Online learning in financial time series

Gabriel Borrageiro

Thesis submitted for the degree of

Doctor of Philosophy

Department of Computer Science

University College London

supervisors: Dr. Paolo Barucca, Dr. Nick Firoozye

November 22, 2022

I, Gabriel Borrageiro, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

We wish to understand if additional learning forms can be combined with sequential optimisation to provide superior benefit over batch learning in various tasks operating in financial time series.

In chapter 4, **Online learning with radial basis function networks**, we provide multi-horizon forecasts on the returns of financial time series. Our sequentially optimised radial basis function network (RBFNet) outperforms a random-walk baseline and several powerful supervised learners. Our RBFNets naturally measure the similarity between test samples and prototypes that capture the characteristics of the feature space.

In chapter 5, **Reinforcement learning for systematic FX trading**, we perform feature representation transfer from an RBFNet to a direct, recurrent reinforcement learning (DRL) agent. Earlier academic work saw mixed results. We use better features, second-order optimisation methods and adapt our model parameters sequentially. As a result, our DRL agents cope better with statistical changes to the data distribution, achieving higher risk-adjusted returns than a funding and a momentum baseline.

In chapter 6, **The recurrent reinforcement learning crypto agent**, we construct a digital assets trading agent that performs feature space representation transfer from an echo state network to a DRL agent. The agent learns to trade the XBTUSD perpetual swap contract on BitMEX. Our meta-model can process data as a stream and learn sequentially; this helps it cope with the nonstationary environment.

In chapter 7, **Sequential asset ranking in nonstationary time series**, we create an online learning long/short portfolio selection algorithm that can detect the best and worst performing portfolio constituents that change over time; in particular, we successfully handle the higher transaction costs associated with using daily-sampled data, and achieve higher total and risk-adjusted returns than the long-only holding of the S&P 500 index with hindsight.

Impact Statement

This thesis adds to the academic research of learning in nonstationary environments or those that experience frequent concept drifts. Chapter 4, **Online learning with radial basis function networks** contributes to the research of feature representation transfer and kernel-based learning methods. Our sequentially optimised, transfer-learning radial basis function networks (RBFNets) outperform various powerful baseline models in multi-horizon, multi-asset forecasting and help the financial industry reduce predictive uncertainty. Furthermore, it is well-known that financial time series exhibit low signal-to-noise ratios. When one uses RBFNets whose hidden units are determined by clustering algorithms or mixture models, the hidden unit outputs retain more remarkable similarity through time, which increases signal-to-noise ratios and ameliorates catastrophic forgetting.

Chapters 5, **Reinforcement learning for systematic FX trading** and 6, **The recurrent reinforcement learning crypto agent** contribute to the academic research of direct reinforcement learning in financial trading systems. Specifically, the models learn to target financial risk positions sequentially and directly without using value function estimation. When regimes change or the funding costs alter to reflect new interest rate differentials, the models transition smoothly to the desired risk positions, accruing lower transaction costs than supervised learning methods.

Finally, chapter 7, **Sequential asset ranking in nonstationary time series**, contributes to the research on prediction with expert advice, with a practical application of learning to rank and select portfolios of assets. Despite the index appreciating intensely during the test period, we outperform the long-only holding of the S&P 500 constituents with hindsight. The financial industry benefits from this research as we can identify subsets of assets to trade on a long/short basis through economic downturns, black swan events and bullish markets. We do so while drastically reducing the high transaction costs that arise with so-called regress-then-rank algorithms.

Acknowledgements

I want to thank my supervisors, Paolo Barucca and Nick Firoozye, for agreeing to supervise me and investing their time and energy into this endeavour. Completing a part-time PhD as a family man is a challenging effort. I thank my wife Marion and my daughters Juno and Thea for their patience, especially over most weekends when I was working on my thesis out of sight. Finally, I would like to dedicate this thesis to the memory of my father, Manuel.

Publications

Chapters 4 through 7 of this thesis were initially submitted as individual manuscripts that have been peer-reviewed and published.

- G. Borrageiro, N. Firoozye and P. Barucca, **Online learning with radial basis function networks**, to appear in The Journal of Financial Data Science, February 2023.
- G. Borrageiro, N. Firoozye and P. Barucca, **Reinforcement learning for systematic FX trading**, in IEEE Access, vol. 10, pp. 5024-5036, 2022, doi: 10.1109/ACCESS.2021.3139510.
- G. Borrageiro, N. Firoozye and P. Barucca, **The recurrent reinforcement learning crypto agent**, in IEEE Access, vol. 10, pp. 38590-38599, 2022, doi: 10.1109/ACCESS.2022.3166599.
- G. Borrageiro, N. Firoozye and P. Barucca, **Sequential asset ranking in nonstationary time series**, in 3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA. ACM, doi: 10.1145/3533271.3561666.

Contents

List of Figures

List of Tables

List of Algorithms

Notation

Numbers and Arrays

a	a real or integer scalar
$\mathbf{a}$	a vector
$\mathbf{A}$	a matrix
$\mathbf{I}_d$	an identity matrix with d rows and d columns
$\mathbf{0}_d$	a zero vector of length d
$\mathbf{0}_{d\times d}$	a zero matrix with d rows and d columns
$diag(a)$	a square, diagonal matrix with a along the diagonal

Sets and Graphs

$\mathbb{A}$	a set
$\mathbb{R}$	the set of real numbers
$\mathbb{Z}^+$	the set of positive integers $\{0,1,2,3,...\}$
$\{0,1,...,n\}$	the set of all integers between 0 and n

Indexing

$\mathbf{a}_i$	element i of vector $\mathbf{a}$
$\mathbf{a}_{-i}$	all elements of $\mathbf{a}$ except for element i
$\mathbf{A}_{i,j}$	element i, j of matrix $\mathbf{A}$
$\mathbf{A}_{:,i}$	column i of matrix $\mathbf{A}$
$\{\boldsymbol{\mu}_j\}_{j=1}^k$	an operation on vector $\boldsymbol{\mu}_j, \quad j = 1,...,k$

Calculus

$\frac{dy}{dx}$	the derivative of y with respect to x
$\frac{\partial y}{\partial x}$	the partial derivative of y with respect to x
$\nabla_x y$	the gradient of y with respect to x
$\nabla^2_{\mathbf{x}} f(\mathbf{x})$	the Hessian matrix of f evaluated at $\mathbf{x}$

Probability and Statistics

$x \sim P$	a P distributed random variable x
$\mathbb{E}_{x\sim P}[f(x)]$	the expectation of $f(x)$ with respect to $P(x)$
$Var[x]$	the variance of x
$Cov[x,y]$	the covariance of x and y
$\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$	Gaussian distribution over $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
R^2	the proportion of the variation in the response that is explained by the predictors
mse	mean square error
iid	independent and identically distributed random variables

Functions

$f(\mathbf{x};\boldsymbol{\theta})$	a function of $\mathbf{x}$ parameterised by $\boldsymbol{\theta}$
$\|\mathbf{x}\|$	L^2 norm of $\mathbf{x}$
$I(.)$	an indicator function that returns 1 for a true condition, or else 0
$sign(x)$	$sign(x) = 1$ if $x > 0$, $\quad 0$ if $x = 0$, $\quad -1$ if $x < 0$
$argSort(.)$	returns the indices that would sort an array from largest to smallest value

Chapter 1

Introduction

This thesis is concerned with online learning in financial time series. In this chapter, we provide context for our study, provide a statement of the problem, discuss our aim and scope, detail our contributions to science and complete the chapter with an outline of the thesis structure. Figure 1.1 succinctly describes our thesis setup. Our thesis is motivated by the difficulty in modelling financial time series, which are typically nonstationary and serially correlated. Even if one uses returns, which are technically stationary as determined by unit root tests, they typically experience regime shifts. Thus, the traditional batch-learning OLS approach has obvious limitations. On various tasks, including forecasting, proprietary risk-taking and portfolio selection, we show that intelligent feature transfer combined with online learning will likely outperform historically difficult-to-beat baselines and powerful batch learners.

1.1 Context of the study

Time-series econometrics is concerned with estimating difference equations containing stochastic components (Enders, 2014). The classical approach in this domain is the Box and Jenkins (1970) ARIMA technique, which broadly involves differencing a time series until it is stationary and then applying a structural, offline learning model, which is valued for its ease of interpretation and hypothesis testing. Financial time series are usually nonstationary and serially correlated. Taking first differences renders such time series stationary, and learning a model of the returns generally leads to regression residuals that show no recognisable structure. The stationary, differenced time series facilitate using a broad class of batch learning models. However, differencing these time series usually removes long-memory properties in the data. de Prado (2015) argues that financial time series exhibit low

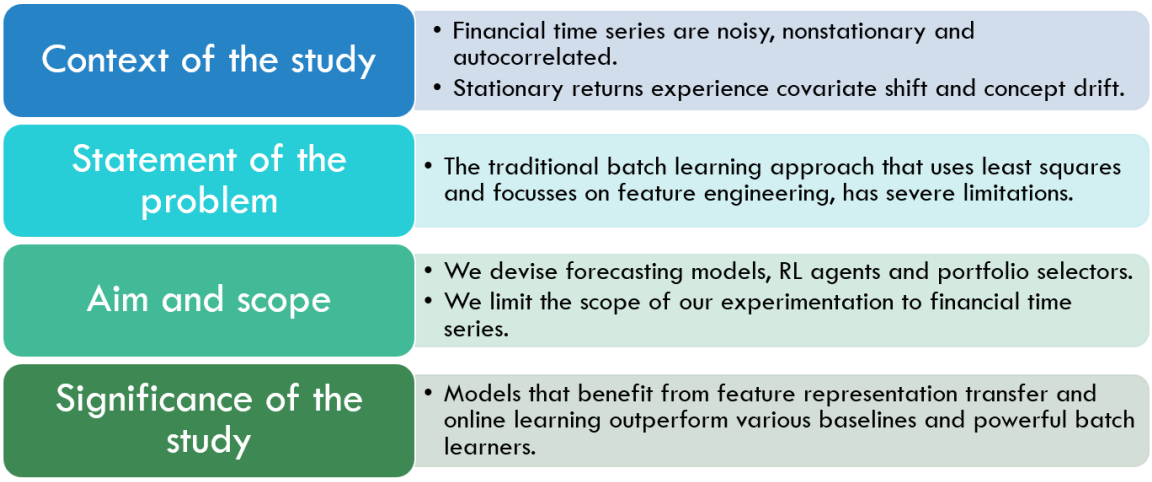


Figure 1.1: Our thesis is motivated by the difficulty in modelling financial time series, which are typically nonstationary and serially correlated. Even if one uses returns, which are technically stationary as determined by unit root tests, they typically experience regime shifts. Thus, the traditional batch-learning OLS approach has obvious limitations. On various tasks, including forecasting, proprietary risk-taking and portfolio selection, we show that intelligent feature transfer combined with online learning will likely outperform historically difficult-to-beat baselines and powerful batch learners.

signal-to-noise ratios due to arbitrage forces, which integer differencing reduces further. The emphasis of his work in de Prado (2018) is to apply a wide range of modern machine learning techniques to model financial time series and to move the emphasis away from applying structural models to integer differenced data.

An approach that copes with nonstationarity is online learning, broadly defined as sequential fitting and prediction. Historically, we find some use of online learning in financial time series. Examples include discounted least squares (Abraham and Ledolter, 1983), which uses a mini-batch sliding-window fitting approach, and recursive least squares (RLS) (Harvey, 1993), which adopts sequential fitting and prediction. More recently, we find exponentially weighted versions of RLS (EWRLS) which use nonlinear kernels (Liu et al., 2010). The use of such models is emphasised in the context of sequential forecasting. However, no claims are made that such models are strictly necessary or work better than offline learning models. It is often viewed that online learning is far noisier than offline learning. For example, Polyak-Ruppert averaging (Ruppert, 1988; Polyak, 1990) is applied to the model weight updates learnt through stochastic gradient descent, reducing the impact of noise on the problem solution and improving convergence rates. More recently, in econometric time

series forecasting, views have shifted toward the hypothesis that online learning is essential, specifically as the online learning approaches can adapt better to dynamic statistical behaviours. For example, Anava et al. (2013) use regret minimisation techniques (Cesa-Bianchi and Lugosi, 2006) to develop practical online learning algorithms for the prediction problem without assuming that the noise terms are iid. Furthermore, they show that their algorithms' performances asymptotically approach the performance of the best arma model with hindsight.

1.2 Statement of the problem

At the core of this thesis, we wish to understand if other forms of learning can be combined with sequential optimisation to provide superior benefit over batch learning in various tasks operating in financial time series. Such tasks include multi-horizon forecasting, proprietary trading and portfolio selection. The traditional approach in this domain is to initially perform feature engineering on stationary returns, apply a batch, supervised learning technique such as regression, and use the model for a while before retraining it. Some thought is given to identifying a training data period that reflects current market conditions. However, due to a great deal of predictive uncertainty in financial time series, market behaviour often adapts a priori. The batch learning models that operate on them are often too slow to react to or cannot identify the behavioural change.

Whilst we expect sequential optimisation to provide a benefit, we anticipate that it will need to be combined with other forms of learning to provide consistent outperformance over offline learning techniques. Offline learning can operate advantageously with a forward-looking bias on the entire batch of training data. For example, if we wish to model $y_{t+h} = f(\mathbf{x}_t, \boldsymbol{\theta}) + \varepsilon_t$, that is, we wish to forecast a target y_{t+h} that occurs h steps ahead in time using inputs $\mathbf{x}_t$, we can learn this mapping using parameters $\boldsymbol{\theta}$ as all future target values are at our disposal. Online learning approaches cannot immediately learn the asymptotic behavioural mapping of current inputs to future targets. We must wait for future target values and even longer for parameter convergence, with several rounds of model fitting. However, online learning can adapt learning in response to dynamic changes in the data's statistical distribution.

1.3 Aim and scope

We aim to combine other forms of learning, such as transfer learning, reinforcement learning or prediction with expert advice, with powerful sequential optimisation techniques to outperform batch learning on tasks such as multi-horizon forecasting, proprietary trading and portfolio selection. In the absence of changes to the statistical distribution of the modelled data, we aim to achieve a similar level of performance to the batch learning models. However, when regime changes or concept drifts occur, we aim for our online learning meta-models to outperform the batch learning ones. Finally, we limit the scope of our modelling to financial time series, which often exhibit small changes over time and occasional large jumps.

1.4 Significance of the study

Collectively our experiments on financial time series provide evidence that combining additional forms of learning, such as transfer learning or reinforcement learning with sequential optimisation, outperforms batch learning models of high learning capacity on various tasks. These various tasks include multi-horizon returns forecasting or risk-adjusted returns maximisation. In addition, all four of our experiments use some form of feature representation transfer from clustering algorithms, mixture models or echo state networks. This feature representation transfer provides upstream models with a somewhat prescient feature representation that learns the salient properties of the input/predictor space.

In section 2.1, we show that even if we use the stationary returns of the S&P 500 dataset (section 3.4), the batch learning models that operate on them have parameters that need to be constantly adapted. We use statistical tests such as the t-test for equal means to show that the further the distance in time between the model parameters being compared, the greater the probability is that these tests will reject the hypothesis that the parameters were generated from the same statistical distribution.

Section 4.4 demonstrates that our extensive use of feature representation transfer from clustering algorithms to the various upstream models is fully justified. For the Refinitiv cross-asset class dataset (section 3.1), we show that the training set returns have low similarity with the test set returns. In contrast, the training set cluster-derived hidden unit outputs

retain greater similarity with their test set counterparts.

In chapter 4, **Online learning with radial basis function networks**, we experiment with the returns of financial time series, providing multi-horizon forecasts with a selection of robust supervised learners.

1. We devise an external input selection algorithm that aims to maximise regression R^2 whilst minimising feature correlations and can operate efficiently in a high-dimensional setting.

2. We improve upon the earlier work on radial basis function networks (RBFNets), which applies feature representation transfer from clustering algorithms to supervised learners.

   (a) In the initial training phase, rather than using a randomised, scalar standard deviation for each hidden unit's radial basis function, we use a covariance matrix estimated via a Bayesian maximum a posteriori approach. If many training data points are assigned to the $j'th$ cluster, the $j'th$ covariance matrix will resemble the maximum likelihood estimate; otherwise, it will resemble the diagonalised variance prior.

   (b) In the online learning phase, our radial basis functions use sequentially optimised precision (inverse covariance) matrices. This approach leads to an improved test time fitting time-complexity of $\mathcal{O}(kd^2)$ from $\mathcal{O}(kd^3)$, where k is the number of hidden units, and d is the external input dimensionality. These parameters are sequentially updated with an exponential decay to facilitate regime changes or concept drifts.

3. Our online RBFNet outperforms a random-walk baseline, a historically tricky challenge, and several powerful batch learners. The outperformance is not purely down to sequential updating in the test set; a competitor exponentially weighted recursive least squares (EWRLS) model is updated similarly and performs less well than several batch-learners.

4. We demonstrate that our local-learning RBFNets are naturally designed to measure the similarity between test samples and continuously updated prototypes that capture the

characteristics of the feature space. The models are robust in mitigating catastrophic forgetting in a way that the global learning feed-forward multilayer perceptrons cannot.

In chapter 5, **Reinforcement learning for systematic FX trading**, we explore online inductive transfer learning, with a feature representation transfer from a RBFNet formed of Gaussian mixture model (GMM) hidden units to a direct, recurrent reinforcement learning (DRL) agent. This agent is put to work in an experiment, trading the major spot market currency pairs, where we accurately account for transaction and funding costs.

1. Earlier academic work that applies direct reinforcement to currency pair trading saw mixed results. These mixed performances result from weak external inputs to the DRL agent, first-order optimisation methods (stochastic gradient ascent), and shared hyperparameters such as sliding window sizes for mini-batch fitting. In contrast, we obtain good experiment results. We use more powerful features such as RBFNets formed by Gaussian mixtures. We use second-order optimisation methods such as extended Kalman filters. We adapt our model parameters sequentially rather than use mini-batch learning train/test sliding window splits. As a result, our DRL agents cope better with statistical changes to the data distribution.

2. A further point of differentiation is that we use a quadratic utility rather than the Sharpe ratio. In earlier work, it was already identified that the Sharpe ratio penalises returns that are larger than $\frac{\mathbb{E}[r^2]}{\mathbb{E}[r]}$, that is, the ratio of expected square returns to expected returns. This result is counter-intuitive as rational investors are happy to accept the volatility of right-tail returns and less happy to accept the volatility of left-tail returns.

3. Our experiments are more rigorous than the earlier work on DRL agents, which rarely use baselines or do not use them. Instead, we adopt two baselines: a funding/carry trader and a momentum trader. The momentum trader is a supervised learning RBFNet that maps the hidden units to the response (one-step ahead returns) using an EWRLS model. Finally, our transfer learning DRL agent achieves better risk-adjusted returns than either baseline.

4. We demonstrate how the DRL agent maintains an essential advantage over the supervised learning momentum trader. The DRL agent's learning can be adapted in response

to impacts on the utility function it seeks to optimise. For example, we compare the realised positions of a USDRUB trader where in the former case, transaction costs and carry are removed (figure 5.8a) and in the latter case, transaction costs and carry are included (figure 5.8b). Without cost, the DRL agent realises a long position broadly (buying USD and selling RUB), as the Ruble depreciates over time. The momentum trader behaves similarly. However, when funding costs are accurately included, the DRL agent learns a short position (selling USD and buying RUB), capturing this positive carry. The momentum trader cannot learn from this additional impact on profit and loss.

In chapter 6, **The recurrent reinforcement learning crypto agent**, we demonstrate a novel application of online transfer learning for a digital assets trading agent. This agent uses a powerful feature space representation in the form of an echo state network (ESN), the output of which is made available to a DRL agent. The agent learns to trade the XBTUSD (Bitcoin versus US Dollars) perpetual swap derivatives contract on BitMEX on an intraday basis.

1. We extend our earlier work into DRL agents that utilise feature representation transfer, except this time, replace the GMM RBFNet for an ESN.

2. Our meta-model can process data as a stream and learn sequentially; this helps it cope with the nonstationarity of the high-frequency order book and trading data.

3. Furthermore, by using the vast high-frequency data, our model, which has a high learning capacity, avoids the kind of overfitting on a lack of data points that occurs with down-sampled data.

4. We escape the over-trading typically seen with supervised learning models by learning the sensitivity of the change in risk-adjusted returns to the model's parameterisation. Stated another way, our model learns from the multiple impact sources on profit and loss and targets the appropriate risk position.

5. Our crypto agent realises a total return of 350%, net of transaction costs, over roughly five years, 71% of which is down to funding profit. The annualised information ratio

that it achieves is 1.46.

6. The scientific experiment that we conduct is representative of the conditions observed in live trading; thus, we are confident that the resulting performance can realistically be transferred to industry use.

In chapter 7, **Sequential asset ranking in nonstationary time series**, we experiment with the S&P 500 constituents, extending the academic research into cross-sectional momentum trading strategies.

1. The main result of this experiment is our ranking algorithm, the naive Bayes asset ranker (NBAR), which we use to select subsets of assets to trade from the S&P 500 index in either a long-only or a long/short (cross-sectional momentum) capacity. Our ranking algorithm computes the sequential posterior probability that individual assets will be ranked higher than other constituents in the portfolio.

2. Our NBAR ranks the predictions of an online learning multivariate regression model, the curds and whey (CAW) model with EWRLS updating. The CAW model benefits from feature representation transfer from RBFNets formed by sequentially optimised k-means clusters, as more signal is extracted from the noisy external input daily returns.

3. Unlike earlier algorithms such as the weighted majority algorithm (Littlestone and Warmuth, 1994), which deals with nonstationarity by ensuring the weights assigned to each expert never dip below a minimum threshold, our ranking algorithm allows experts who performed poorly previously to have increased weight when they start performing well. Our algorithm computes the posterior ranking probabilities with exponential decay and is better suited to learning in nonstationary environments.

4. Comparing the NBAR to Freund et al. (1997)'s Bayes algorithm, the latter also gives the opportunity for hitherto poor performing experts to have increased weight when they start performing well; yet, there is no memory per se with respect to time. The Bayes algorithm incurs a weight increase or decay relative to the weighted majority.

Thus if the weighted majority is noisy, frequent trading will occur and higher transaction costs are expected relative to the NBAR.

5. Our algorithm achieves higher risk-adjusted and total returns than a strategy that would hold the long-only S&P 500 index with hindsight, despite the index appreciating by 205% during the test period. It also outperforms a regress-then-rank baseline, a sequentially fitted curds and whey multivariate regression model.

1.5 Summary of contributions

We summarise our contributions to science, highlighting what is both novel and beneficial.

- We demonstrate that combining intelligent feature representation transfer with online learning is likely to outperform historically difficult-to-beat baselines and powerful batch learners, on a range of tasks operating on financial time series.

- We create a procedure that demonstrates that models operating on technically stationary financial time series returns require regular fitting, in large part due to regime shifts, highlighting the rationale for online learning.

- We show that the training set financial time series returns have low similarity with their test set counterparts, highlighting the challenges faced in particular by kernel-based methods that use the training set returns as test-time prototypes; in contrast, our online learning RBFNets have hidden units that retain greater similarity across time.

- We design bespoke quadratic utility functions for direct reinforcement learning purposes that capture all impacts to PNL, including price discovery, funding and execution cost.

- We create an online learning long/short portfolio selection algorithm that can detect the best and worst performing portfolio constituents that change over time; in particular, we successfully handle the higher transaction costs associated with using daily-sampled data, and achieve higher total and risk-adjusted returns than the long-only holding of the S&P 500 index with hindsight.

1.6 Overview

We complete this chapter with an outline of the structure of this thesis. Chapter 2 contains background information central to our modelling and performs a literature review. We establish a rationale for online learning in economic time series and then discuss several topics, including transfer learning, neural networks, multivariate regression techniques and reinforcement learning. The chapter ends with a discussion of the prediction with expert advice framework and its impact on online portfolio selection.

Chapter 3 provides details of the datasets we use in our various experiments. Chapters 4 to 7 contain our four thesis experiments, where each chapter contains a primary model that combines some form of additional learning with sequential optimisation. In chapter 4, we perform feature representation transfer from clustering algorithms to supervised learners whose goal is to provide multi-horizon forecasts in cross-asset financial time series.

In chapter 5, we perform feature representation transfer from Gaussian mixture models to DRL agents who systematically learn to trade the major currency pairs. Chapter 6 extends this research further, performing feature representation transfer from echo state networks to DRL agents who learn to trade cryptocurrency futures. Ultimately, these DRL agents aim to learn from the multiple sources of impact on profit and loss in a way that supervised learning cannot and to provide superior risk-adjusted returns.

Our final experiment in chapter 7 takes place in the context of the prediction with expert advice framework, where our novel ranking algorithm learns to select subsets of constituents of the S&P 500 index. Our algorithm aims to outperform the long-only holding of the index with hindsight. Finally, concluding remarks are made in chapter 8, including a summary of our contributions and directions for future research.

Finally, figure 1.2 describes how all four of our experiments relate. Online learning, combined with some form of feature representation transfer, underpins all four experiments that we conduct. In addition, chapter 4 sees an additional exercise of external input selection via an algorithm that combines forward stepwise selection with variance inflation factor minimisation.

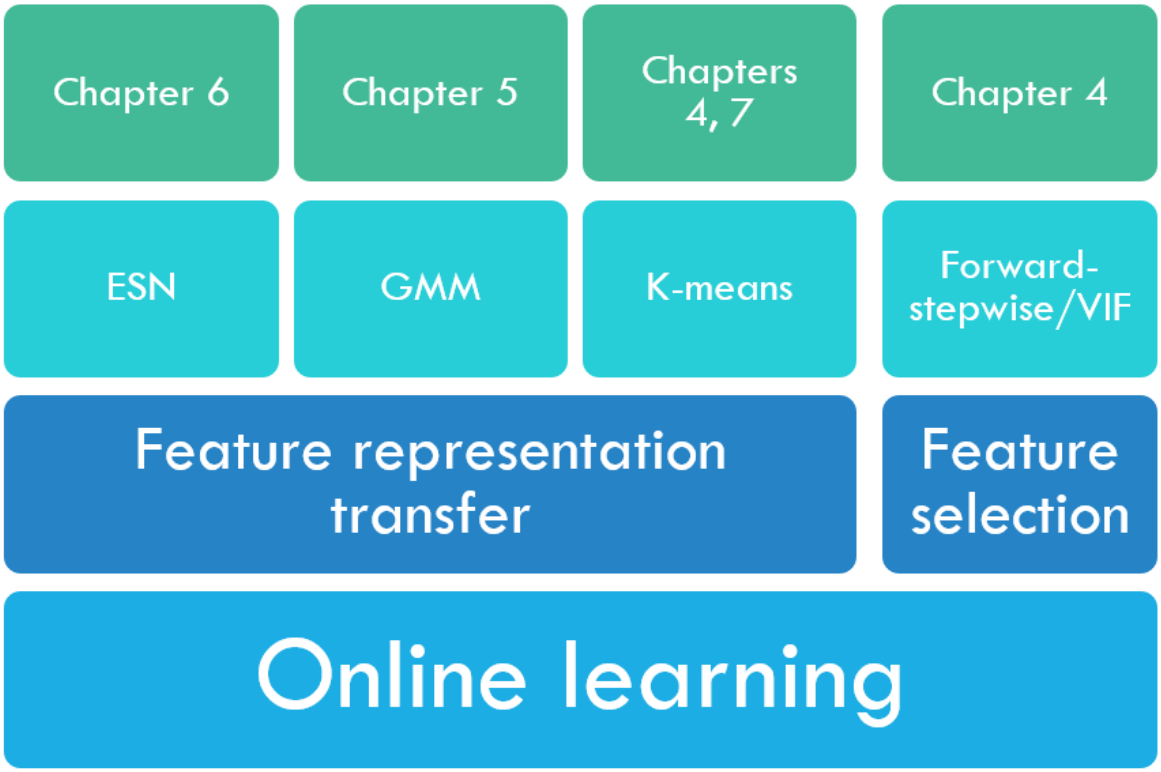


Figure 1.2: Online learning, combined with some form of feature representation transfer, underpins all four experiments that we conduct. In addition, chapter 4 sees an additional exercise of external input selection via an algorithm that combines forward stepwise selection with variance inflation factor minimisation.

Chapter 2

Background and literature review

This chapter introduces the various models and topics that are the foundations of our four experiments. The major issues discussed are feature representation transfer, multivariate regression, reservoir computing, direct, recurrent reinforcement learning, prediction with expert advice and ranking algorithms. We begin by discussing the rationale for online learning in financial time series.

2.1 The rationale for online learning in financial time series

Financial time series exhibit the characteristics of high serial correlation and nonstationarity. The impact of high serial correlation on time series where forecasts are made using regression-based models is that the R^2 of model fit will be spuriously high (Granger and Newbold, 1974). The ramification of nonstationarity can be more severe in that models fitted offline on training data generalise poorly on hitherto unseen test data. Merton (1976) models the dynamics of financial assets, specifically option prices, as a jump-diffusion process, which implies that economic time series should observe small changes over time, so-called continuous changes, and occasional jumps. The model superimposes a diffusion component modelled by geometric Brownian motion, with a jump component driven by a Poisson process. The goal is to model abrupt changes in prices due to the arrival of new information, facilitating the increased likelihood of tail events compared to the normal distribution. Financial time series have also been modelled as a random-walk process (Markowitz and Cootner, 1965). The random-walk process implies the unpredictability of economic time series returns and their time-varying random nature, which Nakamura and Small (2007) find exists in equities, currencies, precious metals and energy returns.

One approach for coping with nonstationarity is to learn online continuously. One may

combine sequential optimisation with states-of-nature/transitional learning approaches such as reinforcement learning or continual learning approaches such as transfer learning. Sequential learning in time is an intuitive concept and has a rich history. The Kalman filter (Kalman, 1960) is a state-space model originally designed for tracking objects in time from noisy measurements. Several approaches exist for sequential learning in nonstationary data; these include discounted least squares (Abraham and Ledolter, 1983) and exponentially weighted recursive least squares (EWRLS) (Liu et al., 2010). Barber et al. (2011) consider Bayesian approaches to time series modelling, which are amenable to sequential learning. Reinforcement learning allows agents to interact with their environment, mapping situations to actions to maximise numerical rewards. Well-known sequentially optimised reinforcement learning algorithms include the so-called temporal-difference learners, such as q-learning (Watkins, 1989) and sarsa (Rummery and Niranjan, 1994). The classical k-armed bandit problem is formulated as an online learning problem, where one is faced with a choice amongst k possible options. After each selection, a reward is assigned. The interplay between exploitation and exploration leads to several algorithms such as ε-greedy, stochastic gradient ascent, and upper confidence bound bandits (Sutton and Barto, 2018). Continual learning is an area of study that asks how artificial systems might learn sequentially, as biological systems do, from a continuous stream of correlated data (Hadsell et al., 2020).They include gradient-based methods (Kaplanis et al., 2018; Kirkpatrick et al., 2017), meta-learning (Wang et al., 2017) and transfer learning (Yang et al., 2020a).

Kroner and Sultan (1993) find that currencies have time-varying distributions that impact the hedging of spot exposures using futures. Denote the optimal hedge ratio as

$$\gamma_t^* = \frac{Cov[s_t, f_t]}{Var[f_t]}, \tag{2.1}$$

where s_t, f_t denote the spot and futures currency returns at time t. Wang et al. (2015) extend the research into optimal hedge ratios, experimenting with a range of time-invariant and time-varying hedge ratio models. They find that these models struggle to outperform the naive hedging strategy that sets $\gamma_t^* = 1$. They attribute this to both covariance estimation error and model misspecification. They conclude that no particular model specification can flexibly capture the dynamic nature of spot and futures returns over time. An equivalent

issue when predicting asset returns is that it is difficult to beat a random-walk baseline, as we discuss in section 4.2.1. However, in chapter 4, we show that feature representation transfer can boost model learning capacity and, when combined with sequential optimisation, can outperform biased baselines such as the random walk.

Nonstationarity in time series can be identified through unit root tests such as the augmented Dickey-Fuller (ADF) test (Said and Dickey, 1984). In section 7, we experiment with the last 21 years of daily sampled S&P 500 data, whose constituents are summarised in table 3.3. When running the ADF test against the S&P 500 daily transaction prices, 90.5% of the time series are considered nonstationary and when running the test against their linear returns of the form

$$y_{j,t} = p_{j,t}/p_{j,t-1} - 1, \tag{2.2}$$

where j denotes the constituent and t denotes the time-step, 100% of the time series are now considered stationary. One could now assume that performing any modelling on the daily returns is safe, and nonstationarity will no longer present a problem. A simple experiment calls this view into question, as whilst the returns might be stationary, the models that operate on such time series require parameters that must be adapted over time. We fit AR(1) models with structural form

$$y_{j,t} = w_{j,t} y_{j,t-1} + \varepsilon_{j,t},$$

to each constituent's daily returns. The models are trained daily using the past $n = 252$ days on a rolling window basis. Using one-week non-overlapping time buckets, we average the AR(1) coefficients and denote them as $\overline{w}_{j,t}$. Finally, we conduct two-sample t-tests for equal means (Hirotsu, 2017) where we compare $\overline{w}_{j,t}$ with $\overline{w}_{j,t+h}$ for $h = 1, ..., 52$, setting the significance level of the test, $\alpha = 0.05$. This t-test assumes that the compared variables are normally distributed[1], $w_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$. We find that as the shift h increases, the probability

[1]The t-test for equal means with unequal variances is a minimum variance unbiased estimator of $\mu_1 - \mu_2$ with variance $\sigma_1^2/n_1 + \sigma_2^2/n_2$, where n_j denotes the number of samples for the $j'th$ variable. The test statistic is intuitively

$$t = (\bar{\mu}_1 - \bar{\mu}_2)/\sqrt{\hat{\sigma}_1^2/n_1 + \hat{\sigma}_2^2/n_2}.$$

The normality assumption of the AR(1) coefficients is verified here by conducting an omnibus test of normality that combines skew and kurtosis considerations (D'Agostino, 1971; D'Agostino and Pearson, 1973). We find that the normality test accepts the normality of the AR(1) coefficients 92% of the time at the 5% significance level.

that the null hypothesis $H_0 : \overline{w}_{j,t} = \overline{w}_{j,t+h}$ is rejected increases considerably. Figure 2.1 averages the results across the S&P 500 constituents and time shifts; it demonstrates that the AR(1) parameters are changing over time, despite the training data being almost the same for small shifts h. Specifically, the fraction of matched training days between coefficients $\overline{w}_{j,t}$ and $\overline{w}_{j,t+h}$ is $max\left(0, \frac{n-7h}{n}\right)$. This experiment provides evidence supporting the use of online learning when modelling financial time series returns.

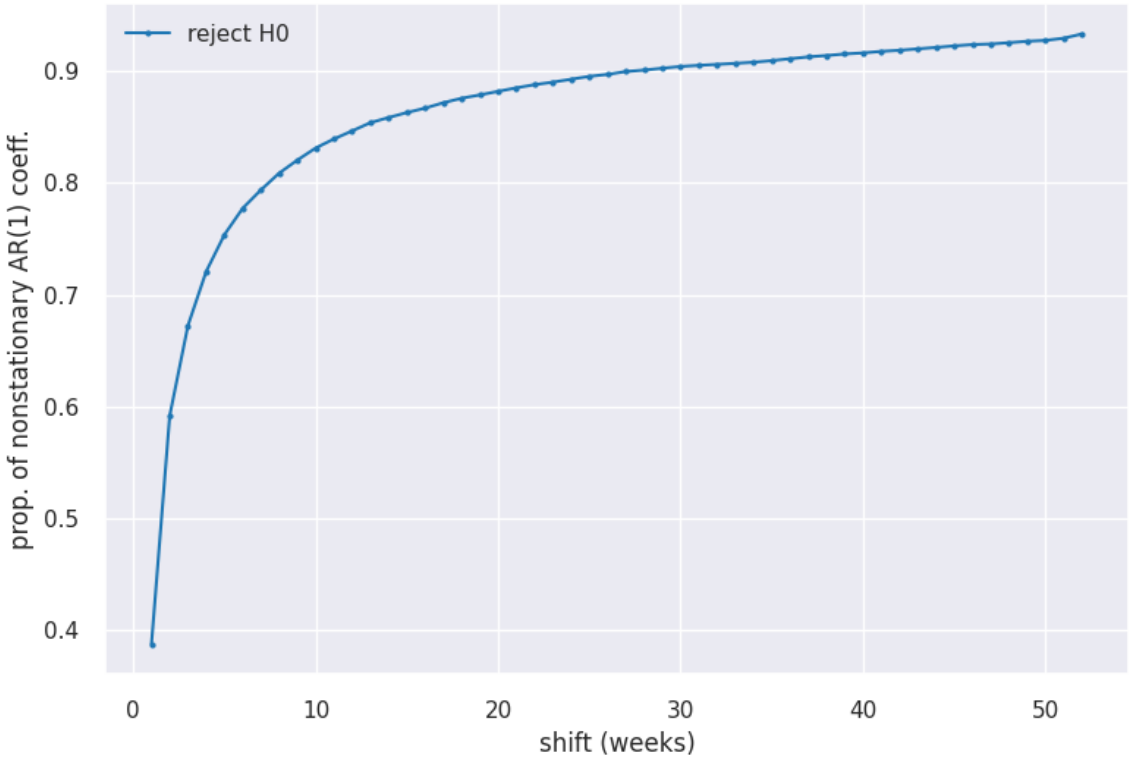


Figure 2.1: For the S&P 500 dataset, we plot the probability that $H_0 : \overline{w}_{j,t} = \overline{w}_{j,t+h}$ is rejected by the two-sample t-test for equal means. Specifically, the weekly averaged AR(1) coefficients are compared. Unit root tests indicate that the daily returns are stationary. However, the models that operate on the returns require constant refitting; this motivates the need for online learning.

Use of equation 2.2 typically leads to data that is differenced by more than is required to reject the null hypothesis of a unit root in the ADF test. Furthermore, over-differencing can turn a long memory process into a white noise one. de Prado (2018) fractionally differences financial time series by increasing order of magnitude until the ADF test indicates stationarity. A sliding window of weights generates the fractionally differenced time series at prediction time. However, time series generated in this way can appear chaotic, espe-

cially for a fractional difference parameter $d \to 0$, taking a few thousand observations to settle down. Thus, we avoid using this technique data in our experiments, especially with the limited availability of daily-sampled data.

2.2 Transfer learning

Transfer learning refers to the machine learning paradigm in which an algorithm extracts knowledge from one or more application scenarios to help boost the learning performance in a target scenario (Yang et al., 2020a). Typically, traditional machine learning requires significant amounts of training data. Transfer learning copes better with data sparsity by looking at related learning domains where data is sufficient. Even with large datasets, including streaming data, transfer learning provides benefits by learning the adaptive statistical relationship of the predictors and the response. Following Pan and Yang (2010):

> **Definition 1 (transfer learning)** *Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.*

Sub-paradigms of transfer learning include inductive, where labelled data is available in the target domain and transductive, where labelled data is available only in the source domain. Feature-based approaches transform the original features to create a new representation and traverse all transfer learning sub-paradigms. An increasing number of papers focus on online transfer learning (Zhao et al., 2014; Salvalaio and de Oliveira Ramos, 2019; Wang et al., 2020).

2.3 The radial basis function network

The RBFNet is typically a single-layer network whose hidden units are radial basis functions (RBF) of the form

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}[\mathbf{x}-\boldsymbol{\mu}_j]^T\boldsymbol{\Sigma}_j^{-1}[\mathbf{x}-\boldsymbol{\mu}_j]\right). \tag{2.3}$$

The hidden unit means and covariances are typically learnt through clustering algorithms such as k-means (Lloyd, 1982). The hidden unit outputs are aggregated into a feature vector

$$\boldsymbol{\phi}_t = [1, \phi_1(\mathbf{x}), ..., \phi_k(\mathbf{x})],$$

and mapped to the response via regression

$$y_t = \boldsymbol{\theta}^T \boldsymbol{\phi}_t + \varepsilon.$$

A schematic of the model is shown in figure 2.2. Figure 2.3 compares several scikit-learn (Pedregosa et al., 2011) classifiers on synthetic datasets. A multilayer perceptron separates classes using hidden units that form hyperplanes in the input space. The separation of class distributions modelled by local radial basis functions is probabilistic. The predictive uncertainty increases where there is class-conditional distribution overlap. The RBF activations can be thought of as the posterior feature probabilities, and the weights can be interpreted as the posterior probabilities of class membership, given the presence of the features (Bishop, 1995).

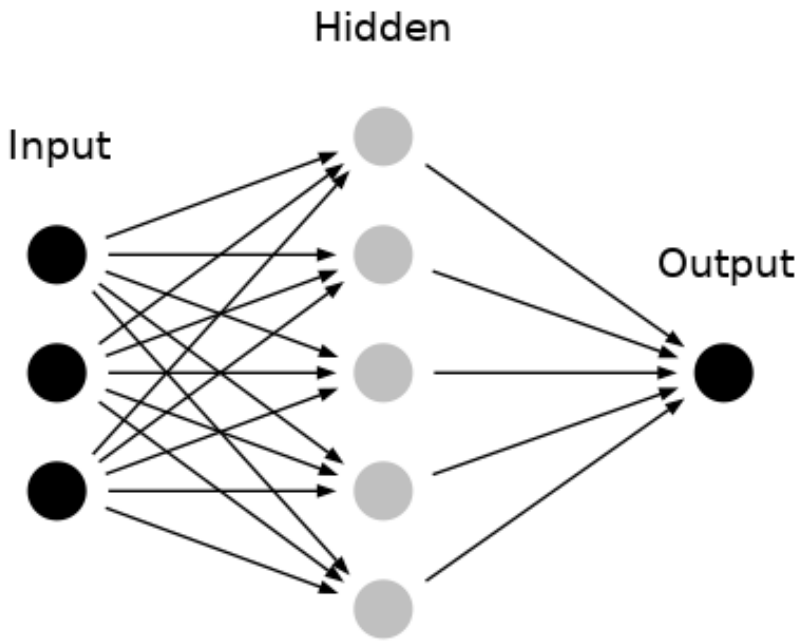


Figure 2.2: Architecture of the radial basis function network.

Rbfnets have a 'best approximation' property (Girosi and Poggio, 1990). In the set of approximating functions corresponding to all possible choices of parameters, the RBF has a minimum approximating error. Seminal papers on RBFNets include those by Broomhead and Lowe (1988) and Moody and Darken (1989). It is first with Moody and Darken (1989)

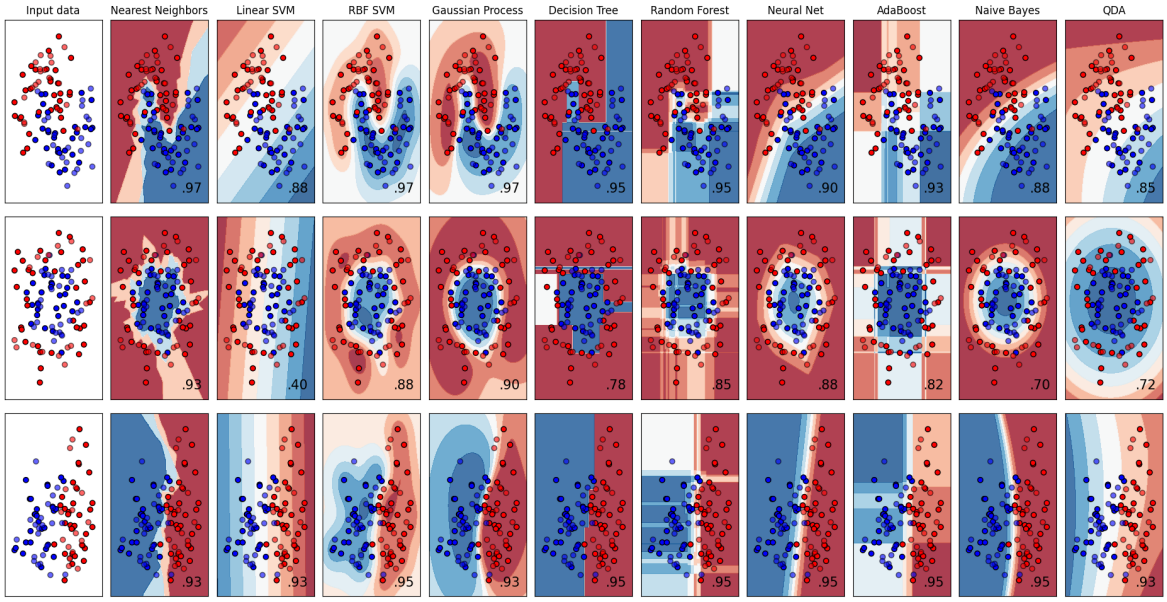


Figure 2.3: Here, we compare the posterior predictive densities of several scikit-learn classifiers on synthetic datasets. A multilayer perceptron separates classes using hidden units that form hyperplanes in the input space. In contrast, the separation of class distributions modelled by local radial basis functions is probabilistic, with each class fitted with a kernel function.

that we see the formulation of the RBFNet as a batch-learning combination of k-means and linear regression. A related, although slower method, is considered by Billings et al. (1996), where they initialise a large RBFNet and rely on orthogonal least squares (Markovsky and Van Huffel, 2007) and forward stepwise selection (Derksen and Keselman, 1992) to select the hidden units. They directly compare the narx model (Chen and Billings, 1989) and the RBFNet, demonstrating an application to multiple-input multiple-output modelling (Bontempi, 2008) in simulated dynamic time series. Kanazawa (2020) applies an RBFNet to macroeconomic forecasting, outperforming benchmarks such as vector autoregression (Myers and Thompson, 1989) and threshold-var (Enders, 2015) estimators. Khosravi (2011) demonstrates an approach to RBFNet training, similar to backpropagation for neural networks (Rumelhart et al., 1986). However, he sets some weights between the inputs and hidden layer, rather than the traditional approach, which has weights from the hidden layer to the outputs. He calls this his weighted RBFNet and finds improved accuracy on the UCI letter classification and Hoda digit recognition datasets.

The RBFNet relates to the Gaussian process regression (GPR) model (Rasmussen and

Williams, 2006b), in that they are both kernel learning methods. However, GPR is more expensive to fit than the RBFNet as it uses all training vectors as test-time prototypes, with a time-complexity of $\mathcal{O}(n^3)$ and memory complexity of $\mathcal{O}(n^2)$, where n is the number of training examples. When it comes to larger datasets, $n >= 10000$, storing the GPR's kernel Gram matrix and solving the associated linear systems become prohibitive. Rasmussen and Williams (2006a) discuss several procedures to learn in large datasets. They consider reduced-rank approximations to the kernel Gram matrix, discuss general strategies for greedy approximations, approximate the GP regression problem for fixed hyperparameters, and describe methods to approximate the marginal likelihood and its derivatives. Most of these methods use a subset of size $m < n$ of the training examples. More recently, online learning GPRs are considered. Koppel et al. (2021) study sequentially optimised Gaussian process approximation that preserves convergence to the population posterior, i.e., asymptotic posterior consistency.

The RBFNet has a time-complexity of $\mathcal{O}(knr)$ for the k-means part, where k is the number of clusters and r is the number of fitting iterations and $\mathcal{O}(n^2k + k^3)$ for the linear regression part. The RBFNet also relates to k-nearest neighbours (KNN) regression (Takezawa, 2006) as a local learning technique and kernel ridge regression (KRR) (Cristianini and Shawe-Taylor, 2000) as a kernel learning method. However, KNN, KRR and GPR typically use all training examples as prototypes at test time, rendering these techniques less useful for financial time series, which experience regular regime shifts.

2.4 Exponentially weighted recursive least squares

Exponentially Weighted Recursive Least Squares (EWRLS) is a well-known algorithm but is included here in our background section as it appears in several experiments. The recursive least squares (RLS) estimator is a particular case of the Kalman filter (Harvey, 1993). The exponentially weighted formulation (EWRLS) facilitates adaptation sequentially in time with exponential decay (Liu et al., 2010). The goal of EWRLS algorithm 2.1 is to forecast a target y_{t+h} using predictors $\mathbf{x}_t$. The predictors are mapped to the response using parameters $\boldsymbol{\theta}_t$. Thus, $y_{t+h} = f(\mathbf{x}_t; \boldsymbol{\theta}_t) + \varepsilon_t$. Algorithm 2.1 includes a variance stabilisation update, which ameliorates the build-up of large values along the diagonal of the precision matrix $\mathbf{P}$,

which may occur if the response y has low variance. See, for example, Gunnarsson (1996) for a further discussion on the regularisation of recursive least squares. Similar regularisation approaches for online learning and nonstationary data are studied by Moroshko et al. (2015).

Algorithm 2.1 Exponentially weighted recursive least squares.

Require: α `// the ridge penalty`
$0 \ll \tau < 1$ `// an exponential forgetting factor`
h `// a forecast horizon,` $h \in \mathbb{Z}^+$
Initialise: $\boldsymbol{\theta} = \mathbf{0}_d$, $\mathbf{P} = \mathbf{I}_d/\alpha$
Input: $\mathbf{x}_t \in \mathbb{R}^d$, y_t
Output: $\hat{y}_t$

$r = 1 + \mathbf{x}_{t-h}^T \mathbf{P}_{t-1} \mathbf{x}_{t-h}/\tau$
$\mathbf{k} = \mathbf{P}_{t-1}\mathbf{x}_{t-h}/(r\tau)$
$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{k}(y_t - \boldsymbol{\theta}_{t-1}^T \mathbf{x}_{t-h})$
$\mathbf{P}_t = \mathbf{P}_{t-1}/\tau - \mathbf{k}\mathbf{k}^T r$
$\mathbf{P}_t = \mathbf{P}_t \tau$ `// variance stabilisation`
$\hat{y}_t = \boldsymbol{\theta}_t^T \mathbf{x}_t$

2.5 Curds and whey multivariate regression

The curds and whey (CAW) procedure due to Breiman and Friedman (1997) is a suitable way of predicting several response variables from the same set of explanatory variables. The method takes advantage of the correlations between the response variables to improve predictive accuracy compared with the usual procedure of doing individual regressions of each response variable on the shared set of predictor variables. Assume there are n training examples, d predictors $\mathbf{X} \in \mathbb{R}^{n \times d}$ and q targets $\mathbf{Y} \in \mathbb{R}^{n \times q}$. The basic version of the procedure begins with the usual multivariate ridge regression

$$\boldsymbol{\Theta} = (\mathbf{X}^T\mathbf{X} + \alpha \mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{Y},$$

and estimates a shrinkage matrix

$$\hat{\mathbf{Y}} = \mathbf{X}\boldsymbol{\Theta}$$
$$\mathbf{W} = (\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} + \alpha \mathbf{I}_q)^{-1}\hat{\mathbf{Y}}^T\mathbf{Y}.$$

At test time, the vector of forecasts is

$$\tilde{\mathbf{y}}_t = \mathbf{W}\hat{\mathbf{y}}_t \equiv \mathbf{W}[\mathbf{x}_t^T \boldsymbol{\Theta}]^T.$$

Breiman and Friedman also derive estimates of the matrix $\mathbf{W}$ that take the form $\mathbf{W} = \mathbf{A}^{-1}\mathbf{D}\mathbf{A}$ where $\mathbf{A}$ is the $q \times q$ matrix whose rows are the response canonical coordinates and $\mathbf{D}$ is a diagonal 'shrinking' matrix. To estimate $\mathbf{W}$, one performs an eigendecomposition of the matrix

$$\mathbf{Q} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

Denote this eigendecomposition as $\mathbf{A}\boldsymbol{\Lambda}\mathbf{A}^T$, where $\boldsymbol{\Lambda}$ is the diagonal matrix whose $k'th$ diagonal element is λ_k,the $k'th$ eigenvalue of $\mathbf{Q}$. Furthermore, denote as $r = d/n$, the ratio of predictor dimensionality to its count. The diagonal shrinking matrix is estimated as

$$\mathbf{D}_{jj} = max\left[0, \frac{(1-r)(\boldsymbol{\Lambda}_{jj} - r)}{(1-r)^2 + r^2(1-\boldsymbol{\Lambda}_{jj})}\right], \quad j = 1, ..., q.$$

Given the nonstationary data we model, our particular interest is in sequential optimisation. Thus we combine the CAW procedure with EWRLS updating. In algorithm 2.2, we set the forecast horizon $h = 1$ for illustration purposes. The goal of the algorithm is to forecast a target vector $\mathbf{y}_{t+1}$ using predictors $\mathbf{x}_t$. The predictors are mapped to the response using parameters $\boldsymbol{\Theta}_t$. Thus, $\mathbf{y}_{t+1} = f(\mathbf{x}_t; \boldsymbol{\Theta}_t) + \mathbf{E}_t$.

2.6 Echo state networks

A recurrent neural network (RNN) is a multilayer network suitable for modelling dynamical systems with memory. Two well-known RNN architectures are the long short-term memory (LSTM) of Hochreiter and Schmidhuber (1997) and the echo state network (ESN) of Jaeger (2001). The LSTM achieves extended memory capability via hidden units whose activation decay is controlled by trainable gates. LSTMs are typically trained through backpropagation (BP) (Rumelhart et al., 1986) or backpropagation through time (BPTT) (Werbos, 1990), with second-order gradient descent methods called Hessian-free optimisation preferred (Martens and Sutskever, 2011). Whilst the approach is amenable to online optimisation, the ESN still

Algorithm 2.2 Sequentially optimised curds and whey regression.

Require: α, τ
// α is a ridge penalty
// τ is an exponential decay constant
Initialise: $\boldsymbol{\Theta} = \mathbf{0}_{d\times q}, \mathbf{W} = \mathbf{0}_{q\times q}, \mathbf{P} = \mathbf{I}_d/\alpha, \mathbf{Q} = \mathbf{I}_q/\alpha$
Input: $\mathbf{x}_t \in \mathbb{R}^d, \mathbf{y}_t \in \mathbb{R}^q$
Output: $\tilde{\mathbf{y}}_t \in \mathbb{R}^q$

$r_t = 1 + \mathbf{x}_{t-1}^T \mathbf{P}_{t-1} \mathbf{x}_{t-1}/\tau$
$\mathbf{k}_t = \mathbf{P}_{t-1}\mathbf{x}_{t-1}/(r_t\tau)$
$\hat{\mathbf{y}}_{t-1} = [\mathbf{x}_{t-1}^T \boldsymbol{\Theta}_{t-1}]^T$
$\boldsymbol{\Theta}_t = \boldsymbol{\Theta}_{t-1} + \mathbf{k}_t(\mathbf{y}_t - \hat{\mathbf{y}}_{t-1})^T$
$\mathbf{P}_t = \mathbf{P}_{t-1}/\tau - \mathbf{k}_t \mathbf{k}_t^T r_t$
$\mathbf{P}_t = \mathbf{P}_{t-1}\tau$
$\hat{\mathbf{y}}_t = [\mathbf{x}_t^T \boldsymbol{\Theta}_t]^T$
$s_t = 1 + \hat{\mathbf{y}}_{t-1}^T \mathbf{Q}_{t-1} \hat{\mathbf{y}}_{t-1}/\tau$
$\mathbf{m}_t = \mathbf{Q}_{t-1}\hat{\mathbf{y}}_{t-1}/(s_t\tau)$
$\mathbf{W}_t = \mathbf{W}_{t-1} + \mathbf{m}_t(\mathbf{y}_t - \mathbf{W}_{t-1}\hat{\mathbf{y}}_{t-1})^T$
$\mathbf{Q}_t = \mathbf{Q}_{t-1}/\tau - \mathbf{m}_t \mathbf{m}_t^T s_t$
$\mathbf{Q}_t = \mathbf{Q}_{t-1}\tau$
$\tilde{\mathbf{y}}_t = \mathbf{W}_t \hat{\mathbf{y}}_t$

outperforms Hessian-free trained RNNs on a benchmark suite designed to challenge long short-term memory acquisition (Jaeger, 2012). Yu et al. (2021) discuss the prolonged fitting times of BPTT, although they provide a faster training method than BPTT, which separates the LSTM cell into forward and recurrent models. In the context of financial time series modelling, Lim and Gorse (2020) train an LSTM to predict the price movements of Bitcoin and show that the model performs well even when the statistical behaviour of the market changes.

In contrast to the LSTM, the ESN achieves extended memory capability by using large dynamic reservoir networks of leaky integrator neurons with long-time constants. An ESN schematic is shown in figure 2.4. Determination of the optimal output weights is solvable analytically, for example, sequentially with RLS (Jaeger, 2002), and is extremely fast to train. ESNs are an example of the reservoir computing paradigm for understanding and training recurrent neural networks, based on treating the recurrent part (the reservoir) differently than the readouts from it (Lukosevicius et al., 2012). From Jaeger (2002), the echo state property is defined:

Definition 2 (echo state property) *If a network is started from two arbitrary states* $\mathbf{x}_0$, $\tilde{\mathbf{x}}_0$

and is run with the same input sequence in both cases, the resulting state sequences $\mathbf{x}_T$, $\tilde{\mathbf{x}}_T$ *converge to each other. If this condition holds, the reservoir network state will asymptotically depend only on the input history, and the network is said to be an echo state network.*

The echo state property is guaranteed if the dynamic reservoir weight matrix $\mathbf{W}^{hidden}$ is scaled such that its spectral radius

$$\rho(\mathbf{W}^{hidden}) = max\{|\lambda_1|, ..., |\lambda_{n_{hidden}}|\},$$

that is, its largest absolute eigenvalue, satisfies $\rho(\mathbf{W}^{hidden}) < 1$. This ensures that $\mathbf{W}^{hidden}$ is contractive. The mathematically correct connection between the spectral radius and the echo state property is that the latter is violated if $\rho(\mathbf{W}^{hidden}) > 1$ in reservoirs using the tanh function as neuron nonlinearity and for zero input (Lukosevicius and Jaeger, 2009). Fitting details for the model can be found in subsection 6.3.1.1.

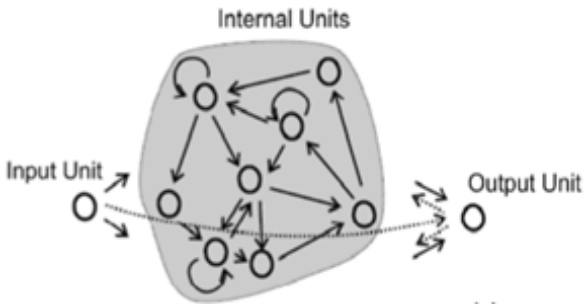


Figure 2.4: A schematic of the echo state network; source: (Jaeger, 2002).

Numerous articles demonstrate ESNs within a reinforcement learning context. For example, Szita et al. (2006) emphasise that ESNs are promising candidates for partially observable problems where information about the past may improve performance. ESNs are linear function approximators acting on the internal state representation built from the previous observations. Therefore, Gordon's results about linear function approximators (Gordon, 2000) can be transferred to the ESN architecture. Building on this, Szita et al. provide proof of convergence to a bounded region for ESN training in the case of k-order Markov decision processes.

Several publications show ESN applications to finance. For example, Lin et al. (2011) forecast prices from the S&P 500 constituents using ESNs, outperforming a buy-and-hold

baseline strategy. Maciel et al. (2014) use ESNs to forecast and trade foreign exchange (FX) rates, outperforming an arma baseline. Liu et al. (2018) create an optimisation algorithm to learn the parameters of an ESN, rather than adopting random weight initialisation, finding improved prediction accuracy when providing forecasts of the daily returns of the Shanghai Composite Index. Finally, Xue et al. (2016) create a scale-free, highly clustered ESN with better echo state property than the conventional ESN of Jaeger (2002). They find improved prediction performance over the conventional ESN when experimenting with the S&P 500 index and the EURUSD FX rate.

2.7 Policy gradient reinforcement learning

Whereas most reinforcement learning algorithms focus on action-value estimation, learning the value of actions and selecting them based on their estimated values, policy gradient (PG) methods learn a parameterised policy that can select actions without using a value function. Williams (1992b) introduces his reinforce algorithm

$$\Delta\boldsymbol{\theta}_{ij} = \eta_{ij}(r - \mathbf{b}_{ij})\ln(\partial\pi_i/\partial\boldsymbol{\theta}_{ij}),$$

where $\boldsymbol{\theta}_{ij}$ is the model weight going from the $j'th$ input to the $i'th$ output, and $\boldsymbol{\theta}_i$ is the weight vector for the $i'th$ hidden unit of a network of such units, whose goal it is to adapt in such a way as to maximise the scalar reward r. For the moment, we exclude the dependence on the time of the weight update to unclutter the notation. Furthermore, η_{ij} is a learning rate, typically applied with gradient ascent, $\mathbf{b}_{ij}$ is a reinforcement baseline, conditionally independent of the model outputs y_i, given the network parameters $\boldsymbol{\theta}$ and inputs $\mathbf{x}_i$. The characteristic eligibility of $\boldsymbol{\theta}_{ij}$ is $\ln(\partial\pi_i/\partial\boldsymbol{\theta}_{ij})$, where $\pi_i(y_i = c, \boldsymbol{\theta}_i, \mathbf{x}_i)$ is a probability mass function determining the value of y_i as a function of the parameters of the unit and its input. Baseline subtraction $r - \mathbf{b}_{ij}$ plays a vital role in reducing the variance of gradient estimators. Sugiyama (2015) shows that the optimal baseline is given as

$$b^* = \frac{\mathbb{E}_{p(r|\boldsymbol{\theta})}\left[r_t \left\| \sum_{t=1}^{T} \nabla \ln \pi(a_t|s_t, \boldsymbol{\theta}) \right\|^2\right]}{\mathbb{E}_{p(r|\boldsymbol{\theta})}\left[\left\| \sum_{t=1}^{T} \nabla \ln \pi(a_t|s_t, \boldsymbol{\theta}) \right\|^2\right]},$$

where the policy function $\pi(a_t|s_t,\boldsymbol{\theta})$ denotes the probability of taking action a_t at time t given state s_t, parameterised by $\boldsymbol{\theta}$. The expectation $\mathbb{E}_{p(r|\boldsymbol{\theta})}$, is distributed over the probability of rewards given the model parameterisation. The main result of Williams' paper is

Theorem 3 *For any reinforce algorithm, the inner product of* $\mathbb{E}[\Delta\boldsymbol{\theta}|\boldsymbol{\theta}]$ *and* $\nabla\mathbb{E}[r|\boldsymbol{\theta}]$ *is non-negative, and if* $\eta_{ij} > 0$*, then this inner product is zero if and only if* $\nabla\mathbb{E}[r|\boldsymbol{\theta}] = 0$*. If* η_{ij} *is independent of i and j, then* $\mathbb{E}[\Delta\boldsymbol{\theta}|\boldsymbol{\theta}] = \eta\nabla\mathbb{E}[r|\boldsymbol{\theta}]$.

This result relates $\nabla\mathbb{E}[r|\boldsymbol{\theta}]$, the gradient in weight space of the performance measure $\mathbb{E}[r|\boldsymbol{\theta}]$, to $\mathbb{E}[\Delta\boldsymbol{\theta}|\boldsymbol{\theta}]$, the average update vector in weight space. Thus for any reinforce algorithm, the average update vector in weight space lies in a direction for which this performance measure is increasing, and the quantity $(r-\mathbf{b}_{ij})\ln(\partial\pi_i/\partial\boldsymbol{\theta}_{ij})$ represents an unbiased estimate of $\partial\mathbb{E}[r|\boldsymbol{\theta}]/\partial\boldsymbol{\theta}_{ij}$.

2.7.1 Policy gradient methods in financial trading

Moody et al. (1998) train trading systems by optimising objective functions that directly measure trading performance, such as the Sharpe ratio (Sharpe, 1966). They train their systems using a direct, recurrent reinforcement learning (DRL) algorithm, an example of the PG method. The *direct* part refers to the fact that the model targets a position directly, with weights that are adapted so that the performance measure is maximised. Denote the annualised Sharpe ratio as

$$sr_T = 252^{0.5} \times \frac{r_T - r_f}{s_T},$$

where r_T is the average daily return for a trading strategy between times $t = 1,...,T$, s_T is the standard deviation of these returns, and r_f is the risk-free rate. The differential Sharpe ratio is defined as

$$\frac{dsr_t}{d\tau} = \frac{b_{t-1}\Delta a_t - 0.5a_{t-1}\Delta b_t}{(a_{t-1} - a_{t-1}^2)^{3/2}}, \tag{2.4}$$

where the quantities a_t and b_t are exponentially weighted estimates of the first and second moments of the reward r_t

$$\begin{aligned}\mathbb{E}[r_T] &\approx a_t = a_{t-1} + \tau(r_t - a_{t-1})\\ Var[r_T] &\approx b_t = b_{t-1} + \tau(r_t^2 - b_{t-1}), \quad t = 1,...,T.\end{aligned}$$

They consider a batch gradient ascent update for the model parameters $\boldsymbol{\theta}$

$$\Delta\boldsymbol{\theta}_T = \eta \frac{dsr_T}{d\boldsymbol{\theta}},$$

where

$$\begin{aligned} \frac{dsr_T}{d\boldsymbol{\theta}} &= \sum_{t=1}^{T} \frac{dsr_T}{dr_t}\frac{dr_t}{d\boldsymbol{\theta}} \\ &= \sum_{t=1}^{T} \left\{ \frac{b_T - a_T r_t}{(b_T - a_T^2)^{3/2}} \right\} \left\{ \frac{dr_t}{df_t}\frac{df_t}{d\boldsymbol{\theta}} + \frac{dr_t}{df_{t-1}}\frac{df_{t-1}}{d\boldsymbol{\theta}} \right\}. \end{aligned} \tag{2.5}$$

The reward

$$r_t = \Delta p_t f_{t-1} - \delta_t |\Delta f_t|$$

depends on the change in reference price p_t, transaction cost δ_t and a differentiable position function

$$f_t \triangleq f(\mathbf{x}_t, \boldsymbol{\theta}_t), \quad -1 \leq f_t \leq 1.$$

Equation 2.5 shows the temporal dependence of the system state on previous decisions, expressed through the derivative of the current return on the previous position.

Moody et al. (1998) present empirical results that demonstrate the efficacy of their methods. Both methods outperform a trading system based on forecasts that minimise mean-square error. However, an undesirable property of the Sharpe ratio is that it penalises a model that produces returns larger than $\frac{\mathbb{E}[r^2]}{\mathbb{E}[r]} \approx \frac{b_t}{a_t}$. Rational traders are typically concerned with the volatility of loss-making trades but not with the volatility of profitable trades. Without providing experimental results, Moody et al. (1998) briefly discuss the possibility of using so-called downside risk measures for DRL purposes, such as semi-variance, downside deviation and Sterling ratios. A likely, unwelcome consequence of using these downside risk measures for DRL optimisation is that the model might learn to trade in quite a defensive manner, spending large amounts of time out of the market. It is more likely that their differential Sharpe ratio approach leads to a model that trades through a larger set of market conditions, therefore allowing for the possibility of increased total returns.

Gold (2003) extends Moody et al. (1998)'s work and investigates high-frequency trading

(hft) in FX with neural networks trained via DRL. He examines the impact of shared system hyper-parameters on performance. In general, he concludes that the trading systems may be effective but that the performance varies widely for different currency markets, and simple statistics of the markets cannot explain this variability. Gorse (2011) applies a batch-learning gradient-based recurrent reinforcement learner to trading the S&P 500 on a daily basis and finds favourable performance compared to a genetic programming baseline. Actor-critic PG methods (Sutton et al., 1999) are used by Tamar et al. (2017) to model dynamic financial risk measures and by Mnih et al. (2016) and Luo et al. (2019) to train reinforcement learning agents to trade futures contracts. Deep, recurrent reinforcement learning networks are considered by Azhikodan et al. (2019), Ye et al. (2020), Aboussalah and Lee (2020), Lei et al. (2020) and Betancourt and Chen (2021) on various trading-related activities such as proprietary risk-taking, algorithmic execution and portfolio management.

2.8 Prediction with expert advice

Prediction with expert advice is a multidisciplinary research area that predicts individual sequences in an online learning setting. Unlike standard statistical approaches, the prediction with expert advice framework imposes no probabilistic assumption on the data-generating mechanism. Instead, it generates predictions that work well for all sequences, with performance nearly as good as the best expert with hindsight. The basic structure of problems in this context is encapsulated in algorithm 2.3, adapted from Rakhlin and Sridharan (2014).

Algorithm 2.3 Sequential prediction with an adaptive environment.

```
// iterate over each time step
```
for $t \leftarrow 1$ **to** T **do**
- The learner chooses the set of predictions $\hat{\mathbf{y}}_t \in \mathcal{D}$, where $\mathcal{D}$ is the decision space.
- Nature simultaneously chooses an outcome $y_t \in \mathcal{A}$, where $\mathcal{A}$ is the outcome space.
- The player suffers a loss $\ell(\hat{\mathbf{y}}_t, y_t)$ and both players observe $(\hat{\mathbf{y}}_t, y_t)$.

Perhaps the most well-known algorithm within this framework is the weighted majority (WM) algorithm of Littlestone and Warmuth (1994), shown in algorithm 2.4. The authors study the construction of prediction algorithms where the learner faces a sequence of trials, and the goal is to make as few mistakes as possible with predictions made at the end of each trial. They are interested in cases where the learner believes some experts will perform

well, but the learner does not know which ones. A simple method based on weighted voting is introduced to minimise the regret concerning the best expert with hindsight, even in the presence of errors in the data. They discuss various versions of the algorithm, proving mistake bounds for them that are closely related to the mistake bounds of the best algorithms of the pool. Finally, given a sequence of trials, if there is an algorithm in the pool d that makes at most m mistakes, then the WM algorithm will make at most $c(log|d| + m)$ mistakes on that sequence, where c is a fixed constant.

Denote the forecaster's cumulative loss as $\hat{L}_T = \sum_{i=1}^{T} \ell(\hat{y}_t, y_t)$ and the cumulative loss of expert i as $\hat{L}_{i,T} = \sum_{i=1}^{T} \ell(\hat{\mathbf{y}}_{i,t}, y_t)$. In algorithm 2.4, we assume the loss function ℓ is convex in its first argument and that it takes values in $[0,1]$. Cesa-Bianchi and Lugosi (2006) show that for any number of experts d and a learning rate $\eta > 0$, the regret of the WM algorithm relative to the best expert with hindsight satisfies

$$\hat{L}_T - \min_{i=1,\ldots,d} L_{i,T} \leq \frac{\ln d}{\eta} + \frac{T\eta}{8}.$$

In particular, with $\eta = \sqrt{8\ln(d)/T}$, the upper bound becomes $\sqrt{(T/2)\ln(d)}$. If convexity in the loss function is not presupposed or no separate training phase is applied, bounded regret concerning the best learner with hindsight is not guaranteed. However, Calliess (2019) provides performance guarantees for increasingly long durations, provided learning has been allowed to take place sufficiently long; this gives rise to a new generalised notion of convergence (and thereby of online-learnability) which he refers to as convergence with increasing permanence.

The WM algorithm is less suited to nonstationary data, despite the authors providing a modification to their base algorithm in section 3 of their paper, which ensures that the weight assigned to the individual experts never dips below η/d, which is the learning rate divided by the number of experts. This fixed, minimum threshold weight is somewhat rudimentary. In the context of financial time series, our preference is for algorithms that assign more weight to experts now performing well, irrespective of whether they performed less well previously. Herbster and Warmuth (1998) tackle prediction with expert advice in the nonstationary setting by introducing an algorithm that determines the best experts for seg-

Algorithm 2.4 The weighted majority algorithm.

Input: outcome $y_t \in \{0,1\}$, expert predictions $\hat{\mathbf{y}}_t \in \{0,1\}^d$ and T rounds
Initialise: $\tilde{\mathbf{w}}_0 = \mathbf{1}^d$, $\eta = \sqrt{8\log(d)/T}$
Output: weighted majority prediction $\hat{y}_t$

for $t \leftarrow 1$ **to** T **do**
- Set $\mathbf{w}_t = \tilde{\mathbf{w}}_t / \sum_{i=1}^{d} \tilde{\mathbf{w}}_{i,t}$.
- Assign $p_0 = \sum_{i=1}^{d} I(\hat{\mathbf{y}}_{i,t} = 0)\mathbf{w}_{i,t}$.
- Allocate the weighted majority prediction $\hat{y}_t = 0$ with probability p_0 or $\hat{y}_t = 1$ with probability $1 - p_0$.
- Suffer loss $\ell(\hat{y}_t, y_t) = |\hat{y}_t - y_t|$.
- Update $\tilde{\mathbf{w}}_{t+1} = \tilde{\mathbf{w}}_t \exp^{-\eta \ell(\hat{y}_t, y_t)}$.

ments of the individual sequences. When the number of segments is $k+1$ and the sequence is of length T, they can bound the additional loss of their algorithm over the best partition by $\mathcal{O}\big(k\log|d| + k\log[T/k]\big)$. An Achilles heel of the algorithm is that knowledge is required a priori of the optimal number of segments k corresponding to the periods when new best experts are required. Freund et al. (1997) study online learning algorithms that combine the predictions of several experts who, like the WM algorithm, belong to the multiplicative weights family of algorithms. They apply their methods to the prediction problem where the best expert may change with time. They derive a specialist algorithm that is as fast as the best-known algorithm of Herbster and Warmuth (1998) and achieves almost as good a loss bound. However, unlike Herbster and Warmuth's algorithm, Freund et al. 's algorithm does not require prior knowledge of the length of the sequence and the number of segments.

2.9 Online financial portfolio selection

The prediction with expert advice framework is valuable in online portfolio selection. Helmbold et al. (1998) present an online investment algorithm that achieves almost the same wealth as the best constant-rebalanced portfolio (BCRP) determined in hindsight from the actual market outcomes. The algorithm employs a multiplicative update rule derived using a framework introduced by Kivinen and Warmuth (1995). They test the performance of their algorithm on actual stock data from the New York Stock Exchange accumulated over 22 years. On these data, their algorithm outperforms the best single stock with hindsight, as well as Cover's universal portfolio selection algorithm (Cover, 1991). Rather than assum-

ing the stationarity of financial time series, Gaivoronski and Stella (2000) propose variable rebalanced portfolios which calculates the BCRP portfolio based on a latest sliding window.

Singer (1998) notes that the earlier work into online portfolio selection algorithms which are competitive with the BCRP determined in hindsight (Cover, 1991; Helmbold et al., 1998; Cover and Ordentlich, 1996), employ the assumption that high yield returns can be achieved using a fixed asset allocation strategy. However, stock markets are nonstationary. The return of a constant rebalanced portfolio is often much smaller than the return of an ad-hoc investment strategy that adapts to changes in the market. In his paper, Singer presents an efficient portfolio selection algorithm that can track a changing market and describes a simple extension of his algorithm for including transaction costs. Finally, he provides a simple analysis of the competitiveness of the algorithm and evaluates its performance on stock data from the New York Stock Exchange accumulated over 22 years; his algorithm outperforms all the algorithms referenced above, with and without transaction costs.

Strategies in the Follow-the-Leader (FTL) approach try to track the BCRP strategy over time. Regularised FTL approaches that ameliorate transaction costs are considered by Agarwal et al. (2006), specifically second-order online Newton step methods. More recently, Li and Hoi (2016) survey prediction with expert advice algorithms in the context of portfolio selection. They design four new algorithms to solve the online portfolio selection problem, including a correlation-driven nonparametric learning approach. Yang et al. (2020b) present a new online portfolio strategy based on the online learning character of a weak aggregating algorithm (WAA), a gradient-based reinforcement learning bandit approach.They consider a number of exponential gradient strategies of different values of parameter η as experts, and then determine the next portfolio by using the WAA to aggregate the experts' advice.

2.10 Learning to rank portfolios of assets

The ranking of a subset of assets to hold in a long/short portfolio is related to the issue of portfolio selection. Poh et al. (2021) apply learning-to-rank algorithms, primarily designed for natural language processing, to cross-sectional momentum trading strategies. Cross-sectional strategies mitigate some of the risk associated with wider market moves by buying the top α-percentile of strategies with the highest expected future returns and selling the bot-

tom α-percentile of strategies with the lowest expected future returns. Classical approaches that rely on the ranking of assets include ranking annualised returns (Jegadeesh and Titman, 1993), or more recent regress-then-rank approaches (Wang and Rasheed, 2018; Kim, 2019; Gu et al., 2020). Poh et al. take a different approach using pair-wise learning-to-rank algorithms such as RankNet (Burges et al., 2005) and LambdaRank (Burges, 2010). Overall, in experiments they conduct on monthly-sampled CRSP 2019 data, the learning-to-rank algorithms achieve higher total and risk-adjusted returns than the traditional regress-then-rank approaches. In terms of assessing the limitations of the work, experimenting with daily sampled data would probably resonate more with the finance community, as this facilitates the possibility of a more active portfolio management style. Grinold and Kahn (2019) demonstrate how active portfolio management can outperform passive portfolio management and discuss associated quantitative procedures. However, working with higher frequency sampled data presents further challenges, such as higher transaction costs due to frequent trading. Additionally, there are slower training times for the learning-to-rank algorithms and, perhaps more crucially, a decrease in signal and an increase in noise from the financial time series.

2.11 Naive Bayes ranking

In chapter 7, we introduce our naive Bayes asset ranker for portfolios comprised of financial assets. First, however, we perform a literature review of naive Bayes ranking. The phrase *naive Bayes ranking* has broad meaning, with different implementations in various contexts. At the core of the idea is the naive Bayes classifier

$$P(y_c|\mathbf{x}) = \frac{P(\mathbf{x}|y_c)P(y_c)}{\sum_j P(\mathbf{x}|y_j)P(y_j)},$$

which predicts that the target y takes on a label value of c if this posterior probability is highest. Independence assumptions in the inputs $\mathbf{x}$ mean that the probabilities can be modelled iteratively and inexpensively. Zhang and Su (2004) study the general performance of naive Bayes in ranking. They use the area under the receiver operating characteristics curve (auc) (Provost and Fawcett, 1997) to evaluate the quality of rankings generated by a classifier. The auc is created by plotting the true-positive rate against the false-positive rate at various threshold settings and has a maximum value of one if no positive example precedes any neg-

ative example. For binary classification, auc is equivalent to the probability that a randomly chosen example of class $-$ will have a more negligible estimated probability of belonging to class $+$ than a randomly chosen example of class $+$. Thus, the auc is a measure of the quality of ranking. Overall, their naive Bayes ranker evaluated using the auc outperforms the C4.4 decision-tree algorithm for ranking (Provost and Domingos, 2003) on various datasets.

Flach and Matsubara (2007) consider binary classification tasks, where a ranker sorts a set of instances from highest to lowest expectation that the instance is positive. They propose a lexicographic ranker, lexrank, whose rankings are derived not from scores but a simple ranking of attribute values obtained from the training data. Using the odds ratio to rank the attribute values, they obtain a restricted version of the naive Bayes ranker. They systematically develop the relationships and differences between classification, ranking, and probability estimation, which leads to a novel connection between the Brier score (Brier, 1950) and auc curves.

Krawczyk and Wozniak (2015) propose a modification to the naive Bayes classifier for mining streams in a nonstationary environment in the presence of the concept drift phenomenon (Gama, 2012). They add a weighting module that automatically assigns an importance factor to each object extracted from the stream; the higher the weight, the more significant the influence the given object exerts on the classifier training procedure. So that their classifier adapts quickly to evolving data, they imbue it with a forgetting principle implemented as weight decay. With each passing iteration, the level of importance of previous objects is decreased until they are discarded from the data collection. In summary, their algorithm works by fitting a naive Bayes classifier on samples they deem essential and removing unnecessary and outdated examples that no longer represent the present state of the analysed data stream. This approach contrasts with our naive Bayes ranker shown in algorithm 7.1; we learn from all training examples on the fly and assign some importance to each expert that offers us advice. Unlike Krawczyk and Wozniak's algorithm, experts that have performed poorly are never discarded, and if they start performing well, they have the opportunity to have more importance assigned to them.

Chapter 3

Datasets

Data for most of our experiments are extracted from Refinitiv using their Data Platform Python client. Each quoted instrument available on Refinitiv is identified by its information code (ric). Note that the Refinitiv data constitutes quote or trade data from multiple vendors. For example, the foreign exchange data is sourced from tradable matching venues such as EBS, Refinitiv Dealing and Matching, market-leading brokers and over-the-counter markets. The remaining cross-asset class data is sourced from multiple global exchanges, including the London Stock Exchange, the New York Stock Exchange, the Chicago Mercantile Exchange, the Intercontinental Exchange and Eurex. Finally, one of our experiments uses data extracted from the digital assets futures exchange, BitMEX.

3.1 Online learning with radial basis function networks

We extract daily sampled data from Refinitiv, including currency pairs, equities, rates, credit, metals, agriculture, energy and crypto. The sampled prices are the last daily traded or quoted limit order book price, with a snapshot taken at 10 PM GMT. The dataset begins on 2018-11-01 and ends on 2022-05-20. We use 649 training and 648 test examples. The full set of constituents, including sector information, is shown in table 3.1.

Table 3.1: The Refinitiv cross-asset class dataset.

	ric	description	sector
0	BTC=	Bitcoin/US Dollar	crypto
1	ETH=	Ethereum/US Dollar	crypto
2	LTC=	Litecoin/US Dollar	crypto
3	AUD=	Australian Dollar/US Dollar	FX
4	AUDCHF=	Australian Dollar/Swiss Franc	FX
5	AUDJPY=	Australian Dollar/Japanese Yen	FX

Continued on next page

Table 3.1 – continued from previous page

	ric	description	sector
6	BRL=	US Dollar/Brazilian Real	FX
7	CAD=	US Dollar/Canadian Dollar	FX
8	CADCHF=	Canadian Dollar/Swiss Franc	FX
9	CADJPY=	Canadian Dollar/Japanese Yen	FX
10	CHF=	US Dollar/Swiss Franc	FX
11	CHFJPY=	Swiss Franc/Japanese Yen	FX
12	CNY=	US Dollar/Chinese Yuan	FX
13	EUR=	Euro/US Dollar	FX
14	EURAUD=	Euro/Australian Dollar	FX
15	EURCAD=	Euro/Canadian Dollar	FX
16	EURCHF=	Euro/Swiss Franc	FX
17	EURGBP=	Euro/British Pound	FX
18	EURJPY=	Euro/Japanese Yen	FX
19	GBP=	British Pound/US Dollar	FX
20	GBPAUD=	British Pound/Australian Dollar	FX
21	GBPCAD=	British Pound/Canadian Dollar	FX
22	GBPCHF=	British Pound/Swiss Franc	FX
23	GBPJPY=	British Pound/Japanese Yen	FX
24	GBPNZD=	British Pound/New Zealand Dollar	FX
25	HKD=	US Dollar/Hong Kong Dollar	FX
26	INR=	US Dollar/Indonesia Rupiah	FX
27	JPY=	US Dollar/Japanese Yen	FX
28	KRW=	US Dollar/Korea Won	FX
29	MXN=	US Dollar/Mexico Peso	FX
30	NOK=	US Dollar/Norwegian Krone	FX
31	NZD=	New Zealand Dollar/US Dollar	FX
32	PLN=	US Dollar/Polish Zloty	FX
33	RUB=	US Dollar/Russian Ruble	FX
34	SEK=	US Dollar/Swedish Krone	FX
35	SGD=	US Dollar/Singapore Dollar	FX
36	TRY=	US Dollar/Turkish Lira	FX
37	TWD=	US Dollar/Taiwanese Dollar	FX
38	ZAR=	US Dollar/South African Rand	FX
39	.BCOM	Bloomberg Commodity	commodities
40	.TRCCRB	Refinitiv CRB	commodities
41	ITEEU5Y=MG	ITRAXX Europe CDS	credit
42	ITEXO5Y=MG	ITRAXX Crossover CDS	credit
43	.TRXFLDGLPUENE	Refinitiv Global Energy	energy
44	.AEX	AEX	equities
45	.AORD	ASX All Ordinaries	equities
46	.AXJO	S&P/ASX 200	equities
47	.BFX	BEL 20	equities
48	.BSESN	S&P Sensex	equities

Continued on next page

Table 3.1 – continued from previous page

	ric	description	sector
49	.BVSP	Brazilian IBOVESPA	equities
50	.FCHI	CAC 40	equities
51	.FTAS	FTSE ALL SHARE	equities
52	.FTJ203	JSE All Share	equities
53	.FTSE	FTSE 100	equities
54	.GDAXI	DAX	equities
55	.GSPTSE	TSX Composite	equities
56	.HSI	Hang Seng	equities
57	.IBEX	IBEX 35	equities
58	.IMOEX	MOEX Russia	equities
59	.IRTS	RTS	equities
60	.IXIC	NASDAQ Composite	equities
61	.KLSE	FTSE Bursa KLSE	equities
62	.KS11	Korea Composite	equities
63	.MID	S&P 400 Mid Cap	equities
64	.MXX	Mexican IPC	equities
65	.NDX	NASDAQ 100	equities
66	.NYA	NYSE Composite	equities
67	.OMXHPI	OMX Helsinki	equities
68	.OMXS30	OMX Stockholm 30	equities
69	.SPX	S&P 500	equities
70	.SSEC	Shanghai Composite	equities
71	.SSMI	Swiss Market	equities
72	.STI	Straits Times	equities
73	.STOXX50	EURO STOXX 50	equities
74	.STOXX50E	EURO STOXX 50	equities
75	.TOPX	TOPIX	equities
76	.TRXFLDGLPU	Refinitiv Global Equities	equities
77	.TRXFLDGLPUHLC	Refinitiv Global Healthcare	equities
78	.TRXFLDUSP	Refinitiv United States	equities
79	AU10YT=RR	Australia 10-year Note	rates
80	CA10YT=RR	Canada 10-year Note	rates
81	CH10YT=RR	Swiss 10-year Note	rates
82	CN10YT=RR	China 10-year Note	rates
83	DE10YT=RR	Germany 10-year Note	rates
84	ES10YT=RR	Spain 10-year Note	rates
85	FR10YT=RR	France 10-year Note	rates
86	GB10YT=RR	United Kingdom 10-year Note	rates
87	IN10YT=RR	India 10-year Note	rates
88	IT10YT=RR	Italy 10-year Note	rates
89	JP10YT=RR	Japan 10-year Note	rates
90	RU10YT=RR	Russia 10-year Note	rates
91	US10YT=RRPS	10-Year Note	rates

Continued on next page

Table 3.1 – continued from previous page

	ric	description	sector
92	US2YT=RRPS	2-Year Note	rates
93	US30YT=RRPS	30-Year Bond	rates
94	US5YT=RRPS	5-Year Note	rates
95	ZA10YT=RR	South Africa 10-year Note	rates
96	XAG=	Silver	metals
97	XAU=	Gold	metals
98	XPD=	Palladium	metals
99	XPT=	Platinum	metals

3.2 Reinforcement learning for systematic FX trading

We extract daily-sampled data from Refinitiv for 36 major cash foreign exchange (FX) pairs with available tomnext forward points and outrights. These FX pairs are listed in table 3.2. The dataset begins on 2010-12-07 and ends on 2021-10-21, with 2,840 observations per pair.

Table 3.2: The major FX pairs experimented with, including rics.

	currency pair	ric	tomnext ric
1	AUDUSD	AUD=	AUDTN=
2	EURAUD	EURAUD=	EURAUDTN=
3	EURCHF	EURCHF=	EURCHFTN=
4	EURCZK	EURCZK=	EURCZKTN=
5	EURDKK	EURDKK=	EURDKKTN=
6	EURGBP	EURGBP=	EURGBPTN=
7	EURHUF	EURHUF=	EURHUFTN=
8	EURJPY	EURJPY=	EURJPYTN=
9	EURNOK	EURNOK=	EURNOKTN=
10	EURPLN	EURPLN=	EURPLNTN=
11	EURSEK	EURSEK=	EURSEKTN=
12	EURUSD	EUR=	EURTN=
13	GBPUSD	GBP=	GBPTN=
14	NZDUSD	NZD=	NZDTN=
15	USDCAD	CAD=	CADTN=
16	USDCHF	CHF=	CHFTN=
17	USDCNH	CNH=	CNHTN=
18	USDCZK	CZK=	CZKTN=
19	USDDKK	DKK=	DKKTN=
20	USDHKD	HKD=	HKDTN=
21	USDHUF	HUF=	HUFTN=

Continued on next page

Table 3.2 – continued from previous page

	currency pair	ric	tomnext ric
22	USDIDR	IDR=	IDRTN=
23	USDILS	ILS=	ILSTN=
24	USDINR	INR=	INRTN=
25	USDJPY	JPY=	JPYTN=
26	USDKRW	KRW=	KRWTN=
27	USDMXN	MXN=	MXNTN=
28	USDNOK	NOK=	NOKTN=
29	USDPLN	PLN=	PLNTN=
30	USDRUB	RUB=	RUBTN=
31	USDSEK	SEK=	SEKTN=
32	USDSGD	SGD=	SGDTN=
33	USDTHB	THB=	THBTN=
34	USDTRY	TRY=	TRYTN=
35	USDTWD	TWD=	TWDTN=
36	USDZAR	ZAR=	ZARTN=

3.3 The recurrent reinforcement learning crypto agent

We extract five minutely sampled trade and order book information for the XBTUSD (Bitcoin vs US Dollar) perpetual swap from BitMEX; this is for March 2017 to December 2021. There are various technical aspects to this contract which are described in section 6.2. The data are extracted using the BitMEX API Explorer, a REST API for the BitMEX trading platform.

3.4 Sequential asset ranking in nonstationary time series

We extract constituent data for the S&P 500 index from Refinitiv. There are 505 constituents of this index, owing to symbols with multiple listings, such as class A and B shares. For each symbol, we extract the closing daily transaction prices. Due to their relatively new trade history, some time series have little data. Therefore, we select a subset of the S&P 500 index, where each constituent contains a trade count greater than or equal to the 25'th percentile of trade counts. This subset, 378 rics, is shown in table 3.3. The dataset begins on 2001-01-26 and ends on 2022-03-25, 5326 days.

Table 3.3: Refinitiv S&P 500 dataset.

	ric	description	sector
1	A	Agilent Technologies, Inc.	life sciences
2	AAPL.O	Apple Inc.	technology
3	ABC	AmerisourceBergen Corporation	trade
4	ABMD.O	ABIOMED, Inc.	life sciences
5	ABT	Abbott Laboratories	life sciences
6	ADBE.O	Adobe Inc.	technology
7	ADI.O	Analog Devices, Inc.	manufacturing
8	ADM	Archer-Daniels-Midland Company	manufacturing
9	ADP.O	Automatic Data Processing, Inc.	technology
10	ADSK.O	Autodesk, Inc.	technology
11	AEE	Ameren Corporation	energy
12	AEP.O	American Electric Power Company, Inc.	energy
13	AES	AES Corporation	energy
14	AFL	Aflac Incorporated	finance
15	AIG	American International Group, Inc.	finance
16	AJG	Arthur J. Gallagher & Co.	finance
17	AKAM.O	Akamai Technologies, Inc.	trade
18	ALB	Albemarle Corporation	life sciences
19	ALGN.O	Align Technology, Inc.	life sciences
20	ALK	Alaska Air Group, Inc.	energy
21	ALL	Allstate Corporation	finance
22	AMAT.O	Applied Materials, Inc.	manufacturing
23	AMD.O	Advanced Micro Devices, Inc.	manufacturing
24	AME	AMETEK, Inc.	life sciences
25	AMGN.O	Amgen Inc.	life sciences
26	AMT	American Tower Corporation	real estate
27	AMZN.O	Amazon.com, Inc.	trade
28	ANSS.O	ANSYS, Inc.	technology
29	AON	Aon Plc Class A	finance
30	AOS	A. O. Smith Corporation	manufacturing
31	APA.O	APA Corp.	energy
32	APD	Air Products and Chemicals, Inc.	life sciences
33	APH	Amphenol Corporation Class A	manufacturing
34	ARE	Alexandria Real Estate Equities, Inc.	real estate
35	ATO	Atmos Energy Corporation	energy
36	ATVI.O	Activision Blizzard, Inc.	technology
37	AVB	AvalonBay Communities, Inc.	real estate
38	AVY	Avery Dennison Corporation	manufacturing
39	AXP	American Express Company	finance
40	AZO	AutoZone, Inc.	trade
41	BA	Boeing Company	manufacturing
42	BAC	Bank of America Corp	finance

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
43	BAX	Baxter International Inc.	life sciences
44	BBWI.K	Bath & Body Works, Inc.	trade
45	BBY	Best Buy Co., Inc.	trade
46	BDX	Becton, Dickinson and Company	life sciences
47	BEN	Franklin Resources, Inc.	finance
48	BFb	Brown-Forman Corporation Class B	manufacturing
49	BIIB.O	Biogen Inc.	life sciences
50	BIO	Bio-Rad Laboratories, Inc. Class A	life sciences
51	BK	Bank of New York Mellon Corporation	finance
52	BKNG.O	Booking Holdings Inc.	energy
53	BKR.O	Baker Hughes Company Class A	technology
54	BLK	BlackRock, Inc.	finance
55	BLL	Ball Corporation	manufacturing
56	BMY	Bristol-Myers Squibb Company	life sciences
57	BRKb	Berkshire Hathaway Inc. Class B	finance
58	BRO	Brown & Brown, Inc.	finance
59	BSX	Boston Scientific Corporation	life sciences
60	BWA	BorgWarner Inc.	manufacturing
61	BXP	Boston Properties, Inc.	real estate
62	C	Citigroup Inc.	finance
63	CAG	Conagra Brands, Inc.	manufacturing
64	CAH	Cardinal Health, Inc.	trade
65	CAT	Caterpillar Inc.	technology
66	CB	Chubb Limited	finance
67	CCI	Crown Castle International Corp	real estate
68	CCL	Carnival Corporation	energy
69	CDNS.O	Cadence Design Systems, Inc.	technology
70	CERN.O	Cerner Corporation	technology
71	CHD	Church & Dwight Co., Inc.	life sciences
72	CHRW.O	C.H. Robinson Worldwide, Inc.	energy
73	CI	Cigna Corporation	finance
74	CINF.O	Cincinnati Financial Corporation	finance
75	CL	Colgate-Palmolive Company	life sciences
76	CLX	Clorox Company	life sciences
77	CMA	Comerica Incorporated	finance
78	CMCSA.O	Comcast Corporation Class A	technology
79	CMI	Cummins Inc.	technology
80	CMS	CMS Energy Corporation	energy
81	CNP	CenterPoint Energy, Inc.	energy
82	COF	Capital One Financial Corporation	finance
83	COO	Cooper Companies, Inc.	life sciences
84	COP	ConocoPhillips	energy
85	COST.O	Costco Wholesale Corporation	trade

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
86	CPB	Campbell Soup Company	manufacturing
87	CPRT.O	Copart, Inc.	trade
88	CRL	Charles River Laboratories International, Inc.	trade
89	CSCO.O	Cisco Systems, Inc.	technology
90	CSX.O	CSX Corporation	energy
91	CTAS.O	Cintas Corporation	manufacturing
92	CTRA.K	Coterra Energy Inc.	energy
93	CTSH.O	Cognizant Technology Solutions Corporation Class A	technology
94	CTXS.O	Citrix Systems, Inc.	technology
95	CVS	CVS Health Corporation	trade
96	CVX	Chevron Corporation	energy
97	D	Dominion Energy Inc	energy
98	DE	Deere & Company	technology
99	DGX	Quest Diagnostics Incorporated	life sciences
100	DHI	D.R. Horton, Inc.	real estate
101	DHR	Danaher Corporation	life sciences
102	DIS	Walt Disney Company	trade
103	DISH.O	DISH Network Corporation Class A	technology
104	DLTR.O	Dollar Tree, Inc.	trade
105	DOV	Dover Corporation	technology
106	DRE	Duke Realty Corporation	real estate
107	DRI	Darden Restaurants, Inc.	trade
108	DTE	DTE Energy Company	energy
109	DUK	Duke Energy Corporation	energy
110	DVA	DaVita Inc.	life sciences
111	DVN	Devon Energy Corporation	energy
112	DXC	DXC Technology Co.	technology
113	EA.O	Electronic Arts Inc.	technology
114	EBAY.O	eBay Inc.	trade
115	ECL	Ecolab Inc.	life sciences
116	ED	Consolidated Edison, Inc.	energy
117	EFX	Equifax Inc.	trade
118	EIX	Edison International	energy
119	EL	Estee Lauder Companies Inc. Class A	life sciences
120	EMN	Eastman Chemical Company	life sciences
121	EMR	Emerson Electric Co.	manufacturing
122	EOG	EOG Resources, Inc.	energy
123	EQR	Equity Residential	real estate
124	ES	Eversource Energy	energy
125	ESS	Essex Property Trust, Inc.	real estate
126	ETN	Eaton Corp. Plc	technology
127	ETR	Entergy Corporation	energy
128	EVRG.K	Evergy, Inc.	energy

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
129	EW	Edwards Lifesciences Corporation	life sciences
130	EXC.O	Exelon Corporation	energy
131	EXPD.O	Expeditors International of Washington, Inc.	energy
132	F	Ford Motor Company	manufacturing
133	FAST.O	Fastenal Company	trade
134	FCX	Freeport-McMoRan, Inc.	energy
135	FDX	FedEx Corporation	energy
136	FE	FirstEnergy Corp.	energy
137	FFIV.O	F5, Inc.	technology
138	FISV.O	Fiserv, Inc.	technology
139	FITB.O	Fifth Third Bancorp	finance
140	FMC	FMC Corporation	life sciences
141	FRT	Federal Realty Investment Trust	real estate
142	GD	General Dynamics Corporation	manufacturing
143	GE	General Electric Company	manufacturing
144	GILD.O	Gilead Sciences, Inc.	life sciences
145	GIS	General Mills, Inc.	manufacturing
146	GL	Globe Life Inc.	finance
147	GLW	Corning Inc	manufacturing
148	GPC	Genuine Parts Company	trade
149	GPN	Global Payments Inc.	trade
150	GPS	Gap, Inc.	trade
151	GRMN.K	Garmin Ltd.	manufacturing
152	GS	Goldman Sachs Group, Inc.	finance
153	GWW	W.W. Grainger, Inc.	trade
154	HAL	Halliburton Company	energy
155	HAS.O	Hasbro, Inc.	manufacturing
156	HBAN.O	Huntington Bancshares Incorporated	finance
157	HD	Home Depot, Inc.	trade
158	HES	Hess Corporation	energy
159	HIG	Hartford Financial Services Group, Inc.	finance
160	HOLX.O	Hologic, Inc.	life sciences
161	HON.O	Honeywell International Inc.	manufacturing
162	HPQ	HP Inc.	technology
163	HRL	Hormel Foods Corporation	manufacturing
164	HSIC.O	Henry Schein, Inc.	trade
165	HST.O	Host Hotels & Resorts, Inc.	real estate
166	HSY	Hershey Company	manufacturing
167	HUM	Humana Inc.	finance
168	HWM	Howmet Aerospace Inc.	manufacturing
169	IBM	International Business Machines Corporation	technology
170	IDXX.O	IDEXX Laboratories, Inc.	life sciences
171	IEX	IDEX Corporation	technology

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
172	IFF	International Flavors & Fragrances Inc.	life sciences
173	ILMN.O	Illumina, Inc.	life sciences
174	INCY.O	Incyte Corporation	trade
175	INTC.O	Intel Corporation	manufacturing
176	INTU.O	Intuit Inc.	technology
177	IP	International Paper Company	manufacturing
178	IPG	Interpublic Group of Companies, Inc.	trade
179	IRM	Iron Mountain, Inc.	real estate
180	ISRG.O	Intuitive Surgical, Inc.	life sciences
181	IT	Gartner, Inc.	trade
182	ITW	Illinois Tool Works Inc.	technology
183	IVZ	Invesco Ltd.	finance
184	J	Jacobs Engineering Group Inc.	real estate
185	JBHT.O	J.B. Hunt Transport Services, Inc.	energy
186	JCI	Johnson Controls International plc	trade
187	JKHY.O	Jack Henry & Associates, Inc.	technology
188	JNJ	Johnson & Johnson	life sciences
189	JNPR.K	Juniper Networks, Inc.	technology
190	JPM	JPMorgan Chase & Co.	finance
191	K	Kellogg Company	manufacturing
192	KEY	KeyCorp	finance
193	KIM	Kimco Realty Corporation	real estate
194	KLAC.O	KLA Corporation	life sciences
195	KMB	Kimberly-Clark Corporation	manufacturing
196	KMX	CarMax, Inc.	trade
197	KO	Coca-Cola Company	manufacturing
198	KR	Kroger Co.	trade
199	L	Loews Corporation	finance
200	LEG	Leggett & Platt, Incorporated	manufacturing
201	LEN	Lennar Corporation Class A	real estate
202	LH	Laboratory Corporation of America Holdings	life sciences
203	LHX	L3Harris Technologies Inc	manufacturing
204	LIN	Linde plc	life sciences
205	LLY	Eli Lilly and Company	life sciences
206	LMT	Lockheed Martin Corporation	manufacturing
207	LNC	Lincoln National Corporation	finance
208	LNT.O	Alliant Energy Corp	energy
209	LOW	Lowe's Companies, Inc.	trade
210	LRCX.O	Lam Research Corporation	technology
211	LUMN.K	Lumen Technologies, Inc.	technology
212	LUV	Southwest Airlines Co.	energy
213	MAA	Mid-America Apartment Communities, Inc.	real estate
214	MAR.O	Marriott International, Inc. Class A	real estate

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
215	MAS	Masco Corporation	manufacturing
216	MCD	McDonald's Corporation	trade
217	MCHP.O	Microchip Technology Incorporated	manufacturing
218	MCK	McKesson Corporation	trade
219	MCO	Moody's Corporation	trade
220	MDT	Medtronic Plc	life sciences
221	MET	MetLife, Inc.	finance
222	MGM	MGM Resorts International	real estate
223	MHK	Mohawk Industries, Inc.	manufacturing
224	MKC	McCormick & Company, Incorporated	manufacturing
225	MLM	Martin Marietta Materials, Inc.	energy
226	MMC	Marsh & McLennan Companies, Inc.	finance
227	MMM	3M Company	life sciences
228	MNST.O	Monster Beverage Corporation	manufacturing
229	MO	Altria Group Inc	manufacturing
230	MRK	Merck & Co., Inc.	life sciences
231	MRO	Marathon Oil Corporation	energy
232	MS	Morgan Stanley	finance
233	MSFT.O	Microsoft Corporation	technology
234	MSI	Motorola Solutions, Inc.	manufacturing
235	MTB	M&T Bank Corporation	finance
236	MTD	Mettler-Toledo International Inc.	life sciences
237	MU.O	Micron Technology, Inc.	manufacturing
238	NEE	NextEra Energy, Inc.	energy
239	NEM	Newmont Corporation	energy
240	NI	NiSource Inc	energy
241	NKE	NIKE, Inc. Class B	manufacturing
242	NLOK.O	NortonLifeLock Inc.	technology
243	NOC	Northrop Grumman Corporation	manufacturing
244	NSC	Norfolk Southern Corporation	energy
245	NTAP.O	NetApp, Inc.	technology
246	NTRS.O	Northern Trust Corporation	finance
247	NUE	Nucor Corporation	manufacturing
248	NVDA.O	NVIDIA Corporation	manufacturing
249	NVR	NVR, Inc.	real estate
250	NWL.O	Newell Brands Inc	manufacturing
251	O	Realty Income Corporation	real estate
252	ODFL.O	Old Dominion Freight Line, Inc.	energy
253	OKE	ONEOK, Inc.	energy
254	OMC	Omnicom Group Inc	trade
255	ORCL.K	Oracle Corporation	technology
256	ORLY.O	O'Reilly Automotive, Inc.	trade
257	OXY	Occidental Petroleum Corporation	energy

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
258	PARA.O	Paramount Global	technology
259	PAYX.O	Paychex, Inc.	trade
260	PBCT.O	People's United Financial, Inc.	finance
261	PCAR.O	PACCAR Inc	manufacturing
262	PEAK.K	Healthpeak Properties, Inc.	real estate
263	PEG	Public Service Enterprise Group Inc	energy
264	PENN.O	Penn National Gaming, Inc.	real estate
265	PEP.O	PepsiCo, Inc.	manufacturing
266	PFE	Pfizer Inc.	life sciences
267	PG	Procter & Gamble Company	life sciences
268	PGR	Progressive Corporation	finance
269	PH	Parker-Hannifin Corporation	manufacturing
270	PHM	PulteGroup, Inc.	real estate
271	PKG	Packaging Corporation of America	manufacturing
272	PKI	PerkinElmer, Inc.	life sciences
273	PLD	Prologis, Inc.	real estate
274	PNC	PNC Financial Services Group, Inc.	finance
275	PNR	Pentair plc	technology
276	PNW	Pinnacle West Capital Corporation	energy
277	POOL.O	Pool Corporation	trade
278	PPG	PPG Industries, Inc.	life sciences
279	PPL	PPL Corporation	energy
280	PSA	Public Storage	real estate
281	PTC.O	PTC Inc.	technology
282	PVH	PVH Corp.	manufacturing
283	PWR	Quanta Services, Inc.	real estate
284	PXD	Pioneer Natural Resources Company	energy
285	QCOM.O	Qualcomm Inc	manufacturing
286	QRVO.O	Qorvo, Inc.	manufacturing
287	RCL	Royal Caribbean Group	energy
288	RE	Everest Re Group, Ltd.	finance
289	REG.O	Regency Centers Corporation	real estate
290	REGN.O	Regeneron Pharmaceuticals, Inc.	life sciences
291	RF	Regions Financial Corporation	finance
292	RHI	Robert Half International Inc.	trade
293	RJF	Raymond James Financial, Inc.	finance
294	RL	Ralph Lauren Corporation Class A	manufacturing
295	RMD	ResMed Inc.	life sciences
296	ROK	Rockwell Automation, Inc.	life sciences
297	ROL	Rollins, Inc.	trade
298	ROP	Roper Technologies, Inc.	life sciences
299	ROST.O	Ross Stores, Inc.	trade
300	RSG	Republic Services, Inc.	energy

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
301	RTX	Raytheon Technologies Corporation	manufacturing
302	SBAC.O	SBA Communications Corp. Class A	real estate
303	SBUX.O	Starbucks Corporation	trade
304	SCHW.K	Charles Schwab Corporation	finance
305	SEE	Sealed Air Corporation	life sciences
306	SHW	Sherwin-Williams Company	trade
307	SIVB.O	SVB Financial Group	finance
308	SLB	Schlumberger NV	energy
309	SNA	Snap-on Incorporated	manufacturing
310	SNPS.O	Synopsys, Inc.	technology
311	SO	Southern Company	energy
312	SPG	Simon Property Group, Inc.	real estate
313	SPGI.K	S&P Global, Inc.	trade
314	SRE	Sempra Energy	energy
315	STE	STERIS Plc	life sciences
316	STT	State Street Corporation	finance
317	STZ	Constellation Brands, Inc. Class A	manufacturing
318	SWK	Stanley Black & Decker, Inc.	manufacturing
319	SWKS.O	Skyworks Solutions, Inc.	manufacturing
320	SYK	Stryker Corporation	life sciences
321	SYY	Sysco Corporation	trade
322	T	AT&T Inc.	technology
323	TAP	Molson Coors Beverage Company Class B	manufacturing
324	TDY	Teledyne Technologies Incorporated	manufacturing
325	TECH.O	Bio-Techne Corporation	life sciences
326	TER.O	Teradyne, Inc.	life sciences
327	TFC	Trust Financial Corporation	finance
328	TFX	Teleflex Incorporated	life sciences
329	TGT	Target Corporation	trade
330	TJX	TJX Companies Inc	trade
331	TMO	Thermo Fisher Scientific Inc.	life sciences
332	TPR	Tapestry, Inc.	manufacturing
333	TRMB.O	Trimble Inc.	life sciences
334	TROW.O	T. Rowe Price Group	finance
335	TRV	Travelers Companies, Inc.	finance
336	TSCO.O	Tractor Supply Company	trade
337	TSN	Tyson Foods, Inc. Class A	manufacturing
338	TT	Trane Technologies plc	life sciences
339	TTWO.O	Take-Two Interactive Software, Inc.	technology
340	TXN.O	Texas Instruments Incorporated	manufacturing
341	TXT	Textron Inc.	manufacturing
342	TYL	Tyler Technologies, Inc.	technology
343	UDR	UDR, Inc.	real estate

Continued on next page

Table 3.3 – continued from previous page

	ric	description	sector
344	UHS	Universal Health Services, Inc. Class B	life sciences
345	UNH	UnitedHealth Group Incorporated	finance
346	UNP	Union Pacific Corporation	energy
347	UPS	United Parcel Service, Inc. Class B	energy
348	URI	United Rentals, Inc.	trade
349	USB	U.S. Bancorp	finance
350	VFC	V.F. Corporation	manufacturing
351	VLO	Valero Energy Corporation	energy
352	VMC	Vulcan Materials Company	energy
353	VNO	Vornado Realty Trust	real estate
354	VRSN.O	VeriSign, Inc.	technology
355	VRTX.O	Vertex Pharmaceuticals Incorporated	life sciences
356	VTR	Ventas, Inc.	real estate
357	VTRS.O	Viatris, Inc.	life sciences
358	VZ	Verizon Communications Inc.	technology
359	WAB	Westinghouse Air Brake Technologies Corporation	manufacturing
360	WAT	Waters Corporation	life sciences
361	WBA.O	Walgreens Boots Alliance Inc	trade
362	WDC.O	Western Digital Corporation	technology
363	WEC	WEC Energy Group Inc	energy
364	WELL.K	Welltower, Inc.	real estate
365	WFC	Wells Fargo & Company	finance
366	WHR	Whirlpool Corporation	manufacturing
367	WM	Waste Management, Inc.	energy
368	WMB	Williams Companies, Inc.	energy
369	WMT	Walmart Inc.	trade
370	WRB	W. R. Berkley Corporation	finance
371	WST	West Pharmaceutical Services, Inc.	manufacturing
372	WY	Weyerhaeuser Company	real estate
373	XEL.O	Xcel Energy Inc.	energy
374	XOM	Exxon Mobil Corporation	energy
375	XRAY.O	Dentsply Sirona, Inc.	life sciences
376	YUM	Yum! Brands, Inc.	trade
377	ZBRA.O	Zebra Technologies Corporation Class A	technology
378	ZION.O	Zions Bancorporation, N.A.	finance

Chapter 4

Online learning with radial basis function networks

We provide multi-horizon forecasts on the returns of financial time series. Our sequentially optimised radial basis function network (RBFNet) outperforms a random-walk baseline and several powerful supervised learners. Our RBFNets naturally measure the similarity between test samples and prototypes that capture the characteristics of the feature space. We show that the training set financial time series returns have low similarity with their test set counterparts, highlighting the challenges faced in particular by kernel-based methods that use the training set returns as test-time prototypes; in contrast, our online learning RBFNets have hidden units that retain greater similarity across time.

This chapter shows the benefits of feature selection and online learning with nonlinear models. We limit the scope of our experimentation to financial time series, which at times exhibit high autocorrelation, nonstationarity, nonlinearity and regime-switching characteristics. This behaviour can be classified as concept drift (Iwashita and Papa, 2019). We find that our online RBFNet obtains the best test set results with minimum NMSE. The multi-layer perceptron (mlp) performs worst. If we compare the local learning of the RBFNet with the global learning technique of the mlp, the latter suffers from catastrophic forgetting (Kirkpatrick et al., 2017; Sukhov et al., 2020). The RBFNets we formulate are naturally designed to measure the similarity between test samples and continuously updated prototypes that capture the characteristics of the feature space. As such, the models are robust in mitigating catastrophic forgetting. In addition, although related to k-nearest neighbours, Gaussian process and kernel ridge regression, our experiments show that the RBFNets, which use clustering algorithms to determine the network's hidden units, are more predictive than using each training vector as test-time prototypes. Section 4.4 demonstrates this visually, with

plots of the cosine similarity between training and test vectors. The original returns space has low similarity, whilst the clustered returns space has high similarity; thus, more signal is extracted from the data.

4.1 Problem formulation

Using the multi-asset dataset 3.1, we consider the problem of forecasting multi-horizon returns $h = 1, .., 30$ days ahead for each asset in the dataset. Multi-horizon forecasting is important in a number of circumstances. For example, if one seeks to purchase a portfolio of risky assets, one will most likely be unsure when the risky positions will be liquidated. Liquidation might occur due to profit taking. Therefore, having a prediction model that works well across multiple forecast horizons is invaluable. Denote the target asset indices as $j = 1, ...d$. In the general regression setting, we wish to learn the mapping of a set of predictors $\mathbf{x}_t$ to an individual horizon/target tuple $y_{t+h,j} = f(\mathbf{x}_t; \boldsymbol{\theta}_t) + \varepsilon_{t,j}$. In this experiment, the predictors $\mathbf{x}_t$ are daily returns for each asset's end of day price

$$\mathbf{x}_t = [\log(p_{t,1}/p_{t-1,1}), ..., \log(p_{t,d}/p_{t-1,d})].$$

The end of day price could be the closing transaction price, settlement price or last quote of the day, depending on what the data vendor Refinitiv provides us with. Instead of a fixed set of predictors, we consider a tailored set of predictors $\mathbf{x}(j)$ for each target y_j, which is determined by feature selection algorithm 4.1 that combines forward stepwise selection with variance inflation factor minimisation. So as to unclutter our notation, assume now that

$$\mathbf{x}_t \triangleq \mathbf{x}_t(j) \quad \forall j = 1, ..., d \quad \mathbf{x} \in \mathbb{R}^q \quad 1 \leq q \leq d.$$

Furthermore, in the case of online learning RBFNet algorithm 4.2, there is an added transformation of the original external input space. We perform online feature representation transfer

from the external inputs $\mathbf{x}_t$ to a new feature space

$$\begin{aligned}\phi_k(\mathbf{x}_t) &= \exp\left(-\frac{1}{2}[\mathbf{x}_t - \boldsymbol{\mu}_{k,t}]^T \boldsymbol{\Lambda}_{k,t}[\mathbf{x}_t - \boldsymbol{\mu}_{k,t}]\right), \quad \forall k = 1,...,K \\ \boldsymbol{\phi}_t &= [1, \phi_1(\mathbf{x}_t), ..., \phi_K(\mathbf{x}_t)]^T, \\ \boldsymbol{\phi}_t &\triangleq \boldsymbol{\phi}_t(j) \quad \forall j = 1,...,d,\end{aligned}$$

via k-means, where $\boldsymbol{\phi}_t$ denotes the hidden units of a single-layer RBFNet. Each $\phi_k(\mathbf{x}_t)$ is parameterised as a k-means cluster conditional multivariate normal $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$. As determination of $\mathbf{x}(j)$ depends on target y_j, there is one RBFNet per target.

4.2 The research experiment

We consider the goal of multi-step forecasting with financial time series. Define the prediction mean squared error (MSE) for the $j'th$ model and $h'th$ forecast horizon as

$$MSE_{h,j} = \frac{1}{t-h}\sum_{i=1}^{t-h}(y_{i+h,j} - \hat{y}_{i,j})^2.$$

There are numerous financial examples where participants seek to minimise MSE over optimal horizons h^* that are not known in advance. For example, a market maker captures *edge*, which is half the bid/ask spread, and only realises a profit when the risk is turned over (going from long to short or vice versa) or flattened. This risk turnover is entirely variable. In the context of execution algos, the algo seeks to minimise the implementation shortfall relative to a benchmark. Even if the execution time is known in advance, such as with time-weighted average price algos, the performance of the algo relative to the benchmark is unknown a priori. A final example is systematic proprietary trading strategies, which rely on statistically driven signals to scale in and out of risk over varying time scales. A prediction model that, on average, performs well over multiple forecast horizons is of great value in all the aforementioned cases.

In our experiment, which uses the returns of the cross-asset Refinitiv dataset introduced in section 3.1, we determine if the RBFNet can provide optimal forecasts over multiple horizons. Optimality is quantified here as the relative performance of the RBFNet to a bench-

mark, which is the random-walk model. The random-walk model is introduced next, in section 4.2.1. Our RBFNet differs from earlier work in that its unsupervised and supervised learning components continue to learn in the test set. Specifically, we facilitate online learning for the k-means++ algorithm that learns the network's hidden units and maps these hidden units to the response using EWRLS. A complete algorithm is shown in section 4.2.3 with algorithm 4.2.

The choice of returns is made for several reasons. Firstly, by constructing returns, all the time series are considered stationary by unit root tests such as the augmented Dickey-Fuller (ADF) test (Said and Dickey, 1984). Secondly, the introduction of returns makes the choice of the random-walk as the baseline model most suitable. Finally, we use several comparator models discussed in section 4.2.4 that rely on the iid random variables assumption. These competitors are traditional batch-learning models and cannot be fitted practically sequentially in several cases. For example, the gradient tree boosters and random forests use underlying trees that are grown using the cart algorithm (Breiman et al., 1984). For regression trees, the cart algorithm will split the training set by feature indices and values as necessary to reduce the total MSE in the tree. Therefore, applying the algorithm at test time to the original training dataset augmented by new test entries is not computationally feasible. In addition, some models use an L-BFGS solver (Liu and Nocedal, 1989) for parameter optimisation, which is purely a batch method. Aside from this, to demonstrate that the results we get with the RBFNet are not purely down to sequential optimisation in the test set, we use a second online learning model, the EWRLS.

4.2.1 The random-walk model

The random-walk model (Harvey, 1993)

$$y_t = y_{t-1} + \varepsilon_t,$$

has stationary expectation

$$\mathbb{E}[y_t] = y_0,$$

yet nonstationary variance and covariance

$$Var[y_t] = t\sigma^2$$
$$Cov[y_t, y_{t-\tau}] = |t-\tau|\sigma^2.$$

A large body of academic literature shows that it is difficult to beat the random-walk model when forecasting returns of financial time series (Meese and Rogoff, 1983; Engel, 1994). Bachelier (1900) considers price series as Gaussian random walks, whose increments are iid Gaussian random variables. Bachelier's first law states that the variation of returns grows with the square root of time. Bouchaud et al. (2018) find that Bachelier's first law holds well for actual financial returns; however, they also find that standard Gaussian random-walk models for financial returns modelling underestimate the extreme fluctuations that are empirically observed. Finally, they find that price changes follow fat-tailed, power-law distributions, with extreme events not as rare as Gaussian models might predict.

4.2.2 Feature selection

In our experiment, we have one hundred assets to choose from in the Refinitiv cross-asset dataset (section 3.1) as external inputs to the models we use. However, there is much redundancy in this external input space. For example, figure 4.1 visually shows the training set log-return correlations, with an overwhelming red colour indicating a positive correlation. Furthermore, table 4.1 shows the distribution of the off-diagonal correlation values, which averages just under 6% and has a maximum value of almost 100%. Correlated features are likely to result in a more significant prediction variance. Denote the singular value decomposition (Jolliffe, 2011) of the training set features as

$$\mathbf{X} = \mathbf{USV}^T,$$

where $\mathbf{S}$ is the diagonal matrix of eigenvalues. The variance of the least squares parameters is:

$$Var[\boldsymbol{\theta}] = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} = \sigma^2\mathbf{VS}^{-2}\mathbf{V}^T.$$

Highly correlated features in $\mathbf{X}$ cause $\mathbf{S}^{-2}$ to be large, which increases $Var[\boldsymbol{\theta}]$.

Table 4.1: Summary statistics for the training set returns correlations for the Refinitiv cross-asset dataset (section 3.1). There are substantial positive correlations, which motivate using an external input feature selection algorithm.

	count	mean	std	min	25%	50%	75%	max
ρ	9900	0.056	0.276	-0.82	-0.12	0.018	0.221	0.996

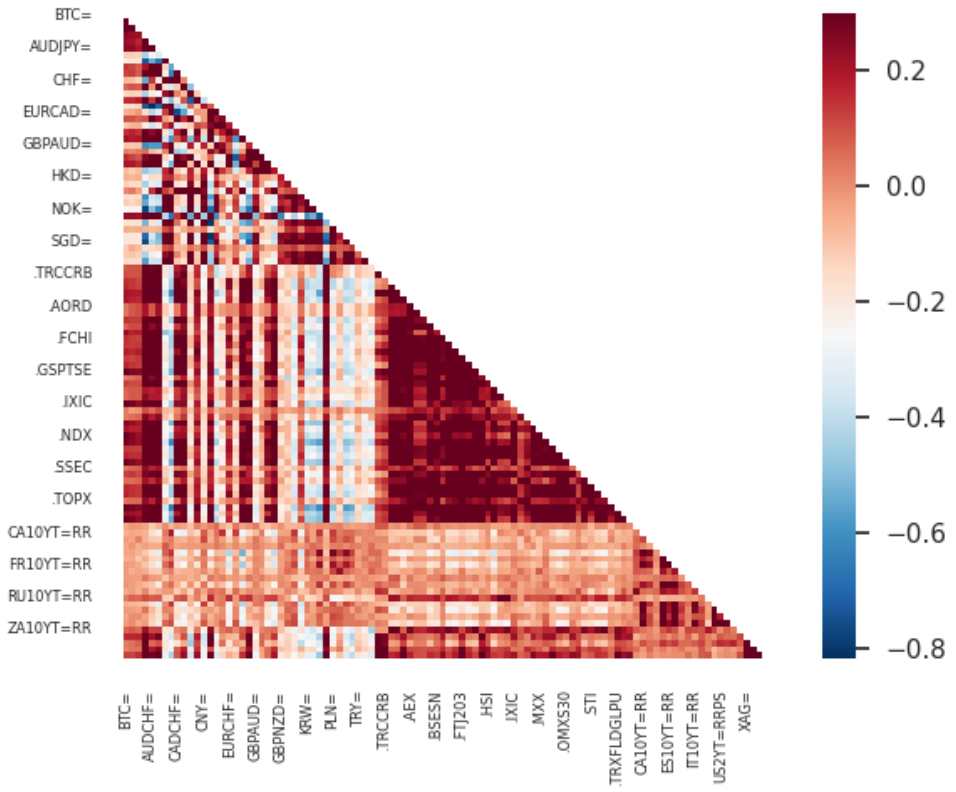


Figure 4.1: We show a heatmap of the training set returns correlations for the Refinitiv cross-asset dataset. Across currency pairs, equities, rates, credit, metals, agriculture, energy and crypto, there is a bias toward positive correlations, which increases systemic risk in the financial markets, particularly during periods of turmoil.

There are various feature selection algorithms, which Hastie et al. (2009) discuss in detail. Forward stepwise selection scales well as the dimensionality d of the feature space $\mathbf{X} \in \mathbb{R}^{n \times d}$ increases. The goal of forward stepwise selection is to choose features that maximise R^2, although the features might be positively correlated. On the other hand, variance inflation factor (VIF) minimisation (James et al., 2013) performs feature selection by minimising the correlation between features. As shown in exhibit 4.1, we combine forward stepwise selection and VIF minimisation to the training set returns. The algorithm is applied to each target, and the target-conditional subset of external inputs is held fixed during the test set evaluation. Therefore, in algorithm 4.1, we novelly combine forward stepwise selection and VIF minimisation to the training set returns. The algorithm is applied to each target, and the target-conditional subset of external inputs is held fixed during test set evaluation. An example of the algorithm's output (to two decimal places) is shown in table 4.2. The target is the training set one-step-ahead daily returns of the EURUSD currency pair. We see that the $R^2 = 0.15$, with the features ranked in order of their contribution to total R^2. Other statistics shown are the regression parameter t-values, which are the estimated regression coefficients divided by their standard errors. The associated p-values can be compared against the 5% critical value and standard frequentist statistics hypotheses of statistical significance inferred. Other statistics included are the normalised MSE (NMSE) which is the regression MSE divided by the variance of the response, which shows predictive quality in that the NMSE is less than 1. Furthermore, the Durbin-Watson (DW) statistic (Durbin and Watson, 1950) for serial autocorrelation is around 2, indicating no serial correlation in the regression residuals. Finally, the ADF statistic shows that the regression residuals are stationary.

4.2.3 The online learning radial basis function network

In section 2.3, we discuss the earlier research into RBFNets. Notably, this earlier work is cast within a batch learning framework, which is unsatisfactory for learning with nonstationary time series or time series that regularly experience concept drifts (Iwashita and Papa, 2019). There is earlier research into online learning RBFNets; for example, Calliess (2019) employs greedy projection (Zinkevich, 2003) to fit his RBFNet sequentially, a no-regret algorithm designed for online convex programming. Whilst Calliess keeps the parameters of the hidden units fixed to scalar values, perhaps for illustration purposes, our online learning

Table 4.2: Forward stepwise selection and VIF minimisation applied to the EURUSD spot FX returns in the training set. Features with a maximum VIF factor $\kappa \leq 5$ are accepted. The contribution to total R^2 is displayed in the final column.

EUR=	value	t-value	crit-value	p-value	VIF	R^2
R^2	0.15					
n	648					
σ^2	0.00					
MSE	0.00					
NMSE	0.79					
DW	2.02		0.05	0.02		
ADF	-25.7		-2.9	0.00		
NOK=	-0.08	-2.18	0.05	0.01	4.62	0.012
.MID	0.04	2.80	0.05	0.00	4.29	0.011
CAD=	-0.09	-1.69	0.05	0.04	2.43	0.010
ITEXO5Y=MG	0.01	0.26	0.05	0.39	4.04	0.009
.FTJ203	0.01	0.77	0.05	0.22	3.46	0.009
NZD=	-0.01	-0.23	0.05	0.40	2.75	0.008
.BSESN	0.03	1.86	0.05	0.03	2.26	0.008
.BVSP	-0.01	-0.76	0.05	0.22	3.07	0.007
XPD=	0.01	1.24	0.05	0.10	1.94	0.006
SEK=	0.02	0.50	0.05	0.30	3.59	0.006
.MXX	0.01	0.68	0.05	0.24	2.12	0.005
PLN=	-0.02	-0.55	0.05	0.28	3.11	0.005
MXN=	0.03	1.06	0.05	0.14	3.12	0.005
XPT=	-0.00	-0.49	0.05	0.30	2.60	0.004
GBP=	-0.00	-0.25	0.05	0.40	2.01	0.004
.TRCCRB	-0.01	-0.86	0.05	0.19	2.03	0.003
.AXJO	-0.00	-0.50	0.05	0.30	2.06	0.003
BTC=	0.01	1.99	0.05	0.02	2.95	0.003
XAG=	-0.01	-0.77	0.05	0.22	3.61	0.003
RU10YT=RR	-0.03	-1.04	0.05	0.14	1.56	0.003
CHFJPY=	0.08	1.90	0.05	0.02	1.55	0.003
BRL=	-0.04	-2.14	0.05	0.01	1.75	0.002
.STI	-0.02	-0.95	0.05	0.16	2.96	0.002
XAU=	0.03	1.30	0.05	0.09	3.26	0.002
ETH=	0.00	0.32	0.05	0.37	3.38	0.002
IT10YT=RR	-0.01	-0.73	0.05	0.23	1.80	0.002
IN10YT=RR	0.09	2.41	0.05	0.00	1.22	0.002
ES10YT=RR	0.02	0.59	0.05	0.27	2.02	0.002
.KLSE	0.02	0.83	0.05	0.20	2.27	0.002

Algorithm 4.1 Forward stepwise selection with variance inflation factor minimisation.

Require: κ `// the maximum VIF`
Initialise: $\mathbb{S} = [1, 2, ..., d]$, $\mathbf{r} = \mathbf{v} = \mathbf{0}_d$
Input: $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{y} \in \mathbb{R}^n$
Output: $\mathbb{S} \in \mathbb{Z}^p, \quad 0 < p \leq d$

for $j \leftarrow 1$ **to** d **do**
  $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}_i$
  $\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_{ij}$
  $\theta_j = \frac{\sum_{i=1}^{n} (\mathbf{y}_i - \bar{y})(\mathbf{X}_{ij} - \bar{x}_j)}{\sum_{i=1}^{n} (\mathbf{X}_{ij} - \bar{x}_j)^2}$
  `//` $\mathbf{r}_j$ `is the` R^2 `of a regression of` $\mathbf{y}$ `on` $\mathbf{x}_j$.
  $\mathbf{r}_j = R^2_{\mathbf{y}|\mathbf{x}_j} = 1 - \frac{\sum_{i=1}^{n} (\mathbf{y}_i - \theta_j \mathbf{X}_{ij})^2}{\sum_{i=1}^{n} (\mathbf{y}_i - \bar{y})^2}$
  `//` $R^2_{\mathbf{x}_j|\mathbf{X}_{-j}}$ `denotes the` R^2 `of a regression of` $\mathbf{x}_j$ `onto the remaining predictors, excluding the` $j'th$ `one.`
  $\mathbf{v}_j = \frac{1}{1 - R^2_{\mathbf{x}_j|\mathbf{X}_{-j}}}$
end
Sort $\mathbf{r}$ in ascending order and use the index to sort $\mathbb{S}$.
while $\forall \mathbf{v} >= \kappa$ **do**
  for $j \leftarrow 1$ **to** d **do**
    if $\mathbf{v}_j >= \kappa$ **then**
      Remove $\mathbb{S}_j$.
      Recompute $\mathbf{v}$.
    end
  end
end

RBFNet shown in algorithm 4.2 partially adopts the approach of Moody and Darken (1989). In the training set, the algorithm uses k-means++ to learn the hidden unit means and then maps the hidden unit output to the response using ridge regression. An innovation we make is that whilst the earlier work uses a randomised, scalar standard deviation σ_j in the RBF equation 2.3, we apply a Bayesian maximum a posteriori (MAP) estimate to the covariance matrices that we use instead. If many training data points are assigned to the $j'th$ cluster, the $j'th$ covariance matrix will resemble the maximum likelihood estimate. In contrast, if few data points are assigned to the $j'th$ cluster, the $j'th$ covariance matrix will resemble the diagonalised variance prior. A further innovation is the online updating of the hidden unit means and covariances, with exponential decay to allow for regime changes or concept drifts. More concretely, we operate with and adapt the precision (inverse covariance) matrices; this leads to a test-time time-complexity of $\mathcal{O}(kd^2)$, a reduction from $\mathcal{O}(kd^3)$ when operating on co-

variance matrices. Finally, we use exponentially weighted recursive least squares (EWRLS) to map the hidden unit outputs to the response; EWRLS operates efficiently with a precision matrix of the hidden unit space.

Algorithm 4.2 The online learning radial basis function network.

Require: k // hidden unit count, α // RR penalty, τ // exponential decay
Initialise: $v_0 = d+2$, $\boldsymbol{\theta} = \mathbf{0}_{k+1}$, $\mathbf{P} = \mathbf{I}_{k+1}/\alpha$
Hidden unit parameterisation via k-means++:
`// This section uses the training set feature matrix` $\mathbf{X} \in \mathbb{R}^{n \times d} = \{\mathbf{x}_i^T\}_{i=1}^n$
Initialise the hidden unit means $\{\boldsymbol{\mu}_j\}_{j=1}^k$.
repeat
  Assign the feature training vector to the nearest hidden unit mean $\delta_{j,i} = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2, \quad \delta_{j,i} = 1 \wedge \delta_{k,i} = 0 \quad \forall k \neq j.$
  Update each hidden unit mean using all the points assigned to it, $\boldsymbol{\mu}_j = \frac{1}{n_j}\sum_{i=1}^n \delta_{j,i}\mathbf{x}_i$.
until *until convergence*;
`// Learn the hidden unit covariances via Bayesian MAP estimation`
Estimate the prior scatter matrix $\mathbf{S}_0 = \frac{1}{k^{1/d}} diag(s_1^2, ..., s_d^2)$ where $s_j = (1/n)\sum_{i=1}^n (x_i - \bar{x}_j)^2$.
Estimate the $j'th$ likelihood scatter matrix $\mathbf{S}_j = \sum_{i=1}^n \delta_{j,i}(\mathbf{x}_i - \bar{\mathbf{x}}_j)(\mathbf{x}_i - \bar{\mathbf{x}}_j)^T$.
The $j'th$ posterior cov is $\boldsymbol{\Sigma}_j = \frac{\mathbf{S}_0 + \mathbf{S}_j}{v_0 + n_j + d + 2}$, where $n_j = \sum_i \delta_{j,i}$, with $\boldsymbol{\Lambda}_j = \boldsymbol{\Sigma}_j^{-1}$.
Output: $\hat{y}_t$
`// This is the online update, with test-time data` $\{\mathbf{x}_t \in \mathbb{R}^d, \ y_t\}$.
$\delta_{j,t} = \arg\min_j \|\mathbf{x}_t - \boldsymbol{\mu}_j\|_2^2$
$\boldsymbol{\mu}_{j,t} = \tau\boldsymbol{\mu}_{j,t-1} + (1-\tau)\mathbf{x}_t$
$a_t = 1 + (\mathbf{x}_t - \boldsymbol{\mu}_{j,t})^T \boldsymbol{\Lambda}_{j,t-1}(\mathbf{x}_t - \boldsymbol{\mu}_{j,t})/\tau$
$\mathbf{k}_t = \boldsymbol{\Lambda}_{j,t-1}(\mathbf{x}_t - \boldsymbol{\mu}_{j,t})/(\tau a_t)$
$\boldsymbol{\Lambda}_{j,t} = \boldsymbol{\Lambda}_{j,t-1}/\tau - \mathbf{k}_t\mathbf{k}_t^T a_t$
`// Map the hidden units to the response using EWRLS.`
$\phi_h(\mathbf{x}_t) = \exp\left(-\frac{1}{2}[\mathbf{x}_t - \boldsymbol{\mu}_{h,t}]^T \boldsymbol{\Lambda}_{h,t}[\mathbf{x}_t - \boldsymbol{\mu}_{h,t}]\right), \quad \forall h = 1, ..., k$
$\boldsymbol{\phi}_t = [1, \phi_1(\mathbf{x}_t), ..., \phi_k(\mathbf{x}_t)]^T$
$b_t = 1 + \boldsymbol{\phi}_{t-1}^T \mathbf{P}_{t-1}\boldsymbol{\phi}_{t-1}/\tau$
$\mathbf{m}_t = \mathbf{P}_{t-1}\boldsymbol{\phi}_{t-1}/(b_t\tau)$
$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{m}_t(y_t - \boldsymbol{\theta}_{t-1}^T\boldsymbol{\phi}_{t-1})$
$\mathbf{P}_t = \mathbf{P}_{t-1}/\tau - \mathbf{m}_t\mathbf{m}_t^T b_t$
$\hat{y}_t = \boldsymbol{\theta}_t^T\boldsymbol{\phi}_t$

4.2.4 Competitor models

The models we consider in our experiment are shown next. We use scikit-learn (Pedregosa et al., 2011) implementations for Gaussian process regression, gradient tree boosting, k-nearest neighbours regression, the multi-layer perceptron, the random forest and support vector regression. We use our implementations of ridge regression, kernel ridge regression,

EWRLS and the RBFNet. The online RBFNet faces a robust assortment of competitor models, which in most cases are suited to or can only be fitted by batch learning.

- **random-walk** - this is the baseline model that is discussed in section 4.2.1.
- **GPR** - the Gaussian process regression model (Rasmussen and Williams, 2005). We combine an RBF kernel with a white noise kernel. Scikit-learn uses an L-BFGS solver by default.
- **GTB** - the gradient tree boosting regression model (Friedman, 2001), with a default of one hundred estimators, a maximum tree depth of 3 and MSE as splitting criteria.
- **KRR** - kernel ridge regression (Cristianini and Shawe-Taylor, 2000).
- **KNNR** - k-nearest neighbours regression (Takezawa, 2006), with a default 5 nearest neighbours and Minkowski distance metric (Du, 2018). When computing the nearest neighbours, the scikit-learn implementation automatically decides between ball-tree and kd-tree algorithms (Munaga and Jarugumalli, 2012).
- **MLP** - multi-layer perceptron regression (Goodfellow et al., 2016). Note that we use the L-BFGS solver, which converges faster and with better solutions on small datasets. We use the default structure of a single layer comprised of one hundred hidden units that use the relu activation.
- **RF** - random forests of regression trees (Breiman, 2001), with a default one hundred estimators. Each tree in the forest is grown to an unrestricted depth and pruned back using minimal cost complexity pruning (Breiman et al., 1984).
- **RR** - the ridge regression model (Hoerl, 1962).
- **SVR** - the support vector regression model, specifically the ν-SVR, where ν controls the number of support vectors (Schölkopf et al., 2000, 2001). This scikit-learn implementation uses the RBF kernel by default.
- **EWRLS** - exponentially weighted recursive least squares, discussed in section 4.2.3.

- **RBFNet** - the online learning radial basis function network detailed in algorithm 4.2. We set $k = 100$ hidden units.

4.2.5 Experiment design

As discussed in section 3.1, we construct daily returns and set half the data aside for training and the other half for testing. We apply the external input selection algorithm 4.1 to the training set returns and end up with a subset of external inputs per target. These are held fixed and used as is in the test set. We set the maximum VIF $\kappa = 5$. For the various models that use RR penalties, we use a default value of $\alpha = 0.0001$. Both the EWRLS and RBFNets use an exponential decay factor $\tau = 0.99$. During test time, these two models continue to be fitted online. The performance criteria that we consider is normalised prediction mean squared error (NMSE) for forecast horizons in days $h = 1, ..., 30$

$$NMSE_{h,j} = \frac{\sum_{i=1}^{t-h}(y_{i+h,j} - \hat{y}_{i,j})^2}{\sum_{i=1}^{t-h}(y_{i+h,j} - y_{0,j})^2}. \tag{4.1}$$

In equation 4.1, the normalisation of MSE occurs relative to the random-walk baseline.

4.3 Results

Table 4.3 shows that several models have average test set NMSE that is better than the random-walk baseline. These include EWRLS, GPR, GTB, KRR, RBFNet and RF. The models that perform worse than the random-walk baseline include KNNR, MLP, RR and SVR, with the MLP the worst performing model. The RBFNet has the lowest average NMSE of 0.636. Comparing this to the second-best result, a NMSE of 0.673 for GPR, we perform a two-sample t-test for equal means (Hirotsu, 2017) and find that the means are considered statistically different, drawn from differently parameterised distributions. For all models, we also perform a Wald test (Wasserman, 2004) for the null hypothesis that the nmspe is no different from 1, tested at the 5% critical value. We find that in all cases, the model-averaged NMSE is statistically different from 1. Figure 4.2 shows the NMSE by model and forecast horizon, as well as NMSE box plots by model. There is a similar performance between the RBFNet and GPR for $h = 1$. For $h = 2, ..., 30$, the RBFNet outperforms GPR, the random-walk baseline and the remaining competitor models. We cannot put the RBFNet

outperformance down to sequential updating alone in the test set; if this were the case, the EWRLS model would outperform the remaining offline learning models. Instead, EWRLS performs worse than GPR, RF, GTB and KRR, all offline learning models.

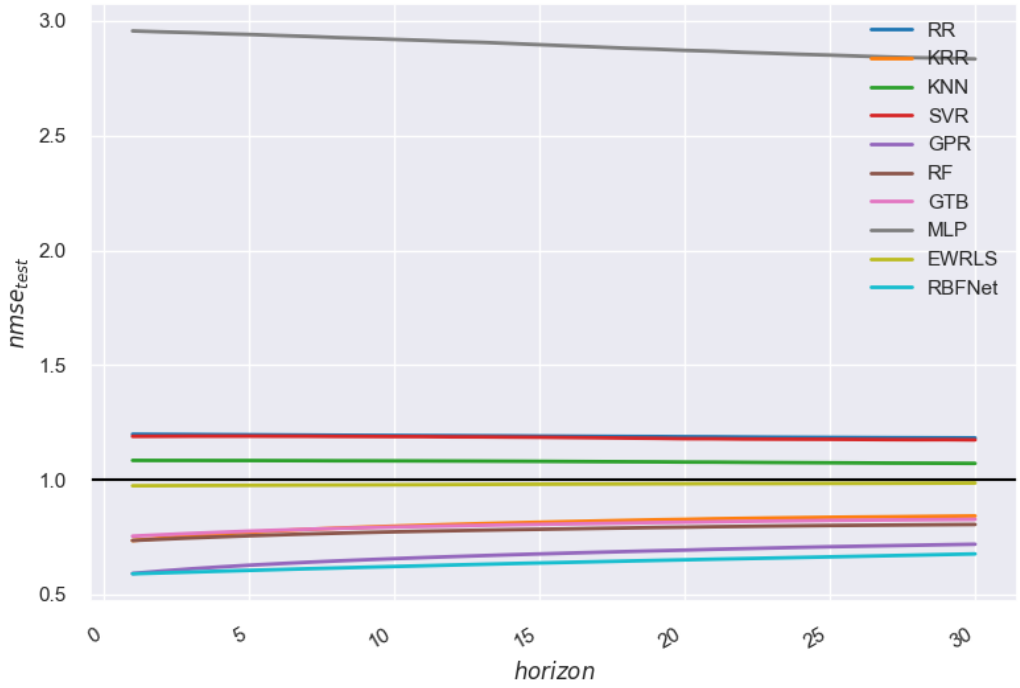
Table 4.3: The models perform multi-step returns forecasting on the Refinitiv cross-asset dataset with horizons from $h = 1, ..., 30$ days ahead. We show the distribution of the test set normalised mean squared error (NMSE) for each model across the multiple forecast horizons. The RBFNet achieves the lowest NMSE.

model	EWRLS	GPR	GTB	KRR	KNNR	MLP	RBFNet	RF	RR	SVR
targets	100	100	100	100	100	100	100	100	100	100
count	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000
mean	0.976	0.673	0.781	0.797	1.068	2.709	**0.636**	0.763	1.221	1.197
std	0.523	0.561	0.894	0.929	0.914	4.188	0.415	0.821	0.843	1.613
min	0.266	0.037	0.041	0.039	0.134	0.081	0.141	0.035	0.476	0.123
25%	0.608	0.315	0.312	0.302	0.423	0.712	0.360	0.322	0.830	0.434
50%	0.855	0.482	0.507	0.547	0.809	1.314	0.514	0.499	1.020	0.682
75%	1.146	0.845	0.900	0.869	1.327	3.064	0.795	0.868	1.359	1.190
max	3.395	3.374	5.703	5.338	5.006	25.7	2.46	5.734	8.332	11.9
se	0.010	0.010	0.016	0.017	0.017	0.076	0.008	0.015	0.015	0.029
t-value	-2.49	-31.9	-13.4	-11.9	4.06	22.3	-48.1	-15.8	14.3	6.70
crit-value	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
p-value	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
reject $H_0 : \mu = 1$	1	1	1	1	1	1	1	1	1	1

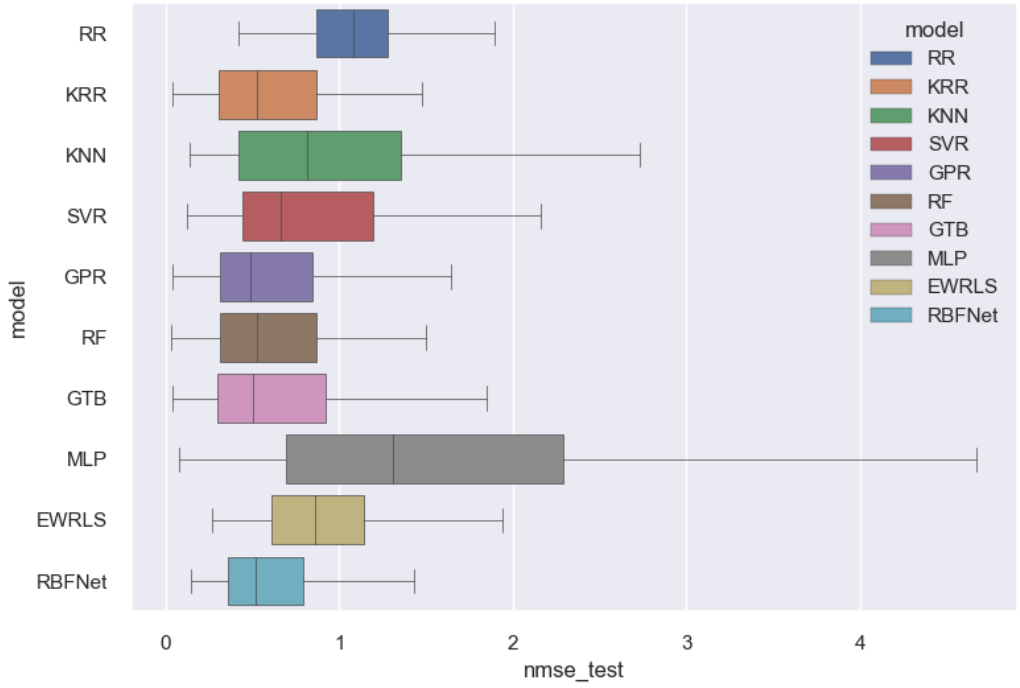
4.4 Discussion

Our RBFNets apply sequentially adapted feature representation transfer from clustering algorithms to supervised learners. Online transfer learning is a relevant area of research for nonstationary time series. Although transfer learning is primarily concerned with transferring knowledge from a source domain to a target domain and may be used offline or online, an increasing number of papers focus on online transfer learning (Zhao et al., 2014; Salvalaio and de Oliveira Ramos, 2019; Wang et al., 2020). Our experiment contributes to the research of continual learning in financial time series by demonstrating that continual learning benefits multi-step forecasting, above and beyond sequential optimisation. If we compare the local learning of the RBFNet with the global learning technique of the MLP, the latter suffers from catastrophic forgetting. Kirkpatrick et al. (2017) and Sukhov et al. (2020) look at ways of improving this issue, specifically at training networks that can maintain expertise on tasks that they have not experienced for a long time. The RBFNets we formulate are naturally

Figure 4.2: The radial basis function network achieves the lowest normalised mean squared error when predicting daily returns of the Refinitiv cross-asset dataset up to $h = 1, ..., 30$ days ahead.



(a) NMSE by model and horizon.



(b) NMSE box plots by model.

designed to measure the similarity between test samples and continuously updated prototypes that capture the characteristics of the feature space. As such, the models are robust in mitigating catastrophic forgetting. To demonstrate this, we conduct a small experiment that measures the cosine similarity

$$cosine\ similarity = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|},$$

between training and test vectors of the financial assets used in the main research experiment. The range of this function is between -1 (complete dissimilarity) and 1 (total similarity). Similar to correlation, a value of 0 indicates no similarity. Figure 4.3 and table 4.4 indicate that the returns have cosine similarity close to zero, whereas the RBFNet hidden unit outputs show much larger cosine similarity, which erodes less quickly with the passage of time. We observe regions of high similarity interspersed with periods of low similarity. For example, a substantial period of low similarity occurred in March 2020, when risky assets such as equities, crypto and commodities sold off massively on the back of global economic shutdowns induced by Covid-19. It is perhaps unsurprising that the RBFNet hidden unit outputs retain greater similarity than the returns. The regime-switching models of Hamilton (1994) rely on Gaussian mixture models to capture the regime-switching characteristics of economic time series. K-means can be thought of as a variant of expectation-maximisation (Dempster et al., 1977), which is used to perform maximum likelihood estimates for Gaussian mixture models. With the similarity plots of the original feature space on the left of figure 4.3 and the clustered feature space on the right of figure 4.3, we see the differences between models such as KRR and the RBFNet. The KRR model will pull the test vectors toward training prototypes that may say little about forecast capability for hitherto unseen data. In contrast, the RBFNets measure the similarity of test vectors with hidden units that have learned the feature space's intrinsic nature.

Table 4.4: Summary statistics for the cosine similarities visualised in figure 4.3.

	mean	std
returns	0.00201	0.386
RBFNet hidden unit output	0.347	0.476

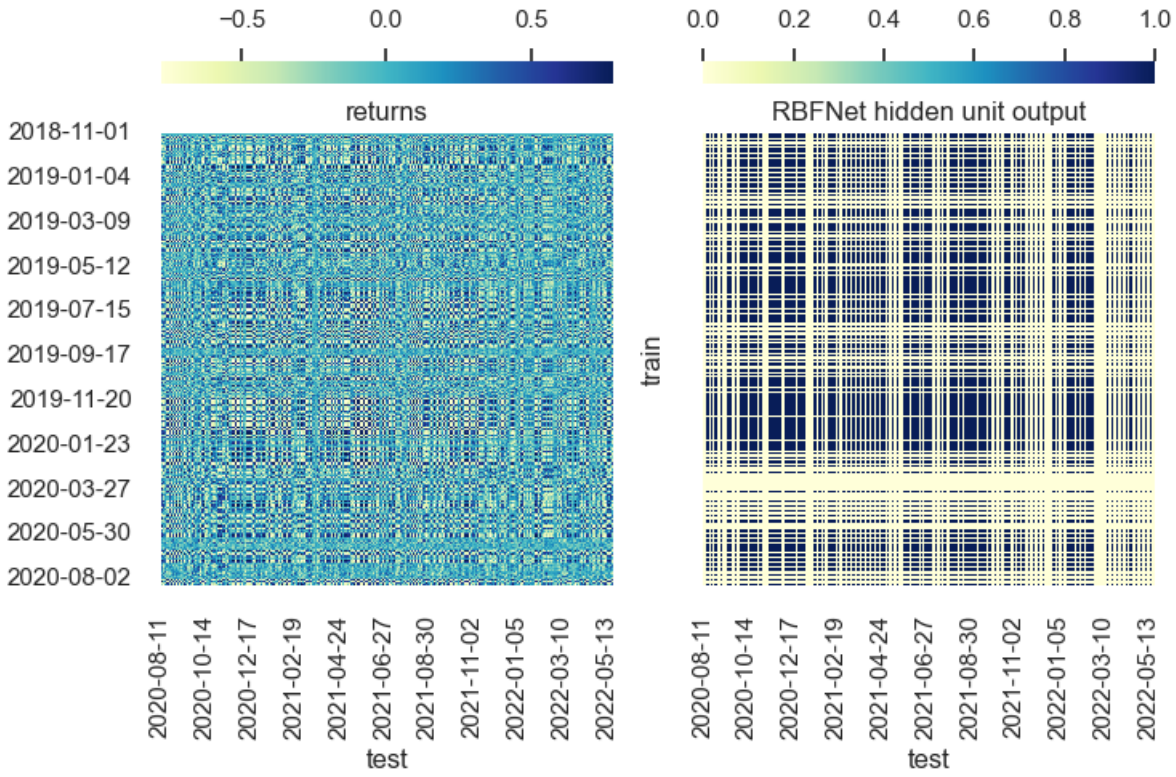


Figure 4.3: We compare the cosine similarities of returns and RBFNet hidden unit outputs; specifically we compare their train/test split. Returns similarity is low and erodes over time. In contrast, the RBFNet hidden outputs retain more remarkable similarity.

Chapter 5

Reinforcement learning for systematic FX trading

We perform feature representation transfer from a radial basis function network to a direct, recurrent reinforcement learning (DRL) agent. Earlier academic work saw mixed results. We use better features, second-order optimisation methods and adapt our model parameters sequentially. As a result, our DRL agents cope better with statistical changes to the data distribution, achieving higher risk-adjusted returns than a funding and a momentum baseline. We design a bespoke quadratic utility function for DRL purposes that captures all impacts to PNL, including price discovery, overnight FX funding and execution cost.

Forecasters of financial time series commonly use supervised learning. For example, Tsay and Chen (2019) apply parametric approaches such as nonlinear state-space models and non-parametric methods such as local learning to nonlinear time series forecasting. In contrast, Bengio (1997) finds that with noisy time series, better results are obtained when the model is trained directly to maximise the financial criterion of interest, such as gains and losses (including those due to transactions) incurred during trading. In this spirit, we extend the earlier work of Moody and Wu (1997) and Gold (2003), where direct, recurrent reinforcement learning (DRL) agents are put to work in financial trading strategies. Rather than optimising for an intermediate performance measure such as maximal forecast accuracy or minimal forecast error, which is still the traditional approach in this domain, we maximise a more direct performance measure such as quadratic economic utility. An advantage of the method is that we can use the risk-adjusted returns of the trading strategy, execution cost and funding cost to influence the learning of the model and update model parameters accordingly.

As discussed in section 2.7.1, the focus of Moody and Wu (1997) is on using the differential Sharpe ratio as a performance measure. In contrast, we adopt the quadratic utility of

Sharpe (2007). This utility ameliorates the undesirable property of the Sharpe ratio in that it penalises a model that produces returns larger than $\frac{\mathbb{E}[r_t^2]}{\mathbb{E}[r_t]}$, that is, the ratio of the expectation of squared returns to the expectation of returns (Moody et al., 1998). Due to using relatively weak features and shared backtest hyper-parameters, Gold (2003) obtained mixed results when experimenting with cash currency pairs. In contrast, our experiment with the major foreign exchange (FX) pairs sees our DRL trading agent achieve an annualised portfolio information ratio of 0.52 with a compound return of 9.3%, net of execution and funding cost, over a seven-year test set. This return is achieved despite forcing the model to trade at the close of the trading day at 5 pm EST, when trading costs are statistically the most expensive.

Aside from the different utility functions, we attribute these improved experiment results to several factors. Firstly, we use more powerful feature engineering in the shape of radial basis function networks (RBFNets). The hidden units of these networks have means, covariances and structures determined by an unsupervised learning procedure for finite Gaussian mixture models (Figueiredo and Jain, 2002). The approach is a form of continual learning, explicitly inductive, feature representation transfer learning, where the knowledge of the mixture model is transferred to upstream models. Secondly, when optimising our utility function with respect to the DRL agent's parameters, we do so sequentially online during the test set, using an extended Kalman filter optimisation procedure (Haykin, 2001). The earlier work uses less powerful offline batch gradient ascent methods, which cope less well with non-stationary financial time series.

To demonstrate that the DRL agent is aware of all impacts to PNL, we compare the realised positions of a USDRUB trader where transaction costs and carry are removed (figure 5.8a) and included (figure 5.8b). Without cost, the DRL agent realises a long position broadly (buying USD and selling RUB), as the Ruble depreciates over time. In contrast, when funding cost is accurately applied, the overnight interest rate differential is roughly 6%, and the DRL agent learns a short position (selling USD and buying RUB), capturing this positive carry. The positive carry is not enough to offset the rapid depreciation of the Ruble.

5.1 Problem formulation

We experiment with the Refinitiv currency pair dataset, section 3.2, specifically taking overnight risk positions. The overnight FX positions attract a funding profit or loss (PNL), depending on the interest rate differential between the two currencies that form the trade pair. There are thus multiple sources of impact on PNL, including directional market moves, funding and transaction costs. These multiple PNL impact sources make it interesting and relevant to experiment with our focus model, the direct reinforcement learning (DRL) agent, that benefits from feature representation transfer via Gaussian mixture models (GMMs). Formulation of the DRL agent is discussed in section 5.3.1; for each currency pair, the model targets a position that maximises expected returns and minimises the variance of returns. The 36 currency pairs that we have available, are all targets for our proprietary trading experiment. Furthermore, these currency pairs are also used as external inputs for our focus model and a supervised learning, momentum baseline (section 5.3.2). Specifically, we construct daily returns from closing mid prices for each currency pair and aggregate them into a vector

$$\mathbf{u}_t = \Big[\frac{p_{t,1}}{p_{t-1,1}} - 1, ..., \frac{p_{t,36}}{p_{t-1,36}} - 1\Big].$$

We then fit a GMM to the external inputs $\mathbf{u}_t$ using a modified expectation-maximisation (EM) procedure due to Figueiredo and Jain (2002). Denote each mixture component as

$$\phi_j(\mathbf{u}_t) \quad j = 1, ..., m.$$

Each mixture component has an associated parameterisation

$$\phi_j(\mathbf{u}_t) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j^{-1}).$$

Thus, we have a single-layer RBFNet formed by the GMM

$$\mathbf{x}_t = [1, \phi_1(\mathbf{u}_t), ..., \phi_m(\mathbf{u}_t), f_{t-1}]^T \in \mathbb{R}^{m+2},$$

where

$$f_{t-1} = tanh\big(\boldsymbol{\theta}_{t-1}^T \mathbf{x}_{t-1}\big)$$

is the previous position of the DRL agent. The $\mathbf{x}_t$ are made available to the DRL agent and the momentum trader baseline. The DRL agent's parameters $\boldsymbol{\theta}_t$ are adjusted such that the quadratic utility is maximised

$$\max_{\boldsymbol{\theta}} \mu_t - \frac{\lambda}{2}\sigma_t^2,$$

where λ is a risk appetite parameter. The expectation of returns $E[r_t] = \mu_t$ and the variance of returns $Var[r_t] = \sigma_t^2$ depend on $\boldsymbol{\theta}_t$ through the position function $f_t(.)$

$$r_t = \Delta p_t f_{t-1} - \delta_t |\Delta f_t| + \kappa_t f_t,$$

where Δp_t is the mid price difference, δ_t is the transaction cost and κ_t is the funding PNL. These returns attributes are constructed individually for each of the 36 currency pairs. The RBFNet is optimised sequentially during the test set, as well as the DRL agent's position function and the exponentially weighted recursive least squares (EWRLS) component of the momentum trader. Finally, we consider the risk-adjusted returns for each of the 36 currency pairs as the primary performance measure. These returns are then aggregated and discussed in sections 5.5 and 5.6.

5.2 Foreign exchange trading

This section describes the FX market and the mechanics of the FX derivatives, which are central to the experimentation we conduct in section 5.4. The global FX market sees over six trillion US dollars traded daily. Figure 5.1 shows this breakdown by instrument type extracted from the Bank of International Settlements Triennial Central Bank Survey, 2019.

FX transactions implicitly involve two currencies: the base currency is quoted conventionally on the left-hand side and the counter currency on the right-hand side. If FX positions are held overnight, the trader will earn the interest rate of the currency bought and pay the interest rate of the currency sold. The interest rates for specific maturities are determined in the inter-bank currency market and are heavily influenced by the base rates typically set by central banks. Currency trades usually settle two business days after the trade date. Clients fund their positions by rolling them forward via tomorrow/next (tomnext) swaps. Tomnext is a short-term FX transaction where a currency pair is bought and sold over two business

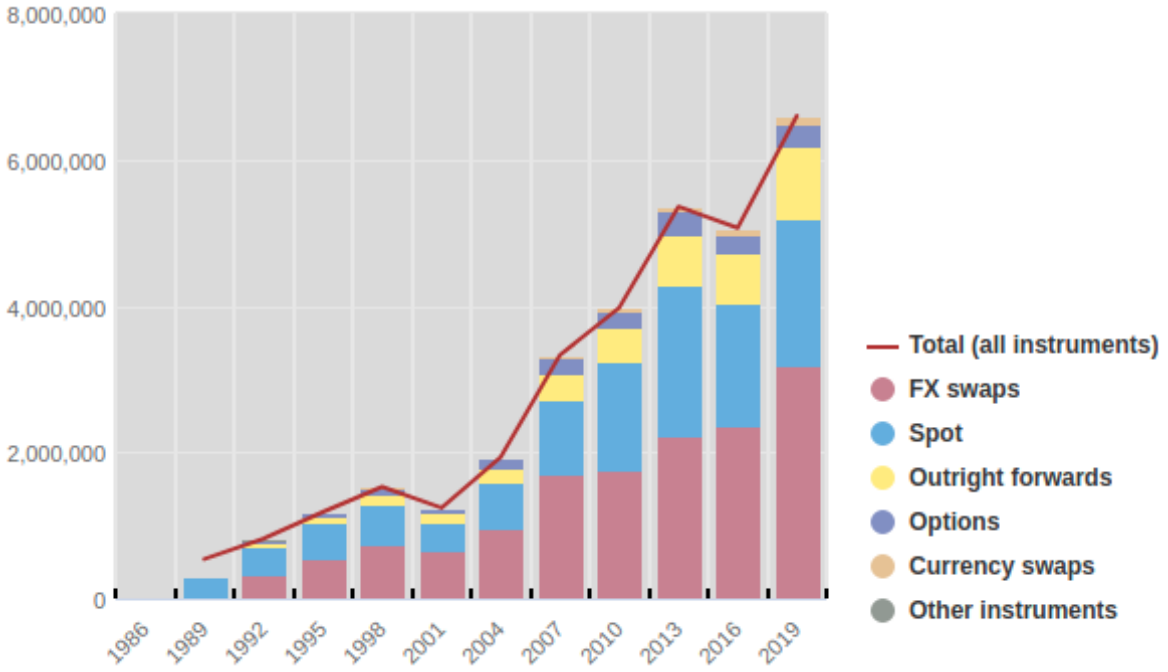


Figure 5.1: Average daily global FX market turnover in USD millions, source: BIS.

days: tomorrow (in one business day) and the following day (two business days from today). The tomnext transaction allows traders to maintain their position without being forced to take physical delivery. To determine this funding cost, one needs to compute the forward prices. Forwards are agreements between two counterparties to exchange currencies at a predetermined rate on some future date.

Forward prices are calculated by adding forward points to a spot price. These points reflect the interest rate differential between the two currencies being traded and the maturity of the trade. Forward points do not represent an expectation of the direction of a currency but rather the interest rate differential. Let bid_t^{spot} denote the spot FX price at which price takers can sell at time t. Similarly, let ask_t^{spot} denote the spot FX price at which price takers can buy at time t. The spot mid-rate is

$$mid_t^{spot} = 0.5 \times (bid_t^{spot} + ask_t^{spot}). \tag{5.1}$$

Forward points are computed as follows

$$mid_t^{fpts} = mid_t^{spot}(e_2 - e_1)\frac{T}{360\iota},$$

where e_2 is the counter currency interest rate, e_1 is the base currency interest rate, T is the number of days till maturity, and ι is the tick size of the FX pair. Example forward points for GBPUSD are shown in figure 5.2. *GBP=* is the Refinitiv information code (ric) for cash GBPUSD and *GBPTND=* is the ric for tomnext GBPUSD forward points. Note that the forward points are quoted as a bid/ask pair, reflecting the interest differential and the additional spread quoted by the FX forwards market maker to compensate them for their quoting risk. The tomnext outrights are computed as

$$bid_t^{tn} = bid_t^{spot} + ask_t^{fpts}\iota$$
$$ask_t^{tn} = ask_t^{spot} + bid_t^{fpts}\iota.$$

When rolling a long GBPUSD position forward, the tomnext swap would involve selling GBPUSD at bid_t^{spot} and repurchasing it at ask_t^{tn}. The cost of this *roll* is thus $notional \times (bid_t^{spot} - ask_t^{tn})$, where *notional* denotes the size of the position taken by the trader. When rolling a short GBPUSD position forward, a trader would buy ask_t^{spot} and sell forward bid_t^{tn}, with the funding cost being $notional \times (bid_t^{tn} - ask_t^{spot})$. This funding may be a loss but also a profit. In addition, traders hold FX positions to capture the favourable interest rate differential between two currency pairs, known as the carry trade.

5.3 Experiment methods

This section describes how our DRL agent targets a position directly. The DRL agent learns the desired risk position via the policy gradient paradigm, discussed in section 2.7. Additionally, we describe the baseline models that are used for returns benchmarking.

5.3.1 Targeting a position with direct recurrent reinforcement

Our DRL agent comprises an unsupervised model and a reinforcement learning component. Specifically, we construct a RBFNet whose hidden units are determined by a Gaussian mixture model (GMM) procedure due to Figueiredo and Jain (2002) which has an identical expectation step to Dempster et al. (1977)'s em algorithm, but a modified log-likelihood function for the maximisation step. The procedure involves the construction of a large mixture and annihilates components from the mixture that are not supported by the data. The GMM hidden unit output is aggregated into a feature vector which includes an additional

GBPF=	GBP Deps & Fwds			
RIC	Bid	Ask	Srce	Time
GBP=	1.3340	1.3344	BCFX	22:00
GBPOND=	-0.06	0.24	ABNA	11:01
GBPTND=	-0.02	0.18	ABNA	11:01
GBPSND=	-0.02	0.18	ABNA	11:01
GBPSWD=	-0.01	0.19	ABNA	11:01
GBP2WD=	-0.01	0.19	ABNA	11:01
GBP3WD=	-0.01	0.19	ABNA	11:01
GBP1MD=	0.02	0.12	KBCB	9:55
GBP2MD=	0.01	0.21	ABNA	11:01
GBP3MD=	0.02	0.22	ABNA	11:01
GBP4MD=	0.06	0.26	ABNA	11:01
GBP5MD=	0.16	0.42	INGI	3:07
GBP6MD=	0.31	0.59	BRKR	8:10
GBP7MD=	0.24	0.44	ABNA	11:01
GBP8MD=	0.29	0.49	ABNA	11:01
GBP9MD=	0.36	0.56	ABNA	11:01
GBP10MD=	0.42	0.62	ABNA	11:01
GBP11MD=	0.47	0.67	ABNA	11:01
GBP1YD=	0.53	0.73	ABNA	11:01
GBP2YD=	1.1	1.3	KLMM	15:59
GBP3YD=	1.3	1.6	KLMM	15:59
GBP4YD=	1.3	1.6	KLMM	15:59
GBP5YD=	1.3	1.6	KLMM	15:59

Figure 5.2: We show Refinitiv GBPUSD forward rates. FX forward rates primarily reflect the interest rate differential between the base and the counter currencies.

bias term and the DRL agent's prior position (equation 5.9). A dot product between the feature vector (equation 5.10) and a weight vector is passed through a nonlinearity, typically the tanh function, which maps this output to the range $-1 \leq f_t \leq 1$, determining the desired risk position of the DRL agent. The DRL agent's weights are determined by an extended Kalman filter (EKF) algorithm 5.1. Figure 5.3 provides a schematic of the feature representation transfer from the RBFNet to the DRL agent. The external input to the transfer learner, represented by the left-most black circles, is a vector of daily returns of the 36 FX pairs used in the experiment, detailed in section 3.2. The grey circles represent the RBFNet hidden unit layer. Additionally, the blue circle represents the previously estimated position of the DRL agent. The DRL output is fed back into the hidden layer recurrently and is represented by the dotted blue line.

Sharpe (2007) discusses asset allocation as a function of expected utility maximisation, where the utility function may be more complex than that associated with mean-variance analysis. Denote the expected utility at time t for a single portfolio constituent as

$$\upsilon_t = \mu_t - \frac{\lambda}{2}\sigma_t^2, \tag{5.2}$$

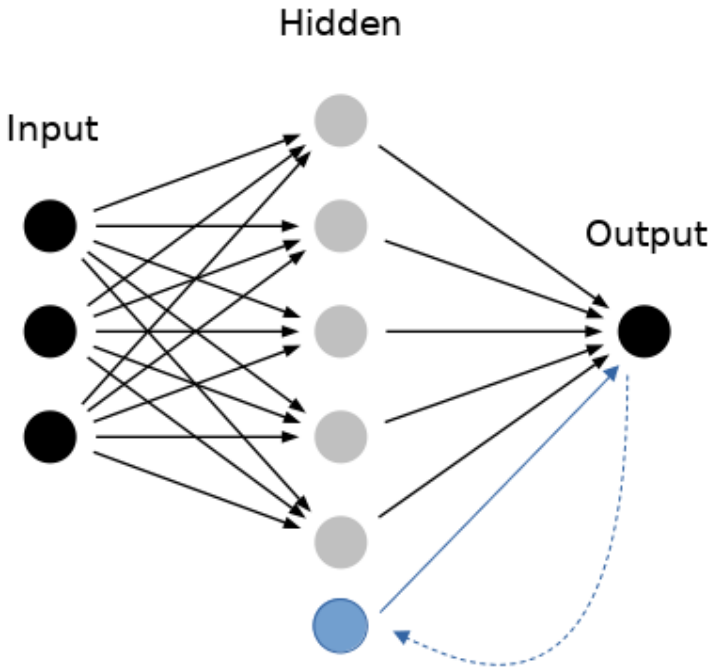


Figure 5.3: Feature representation transfer from a radial basis function network to a direct, recurrent reinforcement learning agent.

where the expected return $\mu_t = \mathbb{E}[r_t]$ and variance of returns $\sigma_t = \mathbb{E}[r_t^2] - \mathbb{E}[r_t]^2$ may be estimated in an online fashion with exponential decay

$$\mu_t = \tau\mu_{t-1} + (1-\tau)r_t, \tag{5.3}$$

$$\sigma_t^2 = \tau\sigma_{t-1}^2 + (1-\tau)(r_t - \mu_t)^2. \tag{5.4}$$

The risk appetite constant $\lambda > 0$ can be set as a function of an investor's desired risk-adjusted return, as demonstrated by Grinold and Kahn (2019). The information ratio is a risk-adjusted differential reward measure, where the difference is taken between the strategy being evaluated and a baseline strategy with expected returns $b_t = \mathbb{E}\left[r_{b,t}\right]$:

$$ir_t = 252^{0.5} \times \frac{\mu_t - b_t}{\sigma_t}. \tag{5.5}$$

The similarity of the information ratio to the Sharpe ratio is apparent. Setting $b_t = 0$, substituting the non-annualised information ratio into equation 5.2 and differentiating with respect

to the risk, we obtain a suitable value for the risk appetite parameter:

$$\begin{aligned} ir_t &= \frac{\mu_t}{\sigma_t} \\ \upsilon_t &= ir_t \times \sigma_t - \frac{\lambda}{2}\sigma_t^2 \\ \frac{d\upsilon_t}{d\sigma_t} &= ir_t - \lambda\sigma_t = 0 \\ \lambda &= \frac{ir_t}{\sigma_t}. \end{aligned} \tag{5.6}$$

The net returns whose expectation and variance we seek to learn are decomposed as

$$r_t = \Delta p_t f_{t-1} - \delta_t |\Delta f_t| + \kappa_t f_t, \tag{5.7}$$

where Δp_t is the change in reference price, typically a mid-price

$$\Delta p_t = 0.5 \times (bid_t + ask_t - bid_{t-1} - ask_{t-1}),$$

δ_t represents the execution cost for a price taker

$$\delta_t = max[0.5 \times (ask_t - bid_t), 0], \tag{5.8}$$

κ_t is the profit or loss of rolling the overnight FX position, the so-called *carry* and f_t is the desired position learnt by the DRL agent

$$f_t = tanh\big(\boldsymbol{\theta}_t^T \mathbf{x}_t\big). \tag{5.9}$$

The model is maximally short when $f_t = -1$ and maximally long when $f_t = 1$. The recurrent nature of the model occurs in the input feature space where the previous position is fed to the model input

$$\mathbf{x}_t = [1, \phi_1(\mathbf{u}_t), ..., \phi_m(\mathbf{u}_t), f_{t-1}]^T \in \mathbb{R}^{m+2}, \tag{5.10}$$

and $\phi_j(.)$ denotes an radial basis function hidden unit, in a network of m such units, which takes as input a feature vector $\mathbf{u}_t$, see section 2.3. The goal of our DRL agent is to maximise the utility in equation 5.2 by targeting a position in equation 5.9. To do this, one may apply

an online stochastic gradient ascent update

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta \nabla \upsilon_t \equiv \Delta \boldsymbol{\theta}_t + \eta \frac{d\upsilon_t}{d\boldsymbol{\theta}_t}.$$

Instead of a fixed learning rate η, one may consider the Adam optimiser (Kingma and Ba, 2017), where an adaptive learning rate is applied. This adaptive learning rate is a function of the gradient expectation and variance. In practice, we find that Adam takes many iterations of model fitting to get the weights to be large enough to take a meaningful position via function 5.9; this is not necessarily an Adam problem but a result of the tanh position function taking a while to saturate. If the weights are too small, then the average position taken by the DRL agent will also be small. Therefore, we perform a gradient-based weight update using an extended Kalman filter (EKF) (Williams, 1992a; Haykin, 2001), modified for reinforcement learning in this context.

Algorithm 5.1 The extended Kalman filter.

Require: α `// a ridge penalty`
1 , τ `//` $0 \ll \tau \leq 1$ `is an exponential decay factor.`
Initialise: $\boldsymbol{\theta} = \mathbf{0}^d$, $\mathbf{P} = \mathbf{I}_d/\alpha$
Input: $\nabla \upsilon_t$
Output: $\boldsymbol{\theta}_t$
2 $z = 1 + \nabla \upsilon_t^T \mathbf{P}_{t-1} \nabla \upsilon_t / \tau$
3 $\mathbf{k} = \mathbf{P}_{t-1} \nabla \upsilon_t / (z\tau)$
4 $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{k}$
5 $\mathbf{P}_t = \mathbf{P}_{t-1}/\tau - \mathbf{k}\mathbf{k}^T z$

In algorithm 5.1, $\mathbf{P}_t$ is an approximation to $[\nabla^2 \upsilon_t]^{-1}$, the inverse Hessian of the utility function υ_t with respect to the model weights $\boldsymbol{\theta}_t$. We decompose $\nabla \upsilon_t = \frac{d\upsilon_t}{d\boldsymbol{\theta}_t}$ as follows:

$$\begin{aligned} \nabla \upsilon_t &= \frac{d\upsilon_t}{dr_t} \left\{ \frac{dr_t}{df_t} \frac{df_t}{d\boldsymbol{\theta}_t} + \frac{dr_t}{df_{t-1}} \frac{df_{t-1}}{d\boldsymbol{\theta}_{t-1}} \right\} \\ &= \frac{d\upsilon_t}{dr_t} \left\{ \frac{dr_t}{df_t} \left(\frac{\partial f_t}{\partial \boldsymbol{\theta}_t} + \frac{\partial f_t}{\partial f_{t-1}} \frac{\partial f_{t-1}}{\partial \boldsymbol{\theta}_{t-1}} \right) \right. \\ &\quad \left. + \frac{dr_t}{df_{t-1}} \left(\frac{\partial f_{t-1}}{\partial \boldsymbol{\theta}_{t-1}} + \frac{\partial f_{t-1}}{\partial f_{t-2}} \frac{\partial f_{t-2}}{\partial \boldsymbol{\theta}_{t-2}} \right) \right\}. \end{aligned} \tag{5.11}$$

The constituent derivatives for the left half of equation 5.11 are:

$$\begin{aligned}
\frac{d\upsilon_t}{dr_t} &= (1-\eta)[1-\lambda(r_t-\mu_t)] \\
\frac{dr_t}{df_t} &= -\delta_t \times sign(\Delta f_t) + \kappa_t \times sign(f_t) \\
\frac{df_t}{d\boldsymbol{\theta}_t} &= \mathbf{x}_t\left[1-\tanh^2\left(\boldsymbol{\theta}_t^T\mathbf{x}_t\right)\right] \\
&\quad + \boldsymbol{\theta}_{t,m+2}\left[1-\tanh^2\left(\boldsymbol{\theta}_t^T\mathbf{x}_t\right)\right] \\
&\quad \times \mathbf{x}_{t-1}\left[1-\tanh^2\left(\boldsymbol{\theta}_{t-1}^T\mathbf{x}_{t-1}\right)\right].
\end{aligned}$$

An alternative procedure to consider, if one still has a strong preference for gradient-only methods, is Lecun (1989)'s modified tanh function

$$f(x) = 1.7159 \times \tanh\left(\frac{2}{3} \times x\right),$$

that allows the target values of ± 1 to be more easily attained, although is bounded on ± 1.7159 and is thus less suitable for our position function, as ± 1 indicates the largest position we take and no leverage is used. A further option is to sequentially compute the volatility of the position function, $Var[tanh(\boldsymbol{\theta}_t^T\mathbf{x}_t)]$, and scale the positions taken by this measure. Volatility scaling of risk in equity portfolios has been shown to generate higher risk-adjusted returns than constant notional portfolios (Harvey et al., 2018).

5.3.2 Baseline models

To assess the comparative strength of the model of section 5.3.1, we employ two baseline models. The first model is a momentum trader, which uses the sign of the next step ahead return forecast as a target position. This model is also a RBFNet, except here, the feature representation transfer of the GMM cluster is to an exponentially weighted recursive least squares (EWRLS) supervised learner. A visual representation of the model is shown in figure 2.2. Our second baseline is the carry trader, which hopes to earn a positive overnight funding rate. Denote the long/short carry as

$$\begin{aligned}
\kappa_t^{long} &= bid_t^{spot} - ask_t^{tn} \\
\kappa_t^{short} &= bid_t^{tn} - ask_t^{spot},
\end{aligned}$$

the position of the carry trader is

$$f_t^{carry} = sign\big(\kappa_t^{long} - \kappa_t^{short}\big).$$

The carry trader goes long (short) the base currency if the base currency has an overnight interest rate higher (lower) than the counter currency. The observed carry may be less than the execution cost; therefore, we allow the carry trader to abstain from trading in such circumstances.

5.4 Experiment design

In this section, we establish the design of the experiment, describing the data we use and the experiment's performance evaluation criteria. Section 3.2 describes the dataset. We have over 11 years of daily data to use in our experiment. From these data, we construct daily returns for each of the 36 currency pairs, reserving the first third as a training set and the final two-thirds as a test set. The returns are inputs to the DRL agent and EWRLS momentum trader. More concretely, we use linear returns of the form

$$ret_{k,t} = \frac{mid_{k,t}}{mid_{k,t-1}} - 1, \quad k = 1, ..., 36.$$

The training set returns are used as an external input to a RBFNet whose structure, hidden unit means and covariances are determined by Figueiredo and Jain (2002)'s GMM algorithm. The DRL agent is also fitted in the training set to each currency pair, explicitly learning the weights in position function 5.9 using the EKF algorithm 5.1. Additionally, the momentum trader of section 5.3.2 is fitted in the training set to each currency pair using EWRLS algorithm 2.1. Both models continue to learn online during the test set. The carry trader baseline does not require any model fitting.

5.4.1 Performance evaluation methods

We force the models to trade as price takers, incurring an immediate execution cost equal to half the bid/ask spread. These data are sampled at the end of the trading day, 10 PM GMT, impacting the risk-adjusted returns performance. At this time, the bid/ask spreads are statistically at their widest, which impacts execution and funding costs. Preferably, we

would extract intraday data; however, Refinitiv restricts us to under 30 business days for these sampled data. Figure 5.4 illustrates the challenge succinctly. It shows relative intraday bid/ask spreads

$$spread_t^{spot} = \frac{ask_t^{spot} - bid_t^{spot}}{mid_t^{spot}},$$

for the 36 currency pairs used in our experiment. The data are sampled minutely over 28 business days ending mid-October 2021. The global maximum bid/ask spread occurs precisely when Refinitiv samples the daily data, namely 10 PM GMT.

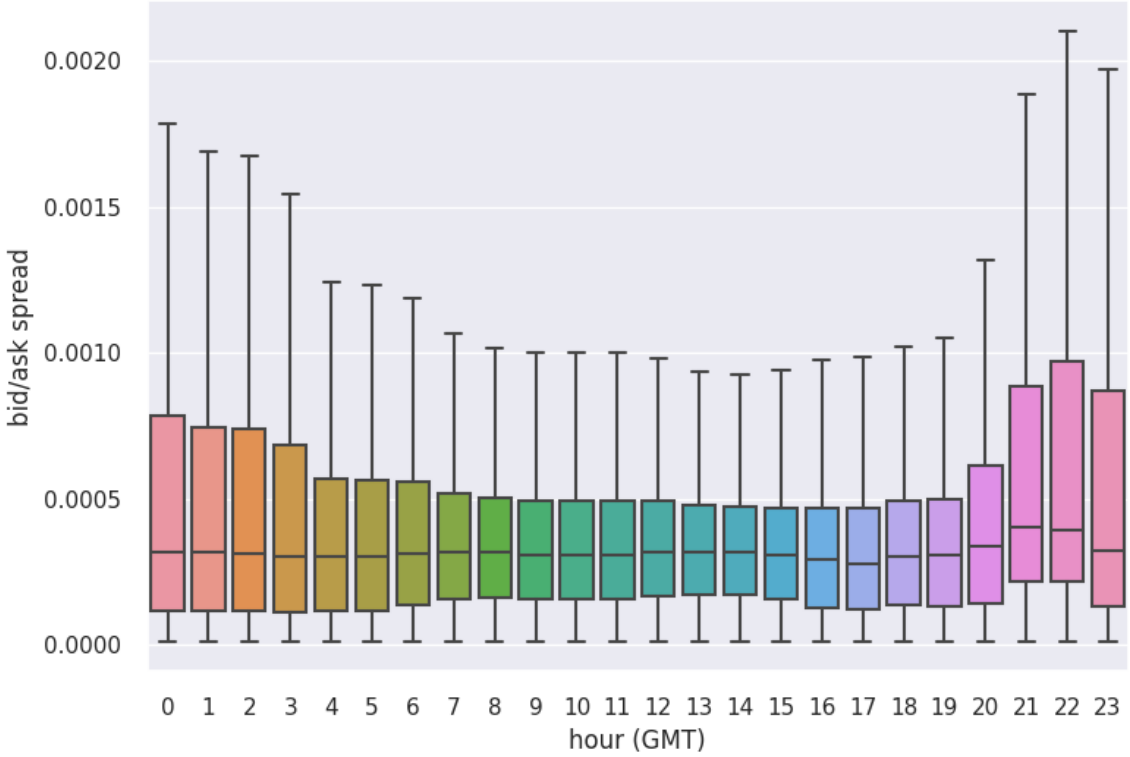


Figure 5.4: We plot relative intra-day bid/ask spreads for our experiment's 36 Refinitiv currency pairs. Execution cost is highest when the trade date rolls onto the next date; this occurs at around 10 PM GMT, precisely when Refinitiv samples their historical daily FX data.

The test set evaluates performance for each currency pair using the net PNL equation 5.7. This reward, net of transaction and funding cost, is in price difference space. We convert to returns space by dividing by the mid-price computed using equation 5.1. These returns are accumulated to produce the results shown in figure 5.6 and the middle sub-plots of figures 5.8a and 5.8b. In addition, the daily returns are described statistically in tables 5.1 and 5.2. In table 5.1, the information ratio (ir) is computed using equation 5.5. We set the baseline

return $b_t = 0$. In summary, we evaluate performance by considering the risk-adjusted daily returns generated by each model, net of transaction and funding costs.

5.4.2 Hyperparameters

The following hyperparameters are set in the experiment:

- $\tau = 0.99$, the exponential decay constant of moving moment equations 5.3, 5.4, 5.7, EKF weight algorithm 5.1 and EWRLS algorithm 2.1.
- $\alpha = 1$, the ridge penalty of EKF weight algorithm 5.1 and an EWRLS algorithm 2.1.
- λ, the risk appetite parameter of equation 5.2, is estimated using equation 5.6.

The hyperparameters are set to reasonable values, rather than optimised, as we wish to preserve as much data for the test set. With larger datasets, we would consider cross-validation of the hyperparameters.

5.5 Results

Figure 5.6 shows the accumulated returns for each strategy. The DRL agent is denoted as *DRL*, the momentum trader is shown as *momentum* and the carry trader is indicated as *carry*. The carry baseline performs poorly, reflecting the low-interest rate differential environment since the 2008 financial crisis. The available funding that can be earnt relative to execution cost is small. Figure 5.5 shows the direction of travel in central bank interest rates over the past 20 years. Central bank rates halved on average during the 2008 global financial crisis and have declined further since. In contrast, the momentum trader achieves the highest return with an annual compound net return of 11.7% and an information ratio of 0.4. Additionally, the DRL agent achieves an annual compound net return of 9.3%, with an information ratio of 0.52. Its information ratio is higher because its daily portfolio returns standard deviation is two-thirds of the momentum trader's. Table 5.1 summarises net PNL (PNL) returns statistics by strategy, with a figure of the distribution of the daily returns in figure 5.7.

Table 5.2 shows the funding or carry in returns space for each strategy. We see that the carry baseline does indeed capture positive carry. However, this return is not enough to offset the execution cost and the PNL associated with holding risk, which moves in a trend-following way, mainly as opposed to the funding PNL. How funding moves opposite to price

trends is expected. Central banks invariably increase overnight rates when currencies depreciate considerably to make their currency more attractive and stem the tide of depreciation. The Turkish Lira and Russian Ruble are two cases in point. We see evidence in table 5.2 that the DRL agent captures more carry relative to the momentum trader. This funding capture is also expected, as the funding PNL makes its way into equation 5.7 and is propagated through to the derivative of the utility function with respect to the model weights, using equation 5.11.

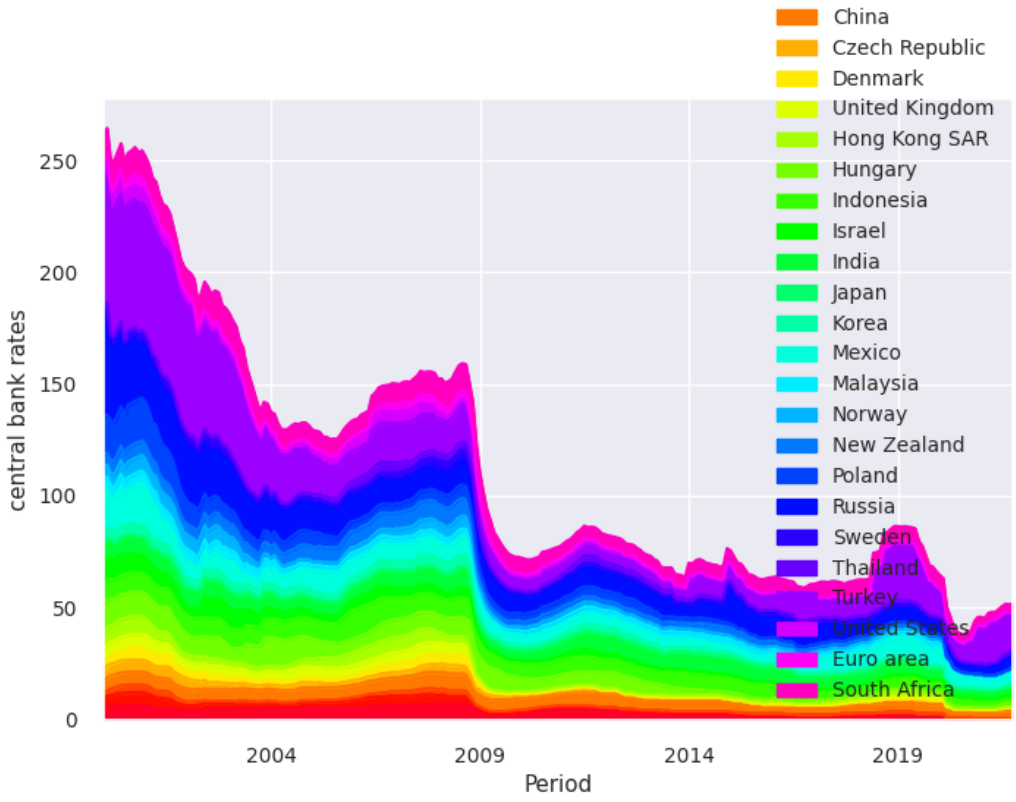


Figure 5.5: Stacked central bank interest rates in percentage points, source: BIS. There has been a downward trajectory in global rates during the period shown, with a sharp contraction in rates shortly after the 2008 financial crisis.

5.6 Discussion

Both baselines make decisions using incomplete information. The momentum trader focuses on learning the FX trends but ignores costs, whilst the carry trader tries to earn funding but ignores price trends. In contrast, the DRL agent optimises the desired position as a function of market moves and funding whilst minimising execution cost. To demonstrate that the DRL agent is indeed learning from these reward inputs, we compare the realised positions of

Table 5.1: Portfolio net PNL returns by strategy. The DRL agent achieves the highest risk-adjusted returns.

	DRL	momentum	carry
count	1888	1888	1888
mean	0.00104	0.00121	-0.002
std	0.032	0.048	0.052
min	-0.141	-0.202	-0.344
25%	-0.019	-0.028	-0.028
50%	-0.000	-0.002	0.000
75%	0.019	0.028	0.026
max	0.245	0.423	0.200
sum	1.953	**2.296**	-4.328
ir	**0.518**	0.403	-0.701

Table 5.2: PNL returns by strategy. The carry baseline naturally achieves the highest funding PNL. However, as table 5.1 shows, this funding PNL cannot offset what is clearly a momentum-driven environment with low-interest rates (see figure 5.5). The DRL agent captures a greater funding profit than the momentum trader. The momentum trader is a supervised learner which forecasts daily returns and cannot adjust its positions based on funding PNL. Funding PNL tends to be negatively correlated with momentum PNL.

	DRL	momentum	carry
count	1888	1888	1888
mean	-0.00030	-0.00050	0.00048
std	0.00019	0.00031	0.00036
min	-0.00395	-0.00576	0.00007
25%	-0.00035	-0.00059	0.00029
50%	-0.00024	-0.00040	0.00035
75%	-0.00019	-0.00032	0.00051
max	0.00153	0.00072	0.00518
sum	-0.56226	-0.94769	**0.90655**

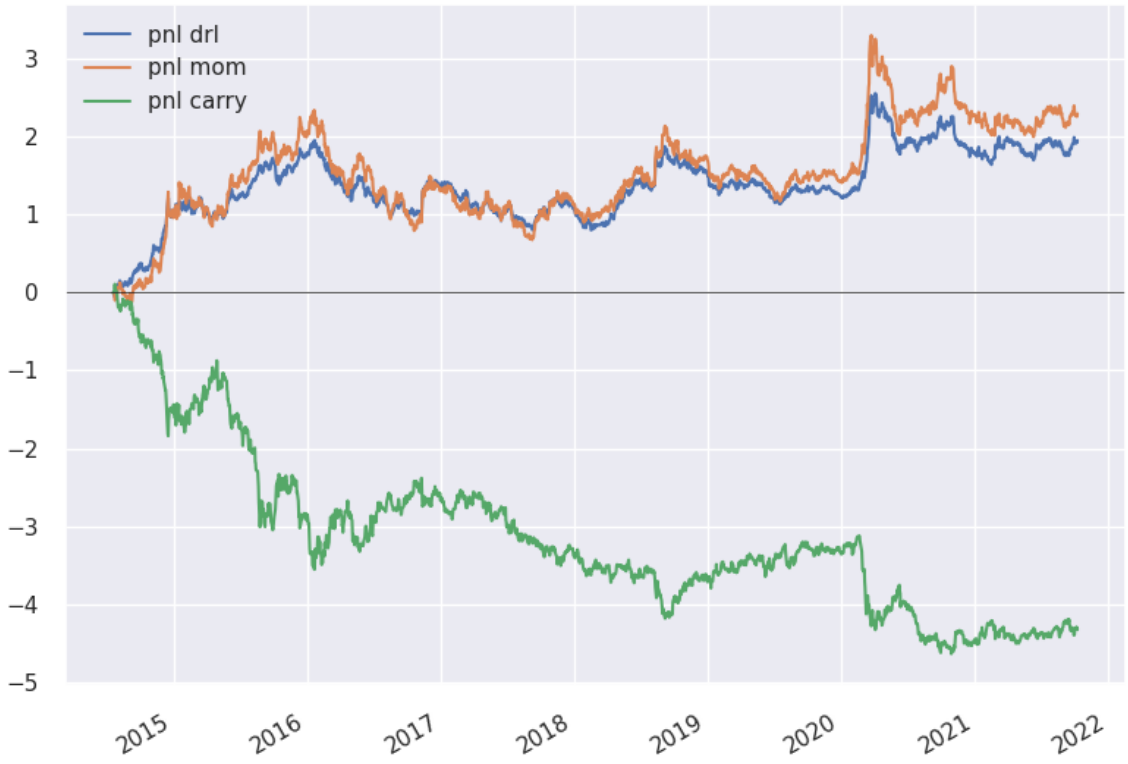


Figure 5.6: Cumulative daily returns across all currency pairs by strategy: **drl** is the DRL agent, **mom** is the supervised-learning momentum trader and **carry**, the carry/funding trader.

a USDRUB trader where transaction costs and carry are removed (figure 5.8a) and included (figure 5.8b). Without cost, the DRL agent realises a long position broadly (buying USD and selling RUB), as the Ruble depreciates over time. In contrast, when funding cost is accurately applied, the overnight interest rate differential is roughly 6%, and the DRL agent learns a short position (selling USD and buying RUB), capturing this positive carry. The positive carry is not enough to offset the rapid depreciation of the Ruble.

How significant are these results? Grinold and Kahn (2019) show table 5.3 of empirical information ratios for US fund managers over the five years from January 2003 through December 2007. Although dated, the results indicate that our DRL agent, which trades statistically at the worst time of day in the FX market, achieves an information ratio at the $75'th$ percentile of information ratios achieved empirically by various fund managers within fixed income and equities. The information ratio measures the probability that a strategy will achieve positive residual returns in every period; it is a measure of consistency. Equation 5.5 shows that the information ratio is the ratio of residual return to residual risk. Denote this

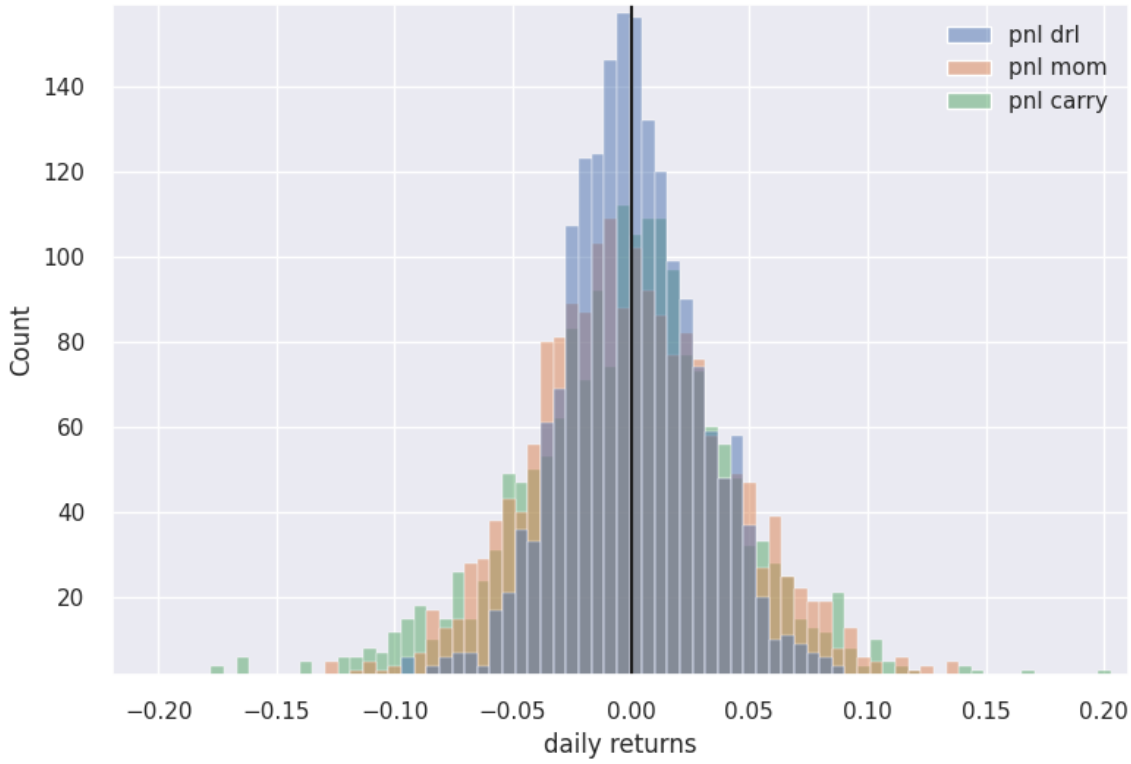


Figure 5.7: The distribution of daily returns by strategy: **drl** is the DRL agent, **mom** is the supervised-learning momentum trader and **carry**, the carry/funding trader.

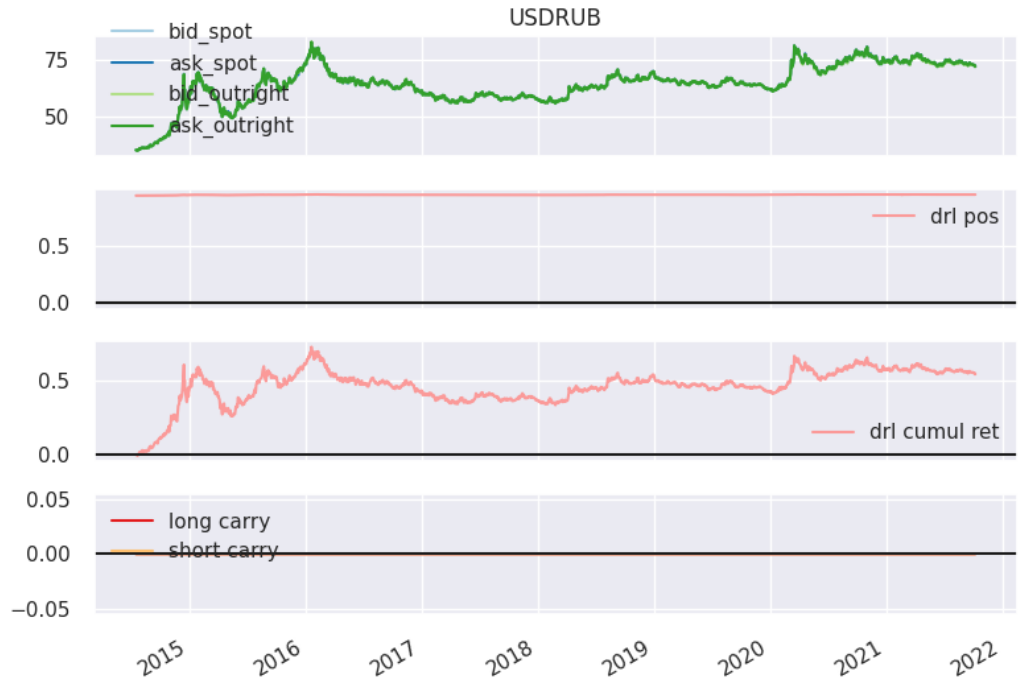
residual return as the strategy's alpha, $\alpha_t = \mu_t - b_t$. The probability of realising a positive residual return is

$$Pr(\alpha_t > 0) = \Phi(ir_t),$$

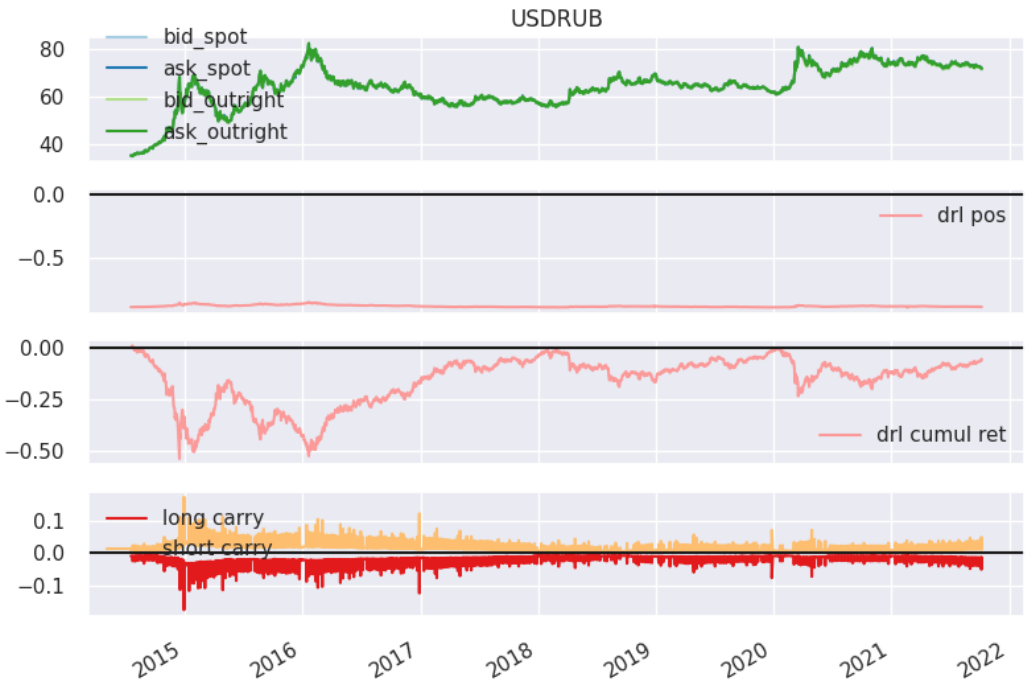
where $\Phi(.)$ denotes the cumulative normal distribution function. We find that the DRL agent has a probability of a positive residual return of 70%, and the momentum baseline has a probability of a positive residual return of 66%.

Table 5.3: Empirical information ratios, source: Blackrock.

asset class	equities	equities	equities	fixed income	both
percentile	mutual funds	long	long/short	institutional	average
90	1.04	0.77	**1.17**	0.96	0.99
75	**0.64**	0.42	0.57	0.50	0.53
50	0.20	0.02	**0.25**	0.01	0.12
25	**–0.21**	–0.38	–0.22	–0.45	–0.32
10	–0.62	–0.77	**–0.58**	–0.90	–0.72



(a) We show a USDRUB DRL agent trading *without* execution or funding cost. Without funding, this transfer learning, direct, recurrent reinforcement learning agent behaves like the supervised learner, the momentum trader.



(b) We show a USDRUB DRL agent trading *with* execution and funding cost. When funding is included, the DRL agent tries to capture a funding profit, going short USD and long RUB.

Figure 5.8: A sensitivity analysis of funding versus position for a USDRUB DRL agent.

Chapter 6

The recurrent reinforcement learning crypto agent

We construct a digital assets trading agent that performs feature space representation transfer from an echo state network to a DRL agent. The agent learns to trade the XBTUSD perpetual swap contract on BitMEX. Our meta-model can process data as a stream and learn sequentially; this helps it cope with the nonstationary environment. We design a bespoke quadratic utility function for DRL purposes that captures all impacts to PNL, including price discovery, intraday funding and execution cost.

In this chapter, we extend our earlier work in chapter 5, where we combine sequential optimisation with feature representation transfer and direct, recurrent reinforcement learning (DRL). We novelly transfer the learning of an echo state network (ESN) to a DRL agent who must learn to trade digital asset futures, specifically the XBTUSD (Bitcoin versus US Dollar) perpetual swap on the BitMEX exchange. Our transfer learner benefits from an ample, dynamic reservoir feature space and can learn from the different impact sources on profit and loss (pnl), including execution costs, exchange fees, funding costs and price moves in the market.

Perhaps the main benefit of this chapter will be for financial industry practitioners. Our meta-model can process data as a stream and learn sequentially; this helps it cope with the nonstationarity of the high-frequency order book and trading data. Furthermore, by using the vast high-frequency data, our model, which has a high learning capacity, avoids the kind of overfitting on a lack of data points that occurs with down-sampled data. We escape the problem of over-trading typically seen with supervised learning models by learning the sensitivity of the change in risk-adjusted returns to the model's parameterisation. Stated another way, our model learns from the multiple impact sources on profit and loss (PNL)

and targets the appropriate risk position. Finally, the scientific experiment that we conduct is representative of actual trading conditions; thus, we are confident of the efficacy of the results for industry use.

6.1 Problem formulation

We consider the goal of proprietary risk taking in the BitMEX XBTUSD perpetual swap, specifically using five-minutely sampled data, as discussed in section 3.3. This crypto derivatives contract attracts a funding PNL every eight hours. The funding mechanism ensures that the contract does not trade too far away from the cash index it tracks. Participants can speculate more easily in the perpetual swap than the underlying cash cryptocurrency pair, as the latter requires ownership or borrowing. In addition, leverage may also be used in these derivatives contracts. As there are multiple PNL impact sources, including directional market moves, execution costs and funding PNL, proprietary risk taking in perpetual swaps is suitably tackled with DRL. Complete specification of the DRL agent, including the use of feature representation transfer from ESNs, is discussed in section 6.3.1. We measure success of the experiment by considering the risk-adjusted returns that the model generates, including the contribution of funding PNL to total PNL. These results are discussed in section 6.3.3. Finally, sensitivity of risk-adjusted returns to ESN initialisation, is examined in section 6.4.

6.2 The BitMEX XBTUSD perpetual swap

The data that we experiment with is from the BitMEX cryptocurrency derivatives exchange. In 2016 they launched the XBTUSD perpetual swap, where clients trade Bitcoin against the US Dollar. The perpetual swap is similar to a traditional futures contract, except for no expiry or settlement. It mimics a margin-based spot market and trades close to the underlying reference index price. A funding mechanism tethers the contract to its underlying spot price. In contrast, a futures contract may trade at a significantly different price due to the basis

$$basis_t = futures_t - index_t.$$

The basis means different things in different markets. For example, in the oil market, the demand for spot oil can outpace the demand for futures oil, leading to a higher spot price.

Crypto futures basis typically increases (decreases) with the time till expiry in a bull (bear) market. Similar effects happen in the equity markets. In both asset classes, participants can take risks in the futures market more easily. The spot markets typically do not offer leverage, and the trader must have inventory in the exchange to trade. In contrast, the futures markets allow traders to use margin-based trading and to go short without borrowing the underlying asset, requiring just a margin (deposit) to fund the position. Figure 6.1 shows the basis in relative terms for the XBTUSD perpetual swap during the bear market of 2018. The mid-price of the perpetual swap is compared against the underlying index it tracks, .BXBT. The relative basis is computed as

$$rbasis_t = \frac{futures_t - index_t}{index_t}. \tag{6.1}$$

Before this bear market, Bitcoin hit a then all-time high of $20,000, and the 100-day exponentially weighted moving average of relative basis was very positive in late 2017. For most of 2018 and 2019, the basis was largely negative, reflecting the cash market sell-off from all-time highs to roughly USD 3,000.

6.2.0.1 Funding

The funding rate for the perpetual swap comprises two parts: an interest rate differential component and the premium or discount of the basis. Denote this funding rate as κ_t. The interest rate differential reflects the borrowing cost of each currency involved in the pair

$$e_t = \frac{e_t^{quote} - e_t^{base}}{T}.$$

For XBTUSD, e_t^{quote} denotes the US Dollar borrowing rate and e_t^{base} denotes the Bitcoin borrowing rate. As funding occurs every 8 hours, $T = 8$. The basis premium/discount component is computed similarly to equation 6.1, with some subtleties applied to minimise market manipulation, such as using time and volume weighted average prices. The funding rate is

$$\kappa_t = b_t + max(min[\zeta e_t - b_t], -\zeta), \tag{6.2}$$

Figure 6.1: We show XBTUSD basis during the bear market of 2019. The basis is negative when the perpetual swap trades below the cash index it tracks. A negative basis in crypto is a reflection of participants reducing risk and is exacerbated by forced liquidations.

where ζ is a basis cap, typically five basis points (0.05%). When the basis is positive, traders with long positions (buy XBT, sell USD) will pay those with short positions (sell XBT, buy USD). Reciprocally, shorts pay longs when the basis is negative.

6.3 The research experiment

In our experiment, we perform feature representation transfer from an ESN to a DRL agent. This meta-model aims to learn to trade digital asset futures on the BitMEX crypto exchange, specifically perpetual contracts. The dynamical reservoir of the ESN acts as a powerful nonlinear feature space; this is fed into the upstream DRL agent, which is aware of the various sources of impact on PNL and learns to target the desired position.

6.3.1 The recurrent reinforcement learning crypto agent

We begin by describing the ESN's dynamic reservoir feature space, the resultant learning of which is transferred to the DRL agent, which targets the desired risk position. Figure 6.2 shows a schematic of the ESN that we combine with a DRL agent. The ESN part is inspired by the schematic of Lukoševičius (2012), and the DRL part is inspired by the McCulloch-Pitts schematic of Bishop (1994). The schematic demonstrates visually that the target labels, the so-called teacher signal, can be fed back into the dynamic reservoir. Equally, one could apply a regression layer to the ESN and loop the resulting forecasts into the ESN reservoir. Both approaches lead to a recurrent ESN. Rather than treating this exercise as a value function estimation task as with Szita et al. (2006), we feed the augmented, dynamic reservoir features of the ESN to a DRL agent. By differentiating a quadratic utility function with respect to the DRL agent's parameters, with feedback connections from the agent's past positions fed back into the ESN dynamic reservoir, the agent learns from the various sources of impact on PNL and targets the appropriate position that maximises the risk-adjusted reward.

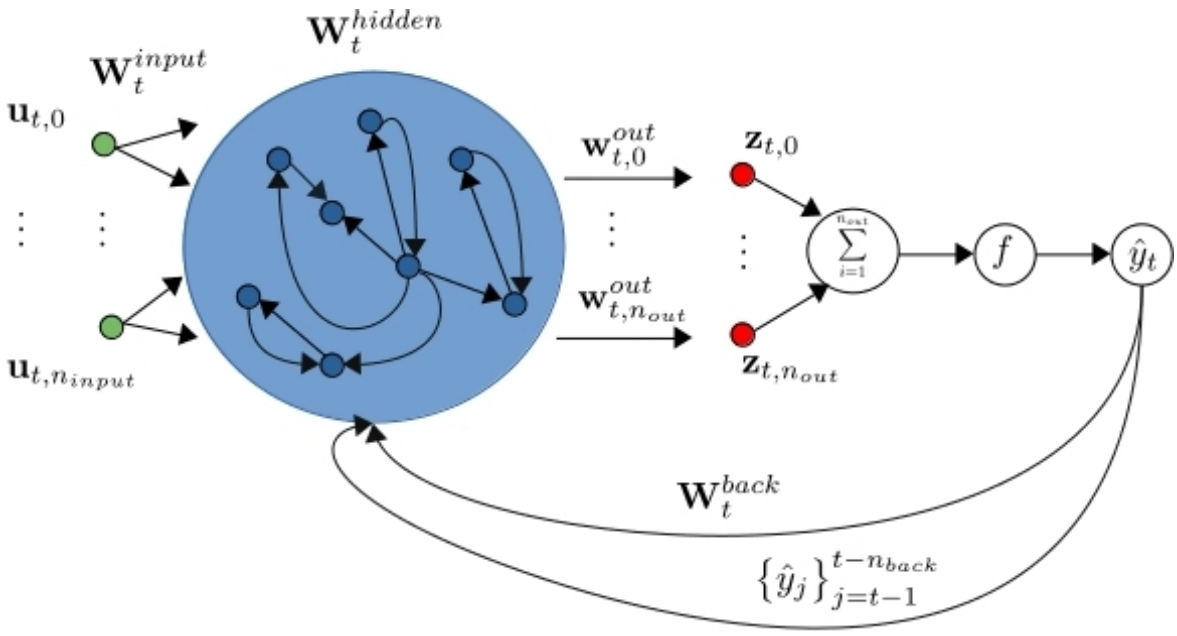


Figure 6.2: We plot a schematic of feature representation transfer from an echo state network to a direct, recurrent reinforcement learning agent.

6.3.1.1 The dynamic reservoir feature space

Denote as $\mathbf{u}_t$, a vector of external inputs to the system, which is observed at time t. In the context of this experiment, such external input would include order book, transaction and funding information. These features may come from the instrument being traded, exogenous instruments, or both. Initialise the external input weight matrix $\mathbf{W}^{input} \in \mathbb{R}^{n_{hidden} \times n_{input}}$, where the weights are drawn at random; a draw from a standard normal would suffice. Here, n_{hidden} denotes the number of hidden units in the internal dynamical reservoir and n_{input} is the number of external inputs, including a bias term. Next, initialise the hidden units weight matrix $\mathbf{W}^{hidden} \in \mathbb{R}^{n_{hidden} \times n_{hidden}}$. The procedure detailed by Yildiz et al. (2012) is

- Initialise a random matrix $\mathbf{W}^{hidden}$, all with non-negative entries.
- Scale $\mathbf{W}^{hidden}$ such that its spectral radius $\rho(\mathbf{W}^{hidden}) < 1$.
- Change the signs of the desired number of entries of $\mathbf{W}^{hidden}$ to get negative connection weights.
- Sparsify $\mathbf{W}^{hidden}$ with probability $P(\beta)$, where $0 \ll \beta < 1$, setting those elements to zero.

This procedure is guaranteed to ensure the echo state property for any input. Intuitively, a recurrent neural network (rnn) has the echo state property concerning an input signal $\mathbf{u}_t$, if any initial network state is forgotten or washed out when the network is driven by $\mathbf{u}_t$ for long enough (Jaeger, 2017). The model supports recurrent connections from either a teacher signal $\mathbf{y}_t \in \mathbb{R}^{n_{back}}$ or model output $\hat{\mathbf{y}}_t \in \mathbb{R}^{n_{back}}$. These are connected to the model via the weight matrix $\mathbf{W}^{back} \in \mathbb{R}^{n_{hidden} \times n_{back}}$, whose weights are initialised at random from a standard normal. Note that $\mathbf{W}^{input}$, $\mathbf{W}^{hidden}$ and $\mathbf{W}^{back}$, have weights that remain fixed.

Finally, initialise the output weight vector $\mathbf{w}_0^{out} \in \mathbb{R}^{n_{input}+n_{hidden}+n_{back}}$. At this point, our procedure differs from the original ESN formulation shown by Jaeger (2001). There, $\mathbf{w}^{out}$ is a matrix $\mathbf{W}^{out} \in \mathbb{R}^{n_{back} \times (n_{input}+n_{hidden}+n_{back})}$ and the performance measure of the model is the quadratic loss

$$\min_{\mathbf{W}^{out}} \frac{1}{2T} \sum_{t=1}^{T} (y_t - \mathbf{W}^{out} \mathbf{z}_t)^2.$$

The reasons will soon appear when we detail the model's DRL part. But first, we describe how we create the augmented system state, $\mathbf{z}_t \in \mathbb{R}^{n_{input}+n_{hidden}+n_{back}}$. Firstly, initialise a zero-valued internal state vector $\mathbf{x}_0 \in \mathbb{R}^{n_{hidden}}$. At time t, compute the recurrent internal state

$$\mathbf{x}_t = f_{hidden}(\mathbf{W}^{input}\mathbf{u}_t + \mathbf{W}^{hidden}\mathbf{x}_{t-1} + \mathbf{W}_t^{back}\hat{\mathbf{y}}_t),$$

where $f_{hidden}(.)$ is typically a squashing function such as tanh. The augmented, recurrent system state is then

$$\mathbf{z}_t = [\mathbf{u}_t, \mathbf{x}_t, \hat{\mathbf{y}}_t]. \tag{6.3}$$

Equation 6.5 defines $\hat{\mathbf{y}}_t$ as the past desired positions of the DRL agent.

6.3.1.2 Direct recurrent reinforcement learning

The augmented, internal feature state, $\mathbf{z}_t$, is now fed into the upstream model, a DRL agent, whose performance measure is a quadratic utility function of reward and risk. The DRL agent learns the desired risk position identically to the procedure detailed in section 5.3.1. The net returns whose expectation and variance we seek to learn are decomposed similarly

$$r_t = \Delta p_t f_{t-1} - \delta_t |\Delta f_t| - \kappa_t f_t, \tag{6.4}$$

except that κ_t is the funding cost of section 6.2. The model is maximally short when $f_t = -1$ and maximally long when $f_t = 1$. The past positions of the model are used as the feedback connections for equation 6.3

$$\hat{\mathbf{y}}_t = [f_{t-n_{back}}, ..., f_{t-1}]. \tag{6.5}$$

The goal of our DRL agent is to maximise the utility in equation 5.2, by targeting a position in equation 5.9. To do this, we apply an online optimisation update of the form

$$\mathbf{w}_t^{out} = \mathbf{w}_{t-1}^{out} + \nabla \upsilon_t \equiv \Delta \mathbf{w}_t^{out} + \frac{d\upsilon_t}{d\mathbf{w}_t^{out}},$$

where, the weight update procedure is an extended Kalman filter (EKF) update, shown in algorithm 5.1.

6.3.2 Experiment design

We put a DRL crypto agent to work by trading the XBTUSD perpetual swap on BitMEX. We transfer the output of the source model, the dynamic reservoir feature space of section 6.3.1.1, to the target model, the direct DRL agent of section 6.3.1.2, who learns to target a risk position directly. Finally, we use five minutely sampled intraday data. The choice of this sampling rate is driven by the throttle imposed by the vendor on retrieving historical data; if we could obtain the raw, asynchronously delivered tick data promptly, we would do so. Nevertheless, we are still using $365 \times 5 \times 1440/5 = 525600$ observations in our experiment. Our performance evaluation procedure involves the following:

- Construct input features from the order book and trade information made available by BitMEX for XBTUSD.
- Feed these input features into an ESN, with $n_{hidden} = 100$, $n_{back} = 10$ and the percentage of reservoir units $\mathbf{W}^{hidden}$ that are sparsified, set to $\beta = 0.75$.
- Feed the output of the ESN (equation 6.3) into a DRL agent (section 6.3.1.2).
- Set the risk appetite constant $\lambda = 0.00001$ for quadratic utility equation 5.2.
- Set the ridge penalty $\alpha = 1$ and the exponential decay factor $\tau = 0.999$ for the EKF of algorithm 5.1.
- Backtest the entire history as a test set, learning online to target the desired position.
- Force the agent to trade as a price taker, who incurs an execution cost equal to equation 5.8 plus exchange fees, which are set to 5 basis points (0.05%).
- For non-zero risk positions, apply the appropriate funding PNL as per equation 6.2.
- Monitor equation 5.3, the expected net reward of the strategy. The agent may trade freely if $\mu_t \geq 0$; otherwise, it must sit out of the market.

6.3.3 Results

Table 6.1 and figure 6.3 show the results of the experiment. The crypto agent achieves a total return of around 350% over a test set that is less than five years. The associated annualised

information ratio (IR) is 1.46. We see that the agent averages a position of $\mathbb{E}[f_t] = 0.41$. Thus there is a bias toward the agent maintaining a long position, which is desirable, as Bitcoin has appreciated against the US Dollar over this period. We see visual evidence in figure 6.3 that on occasion, the crypto agent abstains from trading, or rather is forced to take no position; this will happen during periods when the predictive performance of the agent decreases relative to execution and funding costs. Our crypto agent also captures a 71% cumulative return due to earning funding, which is expected as the agent learns to target the appropriate position that maximises its quadratic utility. Funding is one of the drivers of this utility. The total execution cost and exchange fees the agent pays out is -54%.

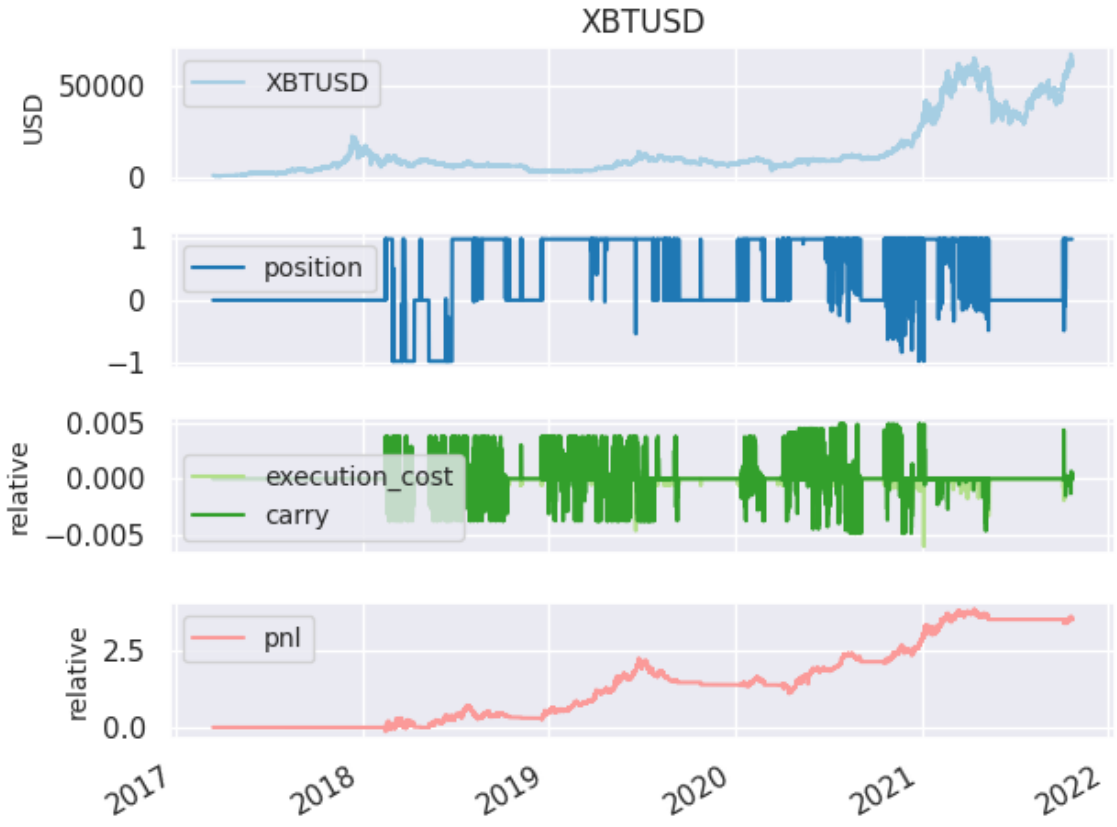


Figure 6.3: We plot cumulative returns for our XBTUSD crypto agent. Although the model averages a mostly long position, it goes short where necessary. Also, it resorts to staying out of the market if it does not have a solid signal indicating to get back in.

Table 6.1: We display daily profit and loss (PNL) statistics for our transfer learning, DRL XBTUSD crypto agent. Our agent averages mostly a long position, which is natural as Bitcoin has appreciated relative to the US Dollar during the test period. We find evidence that our crypto agent can successfully target a positive funding PNL and capture the PNL associated with price trends.

	position	execution	carry	PNL
count	1684	1684	1684	1684
mean	0.405	-0.000	0.000	0.002
std	0.579	0.001	0.005	0.027
sum		-0.542	0.713	3.498
IR				1.46

6.4 Discussion

The ESN provides a robust and scalable feature space representation. We transfer this learning representation to a DRL agent that learns to target a position directly. It is possible to use the ESN as a reinforcement learning agent directly, as shown by Szita et al. (2006). However, the approach may lead to undesired behaviour in a trading context. Specifically, they use an ESN to estimate the state-action value function of a temporal difference learning sarsa model (Sutton, 1988; Sutton and Barto, 2018). This value function takes the form

$$q_\pi(s,a) = \mathbb{E}_\pi\{r_t|s_t,a_t\},$$

where the expected return r_t depends on the transition to state s_t having taken action a_t under policy π. Sarsa estimates this value function sequentially as

$$q(s_t,a_t) = (1-\eta)q(s_t,a_t) + \eta[r_{t+1} + \gamma q(s_{t+1},a_{t+1})], \tag{6.6}$$

where $0 < \gamma \leq 1$ is a discount factor for multi-step rewards, and $\eta > 0$ is a learning rate. Equation 6.6 shows that the state transition reward is passed back to the starting state, which is perfectly desirable in activities such as maze traversal or board games. However, in the context of trading, where the value function $q(s_t,a_t)$ represents the value of a position s_t, with the possibility of switching or remaining in the same position denoted by action a_t, we will on occasion find that a larger utility is assigned to the wrong state. For example, imagine the current state is $s_t = 0$; the model has no position. Now we observe a large positive price

jump leading to reward $r_{t+1} \gg 0$. Equation 6.6 passes the state transition reward r_{t+1} to the initial state $q(s_t = 0, a_t = 0)$. At the next iteration, with probability $Pr(1-\varepsilon)$, $s_{t+1} = 0$ as $q(s_{t+1} = 0, a_{t+1} = 0)$ is the highest value function. However, if the position is zero, the model cannot hope to earn a profit. Even if one excludes the zero state, there is still the possibility of observing this problem for a reversal strategy with possible states $s = \{-1, 1\}$. Our DRL agent does not suffer from these problems.

In assessing the DRL agent's limitations, figure 6.4 and table 6.2 measure the impact of ESN initialisation on test set performance. We run a Monte Carlo simulation of 250 trials, with the network parameterisation fixed as before. There is evidence of the variability of total and risk-adjusted returns, however the distribution of returns is skewed to the right.

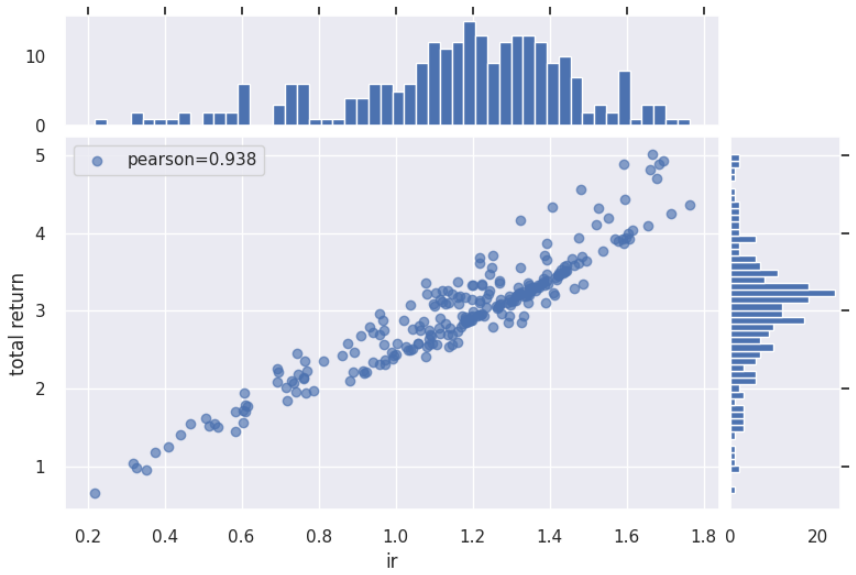


Figure 6.4: A scatter plot of IR versus total return. Each circle represents an outcome from a Monte Carlo simulation of 250 trials which assesses the impact of ESN initialisation on our DRL crypto agent.

Table 6.2: Summary statistics relating to the figure 6.4.

	IR	total return
mean	1.160	2.977
std	0.299	0.740
25%	1.023	2.572
50%	1.199	3.076
75%	1.355	3.352

Chapter 7

Sequential asset ranking in nonstationary time series

We create an online learning long/short portfolio selection algorithm that can detect the best and worst performing portfolio constituents that change over time; in particular, we successfully handle the higher transaction costs associated with using daily-sampled data, and achieve higher total and risk-adjusted returns than the long-only holding of the S&P 500 index with hindsight.

Our particular modelling interest is in financial time series, which are typically nonstationary. Nonstationarity implies statistical distributions that adapt over time and violates the independent and identically distributed (iid) random variables assumption of most regression and classification models. We require approaches that adopt sequential optimisation methods, preferably methods that make little or no assumptions about the data-generating process. The prediction with expert advice framework (Cesa-Bianchi and Lugosi, 2006) is a multidisciplinary area of research suited to predicting sequences sequentially, where statistical distribution assumptions are not made. This framework minimises the regret concerning the best available expert with hindsight and is well-suited to portfolio selection problems.

The main result of this chapter is our novel ranking algorithm, the naive Bayes asset ranker, which we use to select subsets of assets to trade from the S&P 500 index in either a long-only or a long/short (cross-sectional momentum) capacity. Our ranking algorithm forecasts the one-step-ahead sequential posterior probability that individual assets will be ranked higher than other constituents in the portfolio. Earlier algorithms, such as the weighted majority algorithm (Littlestone and Warmuth, 1994), deal with nonstationarity by ensuring the weights assigned to each expert never dip below a minimum threshold without ever increasing weights again. In contrast, our ranking algorithm allows experts who performed poorly

previously to have increased weight when they start performing well. Finally, our algorithm computes the posterior ranking probabilities with exponential decay and is better suited to learning in nonstationary environments.

We achieve higher risk-adjusted and total returns than a strategy that would hold the long-only S&P 500 index with hindsight, despite the index appreciating by 205% during the test period. We also outperform a regress-then-rank baseline, a sequentially fitted curds and whey multivariate regression model.

7.1 Problem formulation

We consider the problem of selecting so-called cross-sectional momentum portfolios using constituents of the S&P 500 index. Cross-sectional momentum portfolios are effectively long/short portfolios, comprising of a subset of assets that are expected to increase in value, and a subset of assets that are expected to decrease in value. The long/short portfolios facilitate trading during bull (increasing asset prices) and bear (decreasing asset prices) markets, and more generally, ameliorate the sensitivity to overall market moves. We select the long/short portfolios using our focus model, the naive Bayes asset ranker (NBAR), discussed in section 7.2. As section 7.3 and schematic 7.1 shows, there are in fact three parts to our focus model setup. We use the daily returns of the S&P 500 constituents as both external input features, denoted $\mathbf{x}_t \in \mathbb{R}^d$, and supervised learning targets, denoted $\mathbf{y}_{t+1} \in \mathbb{R}^d$. These daily returns are provided as external inputs to a solitary, single-layer radial basis function network (RBFNet) comprised of five hundred hidden units. Each hidden unit is parameterised as a multivariate normal

$$\begin{aligned}\phi_k(\mathbf{x}_t) &= \exp\left(-\frac{1}{2}[\mathbf{x}_t - \boldsymbol{\mu}_{k,t}]^T \boldsymbol{\Lambda}_{k,t}[\mathbf{x}_t - \boldsymbol{\mu}_{k,t}]\right), \quad \forall k = 1,...,K \\ \boldsymbol{\phi}_t &= [1, \phi_1(\mathbf{x}_t), ..., \phi_K(\mathbf{x}_t)]^T,\end{aligned}$$

via k-means, where $\boldsymbol{\phi}_t$ denotes the hidden units of the single-layer RBFNet. Each $\phi_k(\mathbf{x}_t)$ is parameterised as a k-means cluster conditional multivariate normal $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$. The one-step-ahead target returns are estimated as

$$\mathbf{y}_{t+1} \approx \hat{\mathbf{y}}_t = f(\boldsymbol{\phi}_t; \boldsymbol{\Theta}_t) + \mathbf{E}_t,$$

where $\mathbf{\Theta}_t$ denotes the curds and whey (CAW) multivariate regression parameters and $\mathbf{E}_t$ denotes a covariance error matrix. The CAW model is discussed in section 2.5. Finally, the CAW output $\hat{\mathbf{y}}_t$ is ranked via the NBAR; a complete specification is shown in algorithm 7.1. We consider risk-adjusted returns as the most important measure, and compare these risk-adjusted returns to a passive, long-only baseline which purchases the index constituents with equal weighting.

7.2 The naive Bayes asset ranker

Our ranking algorithm is the NBAR, which is succinctly displayed in algorithm 7.1. The algorithm sequentially ranks a set of experts; it does so by forecasting the one-step-ahead posterior probability that individual experts will be ranked higher than the set of available experts at its disposal. In the context of the experiment described in section 7.3, each expert is a forecasted return for an individual portfolio constituent of the S&P 500. The forecasted returns come from the CAW multivariate regression model (algorithm 2.2), which utilises feature representation transfer from the constituent S&P 500 returns to RBFNets whose k-means++ (Arthur and Vassilvitskii, 2007) clusters form hidden units. Assume that the algorithm is presented with a set of q forecasts. The goal is to select a subset of experts $1 \leq k \leq q$ such that the reward of the k experts is expected to be the highest; this is achieved by estimating the sequential posterior probability that expert $j \in 1, ..., q$ is ranked higher than each of the remaining $q-1$ experts. This posterior probability is computed with exponential decay, allowing experts who performed poorly and now perform well to be selected with greater weight than previously. The inputs to the algorithm are $\beta = k/q$, with $k \leq q$, the β-percentile subset of assets to trade, and τ, an exponential half-life. Denote as

$$p(\mathbf{r}_{j,t} \geq \mathbf{r}_t) = p(\mathbf{a}_t) = \prod_{i=1}^{q} p(\mathbf{r}_{j,t} \geq \mathbf{r}_{i,t}),$$

the probability that the forecasted returns of asset j will be ranked higher than the q assets considered, where $\mathbf{r}_t \in \mathbb{R}^q$ is the vector of forecasts at time t. Furthermore, denote as $p(\mathbf{r}_{j,t} > 0) = p(\mathbf{b}_t)$, the probability that asset j has a forecasted return at time t that is greater than zero; this condition is required so that we do not naively select k assets to go long if there are fewer assets with expected positive returns. The sequential posterior probability that

algorithm 7.1 computes is

$$p(\mathbf{a}_t|\mathbf{b}_t) = \frac{p(\mathbf{b}_t|\mathbf{a}_t)p(\mathbf{a}_t)}{p(\mathbf{b}_t|\mathbf{a}_t)p(\mathbf{a}_t) + p(\mathbf{b}_t|\mathbf{a}_t^c)p(\mathbf{a}_t^c)}. \tag{7.1}$$

Finally, the algorithm returns the set of indices that would sort $p(\mathbf{a}_t|\mathbf{b}_t)$ from largest to smallest.

Algorithm 7.1 The naive Bayes asset ranker

Input: $\mathbf{r}_t$, τ, β

Initialise: $\mathbf{s} = \mathbf{1}_q$, $\mathbf{S} = \mathbf{1}_{q\times q}$, $\mathbf{p} = \mathbf{0}_q$, $n = 0$, $k = \lfloor \beta q \rfloor$

// $\mathbf{r}_t$ are the forecast one-step-ahead daily returns from the CAW algorithm 2.2

// τ is an exponential decay constant

// β is the maximum percentile of experts that can be chosen

Output: $\mathbf{z}_t = argSort(\mathbf{p}_t)$

if $\tau = 1$ **then**
  $n = n + 1$ $w_t = (n-1)/n$
else
  $w_t = \tau$

// $I(.)$ is an indicator function that returns 1 for a true condition, or else 0

for $j \leftarrow 1$ **to** q **do**
  $\mathbf{s}_{j,t} = w_t \mathbf{s}_{j,t} + (1-w_t) I\left(\mathbf{r}_{j,t} \geq 0\right)$ **for** $i \leftarrow 1$ **to** q **do**
    $\mathbf{S}_{ij,t} = w_t \mathbf{S}_{ij,t} + (1-w_t) I\left(\mathbf{r}_{j,t} \geq \mathbf{r}_{i,t}\right)$

Initialise zero vectors $\mathbf{a} = \mathbf{b} = \mathbf{0}_q$

for $j \leftarrow 1$ **to** q **do**
  $\mathbf{b}_j = \mathbf{s}_{j,t} / \sum_{i=1}^{q} \mathbf{s}_{i,t}$
  $\mathbf{a}_j = \frac{\sum_{i=1}^{q} \mathbf{S}_{ij,t}}{\sum_{h=1}^{q}\sum_{k=1}^{q} \mathbf{S}_{hk,t}}$

$\mathbf{a}^c = 1 - \mathbf{a}$

$\mathbf{p}_t = \frac{\mathbf{ba}}{\mathbf{ba} + \mathbf{ba}^c}$

// $argSort(.)$ returns the indices that would sort an array from largest to smallest value

$\mathbf{z}_t = argSort(\mathbf{p}_t)$

Denote $\mathbf{z}_{j,t} = j^*$, the test time return is $\sum_{j=1}^{k} \mathbf{r}_{j^*,t+1} \mathbf{p}_{j^*,t}$.

7.3 The research experiment

Our research experiment aims to assess the benefits of sequentially optimised ranking algorithms to select subsets or portfolios of financial assets to hold in either a long-only or long/short (cross-sectional momentum) capacity. More concretely, we experiment with the

constituents of the S&P 500 index. We use our NBAR algorithm as the sequentially optimised ranker, with the posterior ranking probabilities of equation 7.1 estimated continuously during the test set. The NBAR inputs are the one-step-ahead predicted daily returns estimated by the CAW multivariate regression model. In turn, the CAW model collects individual one-step-ahead predicted daily returns from RBFNets, one per S&P 500 constituent. The RBFNets have hidden units whose means are determined via k-means++; thus, we perform feature representation transfer from external inputs, the S&P 500 constituent daily returns, to hidden unit outputs determined by clustering algorithms. These hidden unit outputs are mapped to the response, individual constituent one-step-ahead daily return forecasts, using exponentially weighted recursive least squares (EWRLS). A schematic in the form of graphical probability models is shown in figure 7.1.

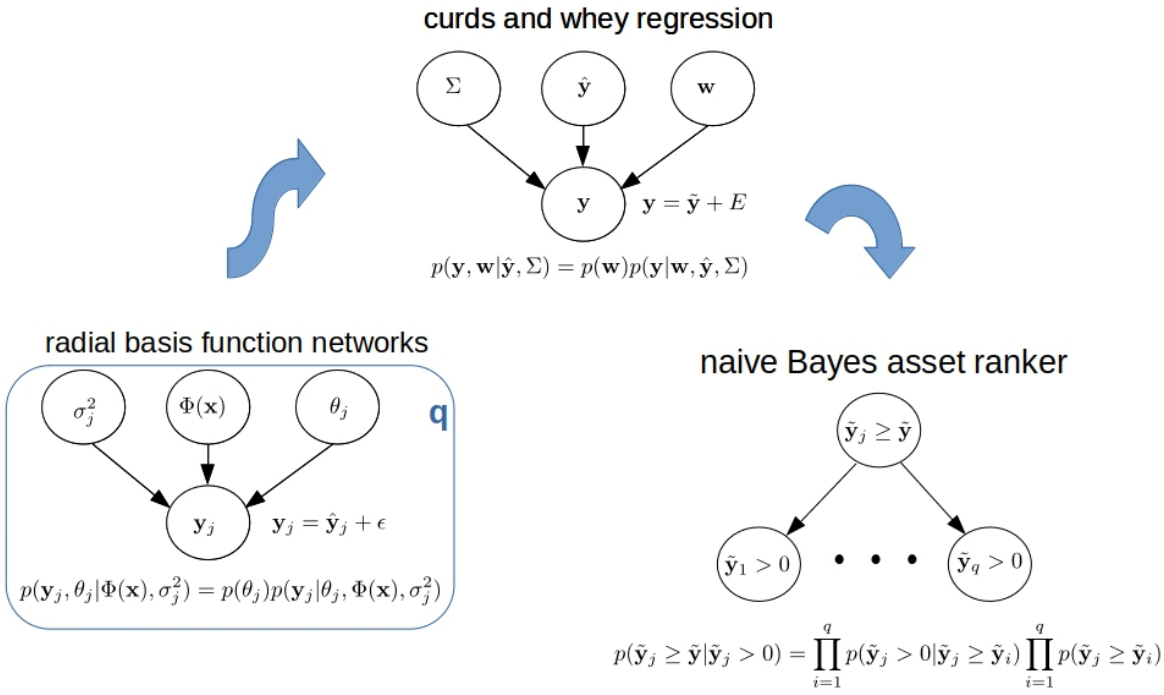


Figure 7.1: The RBFNet forecasts are fed into the CAW multivariate regression model, whose output is ranked and selected by the NBAR.

At first glance, this meta-model setup might seem overly or unnecessarily complicated. We could, for example, use the raw S&P 500 constituent daily returns as inputs to the NBAR. However, we are building on the research of regress-then-rank algorithms in the context of portfolio selection (Jegadeesh and Titman, 1993; Gu et al., 2020), which motivates our use of the RBFNets. We are also building on the work of Breiman and Friedman (1997) to take

advantage of the correlations between the response variables to improve predictive accuracy compared with the usual procedure of doing individual regressions of each response variable on the shared set of predictor variables. We use their CAW procedure but combine it with EWRLS to facilitate sequential optimisation in the test set without forward-looking bias. Finally, we make use of online learning RBFNets as these models retain more remarkable knowledge of the input feature space. They also respond better to regime changes or concept drifts than models that do not use feature representation transfer.

7.3.1 Baseline models

In order to assess the true value of the meta-model shown in figure 7.1, we adopt two baseline models. The first model intuitively is the long-only holding of the S&P 500 constituents with equal weighting and replicates a passive, index-tracking investment strategy. A second baseline is our proxy for the regress-then-rank models, the sequentially optimised CAW multivariate regression model, algorithm 2.2. This baseline also uses the individual predicted daily returns from online learning RBFNets.

7.3.2 The S&P 500 dataset

We conduct this research experiment using the daily closing constituent prices for the S&P 500 index, which we extract from Refinitiv. Due to their relatively new trade history, some time series have little data. Therefore, we select a subset of the S&P 500 index, where each constituent contains a trade count greater than or equal to the 25'th percentile of trade counts; this leaves us with a subset of 378 Refinitiv information codes (rics). The dataset begins on 2001-01-26 and ends on 2022-03-25, 5326 days.

7.3.3 Experiment design

We use the first 25% of the data as a training set and the remaining data as a test set. In the training set, algorithms 2.2 and 7.1 are initialised and fitted. These models are also sequentially optimised without forward-looking bias in the test set. The hyperparameters that are set for this experiment are:

- Exponential decay, $\tau = 0.99$.
- Ridge penalty, $\alpha = 0.001$.

- Radial basis function networks with 500 hidden units determined by k-means++.

- Maximum percentile of assets to trade either long or short, $\beta = 0.05$.

Once the training data are assigned to their nearest cluster centres, the cluster-conditional covariance matrices and their inverses are estimated. Cluster centres with few training data vectors assigned to them are regularised to a diagonal variance prior. Thus, we are adopting a Bayesian maximum a posteriori procedure here.

We use the forecasts of the CAW model, algorithm 2.2, as the basis for taking risk in a subset of constituents in the S&P 500 index. Specifically, the long-only CAW model buys the expected top five per cent of performing assets $\tilde{\mathbf{y}}_t$. For example, if there are k assets in this top five percentile, then a weight of $1/k$ is applied per constituent. The long/short CAW model works similarly, except that it includes the short-selling of the bottom five per cent of most negative forecasts, with a weight of $-1/k$.

A second forecaster we consider is the NBAR algorithm 7.1, applied to the one-step-ahead forecasts of the CAW model. Algorithm 7.1 outputs $\mathbf{z}_t$, the set of indices that would sort $\mathbf{p}_t$ in descending order, which is the posterior probability of highest ranked assets. Denoting $\mathbf{z}_{j,t} = j^*$ and assuming there are k assets in the expected top five per cent of performing assets, the NBAR assigns a weight to the $j'th$ constituent $(1 \leq j \leq k)$ of

$$\mathbf{p}_{j^*,t} \Big/ \sum_{i^*=1}^{k} \mathbf{p}_{i^*,t}. \tag{7.2}$$

Similarly, for short positions, assuming there are k assets we wish to go short, the weight assigned to the $j'th$ constituent $(q-k \leq j \leq q)$ is

$$-\left(1-\mathbf{p}_{j^*,t}\right) \Big/ \left(\sum_{i^*=q-k}^{q} 1-\mathbf{p}_{i^*,t}\right). \tag{7.3}$$

We must also consider execution costs. We force the CAW and NBAR models to trade as price takers, meaning that the models incur a cost equal to half the bid/ask spread times the change in absolute position. Specifically, we apply the average transaction cost per S&P 500

constituent, whose distribution of relative basis point costs

$$bpcost_j = \frac{10000}{T} \sum_{t=1}^{T} \frac{ask_{j,t} - bid_{j,t}}{mid_{j,t}},$$

is shown in figure 7.2. Furthermore, as these data are sampled daily, any portfolio rebalanc-

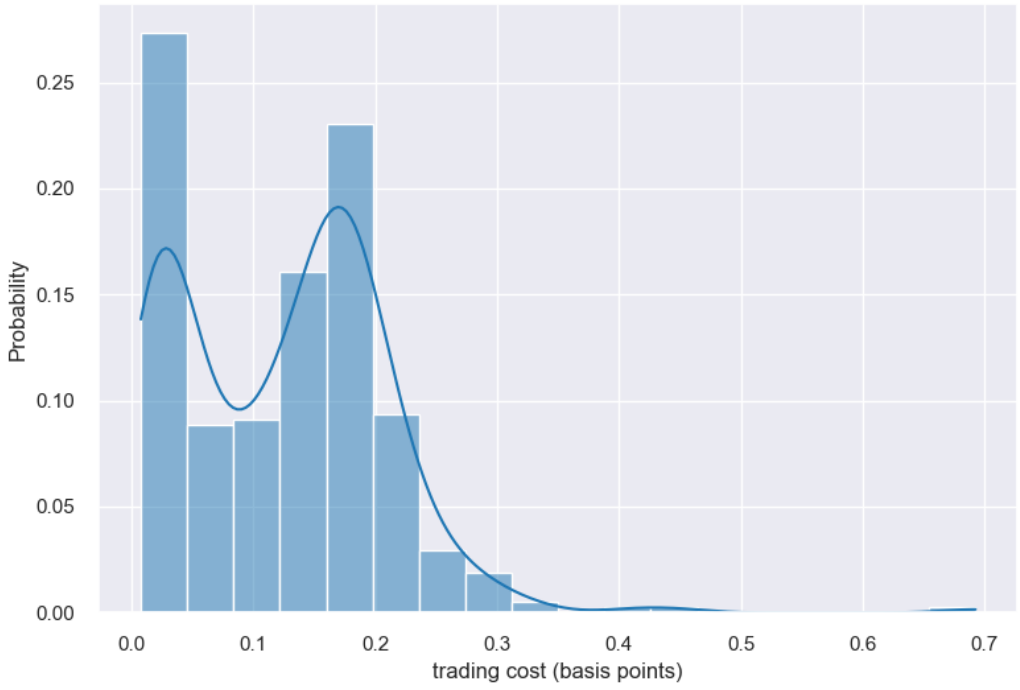


Figure 7.2: The distribution of transaction costs, where the distribution is taken over the average transaction cost per S&P 500 constituent.

ing is applied at most once a day, at the close of trading circa 4 pm EST.

7.3.4 Results

The passive index tracking baseline purchases each constituent with equal weighting at $t = 0$ and holds them till the end of the experiment. This strategy pays transaction costs once and therefore has the least fees, as shown in table 7.1. Table 7.2 and figure 7.3 also show that the cumulative returns generated by this strategy are 205%, the compound annual growth rate (cagr) is 7.3% and the risk-adjusted annualised Sharpe ratio (sr) is a little under 0.8. Assuming normally distributed returns, the Sharpe ratio implies a probability of positive annual returns of 71%. The largest peak-to-trough drawdown for the strategy is just under 72%, and the total return to maximum drawdown is around 2.9. Finally, by simply holding the in-

dex, the percentage of days with positive returns is 55%. The same performance metrics are available for the CAW and NBAR models. Both long-only and long/short CAW and NBAR models outperform the passive index tracking baseline, with the long/short models showing higher risk-adjusted performance measures indicated by the Sharpe ratios. The NBAR performs best, with the long/short NBAR showing the highest total and risk-adjusted returns. Table 7.1 shows that despite the CAW and NBAR models being actively managed strategies that rebalance the portfolios daily, only the CAW models show high transaction costs. The NBAR models rebalance less often and do a better job of picking portfolio constituents.

Figure 7.4 shows a bird's eye view of the NBAR cross-sectional momentum weights across time. Long positions show up as dark blue specks, and short positions show up as yellow specks. We see evidence of the NBAR dynamically shifting weights over time to find the best candidates to hold on a cross-sectional momentum basis, given the fixed constraint that a maximum of five per cent of total assets can be held long or short. The weight range in figure 7.4 indicates a relatively diffuse weight choice; in other words, no single constituent appears to dominate the others regarding predicted returns performance. We can zoom into a specific portfolio constituent, namely Electronic Arts Inc. In figure 7.5, we see that the long/short NBAR switches between long, short and flat positions as necessary, without an exponential decay of the weights permanently, as with the weighted majority algorithm.

Figure 7.6 displays the sensitivity of total returns and Sharpe ratios to the selection percentile, that is, the fraction of assets that are held in either a long-only or long/short manner. We draw similar conclusions with the fixed experiment that selects five per cent of the expected best-performing assets. The NBAR models perform best, and the long/short models perform better than their long-only counterparts. The total return increases as fewer assets are selected, particularly for the NBAR, which shows a $\times 5$ improvement over the baseline when trading just a pair of assets in a long/short manner. However, such a strategy is not scalable if a large amount of investment capital needs to be allocated to it. Furthermore, we have not modelled the trade impact that would invariably appear if we were executing prominent positions relative to each asset's average daily volume turnover. Figure 7.6 also shows that the Sharpe ratios increase toward a selection percentile of around five per cent and, depending on the model, decrease or plateau after that.

Table 7.1: Relative transaction costs incurred by each model in the test set. A buy-and-hold strategy on the S&P 500 achieves the lowest transaction costs. However, from the perspective of a more active portfolio management standpoint, our ranking algorithm incurs far lower transaction costs than the regress-then-rank baseline.

	transaction costs
long S&P 500	**-0.003**
long CAW	-0.933
long/short CAW	-1.966
long NBAR	-0.050
long/short NBAR	-0.104

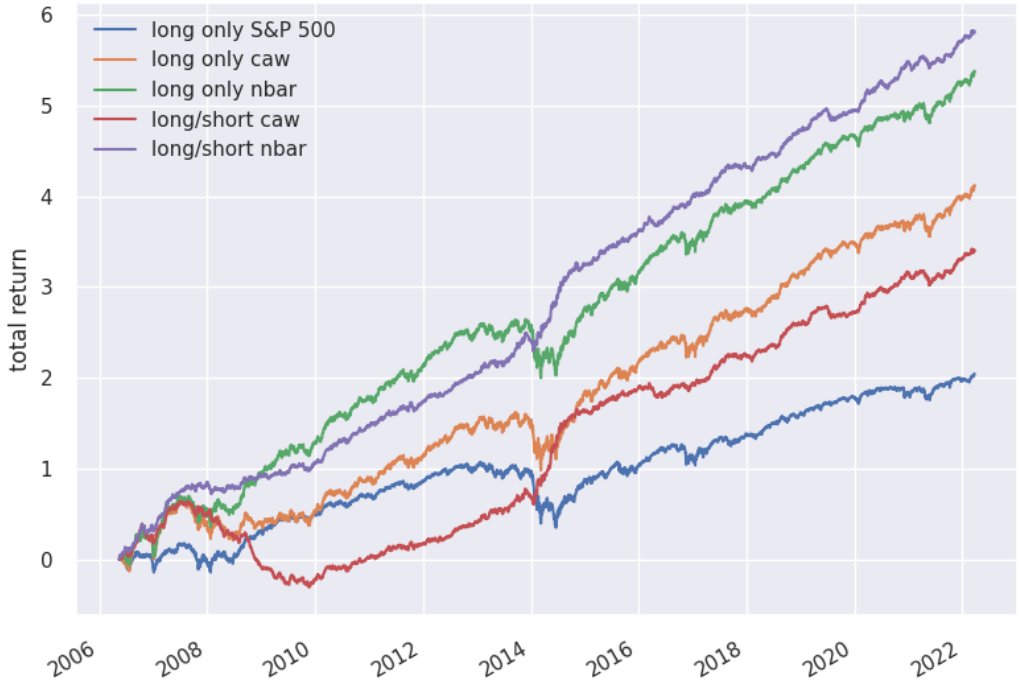


Figure 7.3: Total return by each model in the test set where the maximum selection percentile is set to 5% of the total number of portfolio constituents. The naive Bayes asset ranker performs best, particularly the cross-sectional momentum version.

Table 7.2: Summary returns statistics are shown in relation to the experiment, shown visually in figure 7.3. The cross-sectional momentum naive Bayes asset ranker has the highest total and risk-adjusted returns.

	long S&P 500	long CAW	long NBAR	long/short CAW	long/short NBAR
mean	0.0005	0.001	0.0013	0.0009	0.0015
std	0.012	0.016	0.016	0.010	0.010
total ret	2.047	4.113	5.372	3.397	**5.806**
cagr	0.073	0.108	0.124	0.098	0.128
sr	0.798	1.243	1.636	1.624	**2.879**
$pr(\text{ann. ret} > 0)$	0.71	0.816	0.895	0.893	0.994
max dd	0.717	0.64	0.646	0.942	0.202
total ret / max dd	2.853	6.423	8.311	3.607	28.7
win ratio %	0.549	0.553	0.563	0.547	0.575

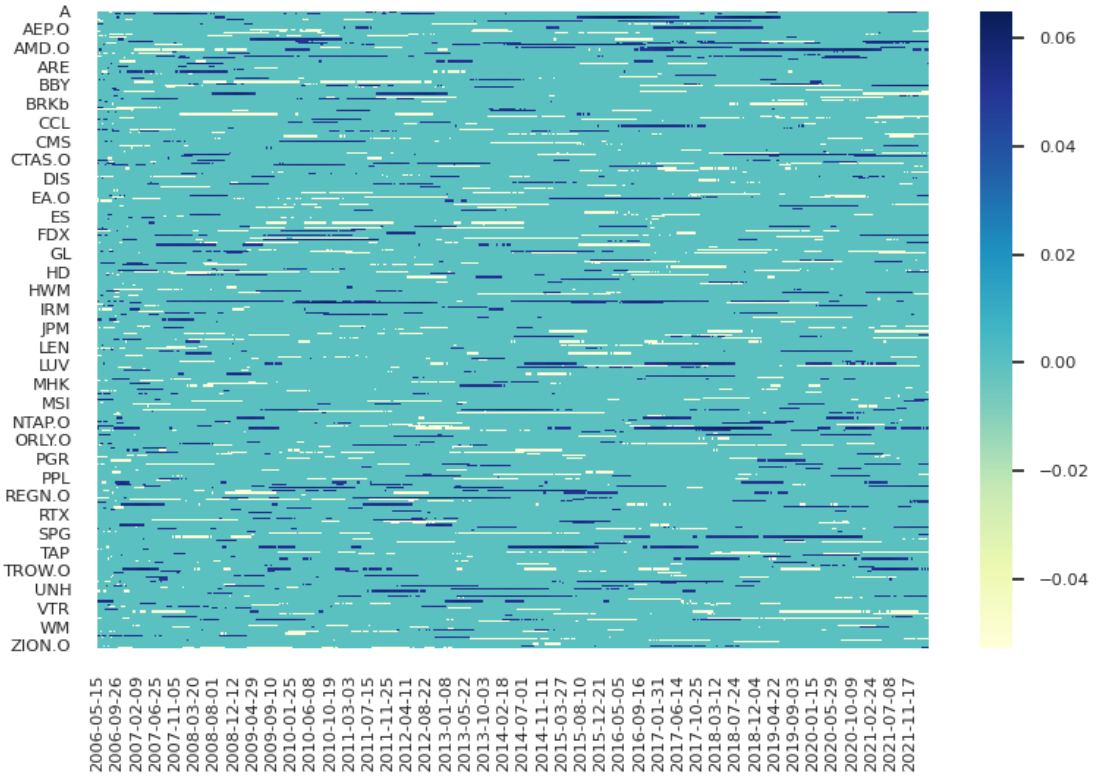


Figure 7.4: The NBAR cross-sectional momentum weights across time. We find visual evidence that the portfolio selection is dynamic and changing over time.

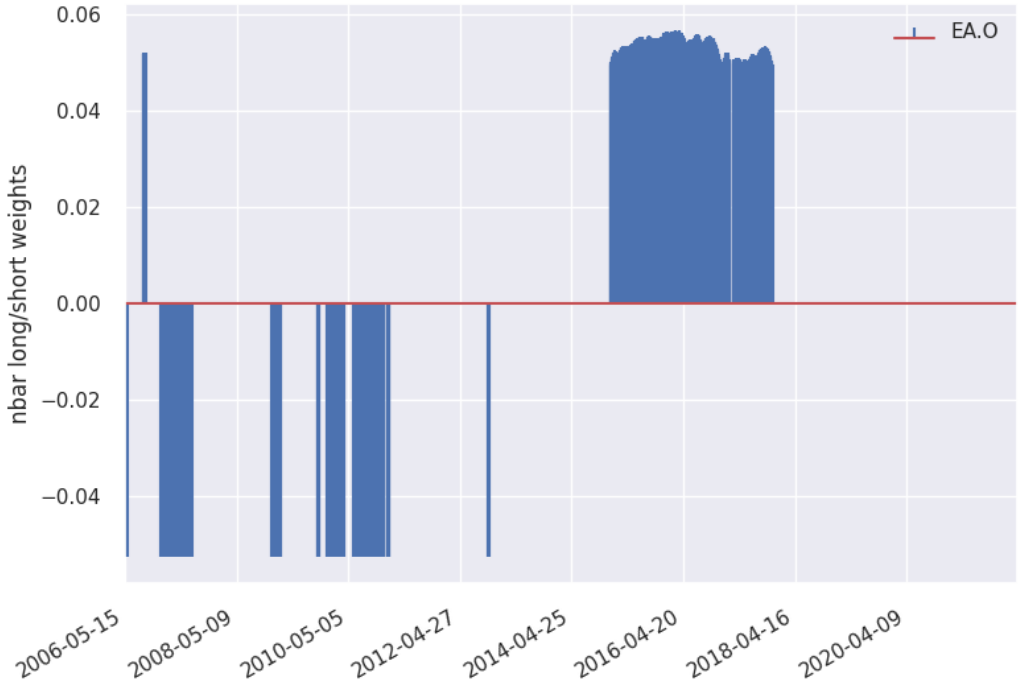


Figure 7.5: NBAR cross-sectional momentum weights across time for Electronic Arts Inc.

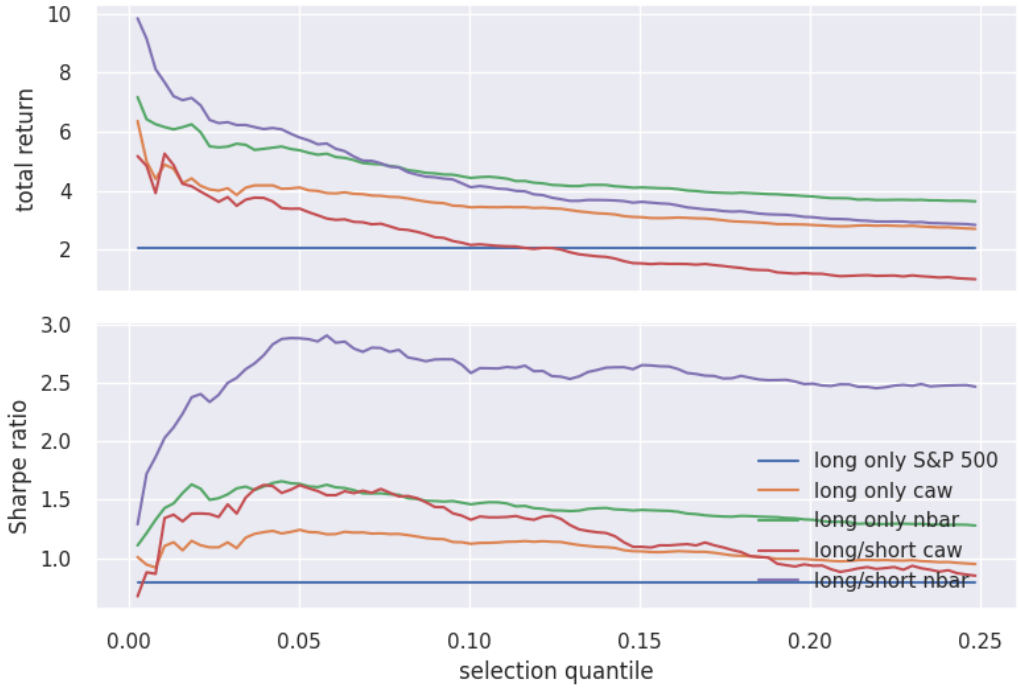


Figure 7.6: Test returns by model and selection percentile. Restricting the maximum selection percentile results in the highest total returns but is not particularly useful for portfolio managers that need to allocate substantial investment capital. The risk-adjusted returns for this test set peak near an upper-bound selection percentile of 5% of total constituents.

7.4 Discussion

How might we rationalise the outstanding performance of the NBAR relative to the long-only, passive baseline? Furthermore, why does the CAW perform worse than the NBAR? Several academic papers discuss the shortcomings of regression models compared to classification models in the financial time series prediction setting. For example, Satchell and Timmermann (1995) show that regression models that typically minimise prediction MSE obtain worse performance than a random-walk model when forecasting daily FX returns. Furthermore, they show that the probability of correctly predicting the sign of the change in daily FX rates is higher for the regression models than the random-walk baseline, even though the mse of the regression models exceeds that of the random-walk model. They conclude that mse is not always an appropriate performance measure for evaluating predictive performance. More recently, Amjad and Shah (2017) find that classical time series regression algorithms, such as ARIMA models, have poor performance when forecasting Bitcoin returns. However, they find that the probability distribution of the sign of future price changes is adequately approximated from finite data, specifically classification algorithms that estimate this conditional probability distribution. In summary, the NBAR could be interpreted easily as a classification model in that it endeavours to predict the posterior ranking probabilities; this task seems more beneficial in the context of long/short portfolio selection than regress-then-rank approaches.

Chapter 8

Conclusion and future work

We conclude this thesis with our final set of inferences based on the experiments conducted, briefly summarise our main contributions once more and end with directions for future research.

8.1 Conclusions

Financial time series exhibit the attributes of autocorrelation, nonstationarity and nonlinearity. Our experiment in chapter 4 demonstrates the added value of feature selection, nonlinear modelling and online learning when providing multi-horizon forecasts. Technically, by constructing returns, the time series become stationary as measured by unit root tests; therefore, offline batch learning is possible. However, we find experimental evidence to support the justified use of sequentially optimised radial basis function networks (RBFNets). The RBFNets obtain the best experiment results, which can be attributed primarily to their clustering algorithms, which learn the intrinsic nature of the feature space. The resulting hidden units provide predictive prototypes for unseen test data, which retain high similarity across time.

Our experiment on the major currency pairs in chapter 5 sees us construct a meta-model that uses feature representation transfer from Gaussian mixture models (gmms) to direct, recurrent, reinforcement learning (DRL) agents. These agents achieve higher funding and risk-adjusted returns than a momentum trader baseline, a supervised learning RBFNet. The momentum trader uses the same feature representation transfer methodology. However, our DRL agent maintains an essential advantage over the momentum trader. The agent's weights can be adapted to reflect the different profit and loss (PNL) variation sources, including returns momentum, transaction costs, and funding costs. We demonstrate this visually in figures 5.8a and 5.8b, where a USDRUB trading agent learns to target different positions

that reflect trading in the absence or presence of cost.

In chapter 6, we demonstrate that the echo state network (esn) provides a robust and scalable feature space representation. We transfer this learning representation to a DRL agent that learns to target a position directly. We argue in section 6.4 that direct reinforcement is likely to perform better than state-action value estimation approaches, mainly when the state represents an agent's position and the action represents the desire to switch positions. Furthermore, in markets that experience frequent regime changes, state-action value approaches may significantly underperform direct reinforcement approaches that target financial risk positions directly.

The main inference that can be drawn from chapters 5 and 6 is that DRL agents achieve good risk-adjusted returns, primarily as these models have adaptive learning capability. Specifically, learning is adapted in response to impacts on the utility functions they maximise. These impacts include funding cost, execution cost or trends in the financial time series. However, it would not be possible to obtain consistently good performance without performing feature representation transfer from either clustering algorithms, mixture models or ESNs. Finally, combining this additional learning with sequential optimisation in the test set is essential, as is the choice of the optimisation method. We use a robust second-order optimisation method which relies on an extended Kalman filter.

Our final experiment is detailed in chapter 7. The experimental objective is to outperform a baseline model that holds the S&P 500 index constituents with equal weighting. Historically, this task has been tackled using so-called regress-then-rank models, which select a subset of assets to hold using regression-estimated returns that provide the ranked portfolio assets. Our experiment employs a robust regress-then-rank baseline, the curds and whey (CAW) model, which takes as external input the one-step-ahead predictions of large RBFNets formed by k-means++ clusters. The risk-adjusted returns of these baselines are compared with those of our novel ranking algorithm, the naive Bayes asset ranker (NBAR). Our ranking algorithm uses the CAW outputs as estimates for predicted asset returns. These predicted returns are then ranked by estimating the sequential probability that individual assets will achieve higher positive returns than the remaining portfolio constituents. Our ranking algorithm achieves the best results, particularly the long/short portfolio version. The

performance holds for a wide range of β-quantile subsets of assets to trade. We can think of the output of our ranking algorithm as experts we wish to follow. Unlike earlier algorithms, such as the weighted majority algorithm, our ranking algorithm is better suited to learning in nonstationary data. This ability to operate in nonstationary environments is achieved by computing the posterior ranking probabilities with exponential decay. Therefore, experts who previously performed poorly have the chance to have more prominence when they start performing well.

8.2 Summary of contributions

We wish to understand if other forms of learning can be combined with sequential optimisation to provide superior benefit over batch learning in various tasks operating in financial time series, including multi-horizon forecasting, proprietary trading and portfolio selection.

In section 2.1, we show that even if we use the stationary returns of the S&P 500 dataset (section 3.4), the batch learning models that operate on them have parameters that need to be constantly adapted. We use statistical tests such as the t-test for equal means to show that the further the distance in time between the model parameters being compared, the greater the probability is that these tests will reject the hypothesis that the parameters were generated from the same statistical distribution.

Section 4.4 demonstrates that our extensive use of feature representation transfer from clustering algorithms to the various upstream models is justified. For the Refinitiv cross-asset class dataset (section 3.1), we show that the training set returns have low similarity with the test set returns. In contrast, the training set cluster-derived hidden unit outputs retain greater similarity with their test set counterparts.

In chapter 4, **Online learning with radial basis function networks**, we experiment with the returns of financial time series, providing multi-horizon forecasts with a selection of powerful supervised learners. We devise an external input selection algorithm that maximises regression R^2, minimises feature correlations and can operate efficiently in a high-dimensional setting. We improve upon the earlier work on RBFNets, which applies feature representation transfer from clustering algorithms to supervised learners. We use sequentially optimised and regularised Bayesian maximum a posteriori precision matrices rather

than a randomised, scalar standard deviation for each hidden unit's radial basis function. Our online RBFNet outperforms a random-walk baseline, a historically tricky challenge, and the remaining models. The outperformance is not purely down to sequential updating in the test set; a competitor EWRLS model is updated similarly and performs less well than several batch-learners. Our RBFNets are naturally designed to measure the similarity between test samples and continuously updated prototypes that capture the characteristics of the feature space.

In chapter 5, **Reinforcement learning for systematic FX trading**, we perform feature representation transfer from an RBFNet formed of Gaussian mixture model hidden units to a direct, recurrent reinforcement learning (DRL) agent. This agent is put to work in an experiment, trading the major FX pairs. Earlier academic work that applied direct reinforcement to FX trading saw mixed results. These mixed performances come from weak external inputs to the DRL agent, first-order optimisation methods, and shared hyperparameters. We use more powerful features such as RBFNets and second-order optimisation methods such as extended Kalman filters and adapt our model parameters sequentially instead of using batch learning. As a result, our DRL agents cope better with statistical changes to the data distribution. Our DRL agent achieves higher risk-adjusted returns than a funding/carry baseline and a supervised learning momentum trader.

In chapter 6, **The recurrent reinforcement learning crypto agent**, we demonstrate a novel application of online transfer learning for a digital assets trading agent. This agent uses a powerful feature space representation in the form of an echo state network, the output of which is made available to a DRL agent. The agent learns to trade the XBTUSD perpetual swap derivatives contract on BitMEX on an intraday basis. Our meta-model can process data as a stream and learn sequentially; this helps it cope with the nonstationarity of the high-frequency order book and trading data. Furthermore, by using the vast high-frequency data, our model, which has a high learning capacity, avoids the kind of overfitting on a lack of data points that occurs with down-sampled data. We escape the problem of over-trading typically seen with the supervised learning model by learning the sensitivity of the change in risk-adjusted returns to the model's parameterisation. Stated another way, our model learns from the multiple impact sources on profit and loss and targets the appropriate risk position.

In chapter 7, **Sequential asset ranking in nonstationary time series**, we extend the academic research into cross-sectional momentum trading strategies. The main result of this experiment is our novel ranking algorithm, the naive Bayes asset ranker, which we use to select subsets of assets to trade from the S&P 500 index. Our ranking algorithm computes the sequential posterior probability that individual assets will be ranked higher than other constituents in the portfolio. Unlike earlier algorithms such as the weighted majority algorithm, which deals with nonstationarity by ensuring the weights assigned to each expert never dip below a minimum threshold, our ranking algorithm computes the posterior ranking probabilities with exponential decay, allowing experts who previously performed poorly to have increased weights when they start performing well. Our algorithm outperforms a strategy that would hold the long-only S&P 500 index with hindsight, despite the index appreciating by 205% during the test period. It also outperforms a regress-then-rank baseline, a sequentially fitted curds and whey multivariate regression model.

8.3 Future research

In chapter 4, our RBFNets are comprised of k-means++ clusters that transfer their learning to EWRLS supervised learners. In terms of future work, one could consider using other unsupervised learning approaches to learn the hidden units of the RBFNet. One such example is stacked denoising autoencoders (Vincent et al., 2010). Another option is variational Bayes expectation maximisation, which Murphy (2012) shows is sparsity promoting, typically requiring fewer clusters than the predefined set required for a GMM. Kingma and Welling (2014) combine autoencoders with variational Bayes.

In chapter 5, where we explore feature representation transfer from gmms to DRL agents that are sequentially optimised. One may improve the results by applying a portfolio overlay. The utility function of equation 5.2 is readily treated as a portfolio problem

$$\upsilon_t = \mathbf{h}^T \boldsymbol{\mu}_t - \frac{\lambda}{2}\mathbf{h}^T \boldsymbol{\Sigma}_t \mathbf{h},$$

where the optimal, unconstrained portfolio weights are obtained by differentiating the port-

folio utility with respect to the weight vector

$$\mathbf{h}^* = \frac{1}{\lambda}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t.$$

Another approach is to treat portfolio selection as a policy gradient problem, where the policy of picking actions, or in this case, portfolio constituents, is estimated via function approximation techniques.

As we discuss in chapter 5, our DRL agent comprises an unsupervised model and a reinforcement learning component. Specifically, we construct an RBFNet whose hidden units are determined by a Gaussian mixture model (GMM) procedure due to Figueiredo and Jain (2002) which has an identical expectation step to Dempster et al. (1977)'s em algorithm, but a modified log-likelihood function for the maximisation step. The procedure involves the construction of a large mixture and annihilates components from the mixture that are not supported by the data. This procedure is purely a batch-learning approach. An attractive alternative, which ties more closely to our philosophy of online learning in financial time series, is the incremental Gaussian mixture network (IGMN) approach of Pinto and Engel (2015). Like Pinto and Engel, we also sequentially update the mixture-conditional precision matrices of the GMM. However, Pinto and Engel take the GMM learning a step further and select mixture components incrementally. At test time, if the current mixtures have low similarity with the available external input vector, then a new mixture component is added. Adding mixture components incrementally when there is low external input similarity with existing prototypes seems like a sensible way to tackle concept drift (Gama, 2012) or covariate shift (Tibshirani et al., 2019).

In section 6.4, part of the experiment that performs feature transfer from ESNs to DRL agents, we discuss how the various esn parameters are initialised at random. We then perform a Monte Carlo simulation of 250 trials, with the network parameterisation fixed as before. We find evidence of the variability of total and risk-adjusted returns caused by the random initialisation of the various esn parameters. In the same way that k-means++ has improved the performance of k-means, more research could be done that improves esn initialisation, with a demonstrably positive impact on risk-adjusted returns.

Finally, in chapter 7, we use the NBAR's output to determine the amount of risk to

take. The risk taken is proportional to the probability that individual assets are ranked higher than the other portfolio constituents. This is shown in equations 7.2 and 7.3. The approach is reasonable and sensible. Furthermore, it works better than portfolio selection using a regress-then-rank approach. However, in terms of future work, one might compare the position sizes derived by the NBAR with position sizing algorithms such as the Kelly criterion or more general models for this purpose (Lototsky and Pollok, 2021).

Appendix A

Colophon

This LaTeX document was created with Overleaf. The references were stored in Mendeley, nicely integrated with UCL's online library services. The experiments were coded up in Python using the integrated desktop environment, Pycharm. Most of our experiment models were coded up from scratch, and occasionally we used some of the scikit-learn (Pedregosa et al., 2011) implementations. The RBFNet schematic, figure 2.2, was generated using Python and Graphviz (Isaksson, 2017). Figures 6.2 and 7.1 were generated using LibreOffice Impress and GNU Image Manipulation Program (GIMP).

Bibliography

Aboussalah, A. M. and Lee, C. (2020). Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. *Expert Systems with Applications*, 140:112891.

Abraham, B. and Ledolter, J. (1983). *Statistical methods for forecasting*. Wiley.

Agarwal, A., Hazan, E., Kale, S., and Schapire, R. E. (2006). Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 9–16, New York, NY, USA. Association for Computing Machinery.

Amjad, M. and Shah, D. (2017). Trading bitcoin and online time series prediction. In Anava, O., Khaleghi, A., Cuturi, M., Kuznetsov, V., and Rakhlin, A., editors, *Proceedings of the Time Series Workshop at NIPS 2016*, volume 55 of *Proceedings of Machine Learning Research*, pages 1–15, Barcelona, Spain. PMLR.

Anava, O., Hazan, E., Mannor, S., and Shamir, O. (2013). Online learning for time series prediction. In *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 172–184, Princeton, NJ, USA. PMLR.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Azhikodan, A. R., Bhat, A. G. K., and Jadhav, M. V. (2019). Stock trading bot using deep reinforcement learning. In Saini, H. S., Sayal, R., Govardhan, A., and Buyya, R., editors, *Innovations in Computer Science and Engineering*. Springer Singapore.

Bachelier, L. (1900). Théorie de la spéculation. *Annales Scientifiques de L'Ecole Normale Supérieure*, 17:21–88.

Barber, D., Taylan, C., and Silvia, C. (2011). *Bayesian Time Series Models*. Cambridge University Press.

Bengio, Y. (1997). Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, 08.

Betancourt, C. and Chen, W. (2021). Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, 164:114002.

Billings, S. A., Fung, C. F., , and Luo, W. (1996). *On-Line Supervised Adaptive Training Using Radial Basis Function Networks*. Department of Automatic Control and Systems Engineering.

Bishop, C. M. (1994). Neural networks and their applications. *Review of Scientific Instruments*, 65(6):1803–1832.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Bontempi, G. (2008). Long term time series prediction with multi-input multi-output local learning. *Proceedings of the 2nd European Symposium on Time Series Prediction (TSP), ESTSP08*.

Bouchaud, J., Bonart, J., Donier, J., and Gould, M. (2018). *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press.

Box, G. and Jenkins, G. (1970). *Time series analysis : forecasting and control*. Holden-Day series in time series analysis. Holden-Day ; Distributed by McGraw-Hill.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L. and Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):3–54.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. The Wadsworth statistics/probability series. Wadsworth International Group.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1 – 3.

Broomhead, D. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA. Association for Computing Machinery.

Burges, C. J. C. (2010). From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research.

Calliess, J. (2019). Online optimisation for online learning and control - from no-regret to generalised error convergence. In *17th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*, pages 2480–2485. IEEE.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press, USA.

Chen, S. and Billings, S. A. (1989). Representations of non-linear systems: the narmax model. *International Journal of Control*, 49(3):1013–1032.

Cover, T. and Ordentlich, E. (1996). Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363.

Cover, T. M. (1991). Universal portfolios. *Mathematical Finance*, 1(1):1–29.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

D'Agostino, R. and Pearson, E. S. (1973). Tests for departure from normality. empirical results for the distributions of b_2 and $\checkmark b_1$. *Biometrika*, 60(3):613–622.

D'Agostino, R. B. (1971). An omnibus test of normality for moderate and large size samples. *Biometrika*, 58(2):341–348.

de Prado, M. L. (2015). The future of empirical finance. *Journal of portfolio management*, 41(4):140–.

de Prado, M. L. (2018). *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38.

Derksen, S. and Keselman, H. J. (1992). Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables. *British journal of mathematical & statistical psychology*, 45(2):265–282.

Du, W. S. (2018). Minkowski-type distance measures for generalized orthopair fuzzy sets. *International journal of intelligent systems*, 33(4):802–817.

Durbin, J. and Watson, G. S. (1950). Testing for serial correlation in least squares regression. i. *Biometrika*, 37:409–428.

Enders, W. (2014). *Applied econometric time series*. Wiley series in probability and mathematical statistics. J. Wiley, 4th ed. edition.

Enders, W. (2015). *Applied econometric time series*. Wiley Series in Probability and Statistics. Wiley, 4th ed edition.

Engel, C. (1994). Can the markov switching model forecast exchange rates? *Journal of international economics*, 36(1):151–165.

Figueiredo, M. A. T. and Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 24:381–396.

Flach, P. and Matsubara, E. T. (2007). *A Simple Lexicographic Ranker and Probability Estimator*. Springer Berlin Heidelberg.

Freund, Y., Schapire, R. E., Singer, Y., and Warmuth, M. K. (1997). Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 334–343, New York, NY, USA. Association for Computing Machinery.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232.

Gaivoronski, A. A. and Stella, F. (2000). Stochastic nonstationary optimization for finding universal portfolios. *Annals of operations research*, 100(1-4):165–188.

Gama, J. (2012). A survey on learning from data streams: current and future trends. *Progress in artificial intelligence*, 1(1):45–55.

Girosi, F. and Poggio, T. (1990). Networks and the best approximation property. *Biological cybernetics*, 63(3):169–176.

Gold, C. (2003). Fx trading via recurrent reinforcement learning. In *IEEE*. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. The MIT Press, London, England.

Gordon, G. J. (2000). Reinforcement learning with function approximation converges to a region. In *NIPS*.

Gorse, D. (2011). Application of stochastic recurrent reinforcement learning to index trading. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium*. ESANN.

Granger, C. and Newbold, P. (1974). Spurious regressions in econometrics. *Journal of econometrics*, 2:111–120.

Grinold, R. and Kahn, R. (2019). *Advances in Active Portfolio Management: New Developments in Quantitative Investing*. McGraw-Hill Education.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273.

Gunnarsson, S. (1996). Combining tracking and regularization in recursive least squares identification. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pages 2551–2552 vol.3.

Hadsell, R., Rao, D., Rusu, A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24:1028–1040.

Hamilton, J. D. (1994). *Modeling Time Series with Changes in Regime*, pages 677–677. Princeton University Press.

Harvey, A. (1993). *Time series models*. Financial Times/Prentice Hall.

Harvey, C. R., Hoyle, E., Korgaonkar, R., Rattray, S., Sargaison, M., and Van Hemert, O. (2018). The impact of volatility targeting. *Journal of portfolio management*, 45(1):14–33.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer Verlag, second edition. edition.

Haykin, S. (2001). *Kalman filtering and neural networks*. Wiley.

Helmbold, D. P., Schapire, R. E., Singer, Y., and Warmuth, M. K. (1998). On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347.

Herbster, M. and Warmuth, M. K. (1998). Tracking the best expert. *Mach. Learn.*, 32(2):151–178.

Hirotsu, C. (2017). *Two-Sample Problem*, chapter 5, pages 75–111. John Wiley & Sons, Ltd.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoerl, A. (1962). Application of ridge analysis to regression problems. In *Chemical Engineering Progress*.

Isaksson, M. (2017). Create simple drawings of neural networks using graphviz. https://github.com/martisak/.

Iwashita, A. S. and Papa, J. P. (2019). An overview on concept drift learning. *IEEE access*, 7:1532–1547.

Jaeger, H. (2001). The ”echo state” approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science.

Jaeger, H. (2002). Adaptive nonlinear system identification with echo state networks. In *NIPS*.

Jaeger, H. (2012). Long short-term memory in echo state networks: Details of a simulation study.

Jaeger, H. (2017). Using conceptors to manage neural long-term memories for temporal patterns. *Journal of Machine Learning Research*, 18:13:1–13:43.

James, G. M., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning : with applications in R*. Springer texts in statistics. Springer.

Jegadeesh, N. and Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48:65–91.

Jolliffe, I. (2011). *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35.

Kanazawa, N. (2020). Radial basis functions neural networks for nonlinear time series analysis and time-varying effects of supply shocks. *Journal of macroeconomics*, 64:103210.

Kaplanis, C., Shanahan, M., and Clopath, C. (2018). Continual reinforcement learning with complex synapses.

Khosravi, H. (2011). A novel structure for radial basis function networks—wrbf. *Neural processing letters*, 35(2):177–186.

Kim, S. (2019). Enhancing the momentum strategy through deep regression. *Quantitative Finance*, 19(7):1121–1133.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. In *arXiv*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114:3521–3526.

Kivinen, J. and Warmuth, M. K. (1995). Additive versus exponentiated gradient updates for linear prediction. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, page 209–218, New York, NY, USA. Association for Computing Machinery.

Koppel, A., Pradhan, H., and Rajawat, K. (2021). Consistent online gaussian process regression without the sample complexity bottleneck. *Statistics and computing*, 31(6).

Krawczyk, B. and Wozniak, M. (2015). Weighted naïve bayes classifier with forgetting for drifting data streams. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2147–2152.

Kroner, K. F. and Sultan, J. (1993). Time-varying distributions and dynamic hedging with foreign currency futures. *Journal of financial and quantitative analysis*, 28(4):535–551.

Lecun, Y. (1989). *Generalization and network design strategies*, pages 143–156. Elsevier.

Lei, K., Zhang, B., Li, Y., Yang, M., and Shen, Y. (2020). Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. *Expert Systems with Applications*, 140:112872.

Li, B. and Hoi, S. C. H. (2016). *Online Portfolio Selection: Principles and Algorithms*. CRC Press, Boca Raton.

Lim, Y. and Gorse, D. (2020). Deep recurrent modelling of stationary bitcoin price formation using the order flow. In Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., and Zurada, J., editors, *Artificial Intelligence and Soft Computing*, pages 170–179, Cham. Springer International Publishing.

Lin, X., Yang, Z., and Song, Y. (2011). Intelligent stock trading system based on improved technical analysis and echo state network. *Expert systems with applications*, 38(9):11347–11354.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, 108:212–261.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(3):503–528.

Liu, J., Sun, T., Luo, Y., Fu, Q., Cao, Y., Zhai, J., and Ding, X. (2018). Financial data forecasting using optimized echo state network. In Cheng, L., Leung, A. C. S., and Ozawa, S., editors, *Neural Information Processing*, pages 138–149, Cham. Springer International Publishing.

Liu, W., Principe, J. C., and Haykin, S. (2010). *Kernel adaptive filtering a comprehensive introduction*. Wiley.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Lototsky, S. and Pollok, A. (2021). Kelly criterion: From a simple random walk to lévy processes. *SIAM Journal on Financial Mathematics*, 12(1):342–368.

Lukoševičius, M. (2012). *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lukosevicius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.*, 3:127–149.

Lukosevicius, M., Jaeger, H., and Schrauwen, B. (2012). Reservoir computing trends. *KI - Künstliche Intelligenz*, 26:365–371.

Luo, S., Lin, X., and Zheng, Z. (2019). A novel cnn-ddpg based ai-trader: Performance and roles in business operations. *Transportation research. Part E, Logistics and transportation review*, 131:68–79.

Maciel, L., Gomide, F., Santos, D., and Ballini, R. (2014). Exchange rate forecasting using echo state networks for trading strategies. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr)*, pages 40–47.

Markovsky, I. and Van Huffel, S. (2007). Overview of total least-squares methods. *Signal Processing*, 87(10):2283–2302. Special Section: Total Least Squares and Errors-in-Variables Modeling.

Markowitz, H. and Cootner, P. H. (1965). The random character of stock market prices. *Journal of the American Statistical Association*, 60(309):381–381.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1033–1040, Madison, WI, USA. Omnipress.

Meese, R. A. and Rogoff, K. (1983). Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of international economics*, 14(1):3–24.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125 – 144.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *arXiv*.

Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, 1:281–294.

Moody, J. and Wu, L. (1997). Optimization of trading systems and portfolios. In *IEEE*.

Moody, J., Wu, L., Liao, Y., and Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17:441–470.

Moroshko, E., Vaits, N., and Crammer, K. (2015). Second-order non-stationary online learning for regression. *Journal of Machine Learning Research*, 16(43):1481–1517.

Munaga, H. and Jarugumalli, V. (2012). Performance evaluation: Ball-tree and kd-tree in the context of mst. In *Signal Processing and Information Technology*, pages 225–228, Berlin, Heidelberg. Springer Berlin Heidelberg.

Murphy, K. P. (2012). *Machine learning a probabilistic perspective*. Adaptive Computation and Machine Learning. MIT Press.

Myers, R. J. and Thompson, S. R. (1989). Generalized optimal hedge ratio estimation. *American journal of agricultural economics*, 71(4):858–868.

Nakamura, T. and Small, M. (2007). Tests of the random walk hypothesis for financial data. *Physica A: Statistical Mechanics and its Applications*, 377(2):599–615.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pinto, R. C. and Engel, P. M. (2015). A fast incremental gaussian mixture model. *PLOS ONE*, 10(10):1–1.

Poh, D., Lim, B., Zohren, S., and Roberts, S. (2021). Building cross-sectional systematic strategies by learning to rank. *The Journal of Financial Data Science*.

Polyak, B. T. (1990). A new method of stochastic approximation type. *Autom. Remote Control*, 51(1):937–946.

Provost, F. and Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, 52:199–215.

Provost, F. and Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, page 43–48. AAAI Press.

Rakhlin, S. and Sridharan, K. (2014). Statistical learning theory and sequential prediction. Technical report, MIT. STAT928.

Rasmussen, C. and Williams, C. (2005). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2006a). Approximation Methods for Large Datasets. In *Gaussian Processes for Machine Learning*. The MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2006b). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. In *Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department*.

Ruppert, D. (1988). Efficient estimations from a slowly convergent robbins-monro process. In *technical report, Cornell University Operations Research and Industrial Engineering*.

Said, S. E. and Dickey, D. A. (1984). Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607.

Salvalaio, B. K. and de Oliveira Ramos, G. (2019). Self-adaptive appearance-based eye-tracking with online transfer learning. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 383–388. IEEE.

Satchell, S. and Timmermann, A. (1995). An assessment of the economic value of non-linear foreign exchange rate forecasts. *Journal of Forecasting*, 14(6):477–497.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471.

Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation*, 12(5):1207–1245.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, 39.

Sharpe, W. F. (2007). Expected utility asset allocation. *Financial Analysts Journal*, 63(5):18–30.

Singer, Y. (1998). Switching portfolios. In *International Journal of Neural Systems*, pages 488–495. Morgan Kaufmann.

Sugiyama, M. (2015). *Statistical Reinforcement Learning*. CRC Press, 1st edition edition.

Sukhov, S., Leontev, M., Miheev, A., and Sviatov, K. (2020). Prevention of catastrophic interference and imposing active forgetting with generative methods. *Neurocomputing (Amsterdam)*, 400:73–85.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA. MIT Press.

Szita, I., Gyenes, V., and Lőrincz, A. (2006). Reinforcement learning with echo state networks. In *Artificial Neural Networks – ICANN 2006*, pages 830–839, Berlin, Heidelberg. Springer Berlin Heidelberg.

Takezawa, K. (2006). *Introduction to nonparametric regression Kunio Takezawa.* Wiley series in probability and statistics. Wiley-Interscience, Hoboken, N.J.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. (2017). Sequential decision making with coherent risk. *IEEE transactions on automatic control*, 62(7):3323–3338.

Tibshirani, R. J., Barber, R. F., Candes, E., and Ramdas, A. (2019). Conformal prediction under covariate shift. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tsay, R. S. and Chen, R. (2019). *Nonlinear time series analysis*. Wiley series in probability and statistics. Wiley.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2017). Learning to reinforcement learn.

Wang, L. and Rasheed, K. (2018). Stock ranking with market microstructure, technical indicator and news. *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pages 322–328.

Wang, X., Wang, X., and Zeng, Z. (2020). A novel weight update rule of online transfer learning. In *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*, pages 349–355. IEEE.

Wang, Y., Wu, C., and Yang, L. (2015). Hedging with futures: Does anything beat the naïve hedging strategy? *Management science*, 61(12):2870–2889.

Wasserman, L. (2004). *Hypothesis Testing and p-values*, pages 149–173. Springer New York, New York, NY.

Watkins, C. J. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Williams, R. (1992a). Training recurrent networks using the extended kalman filter. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 241–246 vol.4.

Williams, R. J. (1992b). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Xue, F., Li, Q., Li, X., and Zhou, H. (2016). The application of shesn on financial time series prediction. In *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pages 692–696.

Yang, Q., Zhang, Y., Dai, W., and Pan, S. J. (2020a). *Transfer Learning*. Cambridge University Press.

Yang, X., He, J., Lin, H., and Zhang, Y. (2020b). Boosting exponential gradient strategy for online portfolio selection: An aggregating experts' advice method. *Computational economics*, 55(1):231–251.

Ye, Z., Deng, W., Zhou, S., Xu, Y., and Guan, J. (2020). Optimal trade execution based on deep deterministic policy gradient. In *Database Systems for Advanced Applications*, volume 12112 of *Lecture Notes in Computer Science*, pages 638–654. Springer International Publishing.

Yildiz, I. B., Jaeger, H., and Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural networks: the official journal of the International Neural Network Society*, 35:1–9.

Yu, W., Gonzalez, J., and Li, X. (2021). Fast training of deep lstm networks with guaranteed stability for nonlinear system modeling. *Neurocomputing (Amsterdam)*, 422:85–94.

Zhang, H. and Su, J. (2004). Naive bayesian classifiers for ranking. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Machine Learning: ECML 2004*, pages 501–512, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhao, P., Hoi, S. C. H., Wang, J., and Li, B. (2014). Online transfer learning. *Artificial intelligence*, 216:76–102.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 928–935. AAAI Press.