useful framework for explaining how a dynamic time series model of a specific architecture learns internal representations and how such learned representations can be understood in terms of exogenous information.

There is one immediate avenue to pursue from this starting point. It is to go beyond simple forecasting to embedding such models in an autonomous trading system along with trading strategies. Such trading strategies could be rule-based, or even sequentially optimised using a reinforcement learning framework. The latter is the main focus of Chapter 5.

Chapter 5

Deep Reinforcement Learning for Bond Portfolio Management

In this chapter we apply RL to financial portfolio management activities, maintaining our assets in the bond market arena. The work described here goes towards fulfilling Objective 6 of this thesis (Section 1.2), of testing the concept of RL specifically to this asset class. In more detail, we present the theoretical formulation of a portfolio management problem in RL terms (Section 5.1), describe the bond assets selected for the experimental work (Section 5.2) and the methodology used (Section 5.3). Finally, we outline the results obtained and their discussion (Section 5.4), followed by the main conclusions deduced from this part of the work (Section 5.5).

5.1 Theoretical formulation of portfolio management in RL

In this section we present the standard RL concept as a Markov decision process (MDP), identifying its main components when applied to financial portfolio management.

5.1.1 Introduction

An MDP is a five-tuple object, which includes (Poole and Mackworth, 2017): state space $\boldsymbol{s_t}$ representative of the environment E; action space $\boldsymbol{a_t}$ for the agent; state transition probabilities between adjacent states; reward function r_t and the reward discounting factor γ. We consider an extended version to include other important pieces of the RL process, such as (Fischer, 2018): a policy mapping states to actions; and the value function or the action-value function (the latter more relevant for our specific work).

A standard RL process includes an agent that interacts with the environment in discrete time steps, learning how to take actions that maximise some form of cumulative reward. In this line, at each time step, the agent observes the state of the environment. For simplicity, we consider that the environment is fully observed, so the observations totally represent the state space. Given this state, the agent takes an action, and receives a scalar reward. In the following sections, we proceed to formulate mathematically the components of the MDP presented above, when applied to financial portfolio management.

5.1.2 States and actions

The states are represented by variables that can be observed from financial markets. They may include historical prices of assets and this is the most common situation in the literature (Fischer, 2018). Additional information similar to the one considered in Chapter 3 and listed in Appendix A can be included. However, in the literature only a few exceptions have included additional information, such as Neuneier (1996) and Eilers et al. (2014).

In our case, we aim to test the RL concept and methodologies in portfolio management and consequently we adopt the simplest approach, i.e. considering only the historical prices of the assets selected. At the same time this is the most demanding approach, since additional information can contribute to the learning process, as we have seen in Chapter 3. Finally, more specific details of the data used in our empirical work will be presented in Section 5.2.

Portfolio managers decide the allocation of assets (i.e. the weights for every asset) in the portfolio at each time step, taking into account their performance expectations for each asset. Hence, the actions in the RL process are directly represented by the weights. To be able to calculate the cost of rebalancing the portfolio (i.e. modifying the weights of the assets), following the actions specified by the agent, the trader or portfolio manager requires an internal state information and must therefore be recurrent (Moody and Wu, 1997). Thus, the state at each time step is represented by windows of historical prices of the assets, together with the exposure to each asset in the previous time step, in other words, the previous weights of each asset.

In summary, the states and actions in a portfolio of m assets are represented by the following equations:

$$\begin{aligned} \boldsymbol{s}_t &= (\boldsymbol{w}_{t-1}, \boldsymbol{x}_t) = (\boldsymbol{a}_{t-1}, \boldsymbol{x}_t) && (5.1)\\ \boldsymbol{a}_t &= \boldsymbol{w}_t && (5.2)\\ \text{subject to} \quad & \boldsymbol{1}_m^T \boldsymbol{w}_t = 1 && (5.3)\\ \text{and} \quad & \boldsymbol{w}_t \succeq 0 && (5.4) \end{aligned}$$

where $\boldsymbol{s_t} \in \mathbb{R}^{m\times(n+1)}$ is the state at time step t; m is the number of assets; n is the number of previous time steps of historical prices considered; $\boldsymbol{x_t} \in \mathbb{R}^{m\times n}$ the historical prices of the assets selected in Section 5.2, considering a window of six time steps per asset; $\mathbf{1}_m$ represents an m-dimensional column vector of ones; $\boldsymbol{a_t}$ and $\boldsymbol{a_{t-1}} \in \mathbb{R}^m$ are the agent actions at time step t and $t-1$, respectively. As explained above, these vectors are equal to the weights $\boldsymbol{w_t}$ and $\boldsymbol{w_{t-1}}$. Taking into account that we are considering long-only portfolios with no leverage, the weights are subject to the constraints presented above.

5.1.3 State transition and assumptions

In the case of financial markets, we assume that our transactions do not have an impact on the market overall, which is a reasonable assumption in practice. As a result, the transition between states is not modelled, and is determined by the market itself. Instead, they are an input to the model and obtained via new observations of asset prices.

Apart from no market impact, we also assume that trades can be executed instantly at the desired price. Although the liquidity of the assets selected for this study (Section 5.2) is very high (Bloomberg, 2021), this may not be entirely possible. It is, however, an appropriate simplification given the objective of this work.

5.1.4 Rewards

In portfolio management, the returns of a portfolio can be additive or multiplicative (Moody and Wu, 1997). The latter is more appropriate when investing fixed fractions of the portfolio, which is the case in our approach. For clarity, the alternative additive returns are used when trading a fixed number of securities and consequently the corresponding equation is different. In addition, the general equation for multiplicative returns incorporates a risk-free interest rate term, i.e. the rates of risk-free assets such as US or European treasury bills. We ignore this term, a solution also adopted elsewhere (Koker and Koutmos, 2020). Considering the present market conditions of zero or near-zero interest-rate policy in most developed countries, this is a reasonable assumption to make. Hence, the total value of the portfolio (P_t), at the end of the episode with T time steps, can be calculated using the following equation:

$$P_t = P_0 \prod_{t=1}^{T} \{1 + R_{P,t}\} \tag{5.5}$$

$$\{1 + R_{P,t}\} = \{1 + \boldsymbol{w}_{\boldsymbol{t-1}}^{T} \boldsymbol{r}_{\boldsymbol{A,t}}\}\{1 - \mathbf{1}_m^T C |\boldsymbol{w_t} - \boldsymbol{w_{t-1}}|\} \tag{5.6}$$

where P_0 is the initial value of the portfolio, i.e. the initial capital invested; $R_{P,t}$ is the return of the portfolio at time step t; while $\{1 + R_{P,t}\}$ represents the reward received by the agent at time step t. Finally, $\boldsymbol{r_{A,t}}$ is the vector return of the assets at time step t and is calculated as follows:

$$\boldsymbol{r_{A,t}} = \frac{\boldsymbol{P_{A,t,close}}}{\boldsymbol{P_{A,t-1,close}}} - 1 \tag{5.7}$$

where $P_{A,t,close}$ and $P_{A,t-1,close}$ correspond to the closing prices of the assets at time step t and $t-1$, respectively.

In some studies, the log scale of the portfolio return is proposed as the rewards (Neuneier, 1998; Fischer, 2018). The log scale strongly penalises the negative returns close to zero, i.e. when the ratio of portfolio values in subsequent time steps tends to zero, or is at least below 0.50. For the portfolio value ratio to drop to that level, it would require a negative daily return below -50%, which is highly unlikely. In reality, for the normal range of variations in a bond portfolio, the log and the linear functions would be very close. For this reason, we adopt the simple portfolio return instead of its logarithm.

In summary, the job of a portfolio manager is to maximise the total portfolio value P_t. And the reward is what defines the goal of the RL process, so the reward considered is equal to $\{1 + R_{P,t}\}$ (Equation 5.6). In our experimental work, we will be considering this solution from the literature, together with two variations of this reward as it will be explained in Section 5.3.4.

5.1.5 Policy

The goal of the agent is to learn a policy π that maximises the expected state return, i.e. maximise the objective function J:

$$J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi} [R_1] \tag{5.8}$$

$$R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i) \tag{5.9}$$

where $\mathbb{E}[.]$ denotes the expected value of a random variable; R_t is the state return at time step t, equal to the sum of discounted future rewards (defined in Section 5.1.4); with R_1 referring to the return from the initial state; γ the discounting factor ($\gamma \in [0, 1]$); and $r(s_i, a_i)$ the reward received following state s_i and action a_i.

The policy is considered by Sutton and Barto (2020) as the core of an RL agent given that it is sufficient by itself to determine behaviour. The policy defines the way the agent behaves at a given time, who in turn is responsible for determining the actions. In the most general cases the policy is a function that maps states to a probability distribution over the action (stochastic policy), or directly to actions, if the policy is deterministic.

Additionally, the policy can be parameterised in different forms, as a function $\pi(a|s,\theta)$, as long as that function is differentiable with respect to its parameters. Consequently, we can use function approximators such as deep neural networks, which can be expressed mathematically as follows, for single-asset portfolios:

$$\boldsymbol{a_t} = \pi\big(\boldsymbol{s_t}, \theta_t\big) \tag{5.10}$$

$$\overset{(5.1)}{=} \pi\big(\boldsymbol{a_{t-1}}, \boldsymbol{x_t}, \theta_t\big) \tag{5.11}$$

where $\boldsymbol{\theta_t}$ represents the (learned) function approximator parameters at time step t.

For multi-asset portfolios, two additional requirements are necessary in the RL system (Moody and Wu, 1997; Moody et al., 1998). First, the policy function approximator requires a multi-output network (see Section 2.5). Second, to satisfy the constraint presented in Equations 5.3 and 5.4, one possible approach is to use a softmax layer, so that the output actions become:

$$\boldsymbol{a_t} = softmax\Big(f\big(\boldsymbol{a_{t-1}}, \boldsymbol{x_t}, \theta_t\big)\Big) \tag{5.12}$$

where $f(.)$ denotes the raw or unscaled output, i.e. before the softmax function is applied to scale the weights.

5.1.6 Action-value function

In this section, we closely follow the nomenclature used in Lillicrap et al. (2015), for a smooth transition to our selected algorithm, which we will describe in Section 5.3.2.3. Hence, the action-value function is used in some critic-only and actor-critic algorithms (see Section 2.7.4), and is defined as follows:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi}\left[R_t | s_t, a_t\right] \tag{5.13}$$

where $Q^{\pi}(s_t, a_t)$ is the action-value of the state-action pair (s_t, a_t) at time step t. In words, it quantifies the expected return when in state s_t at time step t, the agent takes action a_t first and then follows policy π thereafter.

Another form to present the action-value function is with the Bellman equation (Bellman, 1957). It states that the value of the start state is equal to the immediate reward, plus the discounted value of the expected next state (Sutton and Barto, 2020). This recursiveness of the Bellman equation is widely used in RL. For action values, the Bellman equation can be defined as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^{\pi}(s_{t+1}, a_{t+1})\right]\right] \tag{5.14}$$

where $r(s_t, a_t)$ is the immediate reward at time step t.

Equation 5.14 refers to the stochastic case, where the policy π maps states to a probability distribution over the actions, i.e. $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$. For deterministic policies, mapping directly states to actions, i.e. $\mu : \mathcal{S} \to \mathcal{A}$, the inner expectation in Equation 5.14 is not necessary, becoming:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))\right] \tag{5.15}$$

Considering that the expectation depends only on the environment, we can use off-policy algorithms (see Section 2.7.5.4) to learn a deterministic target policy μ and the action-value Q^{μ}, using samples from a stochastic behaviour policy β.

5.2 Data

Consistent with the theme of the thesis, this chapter focuses on a portfolio of different types of bonds. Recall that the author has not found published work covering the topic of RL specifically for fixed income portfolio management (Section 2.7.5.4). For this empirical work we considered assets from different sub-classes of the bond market, specifically: governments, corporates and high yield. The selected assets are ETFs representing these sub-classes, and are listed in Table 5.1.

These assets cover a broad spectrum of financial instruments. On the one hand, in terms of interest rate risk, they include a wide range of maturities, from short-term to long-term. On the other hand, in terms of credit and default risk, they include from the highest ratings (government bonds from developed economies tend to fall into this category), to the lowest ratings included in the high yield category. In summary, the resulting portfolio is representative of some of the main risks a fixed income portfolio manager encounters in real bond portfolios (Fabozzi et al., 2021).

Note that the assets listed in Table 5.1 are the totality of assets in our portfolio. In particular, we do not include a cash position, given the inclusion of ETF SHY (Asset 1), which is a short-term government bond fund. This ETF is sufficiently conservative for bond investment, and can be used in market turmoil for portfolio protection, making the cash position unnecessary.

The choice of time scale or frequencies is important and may range from the high-frequency data traditionally used in forex studies, to annual data. In our work we use daily data so that the agent is enabled to take action on a daily basis if required, thus taking more advantage of market opportunities through time.

The dataset for the observations of the four assets (state space) were obtained from Bloomberg database and covering the period from April 2007 to May 2021 (Bloomberg,

TABLE 5.1: Summary of bond ETFs selected.

Asset	**Ticker**	**Strategy**	**Maturity focus**	**Rating focus**
ETF iShares 1-3 Year Treasury Bond				
Asset 1	SHY	Government	Short-term	Investment Grade A or higher
ETF iShares 20+ Year Treasury Bond				
Asset 2	TLT	Government	Long-term	Investment Grade A or higher
ETF iShares iBoxx $ Investment Grade Corporate Bond				
Asset 3	LQD	Corporate	n/a	Investment Grade BBB or higher
ETF iShares iBoxx High Yield Corporate Bond				
Asset 4	HYG	Corporate	n/a	High Yield BB or lower

Note: n/a not applicable. *Assets 3 and 4* do not have a specific maturity focus, instead including bonds with a wide range of maturities.

2021). On the starting date, the range in this case was limited by the inception date of *Asset 4*, the high yield ETF. Moreover, a total of six historical time steps were included for each of the four assets selected. The window of six time steps was already used in previous chapters (3 and 4) and found to give good results for both MLPs and LSTMs. The corresponding time series are presented in Figure 5.1.

In relation to the transaction cost considered, the average cost for these assets is 1.66 bp (this corresponds to the bid-ask spread obtained from Bloomberg). This value was rounded upwards to 2 bp, to be on the conservative side, since we do not know the composition of the portfolio, which will also evolve with time. This composition will be decided autonomously by the RL agent.

As common practice, we decided to split the dataset considering 80% for training and 20% for testing.[1] As a result, the training data includes 2899 time steps, while the testing data is made of 768 time steps. On the one hand, the training data covers a long period (2007–2018), including several cycles of important market movements. On the other hand, the resulting testing data is very demanding in terms of performance taking into account that it comprises a sharp market correction that occurred in February / March

[1]This is different from the 70%/30% split considered in Chapter 3 (Section 3.1.5). A slightly different option was considered due to the importance of having a larger training dataset in RL problems. In addition, the dataset used in this chapter is totally different from the one used in Chapters 3 and 4, and we have enough data for a 80%/20% split.

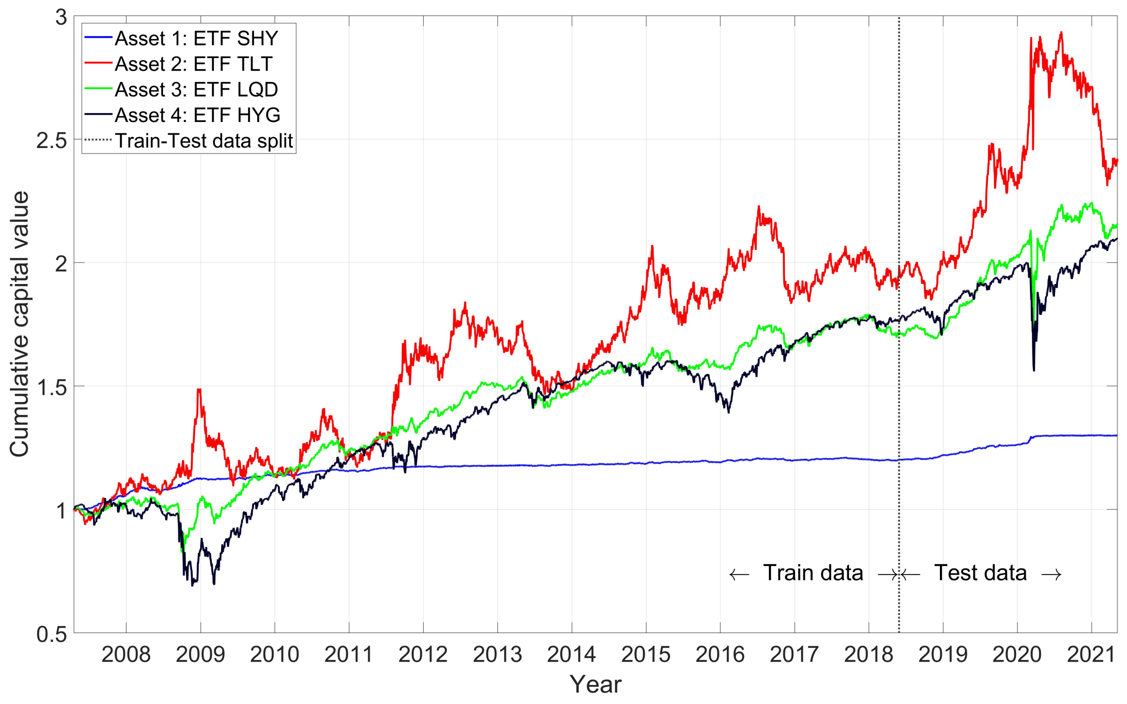


FIGURE 5.1: Cumulative performance of individual assets.

2020 (Figure 5.1). Finally, the fact that throughout the whole dataset some assets go in opposite directions will also be an interesting performance test for the RL system.

5.3 Methodology

The methodology presented in this section is an important contribution of this chapter. Indeed, several alternative approaches were considered that followed the literature more closely, but these did not produce the expected results (as we will show below). As a result, several methodological adaptations were needed until more promising results were achieved. These adaptations will be detailed below. This section is organised as follows. First, we discuss some initial top level options considered impacting subsequent work. Then, we describe in detail the agent, the environment developed for this study and the empirical work carried out.

5.3.1 Top level options

In the following subsections, we start by presenting our choices regarding the type of action space considered, of function approximators and objective function. They result from an initial selection among alternative approaches in RL. Then, we discuss the method used for normalisation, performance evaluation and tuning of hyperparameters.

5.3.1.1 Discrete versus continuous spaces

On the state and action spaces, discrete and continuous solutions can be used. In particular, Pendharkar and Cusatis (2018) used discrete states in their work but recognised that continuous spaces are more realistic, although leading to higher complexity of agent learning. In this vein, Rummery and Niranjan (1994) ascertain that discrete solutions are too restrictive for many practical applications, recommending the use of high-dimensional continuous state spaces. For this reason, we decided to adopt continuous state and action spaces. On the state space side, the observations of financial market variables tend to be continuous by nature, so in this case it would be a requirement.

On the action space side, both discrete and continuous solutions are possible to implement. However, the discrete alternative imposes that the actions, i.e. the weights, can only assume certain predetermined levels, say 0%, 25%, 50%, 75% and 100%. This has known limitations, most notably the "curse of dimensionality" (Lillicrap et al., 2015). In fact, the number of actions increase exponentially with the number of assets in the portfolio. For example, considering the five levels of discretisation referred above and the four assets we are considering, it would result in an action space with dimensionality $5^4 = 625$. Both numbers could be be much higher in real life, so this represents a serious scalability problem. On the portfolio management side, the limitations of this procedure are also evident since the weights for each asset in the portfolio are predetermined and specific, which may not be the optimal exposure to each asset. Moreover, with the discretisation process we are discarding important information from the action space that may be crucial for solving the RL process. In conclusion, the optimal and natural solution for portfolio management is the one where the weights can vary continuous from 0% to 100%. It is the most realistic option and adopted for this work.

5.3.1.2 Policy optimisation and function approximators

In the majority of situations, the agent is confronted with states not seen before and will need to generalise, to make good decisions about the next action to take. The necessary function approximation may be carried out by any of the methods used in supervised learning, which may be integrated in the RL algorithm (Sutton and Barto, 2020). Consequently, all types of solutions can be adopted for function approximation, from simple linear models to complex nonlinear neural networks.

Although linear function approximators have been used in recent studies (Pendharkar and Cusatis, 2018), nonlinear functions may lead to enhanced performance of the RL agent. Thus, some of the models studied in Chapters 3 and 4 can be used for this purpose. Our solution considered a deep neural network for both the actor and critic networks, which will be presented in Section 5.3.2.4.

5.3.1.3 Objective function

The objective function to be maximised by the agent is represented by a target strategy metric which can be selected from a range of metrics (Almahdi and Yang, 2017; Jiang et al., 2017): portfolio return, Sharpe ratio, differential Sharpe ratio, Calmar ratio, among others.

Given that the main objective of this work (Objective 6, Section 1.2) is to test the concept of RL in portfolio management of fixed income portfolios, we opted for the simplest objective function, i.e. {1+portfolio return}, and variations of this one to better differentiate the agents' rewards for positive or negative outcomes. The alternatives to portfolio return will be detailed in Section 5.3.4. The "portfolio return" metric represents a risk-neutral portfolio management, where the maximisation of return is the only objective. Other strategies are also common in the industry, namely the ones considering risk-return trade-off using metrics such as the ones referred above (Luenberger, 1998). However, for our objective this is the simplest and most natural objective function to consider (Moody and Wu, 1997). In future work, it is important to consider other objective functions, which can be incorporated in our RL system once the initial objective is successfully implemented (further details in Section 6.2).

5.3.1.4 Normalisation of observations

The normalisation of observations is an important factor that helps the system to learn. This is equally important in the case of RL (Lillicrap et al., 2015), and in supervised learning (Ioffe and Szegedy (2015) and Section 3.1.5). The observations passed to the agent comprise individual ETF daily returns (and respective lags), together with the respective weights in the previous time step. The weights have an imposed range $[0, 1]$, while most of the ETF returns will fall in the range $[-1, 1]$ when expressed in percentage terms (93%), with a reduced number slightly outside that interval. As a result, in the particular case of bond portfolio management where returns tend to be smaller than is some other assets classes, there is a natural method to normalise the observations. This method was implemented by introducing a Simulink *Gain* block, similar to the one used to scale the rewards (this will be detailed in Section 5.3.3 and Figure 5.3).

5.3.1.5 Performance metric

Considering a risk-neutral portfolio management approach, we selected the performance metric to be the cumulative return of the portfolio during the whole training or testing period, after transaction costs are considered. This metric is frequently presented in the form of total portfolio value (or cumulative capital value) of an initial unit of investment. For bond portfolios in particular, it is especially important to consider both the capital gain component and the income component of returns. This was implicitly taken into account when selecting the assets (Section 5.2 and Table 5.1).

To further support our choice, we refer the recent study by Betancourt and Chen (2021). The authors tested two performance metrics to evaluate the profit-seeking and the risk-aversion behaviours of their actor-critic algorithm (Section 2.7.5.3). For that purpose they tested two different setups, one using total return in the period as performance metric, and a second setup using the differential Shape ratio (Moody and Wu, 1997). They found that the algorithm setup using total return was overall the more robust solution for both profit-seeking and risk-aversion.

5.3.1.6 Hyperparameter tuning

Whenever possible, we considered as reference for the hyperparameter tuning, values used in the literature in other applications of RL to portfolio management or financial trading (Jiang and Liang, 2017; Jiang et al., 2017; Liang et al., 2018; Aboussalah et al., 2021; Pigorsch and Schäfer, 2021). Since hyperparameters are not portable across different applications, this was the first source of information.

Additional recommendations were taken from the literature, in particular with reference to the specific algorithms used in our experimental work (Lillicrap et al., 2015; Mnih et al., 2015), and software platform used (MathWorks, 2021b).

Finally, further tuning was carried out for the learning rate, as it is recognisedly one of the most important hyperparameters in machine learning (Goodfellow et al., 2016). For reference, a list of the hyperparameters used is presented in Appendix C.

5.3.2 Agent

In the following sub-sections we outline the criteria used to select the agent's algorithm, the DDPG. Then, after a brief summary of the algorithms from which it has evolved, we describe the DDPG algorithm in detail. At the end, we specify the neural networks used as function approximators in our portfolio management problem.

5.3.2.1 Algorithm selection

We considered several factors while selecting the algorithm to be used in our experimental work. Our main requirements were:

- Model-free - this is an essential requirement since we do not have a model for the workings of financial markets;

- Can integrate deep function approximators - this is an important feature given the flexibility that neural networks enable and their capacity for modelling complex systems;
- Online - the objective is to easily adapt the algorithm to the financial sources of information and the data frequency;
- Permits high-dimensional problems - this requirements is a results of the complexity of financial markets and the number of features required for a better representation of RL states;
- Continuous state and action spaces - the desirability of continuous spaces was discussed in Section 5.3.1.1;
- Deterministic policies - apart from the direct mapping from states to specific actions, deterministic policy gradients benefit from computing efficiencies (Silver et al., 2014).
- Actor-critic architecture - the advantages of this type of agent are widely present in the literature as outlined in Section 2.7.4. However, it is also a research interest and it follows the fact that considerable less studies in the literature have been carried out with this type of agent, adding some novelty to our study.

A summary of agent types and some of the most recently developed algorithms, which represent important progression of ideas in RL recent history, is presented in Table 5.2.

TABLE 5.2: Summary of agent types and algorithms.
Based on MathWorks (2021a), with changes made by the author. Terminology: SARSA, State-Action-Reward-State-Action; DQN, Deep Q Network; PPO, Proximal Policy Optimization; SAC, Soft Actor-Critic; DPG, Deterministic Policy Gradient; DDPG, Deep Deterministic Policy Gradient; TD3, Twin Delayed DDPG.

		Actor		
		$\longleftarrow$ Discrete actions $\longrightarrow$		
			$\longleftarrow$ Continuous actions	$\longrightarrow$
		None	Stochastic	Deterministic
Critic	None		Policy Gradient	
	Value		Policy Gradient Actor-Critic PPO	
	Q-Value	Q-Learning SARSA DQN	SAC	DPG DDPG TD3

The information presented directly points to three specific algorithms: the DPG (Silver et al., 2014); the DDPG (Lillicrap et al., 2015); and the Twin Delayed DDPG, TD3 (Fujimoto et al., 2018). Our selection is the DDPG, which is a state-of-the-art algorithm and a good compromise between the DPG, one of the models from which the DDPG evolves (further details in Section 5.3.2.2), and the most advanced and complex alternative, the TD3. The latter work also proposes a deterministic algorithm, which is considered an improvement on DDPG (Haarnoja et al., 2018). This algorithm will be considered for future work (Section 6.2).

Finally, the DDPG has been found to be one of the most effective off-policy deep RL methods (Duan et al., 2016). Nonetheless, some potential problems were also pointed out in the same publication, namely: less stability when compared to batch algorithms (methods in which at each iteration, a number of trajectories are generated), and that the performance of the policy can deteriorate significantly during training.

5.3.2.2 Original algorithms leading to the DDPG

The DDPG algorithm presented in a seminal work by Lillicrap et al. (2015), combines the DPG developed by Silver et al. (2014) and the Deep Q Network (DQN), proposed by Mnih et al. (2013, 2015).

To briefly cover the original models, first, the DPG, is also known as Compatible Off-Policy Deterministic Actor-Critic (COPDAC). As the name indicates, it is an actor-critic algorithm, where the critic is, in this case, a linear function approximator estimating the action-values (Q). To ensure sufficient exploration, the action-values are learnt off-policy with samples from a stochastic behaviour policy distinct from the deterministic target policy, using for example Q-learning (Watkins, 1989; Watkins and Dayan, 1992). On the other side of the actor-critic, Silver et al. (2014) demonstrated that the deterministic policy gradient has the form of the expected gradient of the action-value function. Hence, the actor updates its parameters by gradient ascent, in the direction of the critic's action-value gradient. This is more appropriate for continuous action spaces than the typical alternative of greedy maximisation, that requires maximising Q globally at every time step (Silver et al., 2014).

One of the main contributions of the DPG work (Silver et al., 2014) was the extension of the policy gradient framework to deterministic policies, by clearly demonstrating that tools used in stochastic policy gradients could also be applied to deterministic policy gradients, namely: compatible function approximation (Sutton et al., 2000), natural policy gradients (Kakade, 2002), actor-critic architecture (Bhatnagar et al., 2008), and episodic/batch methods (Peters et al., 2005).

The second model in which the DDPG is based, the DQN, is a critic-only algorithm, where the critic function approximator is a deep neural network. Given that RL is known to be unstable and even diverges when neural networks are used as function approximators, the DQN algorithm incorporates two innovative techniques to improve stability (Mnih et al., 2015). First, the critic is trained off-policy from samples stored on an experience replay buffer (also known as experience replay or replay buffer).

Randomised sampling from this buffer is used to remove correlations from the data (observation sequences) and smooth changes in the data distribution. The second innovation, aims to reduce possible correlations between action-value (Q) and the target Q-values, thus improving stability. For this purpose two identical neural networks are considered: one network (Q network) updating its parameters at every time step; while the second network for the optimization target (target Q network), only updates its parameters periodically (at every t time steps). The latter network is frozen between updates, and at the end of every t time steps the weights of the Q network are copied to the target Q network and they become again identical. Hence, only the Q network is trained.

In summary, the DDPG algorithm results from the actor-critic architecture of the DPG combined with the two innovations from DQN, plus the concept of batch normalisation (Ioffe and Szegedy, 2015). Some of these concepts will be further detailed in Section 5.3.2.3.

5.3.2.3 Deep deterministic policy gradient algorithm

The DDPG algorithm was developed at Google Deepmind, UK, by a team of researchers, whose main contribution was "to provide modifications to DPG, inspired by the success of DQN" (Lillicrap et al., 2015). The authors base their idea on the actor-critic architecture of DPG, but instead of using a linear function approximator as the critic, they use a deep neural network. Additionally, based on the DQN algorithm, they duplicate the critic network. The modifications here consist of adapting to the actor-critic architecture, and making "soft" updates of the target network' parameters. Note that, in contrast, in DQN the parameters are updated periodically only, remaining frozen between updates (Section 5.3.2.2).

In what the authors identify as an improvement in stability, they do this duplication of networks for both the critic and the actor. As a result, the DDPG algorithm comprises four deep neural networks, two for the critic - the *Q network*, $Q(s, a|\theta^Q)$ and the *target Q network*, $Q'(s, a|\theta^{Q'})$ and two for the actor - the *behaviour policy network* $\mu(s|\theta^\mu)$ the *target policy network* $\mu'(s|\theta^{\mu'})$ The first critic network, updates its parameters by gradient descent, in order to minimise the loss:

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\mu, a_t \sim \mu, r_t \sim E}\left[\left(y_t - Q(s_t, a_t|\theta^Q)\right)^2\right] \tag{5.16}$$

$$y_t = r(s_t, a_t) + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \tag{5.17}$$

where ρ^μ represents the discounted state visitation distribution for a policy μ; y_t is the target action-value; θ^Q and $\theta^{Q'}$ correspond to the parameters of the Q and target Q networks, respectively; while $\theta^{\mu'}$ are the parameters of the target policy network.

In relation to the first actor network, it updates its parameters by gradient ascent, in the direction of the critic's action-value gradient. Applying the chain rule of differentiation to the expected return, we obtain:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right] \tag{5.18}$$

$$\overset{\text{(chain rule)}}{=} \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t} \right] \tag{5.19}$$

where θ^μ are the parameters of the behaviour policy network; ∇_a and ∇_{θ^μ} denote the gradients with regards to the action a and the parameters of the behaviour policy network θ^μ, respectively.

The second networks for the critic and actor, respectively, conduct "soft" updates, as follows:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{5.20}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{5.21}$$

where the parameter $\tau \ll 1$, is the target smooth factor.

To ensure adequate exploration, and taking into account we are in continuous action spaces, noise is added to the output of the actor behaviour policy μ, obtaining what the authors call the "exploration" policy:

$$a_t = \mu(s_t|\theta_t^\mu) + \mathcal{N} \tag{5.22}$$

where $\mathcal{N}$ denotes stochastic noise sampled from a noise process, for example from an Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930). The complete DDPG algorithm is reproduced in Algorithm 2 from the original paper (Lillicrap et al., 2015).

There are two additional aspects of the DDPG algorithm that should be mentioned due to its importance: the experience replay buffer and normalisation. The experience replay buffer, is another idea based on the DQN algorithm (Mnih et al., 2015). While training, the algorithm stores previous transitions (or previous "experiences") of the type (s_t, a_t, r_t, s_{t+1}), i.e. the sequence state-action-reward-next state. This buffer of past experiences is used to train the model, in an analogy to the human learning process. Hence, at each time step, the agent selects minibatches randomly from this replay buffer to ensure the samples are independent and identically distributed.

Finally, the batch normalisation proposed for RL (Lillicrap et al., 2015) is based on a similar concept from deep learning (Ioffe and Szegedy, 2015). It may be useful in case of low-dimensional observations, with different types of units and ranges, sparing the work of manually scaling the features. It consists of normalising each dimension across samples in one minibatch so that they have unit mean and variance. At the same time,

Algorithm 2: DDPG algorithm, reproduced from Lillicrap et al. (2015).

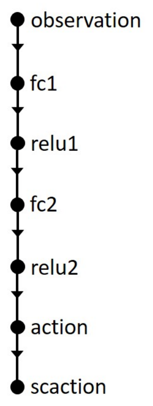
```
1  Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$
     and $\theta^\mu$.
2  Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$.
3  Initialize replay buffer $R$.
4  for episode = 1 to $M$ do
5      Initialize a random process $\mathcal{N}$ for action exploration.
6      Receive initial observation state $s_1$.
7      for $t = 1$ to $T$ do
8          Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and
             exploration noise.
9          Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$.
10         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$.
11         Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$.
12         Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
13         Update critic by minimizing the loss:
               $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
14         Update the actor policy using the sampled policy gradient:
               $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
15         Update the target networks:
               $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
               $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16     end for
17 end for
```

they store the running average of both the mean and the variance, to use during the testing phase.

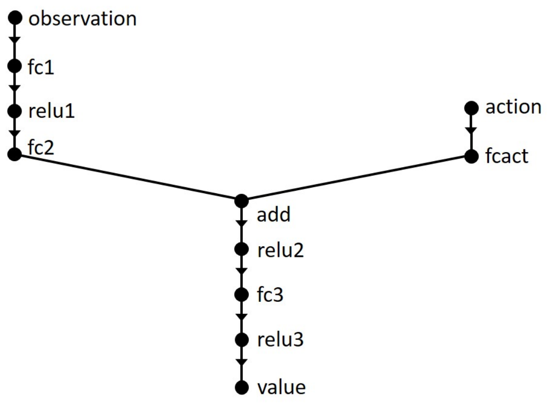
5.3.2.4 Actor and critic deep neural networks

The neural networks for both the actor and critic are presented in Figure 5.2. The actor network is a standard feedforward neural network mapping from state observations to actions, including fully-connected (*fc1* and *fc2*) and ReLU activation (*relu1* and *relu2*) layers (Figure 5.2a). The last layer of the network scales the outputs (*action*) with a softmax layer (*scaction*), given that the actions correspond to the weights of each asset in the portfolio (Section 5.1.2).

Bearing in mind that the critic is part of an agent using a DDPG algorithm, the final output of this network is the state-action value at each time step (*value* in Figure 5.2b). As a consequence, it requires two separate paths: one for the observations and another for the actions. Theses branches converge into a common path via an addition layer



(A) Actor neural network



(B) Critic neural network.

FIGURE 5.2: Diagrams of actor and critic neural networks' paths, where *observation* represents the state observations; *action* the agent' actions; fc is a fully connected layer; $fcact$ is also a fully connected layer in the action path of the network; *relu* the activation function; *scaction* the scaled actions; and *value* is the action value, or Q-value.

(*add* in Figure 5.2b). Apart from this difference, the critic network is also a sequence of fully-connected and ReLU activation layers.

5.3.3 Environment

In this section, we present the environment developed for this work and discuss the selection of platform used for that purpose.

5.3.3.1 Platform selection

The environment is an essential part of the RL system. Although there are a number of notable resources available, such as OpenAI (OpenAI, 2022a,b) and Acme from DeepMind (Hoffman et al., 2020), they tend to cover specific applications, e.g. in games and particular cases for robotics. OpenAI, for example, has developed a set of high-quality implementations of RL algorithms, called OpenAI Baselines. These have been widely used by researchers to compare results and test new ideas against these baselines. However, they also recognise that "*(...) the existing open-source collections of RL environments don't have enough variety, and they are often difficult to even set up and use*" (OpenAI, 2022a), and "*(...) the field of deep RL has a pretty high barrier to entry for new researchers as well as practitioners (...)*" (OpenAI, 2022b). This is a problem that persists, although their contribution to solve it must be celebrated. All these factors make the application of RL to new fields, a task with added complexity and with a very steep learning curve. This is in sharp contrast to supervised and unsupervised learning, where standard libraries exist and may be used in a short period of time (e.g. Scikit-learn, Tensorflow, Keras, Pytorch).

In that context, environments specifically for portfolio management, that could be reliably adapted for the testing of RL agents/algorithms, are not available. Moreover, documentation and support online from these open source frameworks tend not to be ideal. For all these reasons, we decided to develop our own portfolio management environment, using MATLAB and Simulink (MathWorks, 2021b,c), both of which tend to be readily available at higher education and research institutions. This gives us an additional flexibility for adaptations that could not be available otherwise.

5.3.3.2 Simulink environment

A simplified diagram of the Simulink environment system is presented in Figure 5.3, including in the figure the agent (not part of the environment) to identify all connections between agent and environment.

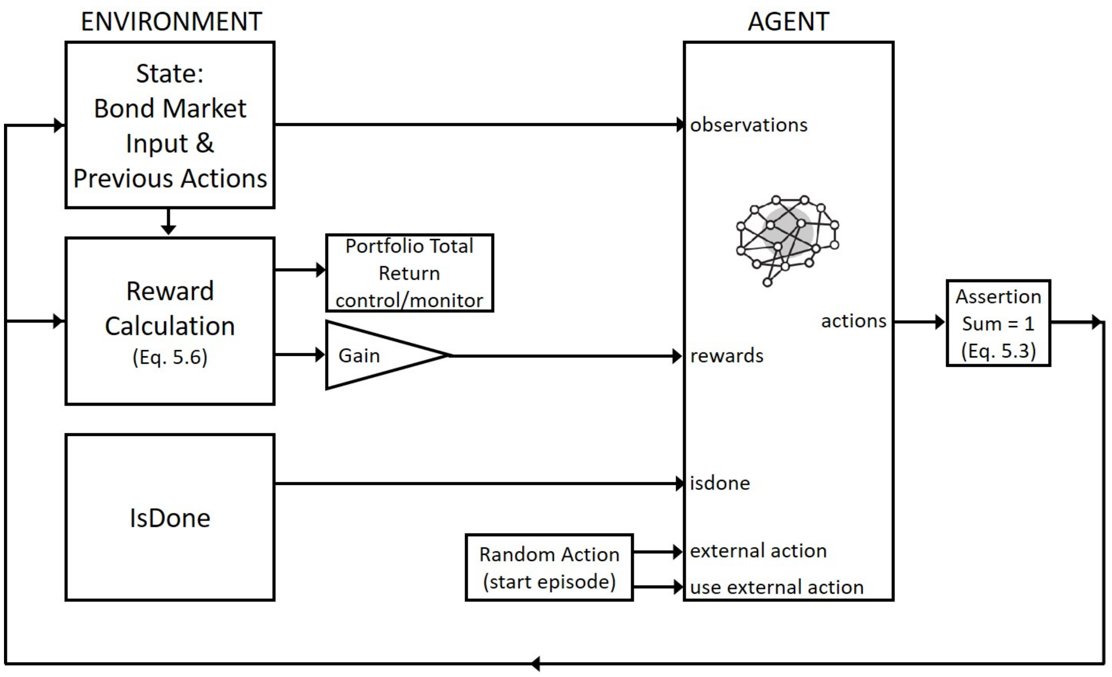

ENVIRONMENT
AGENT
State: Bond Market Input & Previous Actions
observations
Reward Calculation (Eq. 5.6)
Portfolio Total Return control/monitor
Gain
rewards
actions
Assertion Sum = 1 (Eq. 5.3)
IsDone
isdone
Random Action (start episode)
external action
use external action

FIGURE 5.3: Portfolio management Simulink environment and agent.

All the elements connect in a sequence process similar to the one represented simplistically in Figure 2.10. The main subsystems are: the *state*, the *rewards*, and the *IsDone*. The *state* subsystem is responsible for inputting the time series from the four ETFs of our portfolio. Since we consider a window of 6 time steps per asset, this includes 24 time series in total, i.e. one per asset, up to the present time step, and 5 (per asset) lagged versions of the first. In addition, the state also receives the previous actions, i.e. the weights of the portfolio in the previous time step. This subsystem sends this information to the *rewards* subsystem and to the agent. The *rewards* subsystem is responsible for the calculation of rewards, taking into account transaction costs, in accordance with Equation 5.6 and sending the reward signal to the agent. In turn, the *IsDone* subsystem is simply a counter that sends a signal to the agent, identifying if the end of the episode has been reached.

Additional supporting subsystems are included in the diagram. First, the *Portfolio total return* subsystem controls, stores and enables the visualisation of this variable during execution. Second, the *Gain* subsystem enables to apply a factor to the reward signal. The objective of this is to be able to change the scale of rewards to study the sensitivity of the algorithm to their scale (further details in Section 5.3.4).

Third, the *Random action* subsystem is used to randomise the initial weights at the start of each episode. In a typical portfolio management study, the portfolio starts with the whole capital in cash, or in the equivalent asset (Jiang et al., 2017). To force the portfolio to start from a different portfolio exposure and take different trajectories during an investment episode, we introduced this randomisation. This should contribute to avoid local optima in the RL process.

Last, the *Assertion* subsystem is there to ensure that the sum of the actions (weights) is equal to one. It is important to highlight that in the DDPG algorithm (Lillicrap et al., 2015), noise is added to the actor policy to obtain the exploration policy (Equation 5.22). As a result, the simple inclusion of a softmax layer in the actor network does not guarantee that the sum of all weights is equal to one, given that the addition of noise occurs after this point in the cycle. Hence, to ensure that this constraint (equality to one) is not violated, we have considered an additional block in the environment immediately after the agent output in the RL system.

5.3.4 Empirical work to test the RL system

In this section we describe the experimental work carried out with the main objective of testing the RL system, thus evaluating its potential for fixed income portfolio management. A summary of the work is outlined in Table 5.3.

In what concerns the testing data, we consider the final agent, i.e. the resulting agent at the end of the training phase, but also the best performing agents during the training phase. The selection criteria for those best performing agents must be objective in order to avoid any bias in the selection process. Therefore, we select all agents that satisfy the following two conditions: they obtain a total episode reward higher than 90% of the maximum episode reward obtained by all agents during training; and the running average episode reward in the last 10 episodes is also above the 90% level.

TABLE 5.3: Empirical work for RL system testing.

Data	Agent	Objective
Training		Test the system functioning and especially the capacity of learning.
Testing		Test the system in unknown data and compare to the benchmark.
	Final agent	Compare the performance of the final agent resulting from the training phase.
	Best agents	Compare the performance of the best performing agents during training. Selection criteria: episode reward $>$ 90% of the maximum episode reward during training, when the running average episode reward is also above the 90% level.
Reward alternatives		
		$\{1 + R_{P,t}\}$; $\{R_{P,t}\}$; $\{R_{P,t} \times 10^4\}$
Benchmark		
		Buy & Hold strategy of individual ETFs, with particular focus on the performance of the best asset.

The goal of the latter is to exclude agents that could correspond to a spike in the learning process. Overall, a selection of this type would correspond to selecting the best performing portfolio managers.

We considered three alternatives for the rewards. First, the {1+portfolio return} approach as presented in Section 5.1.4 and Equation 5.6. Second, we consider the simple portfolio returns, which are centered at positive but small value close to zero.

Finally, building on the previous alternative we amplify the reward multiplying it by 10^4. The rationale for doing so is connected to the basis point (bp), which is one of the most common units in the bond market, for interest rates, yields and also for returns. One bp is defined as $1/100th$ of 1%, or 10^{-4}. As an example, if the bond portfolio returns move in a range of 0 to 1%, the rewards would have a scale from 0 to 100. Hence, with this alternative approach, the rewards are approximately centered around zero and we ensure a higher differentiation of rewards between positive and negative performance by the agent. On top of that, using this approach, we can test the sensitivity of the RL algorithm to the scaling of rewards, a subject pointed out by Duan et al. (2016) in relation to the DDPG algorithm, but not given considerable attention in subsequent literature.

The benchmark considered is the buy and hold strategy of individual ETFs. However, the main goal is to compare the RL results against the best performing asset in the portfolio, which is a more demanding benchmark.

5.4 Results and discussion

In this section, we present and discuss the results obtained. We start with a brief reporting of the results obtained with the different reward alternatives tested. Then, we proceed with the results from both the training phase and the testing phase, since both phases provide useful information.

5.4.1 Reward alternatives

We considered the three different reward alternatives referred to in Table 5.3. In the first two alternatives, $\{1 + R_{P,t}\}$ and $\{R_{P,t}\}$, the system failed to learn from the starting point, producing learning curves close to flat. On the contrary, in the third alternative, $\{R_{P,t} \times 10^4\}$, there was noticeable and rapid learning from the starting point. These results are not presented here due to the lack of additional information.

This was somewhat unexpected given that those rewards have been used in the literature (Moody and Wu, 1997; Jiang et al., 2017; Filos, 2019; Koker and Koutmos, 2020). However, we highlight that the vast majority of those studies cover other asset classes, where the dimension of the returns tend to be higher. They also use different types of algorithms, which are conceptually different from the actor-critic architecture we are using. Specifically for the DDPG algorithm, we found indications in the literature that it could be susceptible to the scale of the rewards (Duan et al., 2016).

In the case of bonds, which typically have smaller daily returns, the differentiation between rewards given to the agent for positive and negative actions is much higher using the scaled reward, and this may facilitate the learning process. For the reason explained above, the results presented below in Sections 5.4.2 and 5.4.3, concern the last reward alternative studied ($\{R_{P,t} \times 10^4\}$).

5.4.2 Training results

The most important results to test the quality of an algorithm are results obtained in testing data, not known to the model in advance. However, in our case the training results' information is also relevant, because it will show if the RL system is working and the system is able to learn. This is an important part of our objective for this work, given that we have developed some of the components of the system from the beginning. In Figure 5.4 we present the results obtained during the training phase, at evenly spaced episodes.

The results demonstrate a significant learning from a very low portfolio return in the initial run. In fact, we can observe that several agents obtain very high returns, multiplying several times the returns of the best performing asset in the portfolio.

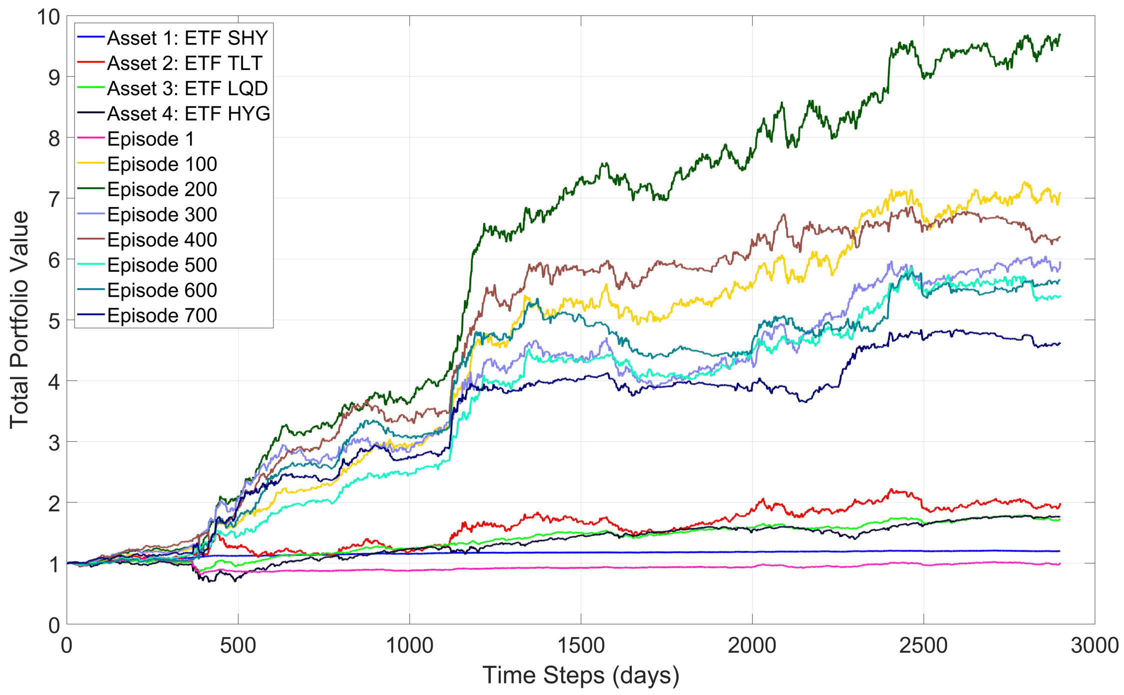


FIGURE 5.4: Portfolio total return at various episodes during the training phase, compared to the total return of the individual assets.

Another observation from the plot is that the curves are not trending to a specific optimum level, although they remain at high levels when compared to the assets forming the portfolio. This is an indication that the algorithm used is not converging to a certain level of portfolio return, or of episode reward, and shows that the learning progress is not totally stable. This is also confirmed by the evolution of the training-progress plot, featuring episode reward against episode number (not presented here to avoid duplication). In one of the few studies using the DDPG algorithm in portfolio management, Snow (2020b) also found this problem. The author notes this RL algorithm can be unstable, is generally hard to converge, and it tends to overfit. Difficulty converging issues have also been reported elsewhere (Liang et al., 2018; Aboussalah et al., 2021), and presented in Section 2.7.5.

5.4.3 Testing results

The results of the testing phase are divided in two parts: the results of the final agent (Figure 5.5), and the results of the best agents (Figure 5.6), selected as explained in Section 5.3.4.

The results from the final agent show a clear underperformance when compared to the total return of the individual assets. Given the observed lack of stability of the algorithm when applied to our portfolio (Section 5.4.2), just leaving the RL system to

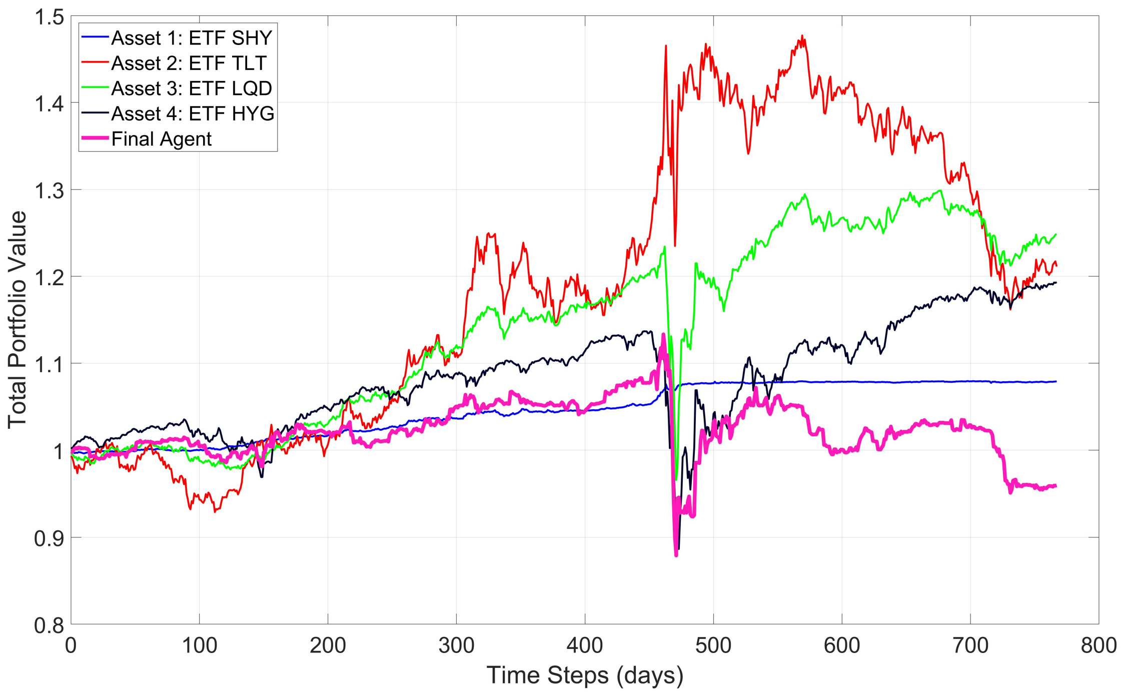


FIGURE 5.5: Portfolio total return with testing data, using the final agent of the training phase, and compared to the total return of the individual assets.

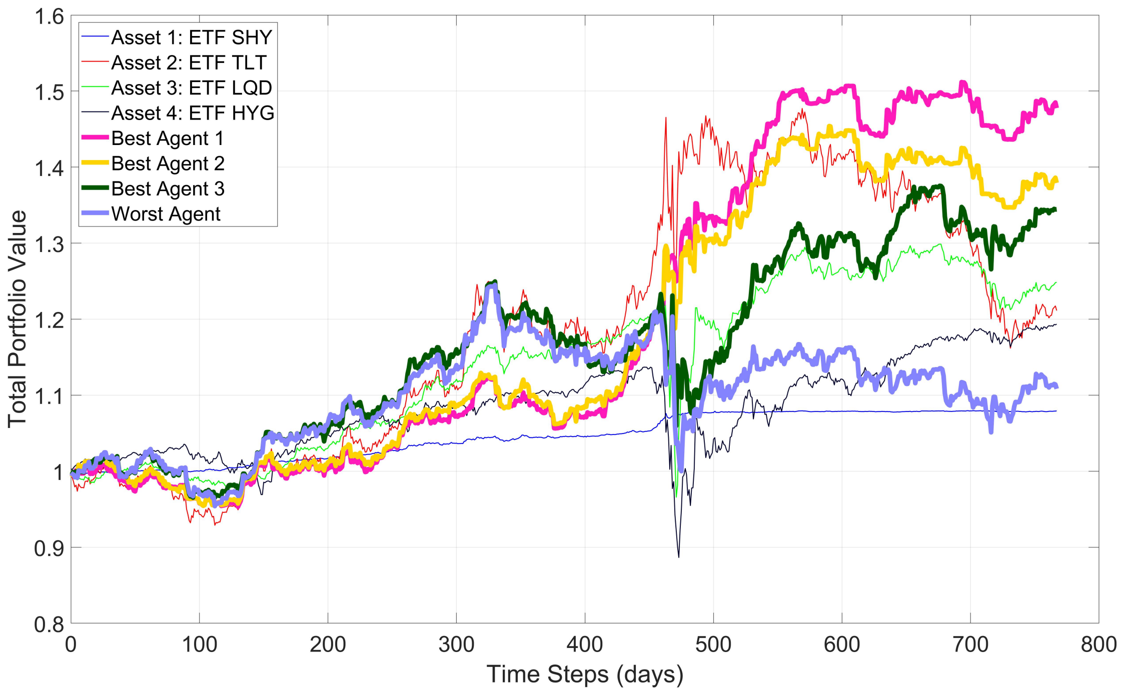


FIGURE 5.6: Portfolio total return with testing data, of the top three agents selected during the training phase, and compared to the total return of the individual assets. For reference, we also add the bottom performer of the selected agents. Selection criteria: agents with episode reward above 90% of the maximum, when the running average episode reward is also above the 90% level.

run until the end of the maximum number of episodes specified for training, may result in a final model far from ideal.

The following phase of the empirical work, with the selected agents (Table 5.3) is able to overcome, at least partially, this issue. Hence, in Figure 5.6 we show the top three agents and also the worst, versus the performance of the individual assets. From this figure, we can observe that by selecting the best agents we are able to outperform the simple buy and hold strategy of any of the assets in our portfolio. Since it is outperforming the best asset, by definition it would also beat the traditional equal weight portfolio strategy.

Nevertheless, some selected agents do underperform in relation to the best asset. It may be because the selected agent is at an early phase of training, as in some cases, or other reasons. Despite that, considering a simple ensemble of all agents satisfying out objective selection criteria (Section 5.3.4 and Table 5.3), with equal weights, we obtain a final portfolio return of 23%. This result is very close to the best performing asset at 25%, while benefitting from a lower drawdown (peak-to-trough decline) during the large correction in the early phases of Covid-19. For a better weighting of the agents in the ensemble, we would need to consider market regimes (more long term, persistent trends). Then, develop a methodology to select from the available agents, representing portfolio managers, the best for the specific regime in which the market trades at the time. This would take us into another area, outside the scope of this work.

In summary, even with an algorithm that demonstrated some stability problems during learning and also overfitting, it is possible to select agents that outperform. So the potential for RL in fixed income portfolio management is clear.

5.5 Summary and conclusions

RL provides an advanced framework for sequential decision making, which is the main task of a portfolio manager when deciding on asset allocation. It also provides an alternative to the difficult problem of directly forecasting financial assets, prices or yields, given the complex nature of financial markets. With RL, we can overcome this issue, with no model of the market required, and focusing directly on the optimisation of our ultimate objective.

When transitioning from supervised learning to RL, there are a number of barriers due to the lack of open-source environments for very specific financial applications and of high quality standard libraries, with the right documentation and online support. In our case, the application for this part of the research (fixed income portfolio management) differs from the traditional applications found in the literature for RL (general multi-asset portfolio management, of equity or equity-bond type portfolio).

Taking this into consideration, we start by formulating the problem of portfolio management in the RL context. Then, we develop an autonomous RL system using an environment created specifically for fixed income portfolio management. The other main component of the system is an actor-critic type of agent that interacts with that environment. The agent incorporates an algorithm, selected following a set of requirements outlined in Section 5.3.2.1, suitable for our specific application. The algorithm selected was the DDPG, a state-of-the-art model developed at DeepMind.

We introduced a number of modifications in our methodology in relation to studies in the literature in the development of our RL system. In particular, we considered a scaled alternative for the rewards, to test the sensitivity of the algorithm to the scale of rewards. Other modifications introduced concern: the normalisation used, particularly useful for bond assets; the *sofmax* block in the environment to assert that the sum of the portfolio weights equal to one; and the randomisation of the initial asset allocation, to improve exploration.

The results showed that the DDPG algorithm is sensitive to the scale of rewards, with the unscaled version not producing significant learning. As to the training results, they demonstrated that the RL system is working and capable of learning. With this data, the agent tends to obtain a final portfolio value several times higher than the return produced by the best individual asset. Another important observation during training, related to the DDPG algorithm, was the lack of convergence to an optimum value. Indeed, the algorithm showed signs of unstable behaviour and overfitting.

Taking that into consideration, to seek outperformance on testing data, we could not wait until the end of the testing phase and use the corresponding final agent. In fact, we had to devise objective criteria for the selection of the best agents, during the training phase. The results showed that it is possible to select agents that outperform all individual assets. Moreover, an unsophisticated ensemble, with equal weights, would achieve a result very similar to the best performing asset in the portfolio.

Overall, we consider our results satisfactory for testing the autonomous system developed and as a proof of concept. They give clear indications of the potential of RL for fixed income portfolio management. Nevertheless, further work is required to improve the results on testing data and the stability of the agent. Concrete steps are outlined in Section 6.2.

Chapter 6

Conclusions and Future Work

In this chapter we present the main conclusions of this thesis, linking them to the initial objectives of the research (Section 1.2). Additionally, we outline potential lines of research for future work.

6.1 Conclusions

The literature review carried out revealed four main gaps and the objective of this research is to fill those gaps. The first concerns the limited amount of publications on the use of machine learning techniques applied specifically to fixed income markets. Other financial areas have been the target of much higher levels of interest from the research community, such as equities and forex. This gap was used to identify an asset class that is relatively under-investigated using machine learning as the target for this research.

The second gap is more specific and relates to the lack of direct solutions in the literature for modelling the yield curve as a whole using machine learning models. For this purpose, multitask learning was identified as a possible solution and was evaluated in this research. This second gap led to our research presented in Chapter 3, which is the first study applying machine learning, specifically multilayer perceptrons, and multitask learning to model the yield curve as a whole.

The third gap is directly linked to the black-box concept behind the neural network type of model. In fact, the lack of explanatory power and of understanding how and why these models work at an internal level, hinders its broader use by financial practitioners. On this issue, past research makes limited attempts at such internal explanations. This gap led to the research presented in Chapter 4, which is the first contribution in that direction.

Finally, the fourth gap in the portfolio management field concerns the reduced number of published studies in the most complex RL architectures, together with the contrasting performances reported. In addition, there is a lack of resources for this type of application, such as specific RL environments, and the absence of studies focusing exclusively on bond portfolios. This gap led to the research presented in Chapter 5.

Therefore, this thesis includes three different topics of research, addressing the gaps identified above. In the first, we conducted a comprehensive study using a wide range of features from macroeconomic and market indicators (Objective 1). We identify as targets for modelling five benchmarks of the European yield curve (3 months, 2, 5, 10 and 30 years), and consider five different forecasting horizons (next day, 5, 10, 15 and 20 days). Two different multitask learning techniques were used (Objective 4): simultaneous modelling of targets (using both yields and forecasting horizon as targets) and transformation into independent single task learning problems. In addition, having the linear regression model as a benchmark, several MLP models were considered (Objectives 2 and 3), specifically: an MLP using all features, a univariate type of model, and an intermediate MLP making use of the most relevant features by target and by forecasting horizon. Additional models were designed using an innovative approach of combining the linear regression and MLP model, by using synthetic data generated by the first as input features in the second (Objective 5).

Our recommended methodology uses a moving window of training data to incorporate the most recent information as it becomes available. Besides, we do the cross-validation dynamically using these moving windows and combine the concepts of bootstrap and bagging. Furthermore, the retraining of models happens at every time step, having the advantage of increased flexibility to changing market conditions. Our results also highlight the vital role of the feature selection process. In particular, we found that the most relevant features change depending on both target and forecasting horizon. This provides important novel insights into the importance of choosing the right methodology in the overall process, in contrast to previous research where a small set of features is pre-determined and used irrespective of the conditions of the problem where they are applied.

In terms of results obtained from the models studied, the MLP using the most relevant features for each target and each forecasting horizon achieved the best results overall (Objectives 1 and 2). While the performance of all models studied is comparable for next-day forecasting, the differences become significant for longer forecasting horizons. Overall, our results are better than the ones in comparable studies from the literature. The results also demonstrate the positive effect of incorporating in the models, via additional features, artificially-generated data from other models (Objective 3). Indeed, the models using this type of data tend to achieve superior performance (Objective 5), especially for longer forecasting horizons and longer maturities (2 to 30 years), suggesting the potential for using hybrid models incorporating data generated by industry-established models. On the comparison of multitask and single task learning, no clear differentiation could be demonstrated (Objective 4). Potential reasons were presented, which are supported by the literature.

In summary, we show that MLPs can be successfully used for this purpose and recommend methodologies that may be used by practitioners to build better forecasting systems for bonds.

The second topic of this thesis can be subdivided in three main parts. First, we conduct an application of LSTM networks to the bond market, specifically for forecasting the 10-year Euro government bond yield, and compare the results to memory-free MLP models (Objective 2). This is among the first studies of its kind. To this end, we model the 10-year bond yield using univariate LSTMs with different input sequences (6, 21 and 61 time steps), considering five forecasting horizons, the next day as well as further into the future, up to next day plus 20 days. Our objective is to compare those LSTM models with univariate MLPs, as well as MLPs using the most relevant features. These are determined using Lasso regression, for each forecasting horizon. For this comparison, we use the same data and methodology as in the previous topic. As previously, we use a training moving windows. This showed to work well because it incorporates the most recent information as it becomes available, thus having the advantage of increased flexibility to changing market conditions.

The direct comparison of models in identical conditions shows that, with the LSTM, we can obtain results that are similar or better and with lower standard deviations. For forecasting horizons equal to 10 and 15 days, the superiority of the LSTMs becomes more evident. In fact, the multivariate MLP needs additional information from markets, to reach the level of accuracy of the univariate LSTM model with longer input sequences. This is a remarkable achievement and a promising result for future work. Furthermore, the results for the univariate LSTM show that shorter forecasting horizons require smaller input sequences and, vice-versa. Therefore, there is a need to adjust the LSTM architecture to the forecasting horizon and in general terms to the conditions of the problem. In summary, the results obtained in the comparative study validate the potential of LSTMs for yield forecasting and identify their memory advantage when compared to memory-free models.

Second, with the objective to analyse the internal functioning of the LSTM model and mitigate the preconceived notion of black-box normally associated with this type of model, we conduct an internal analysis of the information in the memory cell through time (contributing to Objective 2). Our research is the first contribution with that objective. Alternative works are either applied to a different type of model, or conducted an external analysis of the LSTMs (Section 4.1). To achieve this goal, we select several locations within the memory cell to directly calculate and extract the signals (weights) at each time step and hidden unit. Specifically, the locations are as follows: forget gate, product of the outputs from the input gate and input node, output gate, cell state, and hidden state. This analysis is carried out using sequence-to-sequence (6 days) LSTM architectures, with uni and multivariate feature sets and, for interpretability purposes, with reduced number of hidden units (3 units). We used a forecasting horizon of next day plus 5 days. Overall, considering all feature sets, the most remarkable property found consistently in the LSTM signals, is the specialised switching behaviour. This is manifested by the activation / deactivation of units through time, thus contributing or not (respectively) to the forecasting process. Moreover, we found evidence that the LSTM units tend to specialise in different yield ranges or features considered in the model.

As the third and last part of the topic, we investigate the information contained in the signals extracted from the LSTM hidden and cell states, to examine whether the corresponding time series can be explained by external sources of information (contributing to Objective 1). To this effect, we introduce a new methodology, here identified as LSTM-LagLasso, based on both Lasso and Kalman-LagLasso. This methodology is capable of identifying both relevant features and corresponding lags, as the Kalman-LagLasso, but with significant modifications (Section 4.3.3). The findings show that the information contained in the LSTM states is complex, but may be explained by exogenous macroeconomic and markets variables, not known to the model during the learning process. Thus, it is worth exploring this information using the developed LSTM-LagLasso methodology, which may be used as an alternative feature selection method. On the relevant features selected with the LSTM-LagLasso method, they indicate conventional as well as non-conventional market/macro indicators (Section 4.4.3), contributing to the prediction process, but which are not commonly used in forecasting models. In addition, the LSTM-LagLasso identifies lags as important, in particular t, $t-1$ and $t-5$. Above all, LSTM networks can capture this information and maintain it in the long and short-term memories, i.e. cell and hidden states.

In the third topic of this thesis, we apply RL to the portfolio management problem (Objective 6). For this purpose, we develop an autonomous system which includes a newly created environment specifically for fixed income portfolio management, which interacts with an agent using a state-of-the-art algorithm developed by DeepMind, the DDPG. Our work is among a small number of studies testing this algorithm for portfolio management (Section 2.7.5.3). To test this system, we select four bond ETFs to represent our fixed income portfolio and use their historical prices as inputs to the environment. The methodology developed involves a number of modifications in relation to other studies in the literature and are detailed in Section 5.3. These modifications are due to the specific asset class and algorithm we use, but they also reflect aspects previously pointed out in the literature which had no subsequent development. In this particular situation it was the scale of rewards that had direct impact on the learning capacity of our RL system.

The system was successfully tested, demonstrating the capacity to learn from the training data. Hence, the total portfolio return of the agent during training is several times higher than the total return of the best asset in the portfolio. On the agent side, the algorithm showed some signs of instability that require further research. Some potential follow-up on this issue is outlined in Section 6.2. Nevertheless, we demonstrate how to extract the best agents (or portfolio managers) from the system, during training, using an objective selection criteria. Some of these agents are capable of outperforming (in unknown testing data) both the static equal-weight buy and hold strategy and the best asset in the portfolio, considered as benchmarks. In fact, a simple ensemble of those selected agents, with equal weights, performs at a level equivalent to the best asset in the portfolio. Overall the results confirm the potential for RL in fixed income portfolio management.

To finalise, the main practical implications of this thesis is the demonstration that several machine learning models can be used successfully for forecasting and portfolio management in fixed income markets. It represents a powerful tool that the industry is already using, although it is not widespread or completely established yet, especially when applied to bonds. This is the main rationale behind the present study. We are certain that this line of research will continue in the future given its potential and the promising results obtained so far, reported in this thesis and elsewhere in the literature. Overall, we have extended the state-of-the-art literature, developing methodologies and proving concepts that are useful for financial practitioners in those two main areas of forecasting and portfolio management.

6.2 Future work

In this section we outline potential future research, directly related to the work reported in this thesis. In particular, we focus on two distinct areas: the study of internal signals in back-box models and RL.

Statistical significance tests

In Chapter 3, additional statistical tests can be performed on the distributions of predicted results obtained by each model, in order to strengthen our confidence on the statistical significance of the differences observed. In finance, one of the most commonly used methods to compare forecasts and determine statistical significance, is the Diebold–Mariano test (Diebold and Mariano, 1995; Diebold, 2015) and the modified version of this model proposed by Harvey et al. (1997). Both tests could be used to improve the robustness of the conclusions.

Internal signals in back-box models

We observed regime switching behaviour in the internal states in Section 4.4.2. It would be interesting to study if this type of internal unit behaviour can also be identified in other asset classes, for example in equities. This will increase the robustness of these findings and would contribute for a better understanding of the models.

The relationships we uncover between economic variables and internal states of LSTM models (Sections 4.4.3 and 4.4.4), are correlation-based as they result from simple linear regressions. A necessary next step will be to aim to infer causal relationships, casting them in a Granger causality setting. This will increase the explanatory power of the framework and lead to more in-depth understanding of how bond yields (and prices in general) are influenced by the environment in which trading takes place.

Reinforcement learning

From the results obtained in Section 5.4, two different lines of follow-up research ensue: one to improve the stability of the algorithm used by the agent and another to work on improving the results obtained by the best agents. On the first, there several other algorithms that can be tested in the developed platform, in search for a more stable agent. Some of the most immediate candidates for continuous action space are: Twin Delayed DDPG – TD3 (Fujimoto et al., 2018), which is an improved, more complex version of DDPG; Proximal Policy Optimization – PPO (Schulman et al., 2017b) has more stable updates but requires more training; and the Soft Actor-Critic – SAC (Haarnoja et al., 2018), which is also an improved, more complex version of DDPG, that generates stochastic policies. These are all actor-critic type of algorithms, but other alternatives could also be tested in the actor-only and critic-only arena. In comparison, these are simpler architectures, which could prove effective.

On the second line of follow-up research, we would consider the field of ensembles of best agents. In Section 5.4.3, we tested a simple ensemble using equal weights. Better ensembles could be built with better weighting of the agents. Ideally, this line of research should aim at selecting agents for different market regimes. The ensemble would then attribute adequate weights according to the regime in effect at the time.

Finally, future research should consider different objective functions, covering more closely the final set of objectives of an investor or client. Hence, the objective functions can incorporate risk-return metrics, real-world constraints such as individual taxes, investor preferences, and portfolio constraints. These more complex objective functions would go towards a wider application in the asset management industry.

Appendix A

Chapter 3 - Initial List of Features

TABLE A.1: Initial list of features selected from financial markets, macroeconomic and technical indicators.

Group	Subgroup	Ticker	Name
Interest Rates			
Rates	ON	EUDR1T	EUR Deposit O/N
	1M	EUR001M	Euribor 1 Month
	3M	EUR003M	Euribor 3 Month
	6M	EUR006M	Euribor 6 Month
	9M	EUR009M	Euribor 9 Month
	12M	EUR012M	Euribor 12 Month
ECB	ECB Refi Rate	EREU001W	ECB Effective Min Bid Refi
	3M Euribor Fut	ER1	Generic 1st 'ER' Future
	3M Euribor Fut	ER2	Generic 2nd 'ER' Future
	3M Euribor Fut	ER3	Generic 3rd 'ER' Future
	3M Euribor Fut	ER4	Generic 4th 'ER' Future
	3M Euribor Fut	ER5	Generic 5th 'ER' Future
FED	Fed Funds	FDTR	Federal Funds Target Rate
	90day Euro$ Fut	ED1	Generic 1st 'ED' Future
	90day Euro$ Fut	ED2	Generic 2nd 'ED' Future
	90day Euro$ Fut	ED3	Generic 3rd 'ED' Future
	90day Euro$ Fut	ED4	Generic 4th 'ED' Future
	90day Euro$ Fut	ED5	Generic 5th 'ED' Future
Fixed Rate Indices (Bberg/EFFAS)			
	All $>$ 1 Yr	USGATR	Bberg/EFFAS Bond Indices
	All $>$ 1 Yr	EUGATR	Bberg/EFFAS Bond Indices
	1-3 Yr	EUG1TR	Bberg/EFFAS Bond Indices
	3-5 Yr	EUG2TR	Bberg/EFFAS Bond Indices
	5-7 Yr	EUG3TR	Bberg/EFFAS Bond Indices

Continued on next page

Table A.1 – continued from previous page

Group	Subgroup	Ticker	Name
	7-10 Yr	EUG4TR	Bberg/EFFAS Bond Indices
	>10 Yr	EUG5TR	Bberg/EFFAS Bond Indices
Yields			
Europe	3M	GECU3M	Euro Generic Govt 3 Month Yield
	2Y	GECU2YR	Euro Generic Govt 2 Year Yield
	5Y	GECU5YR	Euro Generic Govt 5 Year Yield
	10Y	GECU10YR	Euro Generic Govt 10 Year Yield
	30Y	GECU30YR	Euro Generic Govt 30 Year Yield
US	3M	USGG3M	US Generic Govt 3 Month Yield
	2Y	USGG2YR	US Generic Govt 2 Year Yield
	5Y	USGG5YR	US Generic Govt 5 Year Yield
	10Y	USGG10YR	US Generic Govt 10 Year Yield
	30Y	USGG30YR	US Generic Govt 30 Year Yield
Japan	3M	GJTB3MO	Japan Generic Govt 3 Month Yield
	2Y	GJGB2	Japan Generic Govt 2 Year Yield
	5Y	GJGB5	Japan Generic Govt 5 Year Yield
	10Y	GJGB10	Japan Generic Govt 10 Year Yield
	30Y	GJGB30	Japan Generic Govt 30 Year Yield
UK	2Y	GUKG2	UK Generic Govt 2 Year Yield
	5Y	GUKG5	UK Generic Govt 5 Year Yield
	10Y	GUKG10	UK Generic Govt 10 Year Yield
	30Y	GUKG30	UK Generic Govt 30 Year Yield
Bond Futures			
Europe	2Y	DU1	Generic 1st 'DU' Future
	5Y	OE1	Generic 1st 'OE' Future
	10Y	RX1	Generic 1st 'RX' Future
	30Y	UB1	Generic 1st 'UB' Future
Swap Rates			
Europe	2Y	EUSA2	EUR Swap Annual 2 Yr
	5Y	EUSA5	EUR Swap Annual 5 Yr
	10Y	EUSA10	EUR Swap Annual 10 Yr
	30Y	EUSA30	EUR Swap Annual 30 Yr
US	2Y	USSW2	USD Swap Semi-Annual 30/360 2 Yr
	5Y	USSW5	USD Swap Semi-Annual 30/360 5 Yr
	10Y	USSW10	USD Swap Semi-Annual 30/360 10 Yr
	30Y	USSW30	USD Swap Semi-Annual 30/360 30 Yr
Equities			
		INDU	Dow Jones Industrial Average
		SPX	S&P 500 Index

Continued on next page

Table A.1 – continued from previous page

Group	Subgroup	Ticker	Name
		CCMP	NASDAQ Composite Index
		SX5E	Euro Stoxx 50
		SXXP	Stoxx Europe 600
		SX7E	Euro Stoxx Banks
		UKX	FTSE 100 Index
		TPX	TOPIX Index (Tokyo)
		NKY	NIKKEI 225
		HSI	HANG SENG Index
Currencies			
		EURUSD	EUR-USD X-Rate
		EURJPY	EUR-JPY X-Rate
		EURGBP	EUR-GBP X-Rate
		EURCHF	EUR-CHF X-Rate
Commodities			
		CRY	Core Commodity CRB Index
		CL1	Generic 1st 'CL' Future
		CO1	Generic 1st 'CO' Future
		GC1	Generic 1st 'GC' Future
		SI1	Generic 1st 'SI' Future
		HG1	Generic 1st 'HG' Future
		C 1	Generic 1st 'C' Future
		NG1	Generic 1st 'NG' Future
Volatility			
		MOVE	Merrill Lynch Option Volatility
		VIX	CBOE SPX Volatility Index
		V2X	VSTOXX Index
Economic Indicators			
US		GDP CQOQ	GDP US Chained 2009 Dollars
		CPI XYOY	US CPI Urban Consumers Less F&E
		NAPMPMI	ISM Manufacturing PMI
		IP CHNG	US Industrial Production MoM
		CPTICHNG	US Capacity Utilization %
		USURTOT	U-3 US Unemployment Rate Total
		NFP TCH	US Employees on Nonfarm Payrolls
		INJCJC	US Initial Jobless Claims SA
		LEI CHNG	Conference Board US Leading Index
Europe		EUGNEMUY	Euro Area Gross Domestic Product
		CPEXEMUY	Eurostat Eurozone Core MUICP
		EUIPEMUY	Eurostat Industrial Production
		EUUCEMU	Europ. Commission Capacity Utilization

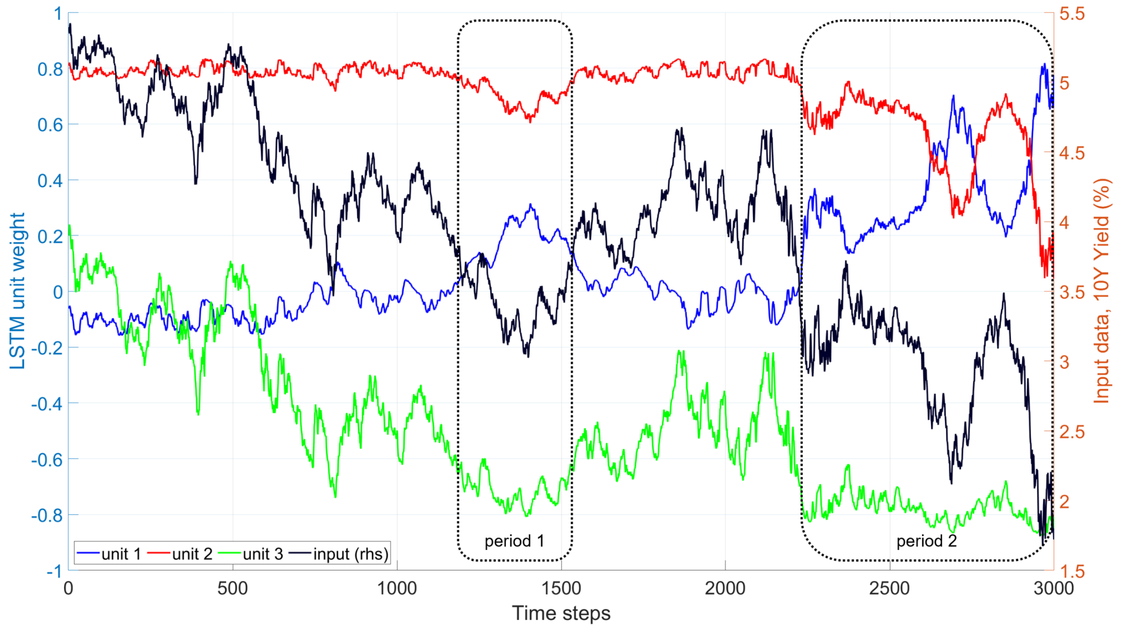
Continued on next page

Table A.1 – continued from previous page

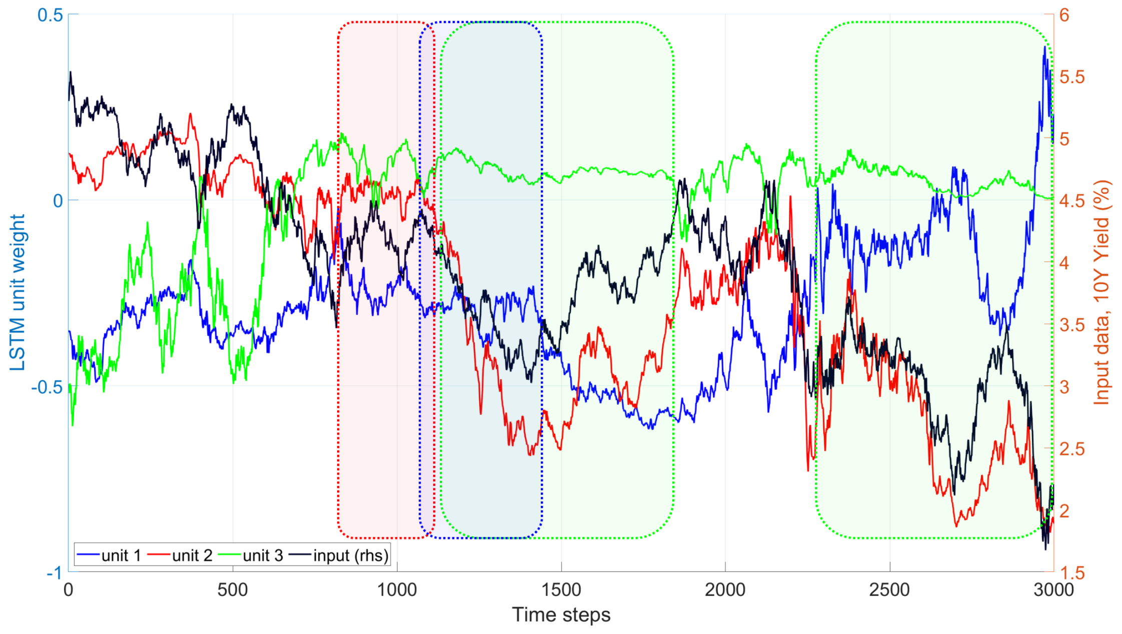
Group	Subgroup	Ticker	Name
		GRZEEUEX	ZEW Euroz. Expectation of Ec. Growth
		UMRTEMU	Eurostat Unemployment Eurozone
Global Indicators		OLE3US	OECD US Composite Leading Indicator
		OLE3EURA	OECD Euro Area Comp. Leading Indic.
		OLE3JAPA	OECD Japan Comp. Leading Indicator
		OLE3BRAZ	OECD Brazil Comp. Leading Indicator
		OLE3INDI	OECD India Comp. Leading Indicator
		OLE3CHIN	OECD China Comp. Leading Indicator
ECB Balance Sheet		EBBSTOTA	ECB Balance Sheet (BS) All Assets
		EBBSDEPF	ECB Balance Sheet Deposit Facility
		EBBSLONG	ECB BS LT Refinancing Operations
		EBBSA050	ECB BS Lending to Euro Area Credit Inst
ECB Money Supply		ECMSM1Y	ECB Money Supply M1 YoY
		ECMSM2Y	ECB Money Supply M2 YoY
		ECMSM3YY	ECB Money Supply M3 YoY
Government Bond Spreads			
Inter-markets	2Y	calculated	US-Europe Govt 2 Year Spread
		calculated	Europe-Japan Govt 2 Year Spread
		calculated	US-Japan Govt 2 Year Spread
		calculated	Europe-UK Govt 2 Year Spread
	10Y	calculated	US-Europe Govt 10 Year Spread
		calculated	Europe-Japan Govt 10 Year Spread
		calculated	US-Japan Govt 10 Year Spread
		calculated	Europe-UK Govt 10 Year Spread
Intra-market	03M10Y	calculated	Europe Govt 3 M-10 Year Spread
		calculated	US Govt 3 M-10 Year Spread
	2Y10Y	calculated	Europe Govt 2-10 Year Spread
		calculated	US Govt 2-10 Year Spread
	10Y30Y	calculated	Europe Govt 10-30 Year Spread
		calculated	US Govt 10-30 Year Spread
Technical Indicators			
Europe	Moving Average	calculated	Moving Average 5 days
2Y, 5Y,		calculated	Moving Average 10 days
10Y, 30Y		calculated	Moving Average 15 days
		calculated	Moving Average 20 days
		calculated	Moving Average 50 days
		calculated	Moving Average 200 days
		calculated	MA 10 days - MA 4 days
		calculated	MA 24 days - MA 14 days
		calculated	MA 48 days - MA 35 days

Appendix B

Chapter 4 - Additional Figures



(A) Feature set 2.



(B) Feature set 3.

FIGURE B.1: Hidden state signals for feature sets 2 and 3.

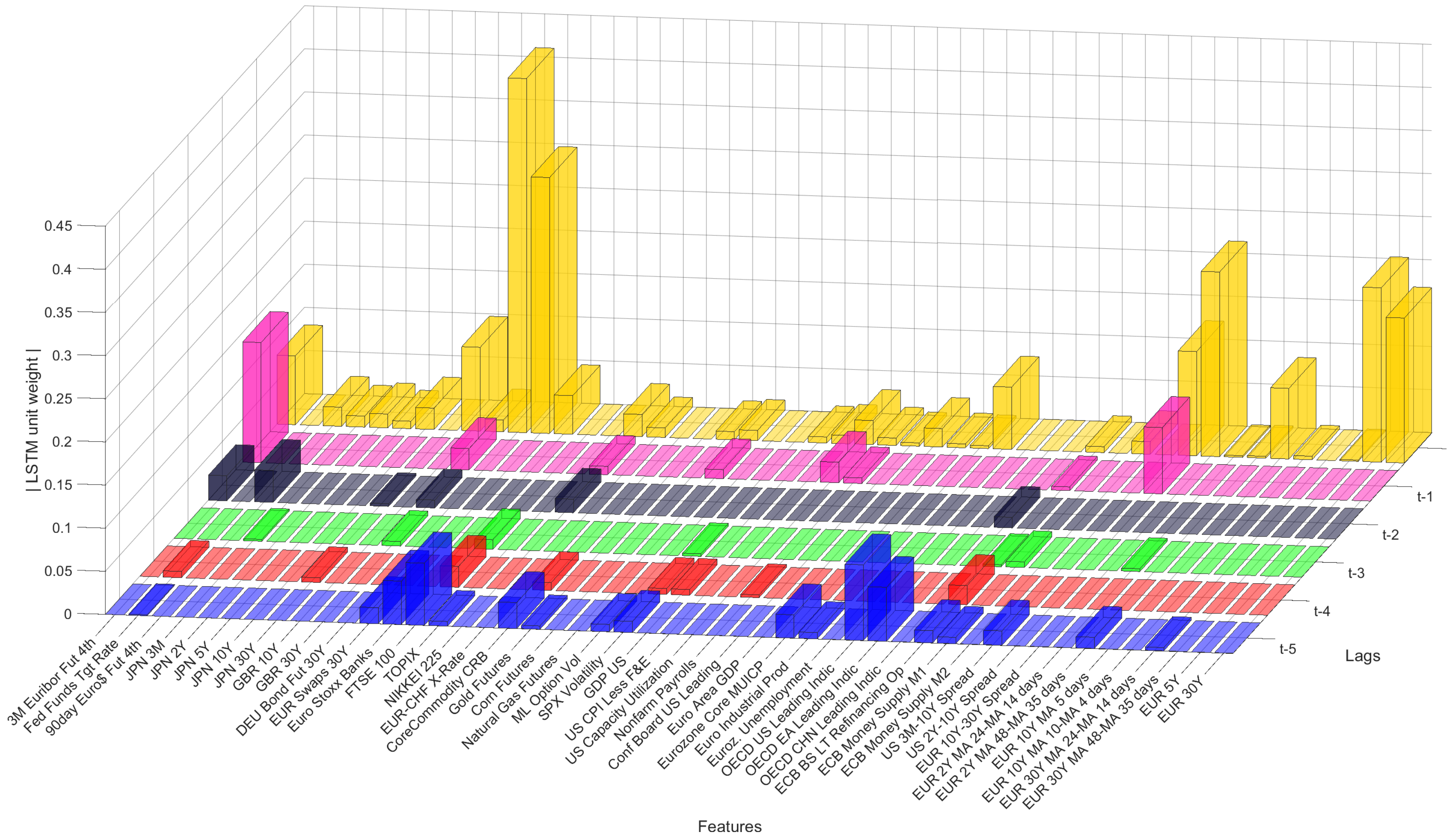


FIGURE B.2: *LSTM-LagLasso* relevant features for the hidden state, unit 3, considering a regularisation parameter γ equal to 1.0. [Terminology: An example of technical analysis indicator used is "EUR 30Y MA 48-MA 35 days" corresponding to the 48-day Moving Average (MA) minus 35-day MA, of the 30-year Euro government bond yield (EUR 30Y)].

Appendix C

Chapter 5 - Additional Information

TABLE C.1: List of hyperparameters used in the RL empirical work.

Hyperparameters	**Value**	**Description**
RL neural networks		
Learning rate	10^{-3}	Learning rate for optimisation.
L2 regularisation factor	10^{-4}	Factor for L2 regularisation (weight decay).
Optimiser	Adam	Adam optimiser.
Epsilon	10^{-8}	Denominator offset.
Gradient decay factor	0.9	Decay rate of gradient moving average.
Squared gradient decay factor	0.999	Decay rate of squared gradient moving average.
Gradient threshold	Inf	Threshold value for the gradient.
Gradient threshold method	L_2-norm	Gradient threshold method used to clip gradient values that exceed the gradient threshold.
RL agent		
Discount factor	0.99	Discount factor applied to future rewards during training.
Experience buffer length	10^6	Size of experience replay buffer.
Minibatch size	64	Size of random experience minibatch.
Ornstein-Uhlenbeck noise options		
Mean	0	Noise model mean.
Mean attraction constant	0.15	Constant that specifies how quickly the noise model output is attracted to the mean.
Standard deviation	0.3	Noise model standard deviation.
Target smooth factor	10^{-3}	Smoothing factor for target actor and critic updates.
Target update frequency	1	Number of steps between target actor and critic updates.

Continued on next page

Table C.1 – continued from previous page

Hyperparameters	**Value**	**Description**
Training		
Maximum episodes	10000	Maximum number of episodes used to train the agent.
Maximum steps per episode	5000	Maximum number of steps to run per episode.
Score averaging window length	10	Length of window for averaging the scores, rewards, and number of steps for each agent.
Testing		
Maximum steps	5000	Maximum number of steps to run the simulation.
Number of simulations	20	Number of simulations run.
Portfolio management		
Window size	6	Number of previous time steps considered.
Maximum trading frequency	daily	Maximum trading frequency enabled.
Transaction cost (bid-ask)	2 bp	Transaction cost considered (per unit of asset traded).

Bibliography

Masaya Abe and Hideki Nakayama. Deep learning for forecasting stock returns in the cross-section. In Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–284. Springer, 2018. ISBN 978-3-319-93034-3. https://doi.org/10.1007/978-3-319-93034-3_22.

Amine Mohamed Aboussalah, Ziyun Xu, and Chi-Guhn Lee. What is the value of the cross-sectional approach to deep reinforcement learning? *Quantitative Finance*, pages 1–21, 2021. https://doi.org/10.1080/14697688.2021.2001032.

J. G. Agrawal, V. S. Chourasia, and A. K. Mittra. State-of-the-art in stock prediction techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(4):1360–1366, 2013.

Saud Almahdi and Steve Y. Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017. https://doi.org/10.1016/j.eswa.2017.06.023.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. https://doi.org/10.1177/0278364919887447.

Oscar Araque, Ignacio Corcuera-Platas, J Fernando Sánchez-Rada, and Carlos A Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236–246, 2017. https://doi.org/10.1016/j.eswa.2017.02.002.

Imanol Arrieta-Ibarra and Ignacio N. Lobato. Testing for predictability in financial returns using statistical learning procedures. *Journal of Time Series Analysis*, 36(5):672–686, 2015.

Adam Atkins, Mahesan Niranjan, and Enrico Gerding. Financial news predicts stock market volatility better than close price. *The Journal of Finance and Data Science*, 4(2): 120–137, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations, ICLR*, pages 1–15, 2015. https://arxiv.org/abs/1409.0473.

Turan G. Bali, Amit Goyal, Dashan Huang, Fuwei Jiang, and Quan Wen. Predicting corporate bond returns: Merton meets machine learning. *Georgetown McDonough School of Business Research Paper No. 3686164, Swiss Finance Institute Research Paper No. 20-110*, 2022. http://dx.doi.org/10.2139/ssrn.3686164.

Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056, 2015. https://doi.org/10.1016/j.eswa.2015.05.013.

Devon K. Barrow and Sven F. Crone. Cross-validation aggregation for combining autoregressive neural network forecasts. *International Journal of Forecasting*, 32(4): 1120–1137, 2016. https://doi.org/10.1016/j.ijforecast.2015.12.011.

Bo Becker and Victoria Ivashina. Reaching for yield in the bond market. *The Journal of Finance*, 70(5):1863–1902, 2015.

Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. http://www.jstor.org/stable/24900506.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection. *arXiv preprint arXiv:1201.0224*, 2012. https://arxiv.org/abs/1201.0224.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *Proceedings of the International Conference on Neural Networks*, pages 1183–1188. IEEE, 1993.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Transactions on Neural Networks*, 5(2):157–166, 1994.

Adrian Benton, Margaret Mitchell, and Dirk Hovy. Multitask learning for mental health conditions with limited social media data. In *Proceedings of the European Chapter of the Association for Computational Linguistics, EACL*, volume 1, pages 152–162, 2017.

Carlos Betancourt and Wen-Hui Chen. Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, 164, 2021. https://doi.org/10.1016/j.eswa.2020.114002.

Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S. Sutton. Incremental natural actor-critic algorithms. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 20, pages 105–112, 2008. https://proceedings.neurips.cc/paper/2007/.

Christopher M. Bishop. *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag New York, 2006. ISBN 978-0-387-31073-2.

Tomas Björk and Bent Jesper Christensen. Interest rate dynamics and consistent forward rate curves. *Mathematical Finance*, 9(4):323–348, 1999.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

Bloomberg. Bloomberg professional database, 2017a. (Accessed on 28-Apr-2017).

Bloomberg. Bloomberg's 6 notable academic contributions in machine learning in 2016. *Tech at Bloomberg | Data Science*, 2017b. https://www.techatbloomberg.com/ (Accessed on 30-May-2017).

Bloomberg. Bloomberg professional database, 2021. (Accessed on 07-May-2021).

John L. G. Board, Charles M. S. Sutcliffe, and William T. Ziemba. Portfolio selection: Markowitz mean-variance model. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2990–2996. Springer, 2009. ISBN 978-0-387-74759-0. https://doi.org/10.1007/978-0-387-74759-0_513.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

Ash Booth. *Automated algorithmic trading: Machine learning and agent-based modelling in complex adaptive financial markets*. PhD thesis, University of Southampton, 2016.

Ash Booth, Enrico Gerding, and Frank McGroarty. Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, 41 (8):3651–3661, 2014a. https://doi.org/10.1016/j.eswa.2013.12.009.

Ash Booth, Enrico Gerding, and Frank McGroarty. Predicting equity market price impact with performance weighted ensembles of random forests. In *Proceedings of the Conference on Computational Intelligence for Financial Engineering & Economics, CIFEr*, pages 286–293. IEEE, 2014b.

Ash Booth, Enrico Gerding, and Frank McGroarty. Performance-weighted ensembles of random forests for predicting price impact. *Quantitative Finance*, 15(11):1823–1835, 2015.

Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

George E. P. Box and Gwilym M. Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968. ISSN 00359254, 14679876.

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time series analysis: Forecasting and control*. Wiley Series in Probability and Statistics. Wiley, fifth edition, 2015. ISBN 978-1-118-67492-5.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Jason Brownlee. *Long short-term memory networks with Python. Develop sequence prediction models with deep learning*. Machine Learning Mastery, Jason Brownlee, 2018.

Jason Brownlee. Random forest for time series forecasting. *Machine Learning Mastery*, 2020. https://machinelearningmastery.com/ (Accessed on 21-May-2021).

Seok-Jun Bu and Sung-Bae Cho. Learning optimal Q-function using deep Boltzmann machine for reliable trading of cryptocurrency. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 468–480. Springer, 2018. ISBN 978-3-030-03493-1. https://doi.org/10.1007/978-3-030-03493-1_49.

Katherine Burton. Inside a moneymaking machine like no other. *Bloomberg*, 21 November, 2016.

Hongyun Cai, Zi Huang, Xiaofeng Zhu, Qing Zhang, and Xuefei Li. Multi-output regression with tag correlation analysis for effective image tagging. In *Proceedings of the International Conference on Database Systems for Advanced Applications, DASFAA*, pages 31–46. Springer, 2014.

João Caldeira and Hudson Torrent. Forecasting the US term structure of interest rates using nonparametric functional data analysis. *Journal of Forecasting*, 36(1):56–73, 2017.

Richard A. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the International Conference on Machine Learning, ICML*, pages 41–48. PMLR, 1993.

Richard A. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Marco Castellani and Emanuel Augusto dos Santos. Forecasting long-term government bond yields: An application of statistical and AI models. *ISEG, Departamento de Economia*, pages 1–34, 2006.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018. https://doi.org/10.1111/ectj.12097.

Taufiq Choudhry, Frank McGroarty, Ke Peng, and Shiyun Wang. High-frequency exchange-rate prediction with an artificial neural network. *Intelligent Systems in Accounting, Finance and Management*, 19(3):170–178, 2012.

Carlo Ciliberto, Youssef Mroueh, Tomaso Poggio, and Lorenzo Rosasco. Convex learning of multiple tasks and their structure. In *Proceedings of the International Conference on Machine Learning, ICML*, volume 37, pages 1548–1557. PMLR, 2015. http://proceedings.mlr.press/v37/.

Eugenio Culurciello. The fall of RNN / LSTM. *Towards Data Science*, 2018. https://towardsdatascience.com/the-fall-of-rnn-lstm (Accessed on 05-Sep-2018).

Michael A. H. Dempster and Vasco Leemans. An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006. https://doi.org/10.1016/j.eswa.2005.10.012.

David Diaz, Babis Theodoulidis, and Carlos Dupouy. Modelling and forecasting interest rates during stages of the economic cycle: A knowledge-discovery approach. *Expert Systems with Applications*, 44:245–264, 2016. https://doi.org/10.1016/j.eswa.2015.09.010.

David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a): 427–431, 1979. https://doi.org/10.1080/01621459.1979.10482531.

Francis X. Diebold. Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of Diebold–Mariano tests. *Journal of Business & Economic Statistics*, 33(1):1–9, 2015. https://doi.org/10.1080/07350015.2014.983236.

Francis X. Diebold and Canlin Li. Forecasting the term structure of government bond yields. *Journal of Econometrics*, 130(2):337–364, 2006. https://doi.org/10.1016/j.jeconom.2005.03.005.

Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–265, 1995.

Francis X. Diebold and Glenn D. Rudebusch. *Yield curve modeling and forecasting: The dynamic Nelson-Siegel approach*. Econometric and Tinbergen Institutes lectures. Princeton University Press, 2013. ISBN 978-0-691-14680-5.

Francis X. Diebold, Glenn D. Rudebusch, and S. Boragan Aruoba. The macroeconomy and the yield curve: A dynamic latent factor approach. *Journal of Econometrics*, 131 (1):309–338, 2006. https://doi.org/10.1016/j.jeconom.2005.01.011.

Juan Peralta Donate, Paulo Cortez, Germán Gutiérrez Sánchez, and Araceli Sanchis de Miguel. Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble. *Neurocomputing*, 109:27–32, 2013. https://doi.org/10.1016/j.neucom.2012.02.053.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the International Conference on Machine Learning, ICML*, volume 48, pages 1329–1338. PMLR, 2016. http://proceedings.mlr.press/v48/.

Christian L. Dunis and Vincent Morrison. The economic value of advanced time series methods for modelling and trading 10-year government bonds. *European Journal of Finance*, 13(4):333–352, 2007.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

Bradley Efron and Robert J. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1993. ISBN 978-0-412-04231-7.

Dennis Eilers, Christian L. Dunis, Hans-Jörg von Mettenheim, and Michael H. Breitner. Intelligent trading of seasonal effects: A decision support algorithm based on reinforcement learning. *Decision Support Systems*, 64:100–108, 2014. https://doi.org/10.1016/j.dss.2014.04.011.

Jeffrey L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Walter Enders. *Applied Econometric Time Series*. Wiley Series in Probability and Statistics. Wiley, fourth edition, 2014. ISBN 978-1-118-91866-1.

Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007, 1982.

Frank J. Fabozzi, Steven V. Mann, and Francesco Fabozzi. *The handbook of fixed income securities*. McGraw-Hill, ninth edition, 2021. ISBN 978-1-260-47390-2.

Eugene F. Fama and Kenneth R. French. Business conditions and expected returns on stocks and bonds. *Journal of Financial Economics*, 25(1):23–49, 1989. https://doi.org/10.1016/0304-405X(89)90095-0.

Eugene F. Fama and Kenneth R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993. https://doi.org/10.1016/0304-405X(93)90023-5.

Eugene F. Fama and Kenneth R. French. A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22, 2015. https://doi.org/10.1016/j.jfineco.2014.10.010.

Helmut Farbmacher, Martin Huber, Lukáš Lafférs, Henrika Langen, and Martin Spindler. Causal mediation analysis with double machine learning. *The Econometrics Journal*, 25(2):277–300, 2022. https://doi.org/10.1093/ectj/utac003.

Guanhao Feng, Stefano Giglio, and Dacheng Xiu. Taming the factor zoo: A test of new factors. *The Journal of Finance*, 75(3):1327–1370, 2020. https://doi.org/10.1111/jofi.12883.

Filipa da S. Fernandes, Charalampos Stasinakis, and Zivile Zekaite. Forecasting government bond spreads with heuristic models: Evidence from the Eurozone periphery. *Annals of Operations Research*, 282(1):87–118, 2019. https://doi.org/10.1007/s10479-018-2808-0.

Angelos Filos. Reinforcement learning for portfolio management. Master's thesis, Imperial College London, 2019. https://arxiv.org/abs/1909.09571.

Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. https://doi.org/10.1016/j.ejor.2017.11.054.

Thomas G. Fischer. Reinforcement learning in financial markets - a survey. FAU Discussion Papers in Economics 12/2018, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2018.

Tristan Fletcher and John Shawe-Taylor. Multiple kernel learning with fisher kernels for high frequency currency prediction. *Computational Economics*, 42(2):217–240, 2013.

Tristan S. B. Fletcher. *Machine learning for financial market prediction*. PhD thesis, University College London, 2012.

Fabio D. Freitas, Alberto F. De Souza, and Ailson R. de Almeida. Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, 72(10-12):2155–2170, 2009. https://doi.org/10.1016/j.neucom.2008.08.019.

FRM. Medallion International Ltd. Investment analysis. *Technical Report, Financial Risk Management*, 2002.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018. https://arxiv.org/abs/1802.09477.

Felix A. Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN*, volume 3, pages 189–194. IEEE, 2000.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. In *Proceedings of the International Conference on Artificial Neural*

Networks, ICANN, volume 2, pages 850–855. Institution of Engineering and Technology, 1999.

Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying LSTM to time series predictable through time-window approaches. In Roberto Tagliaferri and Maria Marinaro, editors, *Proceedings of the Italian Workshop on Neural Nets, WIRN Vietri-01*, Perspectives in Neural Computing, pages 193–200. Springer, 2002. ISBN 978-1-4471-0219-9.

Joumana Ghosn and Yoshua Bengio. Multi-task learning for stock selection. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 9, pages 946–952. MIT Press, 1997. https://proceedings.neurips.cc/paper/1996/.

Stefano Giglio, Bryan T. Kelly, and Dacheng Xiu. Factor models, machine learning, and asset pricing. *Annual Review of Financial Economics*, 2021. http://dx.doi.org/10.2139/ssrn.3943284.

Grazia Gilardoni. Recurrent neural network models for financial distress prediction. Master's thesis, Politecnico di Milano, 2017.

C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, 44(1-2):161–183, 2001. https://doi.org/10.1023/A:1010884214864.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 15, pages 315–323. Proceedings of Machine Learning Research, 2011.

Periklis Gogas, Theophilos Papadimitriou, Maria Matthaiou, and Efthymia Chrysanthidou. Yield curve and recession forecasting in a machine learning framework. *Computational Economics*, 45(4):635–645, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 978-0-262-03561-3. https://www.deeplearningbook.org/.

Nikola Gradojevic and Jing Yang. Non-linear, non-parametric, non-fundamental exchange rate forecasting. *Journal of Forecasting*, 25(4):227–245, 2006.

Alex Graves. *Supervised sequence labelling with recurrent neural networks*. Springer, 2012. ISBN 978-3-642-24796-5.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610, 2005. https://doi.org/10.1016/j.neunet.2005.06.042.

Shihao Gu, Bryan T. Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020. https://doi.org/10.1093/rfs/hhaa009.

Yang Guan, Shengbo Eben Li, Jingliang Duan, Jie Li, Yangang Ren, Qi Sun, and Bo Cheng. Direct and indirect reinforcement learning. *International Journal of Intelligent Systems*, 36(8):4439–4467, 2021. https://doi.org/10.1002/int.22466.

Umut Güçlü and Marcel A. J. van Gerven. Modeling the dynamics of human brain activity with recurrent neural networks. *Frontiers in Computational Neuroscience*, 11: 1–14, 2017.

Cecile Gutscher, Yakob Peterseil, and Dani Burger. The new quant billions are hiding in the bond market. *Bloomberg*, 9 July, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. https://arxiv.org/abs/1801.01290.

Ben M. Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *arXiv preprint arXiv:2112.04553v1*, 2021. https://arxiv.org/abs/2112.04553.

James Douglas Hamilton. *Time series analysis*. Princeton University Press, 1994. ISBN 978-0-691-04289-3.

David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291, 1997. https://doi.org/10.1016/S0169-2070(96)00719-4.

Arman Hassanniakalager, Georgios Sermpinis, and Charalampos Stasinakis. Trading the foreign exchange market with technical analysis and Bayesian Statistics. *Journal of Empirical Finance*, 63:230–251, 2021. https://doi.org/10.1016/j.jempfin.2021.07.006.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer Series in Statistics, second edition, 2013.

Martin B Haugh and Andrew W. Lo. Computational challenges in portfolio management. *Computing in Science & Engineering*, 3(3):54–59, 2001. https://doi.org/10.1109/5992.919267.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen, 1991. Diploma thesis, Technische Universität München.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. https://doi.org/10.1162/neco.1997.9.8.1735.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In Stefan C. Kremer and John F. Kolen, editors, *A Field Guide to Dynamical Recurrent Networks*, pages 1–15. Wiley-IEEE Press, 2001. ISBN 978-0-7803-5369-5.

Robert J. Hodrick and Edward C. Prescott. Postwar US business cycles: An empirical investigation. *Journal of Money, Credit and Banking*, pages 1–16, 1997.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979v1*, 2020. https://arxiv.org/abs/2006.00979.

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Wei Huang, Kin Keung Lai, Yoshiteru Nakamori, Shouyang Wang, and Lean Yu. Neural networks in finance and economics forecasting. *International Journal of Information Technology & Decision Making*, 6(01):113–140, 2007.

James M. Hutchinson, Andrew W. Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning, ICML*, volume 37, pages 448–456. PMLR, 2015. http://proceedings.mlr.press/v37/.

Bruce I. Jacobs and Kenneth N. Levy. Factor modeling: The benefits of disentangling cross-sectionally for explaining stock returns. *The Journal of Portfolio Management*, 47 (6):33–50, 2021. https://doi.org/10.3905/jpm.2021.1.240.

Carlos M. Jarque and Anil K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics letters*, 6(3):255–259, 1980. https://doi.org/10.1016/0165-1765(80)90024-5.

Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *Proceedings of the Intelligent Systems Conference, IntelliSys*, pages 905–913. IEEE, 2017. https://doi.org/10.1109/IntelliSys.2017.8324237.

Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv: 1706.10059v2*, pages 1–31, 2017. https://arxiv.org/abs/1706.10059.

Michael I. Jordan. Serial order: A parallel distributed processing approach. *ICS Report 8604*, 1986.

Sham M. Kakade. A natural policy gradient. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 14. MIT Press, 2002. https://proceedings.neurips.cc/paper/2001/.

Mikhail Kanevski and Vadim Timonin. Machine learning analysis and modeling of interest rate curves. *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, pages 47–52, 2010.

Mikhail Kanevski, Michel Maignan, Alexei Pozdnoukhov, and Vadim Timonin. Interest rates mapping. *Physica A: Statistical Mechanics and its Applications*, 387(15):3897–3903, 2008. https://doi.org/10.1016/j.physa.2008.02.069.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015. https://karpathy.github.io/ (Accessed on 19-Jan-2018).

Bryan T. Kelly, Seth Pruitt, and Yinan Su. Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, 134(3):501–524, 2019. https://doi.org/10.1016/j.jfineco.2019.05.001.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations, ICLR*, pages 1–15, 2015. arXiv preprint arXiv:1412.6980v9.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, volume 2, pages 1137–1143. IJCAI Organization, 1995. ISBN 978-1-55860-363-9.

Thomas E. Koker and Dimitrios Koutmos. Cryptocurrency trading using machine learning. *Journal of Risk and Financial Management*, 13(8), 2020. https://doi.org/10.3390/jrfm13080178.

Marko Kolanovic and Rajesh T. Krishnamachari. Big data and AI strategies: Machine learning and alternative data approach to investing. *J.P. Morgan Quantitative and Derivatives Strategy Report*, 2017.

Mathias Kraus and Stefan Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104:38–48, 2017. https://doi.org/10.1016/j.dss.2017.10.001.

Roman Kräussl, Thorsten Lehnert, and Kalle Rinne. The search for yield: Implications to alternative investments. *Journal of Empirical Finance*, 44:227–236, 2017. https://doi.org/10.1016/j.jempfin.2017.11.001.

Yoram Kroll, Haim Levy, and Harry M. Markowitz. Mean-variance versus direct utility maximization. *The Journal of Finance*, 39(1):47–61, 1984. https://doi.org/10.2307/2327667.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. https://doi.org/10.1038/nature14539.

Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940v3*, pages 1–11, 2018. https://arxiv.org/abs/1808.09940.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971v6*, pages 1–14, 2015. Published as a conference paper at the International Conference on Learning Representations, ICLR-2016. https://arxiv.org/abs/1509.02971.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's thesis, University of Helsinki, 1970.

Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019v4*, pages 1–38, 2015.

David G. Luenberger. *Investment science*. Oxford University Press, 1998. ISBN 978-0-19-510809-5.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1412–1421. Association for Computational Linguistics, 2015. https://doi.org/10.18653/v1/D15-1166.

Nicolas Mahler. Modeling the S&P 500 index using the Kalman filter and the LagLasso. In *Proceedings of the International Workshop on Machine Learning for Signal Processing, MLSP*, pages 1–6. IEEE, 2009. https://doi.org/10.1109/MLSP.2009.5306195.

Moinak Maiti. Introduction to asset pricing factor models. In *Applied Financial Econometrics: Theory, Method and Applications*, pages 113–151. Springer, 2021. https://doi.org/10.1007/978-981-16-4063-6_5.

Nijolė Maknickienė and Algirdas Maknickas. Application of neural network for forecasting of exchange rates and forex trading. In *Proceedings of the International Scientific Conference Business and Management*, pages 122–127, 2012.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x.

MathWorks. Reinforcement learning onramp course. *MATLAB Academy – MathWorks*, 2021a.

MathWorks. *Reinforcement learning toolbox™. User's guide*. MathWorks, 2021b.

MathWorks. *Simulink®. User's guide*. MathWorks, 2021c.

Peter S. Maybeck. *Stochastic models, estimation, and control*. Mathematics in Science and Engineering. Academic Press, 1979. ISBN 978-0-12-480701-3.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Mariana Sampaio e Mello. Search-for-yield in Portuguese fixed-income mutual funds and monetary policy. Master's thesis, Nova School of Business and Economics, 2015.

Robert C. Merton. On the pricing of corporate debt: The risk structure of interest rates. *The Journal of finance*, 29(2):449–470, 1974. https://doi.org/10.1111/j.1540-6261.1974.tb03058.x.

Hans-Jörg von Mettenheim. *Advanced neural networks: Finance, forecast and other applications*. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2010.

Hans-Jörg von Mettenheim. Trading decision support with historically consistent neural networks. In Christian Dunis, Spiros Likothanassis, Andreas Karathanasopoulos, Georgios Sermpinis, and Konstantinos Theofilatos, editors, *Computational Intelligence Techniques for Trading and Investment*, number 6 in Routledge advances in experimental and computable economics, pages 116–130. Routledge, 2014. ISBN 978-0-415-63680-3.

Hans-Jörg von Mettenheim and Michael H. Breitner. Robust decision support systems with matrix forecasts and shared layer perceptrons for finance and other applications. In *Proceedings of the International Conference on Information Systems, Paper 83*, pages 1–17. ICIS, 2010.

Hans-Jörg von Mettenheim and Michael H. Breitner. Forecasting complex systems with shared layer perceptrons. In *Operations Research Proceedings*, pages 15–20. Springer, 2011.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602v1*, pages 1–9, 2013. https://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. https://doi.org/10.1038/nature14236.

Christoph Molnar. *Interpretable machine learning. A guide for making black box models explainable*. Leanpub, second edition, 2022. ISBN 979-8411463330. https://christophm.github.io/interpretable-ml-book/.

Luis Montesdeoca and Mahesan Niranjan. Extending the feature set of a data-driven artificial neural network model of pricing financial option. In *Symposium Series on Computational Intelligence, SSCI*, pages 1–6. IEEE, 2016.

Luis Montesdeoca and Mahesan Niranjan. On comparing the influences of exogenous information on bitcoin prices and stock index values. In Panos Pardalos, Ilias Kotsireas, Yike Guo, and William Knottenbelt, editors, *Mathematical Research for Blockchain Economy, MARBLE*, pages 93–100. Springer, 2020. https://doi.org/10.1007/978-3-030-37110-4_7.

John Moody and Matthew Saffell. Reinforcement learning for trading. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 11, pages 917–923. MIT Press, 1999. https://proceedings.neurips.cc/paper/1998/.

John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, 2001. https://doi.org/10.1109/72.935097.

John Moody and Lizhong Wu. Optimization of trading systems and portfolios. In *Proceedings of the Conference on Computational Intelligence for Financial Engineering & Economics, CIFEr*, pages 300–307. IEEE, 1997. https://doi.org/10.1109/CIFER.1997.618952.

John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998.

Joseph V. T. Morell. The decline in the predictive power of the US term spread: A structural interpretation. *Journal of Macroeconomics*, 55:314–331, 2018. https://doi.org/10.1016/j.jmacro.2017.12.003.

Lkhagvadorj Munkhdalai, Oyun-Erdene Namsrai, and Keun Ho Ryu. A hybrid approach based on long short-term memory networks and vector autoregression for stock market price prediction. *Proceedings of the International Conference on Frontiers of Information Technology, Applications and Tools, FITAT*, pages 1–4, 2017.

Charles R. Nelson and Andrew F. Siegel. Parsimonious modeling of yield curves. *Journal of Business*, pages 473–489, 1987.

Ralph Neuneier. Optimal asset allocation using adaptive dynamic programming. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 8, pages 952–958. MIT Press, 1996. https://proceedings.neurips.cc/paper/1995/.

Ralph Neuneier. Enhancing Q-learning for optimal asset allocation. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 10, pages 936–942. MIT Press, 1998. https://proceedings.neurips.cc/paper/1997/.

Mahesan Niranjan. Sequential tracking in pricing financial options using model based and neural network approaches. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 9, pages 960–966. MIT Press, 1996. https://proceedings.neurips.cc/paper/1996/.

Ana Rita Nogueira, Andrea Pugnana, Salvatore Ruggieri, Dino Pedreschi, and João Gama. Methods and tools for causal discovery and causal inference. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(2):e1449, 2022. https://doi.org/10.1002/widm.1449.

Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting. *Expert Systems with Applications*, 119:362–375, 2019a. https://doi.org/10.1016/j.eswa.2018.11.012.

Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. The memory advantage of long short-term memory networks for bond yield forecasting. *International Conference on Forecasting Financial Markets, FFM*, 2019b. https://doi.org/10.2139/ssrn.3415219.

Manuel Nunes, Enrico Gerding, Frank McGroarty, and Mahesan Niranjan. LSTM-LagLasso for bond yield forecasting: Peeping into the long short-term memory networks' black box. *Workshop Advancing Machine Learning in Finance, Insurance and Economics*, 2020. https://arxiv.org/abs/2005.02217.

OECD. *Business and finance outlook*. OECD Publishing, Paris, 2015a.

OECD. *Pension markets in focus*. OECD Publishing, Paris, 2015b.

Christopher Olah. Understanding LSTM networks, 2015. https://colah.github.io/ (Accessed on 19-Jan-2018).

Mariana Oliveira and Luis Torgo. Ensembles for time series forecasting. In *Proceedings of the Asian Conference on Machine Learning, ACML*, volume 39, pages 360–370. Proceedings of Machine Learning Research, 2014.

OpenAI. Getting started with Gym, 2022a. https://gym.openai.com/docs (Accessed on 07-Jan-2022).

OpenAI. Openai spinning up, 2022b. https://spinningup.openai.com (Accessed on 07-Jan-2022).

Hyungjun Park, Min Kyu Sim, and Dong Gu Choi. An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications*, 158, 2020. https://doi.org/10.1016/j.eswa.2020.113573.

Sebin Park, Myeong-Seon Gil, Hyeonseung Im, and Yang-Sae Moon. Measurement noise recommendation for efficient Kalman filtering over a large amount of sensor data. *Sensors*, 19(5):1–19, 2019. https://doi.org/10.3390/s19051168.

Parag C. Pendharkar and Patrick Cusatis. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1–13, 2018. https://doi.org/10.1016/j.eswa.2018.02.032.

Luca Di Persio and Oleksandr Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, 10:403–413, 2016a.

Luca Di Persio and Oleksandr Honchar. Artificial neural networks approach to the forecast of stock market price movements. *International Journal of Economics and Management Systems*, 1:158–162, 2016b.

Luca Di Persio and Oleksandr Honchar. Recurrent neural networks approach to the financial forecast of google assets. *International Journal of Mathematics and Computers in Simulation*, 11:7–13, 2017.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *European Conference on Machine Learning, ECML*, pages 280–291. Springer, 2005. https://doi.org/10.1007/11564096_29.

Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K. Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J. Bessa, Jakub Bijak, John E. Boylan, et al. Forecasting: Theory and practice. *International Journal of Forecasting*, 2022. https://doi.org/10.1016/j.ijforecast.2021.11.001.

Richard R. Picard and R. Dennis Cook. Cross-validation of regression models. *Journal of the American Statistical Association*, 79(387):575–583, 1984.

Uta Pigorsch and Sebastian Schäfer. High-dimensional stock portfolio trading with deep reinforcement learning. *arXiv preprint arXiv:2112.04755v1*, 2021. https://arxiv.org/abs/2112.04755.

David L. Poole and Alan K. Mackworth. *Artificial intelligence: Foundations of computational agents*. Cambridge University Press, second edition, 2017. https://artint.info/.

Michiel de Pooter. Examining the Nelson-Siegel class of term structure models. *Tinbergen Institute Discussion Paper*, TI 2007-043/4, 2007.

Adam Prügel-Bennett. Advanced machine learning, 2017. University of Southampton, School of Electronics and Computer Science.

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, pages 2627–2633. IJCAI Organization, 2017. https://doi.org/10.24963/ijcai.2017/366.

Soumya Ray and Prasad Tadepalli. Model-based reinforcement learning. In *Encyclopedia of Machine Learning*, pages 690–693. Springer, 2010. ISBN 978-0-387-30164-8. https://doi.org/10.1007/978-0-387-30164-8_556.

Gordon Ritter. Machine learning for trading. *SSRN Electronic Journal*, 2017. https://doi.org/10.2139/ssrn.3015609.

Peter M. Robinson. Root-N-consistent semiparametric regression. *Econometrica*, 56(4): 931–954, 1988. https://doi.org/10.2307/1912705.

Pamela Roux and Katherine Burton. This hedge fund may be poised to create the most billionaires. *Bloomberg*, 25 April, 2017.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098v1*, pages 1–14, 2017.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362. MIT Press, 1987.

Gavin A. Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166, Department of Engineering, University of Cambridge*, 1994.

Rajiv Sambasivan and Sourish Das. A statistical machine learning approach to yield curve forecasting. In *Proceedings of the International Conference on Computational Intelligence in Data Science, ICCIDS*, pages 1–6. IEEE, 2017.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017a. https://arxiv.org/abs/1704.06440.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347v2*, 2017b. https://arxiv.org/abs/1707.06347.

Georgios Sermpinis, Konstantinos Theofilatos, Andreas Karathanasopoulos, Efstratios F. Georgopoulos, and Christian Dunis. Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *European Journal of Operational Research*, 225(3):528–540, 2013. http://dx.doi.org/10.1016/j.ejor.2012.10.020.

Georgios Sermpinis, Andreas Karathanasopoulos, Rafael Rosillo, and David de la Fuente. Neural networks in financial trading. *Annals of Operations Research*, 297(1): 293–308, 2019. https://doi.org/10.1007/s10479-019-03144-y.

Martin Sewell. Characterization of financial time series. *Research Note RN/11/01, Department of Computer Science, University College London*, 2011.

Jun Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):486–494, 1993.

Lloyd S. Shapley. A value for n-person games. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, number 28 in Annals of Mathematics Studies, pages 307–317. Princeton University Press, 1953. https://doi.org/10.1515/9781400881970-018.

William F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442, 1964. https://doi.org/10.1111/j.1540-6261.1964.tb02865.x.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning, ICML*, volume 32, pages 387–395. PMLR, 2014. http://proceedings.mlr.press/v32/.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. https://doi.org/10.1038/nature16961.

Derek Snow. Machine learning in asset management: Part 1: Portfolio construction – trading strategies. *The Journal of Financial Data Science*, 2(1):10–23, 2020a. https://doi.org/10.3905/jfds.2019.1.021.

Derek Snow. Machine learning in asset management: Part 2: Portfolio construction – weight optimization. *The Journal of Financial Data Science*, 2(2):17–24, 2020b. https://doi.org/10.3905/jfds.2020.1.029.

Petar Sorić and Ivana Lolić. A note on forecasting euro area inflation: Leave-h-out cross validation combination as an alternative to model selection. *Central European Journal of Operations Research*, 23(1):205–214, 2015.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

Ilya Sutskever. *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, second edition, 2020. ISBN 978-0-262-03924-6. http://www.incompleteideas.net/book/.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 12, pages 1057–1063. MIT Press, 2000. https://proceedings.neurips.cc/paper/1999/.

Akiko Takeda, Mahesan Niranjan, Jun-ya Gotoh, and Yoshinobu Kawahara. Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. *Computational Management Science*, 10(1):21–49, 2013.

Chi-Sang Tam and Ip-Wing Yu. Modelling sovereign bond yield curves of the US, Japan and Germany. *International Journal of Finance & Economics*, 13(1):82–91, 2008.

Ran Tao, Chi-Wei Su, Yidong Xiao, Ke Dai, and Fahad Khalid. Robo advisors, algorithmic trading and investment management: Wonders of fourth industrial revolution in financial markets. *Technological Forecasting and Social Change*, 163:120421, 2021. https://doi.org/10.1016/j.techfore.2020.120421.

Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173, 2021. https://doi.org/10.1016/j.eswa.2021.114632.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

George E. Uhlenbeck and Leonard S. Ornstein. On the theory of the brownian motion. *Physical Review*, 36(5):823–841, 1930. https://doi.org/10.1103/PhysRev.36.823.

Wali Ullah, Yasumasa Matsuda, and Yoshihiko Tsukuda. Generalized Nelson-Siegel term structure model: Do the second slope and curvature factors improve the in-sample fit and out-of-sample forecasts? *Journal of Applied Statistics*, 42(4):876–904, 2015.

Kerda Varaku. *Essays on causal inference and treatment effects in productivity and finance: Double robust machine learning with deep neural networks and random forests*. PhD thesis, Rice University, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems, NeurIPS*, volume 30, pages 5998–6008, 2017. https://proceedings.neurips.cc/paper/2017/.

Chang Sim Vui, Gan Kim Soon, Chin Kim On, Rayner Alfred, and Patricia Anthony. A review of stock market prediction with artificial neural network (ANN). In *Proceedings of the International Conference on Control System, Computing and Engineering, ICCSCE*, pages 477–482. IEEE, 2013.

Eric A. Wan. Modeling nonlinear dynamics with neural networks: Examples in time series prediction. In *Proceedings of the International Society for Optics and Photonics, SPIE*, pages 327–327. Citeseer, 1993.

Helmut Wasserbacher and Martin Spindler. Machine learning for financial forecasting, planning and analysis: recent developments and pitfalls. *Digital Finance*, 4(1):63–88, 2022. https://doi.org/10.1007/s42521-021-00046-2.

Christopher J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992. https://doi.org/10.1007/BF00992698.

Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Robin Wigglesworth and Laurence Fletcher. A robot named Gekko: 'quant' funds home in on the bond market. *Financial Times*, 12 June, 2019.

Robin Wigglesworth and Laurence Fletcher. The next quant revolution: shaking up the corporate bond market. *Financial Times*, 7 December, 2021.

Peter M. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural computation*, 7(1):117–143, 1995.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Ruoxuan Xiong, Eric P. Nichols, and Yuan Shen. Deep learning stock volatility with google domestic trends. *arXiv preprint arXiv:1512.04916v3*, pages 1–6, 2016. https://arxiv.org/abs/1512.04916.

Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *Workshop on Challenges and Opportunities for AI in Financial Services, Conference on Neural Information Processing Systems, NeurIPS*, 2018. https://arxiv.org/abs/1811.07522.

Jia-Ching Ying, Yu-Bing Wang, Chih-Kai Chang, Ching-Wen Chang, Yu-Han Chen, and Yow-Shin Liou. DeepBonds: A deep learning approach to predicting United States treasury yield. In *International Conference on Ubi-Media Computing, UMEDIA*, pages 245–250. IEEE, 2019. https://doi.org/10.1109/Ubi-Media.2019.00055.

Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*, pages 1–21, 2019. https://arxiv.org/abs/1901.08740.

Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. A universal end-to-end approach to portfolio optimization via deep learning. *arXiv preprint arXiv:2111.09170*, 2021. https://arxiv.org/abs/2111.09170.

Yishen Zhang, Dong Wang, Yuehui Chen, Huijie Shang, and Qi Tian. Credit risk assessment based on long short-term memory model. In *Proceedings of the International Conference on Intelligent Computing, ICIC*, pages 700–712. Springer, 2017.

Yuanyuan Zhang, Xiang Li, and Sini Guo. Portfolio selection problems with Markowitz's mean–variance framework: A review of literature. *Fuzzy Optimization and Decision Making*, 17(2):125–158, 2018. https://doi.org/10.1007/s10700-017-9266-z.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep learning for portfolio optimization. *The Journal of Financial Data Science*, 2(4):8–20, 2020. https://doi.org/10.3905/jfds.2020.1.042.

Yang Zhao, Charalampos Stasinakis, Georgios Sermpinis, and Filipa da S. Fernandes. Revisiting Fama–French factors' predictability with Bayesian modelling and copula-based portfolio optimization. *International Journal of Finance & Economics*, 24(4):1443–1463, 2019. https://doi.org/10.1002/ijfe.1742.