KERNEL ADAPTIVE FILTERING APPROACHES FOR FINANCIAL TIME-SERIES PREDICTION

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2021

Sergio Garcia Vega

Department of Computer Science

Contents

Abbreviations 11

Nomenclature 14

Notations 15

Abstract 16

Declaration 18

Copyright 19

Journal Publications 20

Paper Under Review 21

Acknowledgements 22

1 Introduction 23

1.1 Context and Motivation 23

1.2 Research Aim and Objectives 29

1.3 Thesis Contributions 30

1.4 Thesis Structure 32

2 Background and Related Work 33

2.1 Financial Applications 33

2.1.1 Algorithmic Trading 34

2.1.2 Risk Assessment 35

2.1.3 Fraud Detection 37

2.1.4 Portfolio Management 38

2.1.5 Asset Pricing and Derivatives Market . . . 39
2.1.6 Cryptocurrency and Blockchain Studies . . . 40
2.1.7 Financial Sentiment Analysis . . . 41
2.1.8 Other Financial Applications . . . 42
2.2 Financial Data Analysis Approaches . . . 43
2.2.1 Statistical Techniques . . . 43
2.2.2 Pattern Recognition . . . 44
2.2.3 Machine Learning . . . 45
2.2.4 Hybrid Analysis . . . 50
2.3 Research Gaps and Challenges . . . 53

3 A Kernel-based Sequence Prediction Approach 57
3.1 An Approach for Sequence Prediction . . . 58
3.1.1 Multiple Kernel-Sizes in Online Sequential Learning . . . 58
3.1.2 An Online Technique to Optimize Step-Size . . . 64
3.2 Experimental Design . . . 67
3.2.1 Data Sets . . . 67
3.2.2 Comparative Methods . . . 67
3.2.3 Parameter Settings . . . 68
3.3 Simulation Results and Analysis . . . 72
3.4 Chapter Summary . . . 75

4 An Entropy-based Cost Function for Sequence Prediction 76
4.1 An Entropy-based Prediction approach . . . 77
4.1.1 Adaptation Criterion based on Entropy . . . 77
4.1.2 Entropy-based Bandwidth for Density Estimation . . . 79
4.1.3 Neural Network Architecture using Kernel Machines . . . 80
4.2 Experimental Design . . . 81
4.2.1 Data Sets . . . 82
4.2.2 Comparative Methods . . . 82
4.2.3 Parameter Settings . . . 83
4.3 Simulation Results and Analysis . . . 84
4.4 Chapter Summary . . . 87

5 A Kernel-based Stock Market Interdependence Approach 88
5.1 Sequential and Interdependent Nature of Financial Time-Series . . . 89

5.1.1 Sequential Learning based on Adaptive Filtering 89
5.1.2 A Stock Market Interdependence Approach 93
5.2 Experimental Design 95
5.2.1 Data Sets 96
5.2.2 Comparative Methods 97
5.2.3 Parameter Settings 97
5.3 Simulation Results and Analysis 98
5.4 Chapter Summary 107

6 Conclusions and Future Work **108**
6.1 Conclusions 108
6.2 Future Work 110
6.2.1 Predicting Several Steps Ahead 110
6.2.2 Automated Machine Learning 111
6.2.3 Transfer Learning 112

A Financial Data **113**
A.1 Financial Time-Series 113
A.2 Financial Assets and Markets 114
A.3 Returns on Assets 115
A.4 Stock Prices 116

B Machine Learning Models **118**
B.1 Neural Networks 118
B.1.1 Multilayer Feed-forward Neural Networks 121
B.1.2 Convolutional Neural Networks 122
B.1.3 Recurrent Neural Networks 123
B.2 Linear Adaptive Filters 124
B.2.1 Least Mean Square Algorithm 124
B.2.2 Recursive Least Squares Algorithm 125
B.2.3 Affine Projection Algorithm 126
B.3 Kernel Adaptive Filtering 127
B.3.1 Reproducing Kernel Hilbert Spaces 128
B.3.2 Kernel Least Mean Square Algorithm 129
B.3.3 Kernel Recursive Least Squares Algorithm 130
B.3.4 Kernel Affine Projection Algorithm 131

C Information Theory **133**
C.1 Entropy 134
C.2 Mutual Information 134
C.3 Divergence and Mutual Information 135

D Information-Theoretic Learning **137**
D.1 Renyi Entropy 137
D.2 Quadratic Renyi Entropy 138
D.3 Renyi Divergence and Mutual Information 139

Word Count: 30278

List of Tables

2.1 Summary of literature on algorithmic trading approaches. 35

2.2 Summary of literature on risk assessment approaches. 37

2.3 Summary of literature on fraud detection approaches. 38

2.4 Summary of literature on portfolio management approaches. . . . 39

2.5 Summary of literature on asset pricing approaches. 40

2.6 Summary of literature on cryptocurrencies and block chain studies. 41

2.7 Summary of literature on financial sentiment analysis approaches. 42

2.8 Summary of literature on financial application approaches. 43

3.1 Testing MSE for proposal in considered data sets using different values of initial kernel-sizes σ_1. *FX*-foreign exchange; *TSLA*-Tesla stock price. 69

3.2 Parameter setting of proposed approach in considered data sets. *FX*-foreign exchange; *TSLA*-Tesla stock price; M-input vector size; η_1-initial step size; σ_1-initial kernel size; ρ-kernel size adaptation; β-step size adaptation; δ-centroid threshold. 69

3.3 Parameter setting of comparative methods in considered data sets. *FX*-foreign exchange. *TSLA*-Tesla stock price. M-input vector size, η-step size, η_1-initial step size, σ-kernel size, λ-centroid distance, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{E}$-epochs. 70

3.4 Testing MSE for KAF methods in considered data sets using different kernel-sizes σ. *FX*-foreign exchange. *TSLA*-Tesla stock price. 71

3.5 Testing MSE for NN methods in considered data sets using different epochs $\mathcal{E}$. *FX*-foreign exchange prices. *TSLA*-Tesla stock price. $\mathcal{N}$-neurons. 71

3.6 Testing MSE in considered data sets. *MSE*-mean squared error. *Samples*-average number of samples used to predict test set. *FX*-foreign exchange prices. *TSLA*-Tesla stock price. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 72

4.1 Stocks in the experimental design. 82

4.2 Parameter setting of compared methods. m-input vector size, L-error samples length, η-step size, $\hat{\eta}$-step size density, σ-kernel parameter, $\hat{\sigma}_1$-initial bandwidth density, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{A}$-number of lagged differences in the model, $\mathcal{I}$-cointegration rank. 83

4.3 Testing MSE using different values of step-size for density estimation. 84

4.4 Testing MSE at final iteration on stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 85

4.5 Testing MAE at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 85

5.1 Stocks in the experimental design. 96

5.2 Parameter setting of compared methods. M-input vector size, η-step size, σ-bandwidth, λ-quantization value, β-centroid distance, δ-threshold, ε-quantization value, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{G}$-maximum number of lags, $\mathcal{A}$-number of lagged differences in the model, $\mathcal{I}$-cointegration rank. 98

5.3 Testing MAE at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 99

5.4 Testing MSE at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 100

5.5 Testing SR at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%. 101

List of Figures

1.1 Summary of contributions. 31

2.1 Financial data analysis approaches and their financial applications. 53

3.1 Proposed approach for sequence prediction at iteration t. 63
3.2 Multiple kernel-sizes in online sequential learning. 64
3.3 Performed predictions in Tesla stock prices. 74
3.4 Performed predictions in foreign exchange rates. 74

4.1 Proposed neural network architecture using kernel machines. . . . 80
4.2 Stock returns prediction in the test set of HEI. 86
4.3 Stock returns prediction in the test set of CCL. 86

5.1 Three representative samples of the training set $\mathcal{T}$ when $M = 3$. The blue line represents stock returns of a given stock from which the training samples are selected. The upper, middle and lower graphs show the first, second, and last training samples, respectively. 89
5.2 Sequential learning within a stock market interdependence approach. 94
5.3 Models used by the proposed approach to predict the 24 stocks. . 103
5.4 Stock return predictions in the test sets of Allianz SE. 104
5.5 Stock return predictions in the test sets of Ashtead Group PLC. . 105
5.6 Stock return predictions in the test sets of Alphabet In-CL A. . . 106

B.1 Neural Network diagram with two layers. 119
B.2 Multilayer Feed-forward Neural Network architecture. 122
B.3 Convolutional Neural Network architecture. 123
B.4 Recurrent Neural Network architecture. 123
B.5 Linear adaptive filter structure. 124
B.6 Non-linear adaptive filter structure. 127

B.7 Non-linear mapping from the input space to the feature space. . . 127

C.1 Relationships between entropy and mutual information. 135

Abbreviations

AE	Auto Encoder
AMEX	American Stock Exchange
APA	Affine Projection Algorithm
ARIMA	Auto Regressive Integrated Moving Average
AUROC	Area Under the Receiver Operating Characteristics
BA	Balanced Accuracy
BPNN	Back-propagation Neural Network
BRT	Boosted Regression Trees
BSE	Bombay Stock Exchange
CDS	Credit Default Swaps
CNN	Convolutional Neural Network
CP	Chart Patterns
DBN	Deep Belief Network
DJIA	Dow Jones Industrial Average
DL	Deep Learning
DNN	Deep Neural Network
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
DRSE	Deep Random Subspace Ensembles
DTW	Dynamic Time Warping
ELM	Extreme Learning Machine
FN	False Negative
FNN	Feed-forward Network
FP	False Positive
FTSE	Financial Times Stock Exchange
FX	Foreign Exchange
GA	Genetic Algorithm
GARCH	Generalized Autoregressive Conditional Heteroskedasticity

GP	Genetic Programming
HMM	Hidden Markov Model
HHMM	Hierarchical Hidden Markov Model
ITL	Information-Theoretic Learning
KAF	Kernel Adaptive Filtering
KAPA	Kernel Affine Projection
KL	Kullback–Leibler Divergence
KLMS	Kernel Least Mean Square
KRLS	Kernel Recursive Least Squares
LE	Laplacian Eigen-Maps
LDA	Latent Dirichlet Allocation
LMS	Least Mean Square
LR	Likelihood Ratio
LSE	London Stock Exchange
LSTM	Long Short Term Memory
MACD	Moving Average Convergence and Divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MDA	Multilinear Discriminant Analysis
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NALL	Negative Average Log Likelihood
NASDAQ	National Association of Securities Dealers Automated Quotations
NICE	Nearest Instance Centroid Estimation
NLICA	Non-linear Independent Component Analysis
NN	Neural Network
NSE	National Stock Exchange of India
NYSE	New York Stock Exchange

PCA	Principal Component Analysis
PDF	Probability Density Function
PMF	Probability Mass Function
RBFN	Radial Basis Function Network
RBM	Restricted Boltzmann Machine
RF	Random Forest
RKHS	Reproducing Kernel Hilbert Space
RL	Reinforcement Learning
RLS	Recursive Least Squares
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RSI	Relative Strength Index
SAE	Stacked Auto-Encoder
SR	Sharpe Ratio
STE	Stock Exchange of Thailand
SVM	Support Vector Machines
SZSE	Shenzhen Stock Exchange
TAIEX	Taiwan Capitalization Weighted Stock Index
TM	Topic Models
TN	True Negative
TP	True Positive
TSE	Tokyo Stock Exchange
TSLA	Tesla
VSS	Variable Step Size
WBA	Weighted Balanced Accuracy
WMTR	Weighted Multichannel Time Series Regression
WT	Wavelet Transform

Nomenclature

$f(\cdot)$	continuous input-output mapping
$\mathbb{U}$	input domain
$\mathbb{R}$	the set of real numbers
$\mathcal{T}$	training data
$\boldsymbol{u}$	filter input vector
y	desired output
N	number of samples
L	most recent errors
e	output estimation error
η	step-size parameter
$\kappa_\sigma(\cdot,\cdot)$	Mercer kernel
σ	kernel-size parameter
$\|\cdot\|$	Euclidean norm
$\mathcal{L}$	number of layers
$\mathcal{N}$	neurons per layer
$\mathcal{E}$	number of epochs
$h(\cdot)$	non-linear activation function
ς	logistic sigmoid function
$l(\cdot)$	activation function
$\mathbb{E}\{\cdot\}$	expected value of a random variable
$p(\cdot)$	probability density (or mass) function
$H(\cdot)$	Shannon entropy
$H(\cdot,\cdot)$	Shannon joint entropy
$H(\cdot\|\cdot)$	Shannon conditional entropy
$I(\cdot)$	Shannon information
$I(\cdot,\cdot)$	Shannon mutual information
$D(\cdot\|\|\cdot)$	Kullback–Leibler divergence
$\hat{H}_\alpha(\cdot)$	Renyi entropy
$\hat{I}_\alpha(\cdot)$	Renyi mutual information
$\hat{D}(\cdot\|\|\cdot)$	Renyi divergence
$\ln(\cdot)$	natural logarithm

Notations

Notation	Description	Examples
Scalars	Small *italic* letters	y, p, q
Vectors	Small *italic* ***bold*** letters	$\boldsymbol{w}$, $\boldsymbol{u}$
Matrices	Capital *ITALIC* ***BOLD*** letters	$\boldsymbol{U}$, $\boldsymbol{V}$
Time or iteration	Subscript indices	y_t, $\boldsymbol{u}_t$
Scalar constants	Capital *ITALIC* letters	N, M

Abstract

Kernel Adaptive Filtering Approaches For Financial Time-Series Prediction

Sergio Garcia Vega

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy, 2021

Financial time-series are continuously generated by multiple sources, such as banks and corporations. These time-series are ordered sequences of data records that become available over time, imposing an order that must be considered when training models and making predictions. In addition, financial data represent, and are part of, highly complex systems that depend on and are generated by various factors such as financial policies and national economic growths. Thus, unlike traditional regression, predicting financial time-series requires consideration of both their sequential and interdependent nature. This thesis aims to address three related weaknesses of data-centralized and off-line machine learning methods:

1. **Sequence Learning.** Traditional approaches, developed to optimize performance on static data sets, restrict their flexibility to solve sequence prediction tasks in real-world applications. This means that, as traditional approaches are trained off-line, the profitable conditions of the trained models may disappear when they are tested in on-line environments;

2. **Higher Order Statistics.** The goal of dynamic modelling is to identify the dynamical system that produced a given input-output mapping. The mean square error (MSE), which is a second-order statistical measure, have been traditionally used as cost function when training adaptive systems. However, financial markets are complex and chaotic systems, meaning that second-order measures may be poor descriptors of optimality;

3. **Distributed Learning.** The movement of stock markets may affect the behaviour of stocks in other regions or countries. Traditional approaches do not directly consider inter-dependencies of the financial system, discarding interconnections and correlations that may represent important internal forces of the market.

To address these issues, this work proposes a variety of kernel adaptive filtering approaches, which are data-driven methods for sequence learning that combine the convex optimization of linear adaptive filters and the universal approximation property of neural networks (NNs). The proposed approaches consider interconnections between stock markets, which reduces volatility and maximizes returns while the robustness and simplicity of kernel adaptive filters are maintained. In particular, this thesis proposes: (1) a kernel adaptive filtering approach to support sequence prediction tasks in financial time-series; (2) an entropy-based cost function for kernel adaptive filtering to capture higher-order statistics of financial time-series; (3) a kernel adaptive filtering approach that captures internal forces of the market to improve profitability in real-time applications.

The results show relatively low MSE values and higher Sharpe ratio when compared with recurrent NNs and autoregressive-based models. The proposed approaches, unlike NNs, do not need the whole training set to start learning the model, meaning that predictions are generated while the model is sequentially updated at the same time. In addition, when compared with kernel adaptive filtering methods, there is an improvement between 0.5% and 54% in terms of MSE, showing high tolerance to noisy and non-stationary conditions. The results of this thesis are in line with previous studies, suggesting that the United States market is more influenced by the European and not vice versa.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Journal Publications

- Garcia-Vega, S., Zeng, X.-J., and Keane, J. (2019). Learning from data streams using kernel least-mean-square with multiple kernel-sizes and adaptive step-size. *Neurocomputing*, 339:105-115.
- Garcia-Vega, S., Zeng, X.-J., and Keane, J. (2020). Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach. *Expert Systems with Applications*, 160:113668.

Paper Under Review

- Garcia-Vega, S., Zeng, X.-J., and Keane, J. An entropy-based prediction framework for stock returns using kernel adaptive filtering and a neural network architecture. *Pattern Recognition Letters*.

Acknowledgements

To the memory of my Dad...
I look up to the sky and I talk to you... what I wouldn't give to hear you talk back

I would like to express my gratitude to my supervisor Prof. John Keane and co-supervisor Prof. Xiao-Jun Zeng for their valuable guidance and constant support during this research. Thanks to the BigDataFinance project for giving me the financial support during my PhD. Many thanks to Rytis for sharing interesting scientific discussions with me during our stay in Manchester.

I also owe thanks to my friend Kestutis and his wife Margarita for allowing me to be part of their families and showing me their beautiful Lithuanian culture. I would also like to thank my very good friend Mauricio Orbes who always gives me a different point of view about life and science, allowing me to constantly improve in my professional career during more than 10 years of friendship.

Finally, I would like to express my deepest thanks to my family, thank you for all your love. To my mum Magdalena for her unconditional support and encouraging me to do my best every day. To the love of my life Rafaella for making me happy and being with me during the difficult times. To my heavenly father, '*Porque Jehová da la sabiduría, y de su boca viene el conocimiento y la inteligencia*' Proverbs 2:6.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675044.

Chapter 1

Introduction

This chapter introduces the research problem under investigations and the structure of the thesis. Firstly, the research problem is motivated and its main challenges are introduced. Secondly, the aims and objectives of the investigation are presented. Finally, the main contributions made by this thesis are summarized.

1.1 Context and Motivation

Financial markets operate through a global network of banks, corporations, and individuals (Rejeb and Arfaoui, 2016). This system is composed of a large number of highly interconnected dynamical units, which presents a challenge to identify, model and extrapolate patterns. The financial time-series are ordered sequences of data records that become available over time. Sequence learning, on complex and noisy data records, requires consideration of properties in addition to prediction accuracy (Cao et al., 2019). That is, traditional regression approaches are developed to optimize performance on static data sets, restricting their flexibility to solve sequence prediction tasks in real-world applications (Cui et al., 2016). In contrast, sequence prediction models have memory, can learn temporal dependence between observations, and do not have prior information about future sequence of data records (Ahmad et al., 2017). This means that the sequence imposes an order on the samples, which must be preserved when training models and making predictions (Gueniche et al., 2015). Thus, unlike traditional regression, predicting financial time-series requires consideration of their sequential and interdependent nature (Rejeb and Arfaoui, 2016).

Traders and researchers have developed a variety of methods to predict future market behaviour (Han et al., 2013; Patel et al., 2015). For example, technical analysis aims to combine financial market patterns to improve predictions, while fundamental analysis evaluates securities in an attempt to measure its intrinsic value (Nazário et al., 2017). However, these methods do not provide a consistent investment strategy as their configurations are subject to analyst/trader preference (Olsen et al., 2018). In contrast to technical and fundamental analysis, algorithmic trading aims to provide consistent approaches and solid analysis frameworks by removing emotion (Hu et al., 2015). This kind of investment strategy has become popular in asset management and trading as it minimizes cost, market impact, and risk (Aldridge, 2013). Although, algorithmic trading provides advantages (Kissell, 2013), it is difficult for investment managers and hedge funds to earn consistent annual returns - even if the profit potential is big – on a systematic basis (Olsen et al., 2018). In addition to the strategies mentioned above, several prediction methods use statistical models and variants of regressive approaches (Kovács et al., 2017; Li et al., 2017c). However, their linear models may not work under non-stationary conditions and the stochastic nature of financial time-series (Khashei and Hajirahimi, 2019; Suhermi et al., 2018).

A variety of data-driven approaches based on Feed-forward Neural Networks (FNN) and Recurrent Neural Networks (RNN) have been used to predict financial time-series (Li et al., 2017c; Liu et al., 2015; Doucoure et al., 2016; Hosaka, 2019). However, their training strategies are difficult to interpret, as it is not clear how learning from input data is done or how performance can be consistently ensured, limiting their utility and acceptance in many real-world applications (Svozil et al., 1997; Olden and Jackson, 2002; Ghasemi et al., 2018; Zhang et al., 2018b). In contrast, Kernel Adaptive Filtering (KAF) methods, another data-driven approach based on sequence learning, have proven useful in identifying non-linear systems (Liu et al., 2010; Li and Príncipe, 2017). They combine the universal approximation property of NNs and the convex optimization of linear adaptive filters (Chen et al., 2016a). Their training scheme uses a combination of error-correction and memory-based learning, resulting in a simple convex optimization problem with a unique global optimum (Principe, 2010). However, when working with kernel adaptive filtering, additional challenges arise, for example, selection of kernel-size and step-size parameters (Liu et al., 2010).

The kernel-size parameter is used to define similarity between data points. That is, if the kernel-size is too large, all data will look similar; in contrast, if the kernel-size is too small, all data will look distinct (see Appendix B.3). The kernel-size in KAF only affects the dynamics of learning, i.e., this parameter affects the accuracy in off-line learning (the entire training data set are used at once to generate the best predictor), while the convergence is affected in online learning (ordered sequences of data records become available over time and are used to update the model at each step). The kernel-size is usually calculated using cross-validation (An et al., 2007), penalizing functions (Härdle, 1990), Silverman's rule of thumb (Silverman, 1986), or manual selection. The cross-validation and penalizing functions are not suitable for online kernel learning and have shown to be computationally intensive, while the Silverman's rule assumes a Gaussian distribution and is usually not appropriate for multi-modal distributions (Chen et al., 2016a). The previous methods are unsuitable for determining an optimal kernel-size in KAF, as they are either off-line mode methods or originate from a different problem, such as kernel density estimation.

Multiple-kernel-learning methods have been proposed to address the problem of kernel-size in KAF. For example, mixture KLMS employs a Gaussian kernel with multiple kernel-sizes (Pokharel et al., 2013). However, a pool of predefined kernel-sizes is needed in advance, meaning that the algorithm cannot create new kernel-sizes in an online way. Hence, the algorithm may be unable to adjust to abrupt changes in the system. Despite this, the idea of switching the models at different instances of time may be useful to identify non-stationary systems. The step-size parameter is the compromise between convergence time and misadjustment (Liu et al., 2010). Although many KAF methods use this parameter to compute predictions, they do not optimize step-size within the context of adaptive filtering with kernels (Liu et al., 2008; Chen et al., 2012; Zhao et al., 2013; Zheng et al., 2016); for example, recent work has proposed kernel-size optimization (Chen et al., 2016a), however, that work does not optimize the step-size parameter (though it is noted as "future work"). In practice (Liu et al., 2010; Zhao et al., 2020), this parameter is calculated off-line, hence it remains unchanged through iterations.

In addition, the goal of dynamic modelling is to identify the dynamical system that produced the given input-output mapping. This is usually done with cost functions based on second-order statistical measures, such as mean square error

(MSE) (Erdogmus and Principe, 2002a). The MSE, a criterion that measures the average squared difference between the desired and estimated values, is a widely used criterion when training adaptive systems, such as NNs and KAF (Zhao et al., 2011). The MSE describes real-life random phenomena using second-order statistics, providing analytical tractability, computational simplicity, and under Gaussian assumption is an optimal criterion for linear filters (Erdogmus and Principe, 2002b). However, under non-linear and non-Gaussian conditions, the MSE may be a poor descriptor of optimality (Singh and Príncipe, 2011). Thus, a more appropriate adaptation criterion requires consideration of information content of signals rather than simply their energy (a quantity related to second order statistics) (Xu, 1999; Principe, 2010). Information theoretic learning (ITL), which uses the general framework of information theory in the design of cost functions, has proven useful in training adaptive systems (Giraldo and Principe, 2013). In particular, unlike MSE, ITL uses probability density functions as the adaptation criterion, improving performance in various applications where the signals are non-Gaussian (Deng et al., 2016).

Information theory provides an effective alternative to analyze complex systems by using statistical descriptors such as: *entropy*, which is a scalar quantity that measures the average information contained in a given probability distribution function (Erdogmus and Principe, 2002a); *divergence*, which measures the dissimilarity between two different probability densities (Moreno et al., 2004); *mutual information*, which measures the total decrease in uncertainty between two random variables (Gray, 2011). These statistical descriptors have been used within the context of financial data analysis. For example, a previous study suggested that entropy has a predictive ability with respect to stock market dynamics (Caraiani, 2014). Another work found that entropy has predictive power on the Shenzhen stock market (Gu et al., 2015). Recently, when compared with MSE, mutual information has shown to be a more effective approach to measure the non-linear dynamic relationship between stock prices (Guo et al., 2018). However, estimation of the above statistical descriptors requires knowledge of the data probability density function, which in practice means either assumes an analytical model or use of a non-parametric estimator such as Parzen density estimation (Deng et al., 2016). This, usually leads to poor performance in large dimensional spaces (Principe et al., 2000).

The entropy of Renyi uses any entropy order and kernel function, providing a computationally efficient alternative (Erdogmus and Principe, 2002b). In practice, Renyi's entropy does not need to estimate the probability density function. Instead, non-parametric estimators based on kernels are used. This provides a framework to non-parametrically adapt systems based on entropy, divergence and mutual information (Chen et al., 2011). The Renyi's estimators have been used in a variety of applications such as image segmentation (Nobre et al., 2016), epileptic seizure detection (Mammone et al., 2011; Gao and Hu, 2013), identification of sleep stages (Fraiwan et al., 2012), diagnosis of depression (Faust et al., 2014), and motor imagery systems (Kee et al., 2017). In finance, Renyi's entropy applications are portfolio selection (Rödder et al., 2010; Xu et al., 2011; Zhou et al., 2013), asset pricing (Neri and Schneider, 2012; Ormos and Zibriczky, 2014), and information transfer between financial time-series (Dimpfl and Peter, 2014; Jizba et al., 2012; Korbel et al., 2019). However, the previous works do not aim to use statistical descriptors such as entropy and mutual information within the context of adaptive filtering, restricting their application to sequence prediction tasks.

The generalized stochastic information gradient algorithms in kernel space aim to combine the advantages of adaptive systems and information theoretic quantities, e.g., kernel maximum correntropy criterion (Zhao et al., 2011) and kernel minimum error entropy algorithm (Chen et al., 2013a). They implement the conventional linear adaptive filters in Reproducing Kernel Hilbert Space (RKHS) and obtains the non-linear adaptive filters in original input space (Principe, 2010). Then, statistical descriptors such as entropy and correntropy (a measure that estimates the similarity between two random variables) are used as adaptation criterion to minimize the difference between the desired and estimated values, making them suitable for sequence prediction tasks and providing robustness in the presence of noisy and non-stationary conditions at the same time. However, there remain two main challenges in these approaches: although they have shown superior performance on synthetic data, there is no evidence of such competitive performance in real-world applications such as financial data analysis (thought it is noted as "future research"); and their computational complexity increases with the estimation of the error (or data) distribution in online scenarios.

In addition to the sequence learning limitations mentioned above, most current implementations do not consider cross-dependency of the financial system, meaning that information coming from multiple distributed sources – decentralized models – has been discarded (Tkáč and Verner, 2016). That is, the movement of a stock market in a country is likely to be affected by movement of other stocks in both that country and in other regions (Masih and Masih, 2001). Many strategies have been proposed to analyse the distributed data of complex networks. The names for these strategies vary depending on the research field, e.g., functional and effective connectivity are well-known terms in neuroscience (Cheng et al., 2018; Park et al., 2018; Parker et al., 2018), while co-movement, interdependence, and connectivity analysis are commonly found in finance (Reboredo, 2018; Caraiani, 2018; Jiang et al., 2017a). Regardless of the term used, the main interest is to identify and quantify interactions on these complex systems. This is done using connectivity measures such as correlation (Samuel and Okey, 2015), coherence (Lipping et al., 2017), and Pearson correlation coefficient (Hauke and Kossowski, 2011). However, how to incorporate these inter-dependencies into an analytical model, such as sequential learning, to predict financial markets remains an open issue. In summary, many modelling attempts fail because:

1. The models are trained off-line; when going online the profitable conditions for which the model was optimized disappear (Olsen et al., 2018);

2. Financial markets are complex, non-stationary, non-linear, and chaotic systems (Ince and Trafalis, 2007);

3. Traditional approaches do not consider inter-dependencies of the financial system, that is, financial data arrives every second – even millisecond – from various sources (Fan et al., 2014a; Wu et al., 2013a).

Hence, this research aims to address the weakness of data-centralized and off-line machine learning methods (Fan et al., 2014a), which fail to consider fast time-varying characteristics from data resources in other regions, sectors, and markets.

1.2 Research Aim and Objectives

The overall aims of this thesis are: (1) To develop regression approaches that consider the sequential nature of financial time-series, while improving convergence time; (2) To design adaptive filter models considering non-linear and non-Gaussian conditions of financial time-series, aiming to enhance accuracy in sequence prediction tasks; (3) To identify interdependencies between financial markets and incorporate them into sequence prediction approaches, aiming to improve profitability and accuracy in regression tasks. In more detail, the research objectives of this thesis are:

1. **Propose a sequence prediction approach based on kernel adaptive filtering for financial time-series.**
   The sequential nature of financial time-series imposes an order on the samples, which must be preserved when training models and making predictions. The prediction approach must learn from a continuous sequence of data records and be robust to non-stationary conditions, aiming to improve convergence and prediction accuracy.

2. **Develop a cost function for kernel adaptive filtering using higher order statistics of financial time-series.**
   Second-order statistic measures have been traditionally used as cost functions when training adaptive systems. However, under non-Gaussian conditions, these measures may be poor descriptors of optimality. The cost function must capture higher order statistics of financial time-series rather than simply their energy, aiming to enhance prediction accuracy under non-stationary conditions in sequence learning tasks.

3. **Propose a kernel adaptive filtering approach within a distributed learning paradigm to predict financial time-series.**
   The approach must consider both the sequential and interdependent nature of financial time-series, which is critical for big data analysis and real-time applications. The distributed learning paradigm must incorporate the interdependencies between financial markets into an analytical model, aiming to improve prediction accuracy and profitability.

1.3 Thesis Contributions

Approaches based on kernel adaptive filtering have been proposed to support sequence prediction tasks for financial data analysis. This section overviews the research contributions made by this thesis: **1. Sequence Learning**, we investigate and develop a kernel adaptive filtering approach to support sequence prediction tasks in financial time-series, improving both prediction accuracy and convergence in real-world applications; **2. Higher Order Statistics**, we investigate and develop an entropy-based cost function for kernel adaptive filtering to capture higher order statistics of financial time-series, enhancing prediction accuracy under non-Gaussian and non-stationary conditions; **3. Distributed Learning**, we propose a kernel adaptive filtering approach that considers the interdependent nature of financial systems, capturing internal forces of the market and improving profitability in real-time applications. The following list provides a more detailed description of the research contributions:

1. **A kernel adaptive filtering approach to support sequence prediction tasks in financial time-series**. The approach addresses two main challenges in kernel adaptive filtering, i.e., kernel parameter selection and tuning of the step-size. In this work, the kernel-sizes, unlike traditional kernel adaptive filtering formulations, are both created and updated in an online sequential way. Further, to improve convergence time, we introduce an adaptive step-size strategy that minimizes MSE using a stochastic gradient algorithm. The approach is validated using two publicly available data sets, i.e., the mid-prices of two major currencies in the foreign exchange market and Tesla stock prices from its initial public offering. Results show both faster convergence to relatively low values of MSE and better accuracy when compared with kernel adaptive filtering methods, LSTM, and RNN.

2. **An entropy-based cost function for financial time-series prediction using kernel adaptive filtering.** The cost function uses instantaneous entropy as the adaptation criterion within a stochastic gradient descent optimization approach. Unlike traditional approaches, this minimizes the error entropy between the model output and the desired response, capturing higher-order statistics and information content of financial time-series rather than simply their energy. Further, to enhance predictions we extend

the architecture of NNs to incorporate kernel adaptive filtering by allowing any neuron to be replaced by a kernel machine, which better captures complex patterns behind the data. The approach is evaluated on 10 stocks from different economies over 12 years. Results show that, when compared with kernel adaptive filtering methods, there is an improvement between 0.5% and 54% in terms of MSE.

3. **A kernel adaptive filtering approach within a distributed learning paradigm for financial time-series prediction.** Unlike traditional kernel adaptive filtering formulations, the approach predicts using both local models and the individual local models learned from other markets, enhancing prediction accuracy. The proposal uses a distributed learning paradigm rather than a centralized one in the sense that individual prediction models are learned based solely on a local data store, avoiding expensive and time-consuming data transportation into an integrated, central data store. The approach is validated on 24 different stocks from three major economies, showing higher Sharpe ratio when compared with kernel adaptive filtering methods, LSTM, and autoregressive-based models.

Figure 1.1 depicts the inter-relations of the contributions made by this thesis, which are represented by the green rectangles. Here, the term streaming data refers to an ordered sequence of data records that become available over time, e.g., financial transactions (Cui et al., 2016). Contributions 1, 2, and 3 are described in Chapters 3, 4, and 5, respectively.

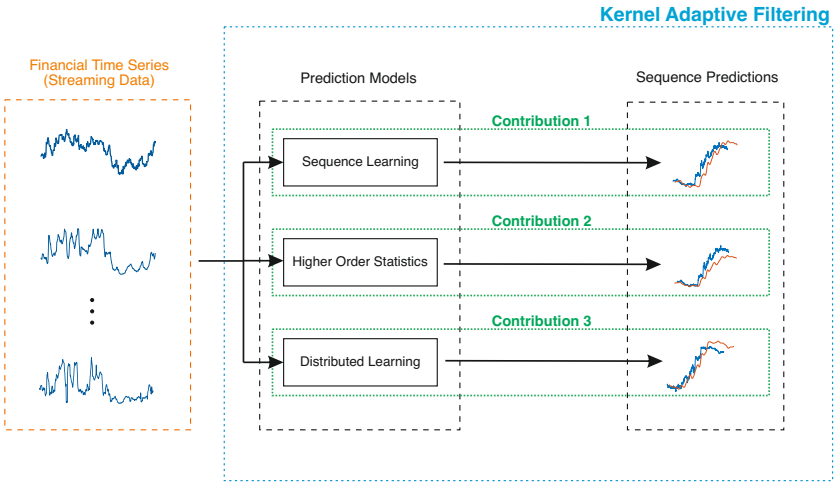


Figure 1.1: Summary of contributions.

1.4 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2. The related work used throughout this thesis is introduced in this chapter. The basic concepts of financial markets such as assets, stock prices, and market value are presented. Then, a literature review of financial data analysis approaches is provided. Finally, the research gaps and challenges are identified, summarised and discussed.

Chapter 3. This chapter develops a sequence prediction approach based on kernel adaptive filtering for financial time-series. The approach addresses two main challenges of kernel adaptive filtering, i.e., determination of kernel-size and tuning of the step-size parameter. The approach is validated on two real-world data sets, i.e., the mid-prices of two major currencies in the foreign exchange market and Tesla stock prices.

Chapter 4. A three-stage approach for stock returns prediction is proposed: (1) an entropy-based cost function is developed; (2) the bandwidth for density estimation is optimized using the quadratic information potential criterion; (3) the architecture and functionality of NNs are extended to include kernel adaptive filtering. The approach is tested on 10 different stocks over a 12 year interval.

Chapter 5. This chapter develops a two-phase approach for stock returns prediction using sequential learning within a stock market interdependence approach. Thus, the underlying models of each stock are learned separately using a kernel-based adaptive filter that encodes different patterns of the input space. The approach is validated on 24 different stocks from three major economies over 12 years.

Chapter 6. The thesis summary is given in this chapter; the three proposed approaches are critically evaluated, and suggestions for future work are included.

Chapter 2

Background and Related Work

Financial markets are influenced by political and economical factors (Bezerra and Albuquerque, 2017; Göçken et al., 2016), making their prediction one of the most challenging problems involving time-series (Chen et al., 2017). The literature shows that financial time-series are predicted using a variety of statistical and machine learning methods, which are mainly evaluated in terms of prediction accuracy, and profitability. The classic financial market prediction techniques are moving averages and autoregressive models (Wang et al., 2012; Kumar and Thenmozhi, 2014). More recently, machine learning techniques have been widely used in financial applications, as they are designed to address non-linearity, randomness, and chaotic data (Chen et al., 2017).

This chapter summarizes the background and related work used throughout this thesis. Firstly, financial applications are introduced in Section 2.1. Then, a literature review of financial data analysis approaches is provided in Section 2.2. Finally, the research gaps and challenges are discussed in Section 2.3.

2.1 Financial Applications

This section highlights the main financial applications of machine learning techniques. Note that, as some studies addressed multiple problems, there may be some overlaps between different implementation areas.

2.1.1 Algorithmic Trading

Algorithmic trading, which is defined as buy-sell decisions made solely by algorithmic models, aims to develop consistent approaches and methods that provide a solid analysis framework (Hu et al., 2015). The decisions made by algorithmic trading can be simple rules, optimized processes, mathematical models, or complex approximation techniques (Ozbayoglu et al., 2020). This kind of investment strategy has become popular in asset management and trading as it minimizes cost, market impact, and risk (Aldridge, 2013). For example, Karaoglu et al. (2017) proposed to predict stock prices using an algorithmic trading strategy based on RNN and Long Short Term Memory (LSTM).

Similarly, Bao et al. (2017) implemented a stock price prediction strategy using technical indicators, Wavelet Transforms (WT), Stacked Autoencoders (SAEs), and LSTM. Zhang et al. (2017a) proposed to predict stock prices using frequency trading pattern and RNN. Tran et al. (2017) developed a deep learning model for mid-price prediction using high frequency limit order book data with tensor representation. Fischer and Krauss (2018) implemented an LSTM-based strategy to predict S&P 500 index. Mourelatos et al. (2018) proposed to analyse the Greek Stock Exchange Index using an LSTM-based model. Si et al. (2017) implemented a trading model using LSTM to predict the Chinese market, while Yong et al. (2017) used a FNN model to predict Singapore Stock Market index.

The focus of other studies have been FX rates (see Appendix A.2) or cryptocurrency. For example, Lu (2017) proposed to trade the currency pair GBP/USD using reinforcement learning and LSTM models. Dixon et al. (2017) implemented a FNN-based model to predict commodities and FX trading prices. Korczak and Hemes (2017) used different input parameters on a multi-agent-based trading environment such as Convolutional Neural Network (CNN) to predict the currency pair GBP/PLN. Spilak (2018) constructed a dynamic portfolio using a variety of methods and cryptocurrencies such as LSTM, RNN, Multilayer Perceptron (MLP), Namecoin, Dogecoin, Litecoin, Monero, Nxt, Ripple, Dash, and Bitcoin.

Jeong and Kim (2019) proposed a deep NN model aiming to solve overfitting, market profit, and predicting number of trading shares. In Sezer et al. (2017) the buy & sell limits on the Relative Strength Index (RSI) were optimized using genetic algorithms. Navon and Keller (2017) proposed to predict the next price using FNN and a trading strategy. Troiano et al. (2018) used the Moving Average Convergence and Divergence (MACD) to predict trending prices on the DOW 30.

Sirignano and Cont (2019) proposed to predict stock price movements using an LSTM-based model and limit order book data, which is similar to the strategy proposed by Tsantekidis et al. (2017). Table 2.1 summarizes algorithmic trading approaches, showing relevant information of each study, such as the employed data set, main methods and performance measures.

Table 2.1: Summary of literature on algorithmic trading approaches.

Reference	Data Set	Main Methods	Performance Measures
Karaoglu et al. (2017)	Stock of Garanti Bank	RNN, LSTM	MSE, RMSE, MAE
Bao et al. (2017)	Nifty, S&P500 Index, DJIA	WT, SAE, LSTM	MAPE, Correlation Coefficient
Zhang et al. (2017a)	50 Stocks from NYSE	RNN	MSE
Tran et al. (2017)	5 Stocks from Finnish Stock Market	WMTR, MDA	Accuracy, Precision, Recall, F1-Score
Si et al. (2017)	Chinese Stock Market	LSTM	Profit and Loss, SR
Yong et al. (2017)	Singapore Stock Market Index	FNN	RMSE, MAPE, SR
Lu (2017)	FX	LSTM	SR
Dixon et al. (2017)	Commodity, FX	FNN	SR
Korczak and Hemes (2017)	FX	CNN	SR
Sezer et al. (2017)	Stocks from Dow30	GA	Annualized Return
Navon and Keller (2017)	Stocks from S&P500	FNN	Cumulative Gain
Tsantekidis et al. (2017)	Nasdaq Nordic	LSTM	Precision, Recall, F1-Score
Fischer and Krauss (2018)	S&P500 Index	LSTM	Return, SR, Accuracy
Mourelatos et al. (2018)	Stock of National Bank of Greece	LSTM	Return, Volatility, SR, Accuracy
Spilak (2018)	Bitcoin, Namecoin, Dogecoin, Litecoin, Dash	LSTM, RNN, MLP	Accuracy, F1-Score
Troiano et al. (2018)	Stocks from Dow30	LSTM, MACD	Accuracy
Jeong and Kim (2019)	S&P500 Index	DNN	Profit, Correlation
Sirignano and Cont (2019)	High-frequency records	LSTM	Accuracy

2.1.2 Risk Assessment

This financial application identifies the risk of any given asset and aims to help eliminate any potential risk-related consequences, e.g., credit scoring, consumer

credit determination, and bankruptcy prediction (Zio, 2018). Kirkos and Manolopoulos (2004); Ravi et al. (2008); Fethi and Pasiouras (2010) used machine learning models to analyse bank performance assessment studies. Lahsasna et al. (2010); Chen et al. (2016b) summarised credit scoring and risk assessment studies that are based on soft computing techniques, while Marques et al. (2013) focused on evolutionary models within a credit scoring context. Kumar and Ravi (2007); Verikas et al. (2010) reviewed bankruptcy studies that used machine learning implementations.

In the same way, Sun et al. (2014) provided a survey with a focus on corporate failures and financial distress. Luo et al. (2017) used Credit Default Swaps (CDS) for credit classification and corporate credit rating, finding that a model based Restricted Boltzmann Machine (RBM) achieved the best performance among the compared methods. Yu et al. (2018a) implemented a Support Vector Machine (SVM) model for credit classification, achieving an accuracy above 80%. Li et al. (2017e) proposed to use MLP and k-means for credit risk classification. Tran et al. (2016) proposed to use Stacked Auto-Encoder (SAE) and Genetic Programming (GP) for credit scoring applications. In another example, Neagoe et al. (2018) used CNN-based models to classify credit scores. Niimi (2015) compared models based on SVM and Random Forest (RF) to provide information about credit fraud and credit approval classification.

There are also some applications in financial distress predictions for banks and corporates. For example, Lanbouri and Achchab (2015) used a Deep Belief Network (DBN) with SVM to identify whether a firm was in trouble or not, while Rawte et al. (2018) studied bank risk classification. Rönnqvist and Sarlin (2015) proposed to determine the bank stress by using semantic vector representations. Rönnqvist and Sarlin (2017) used text mining to identify bank distress by using financial news and deep FNN, which is similar to the strategy proposed by Cerchiello et al. (2017). Malik et al. (2018) predicted the performance of banks using an LSTM network to evaluate the bank stress levels. In addition, some studies focused on bankruptcy or corporate default prediction. Yeh et al. (2015) used RBM and DBN to predict is a company was solvent or default, outperforming SVM-based models. Sirignano et al. (2016) proposed to identify the mortgage risk within 20 years by analysing different factors that affected the mortgage payment structure. Finally, Chatzis et al. (2018) developed several machine learning models to detect events that caused stock market crashes. Table 2.2

summarizes risk assessment approaches in terms of the employed data set, main methods and performance measures.

Table 2.2: Summary of literature on risk assessment approaches.

Reference	Data Set	Main Methods	Performance Measures
Lanbouri and Achchab (2015)	French Companies	SVM	Precision, Recall
Rönnqvist and Sarlin (2015)	European Banks, News Articles	DNN	F1-Score
Yeh et al. (2015)	Stock Returns	DBN	Accuracy
Tran et al. (2016)	Credit Database	GP, SAE	FP
Sirignano et al. (2016)	Mortgate Dataset	NN	NALL
Luo et al. (2017)	Credit Default Swaps	RBM	Accuracy, FN, FP
Li et al. (2017e)	Kaggle Credit Data	MLP	Accuracy, TP, TN
Rönnqvist and Sarlin (2017)	European Banks, News Articles	FNN	F1-Score
Cerchiello et al. (2017)	News Articles	NN	Relative Usefulness
Yu et al. (2018a)	Credit Database	SVM	Accuracy, TN, TP
Neagoe et al. (2018)	Credit Database	CNN	Accuracy
Rawte et al. (2018)	Bank Holding Companies	CNN, LSTM, SVM	Accuracy, F1-Score
Malik et al. (2018)	Bank Holding Companies	LSTM	RMSE
Chatzis et al. (2018)	Exchange Rates	SVM, DNN, RF	LR, BA, WBA

2.1.3 Fraud Detection

There are a variety of financial fraud cases such as tax evasion, money laundering, credit card, and insurance claim fraud, which is why this application is one of the most extensively studied areas of finance for machine learning (Ozbayoglu et al., 2020). For example, Wang (2010); Phua et al. (2010); Ngai et al. (2011); Sharma and Panigrahi (2013); West and Bhattacharya (2016) reviewed financial fraud

detection studies that focused on soft computing and data mining techniques. Heryadi and Warnars (2017) proposed a variety of deep learning models to identify credit card fraud detection in Indonesian banks. They also considered the effects of data imbalance between non-fraud and fraud data.

There are also some studies mainly focused on RNN approaches. For example, Roy et al. (2018) proposed to predict credit card fraud using LSTM models. Jurgovsky et al. (2018) proposed to detect credit card fraud using LSTM and transaction sequences. In addition, Paula et al. (2016) used auto-encoders to identify money laundering and financial fraud on Brazilian companies. Gomes et al. (2017) proposed an anomaly detection model based on auto-encoders to identify anomalies in the Brazilian parliamentary expenditure spending. Wang and Xu (2018) and Li et al. (2017b) used deep neural network models to detect auto-mobile insurance fraud and online payment transaction fraud, respectively. Goumagias et al. (2018) proposed to predict the risk-averse firms' tax evasion behaviours using reinforcement learning. Table 2.3 summarizes fraud detection approaches, showing relevant information of each study.

Table 2.3: Summary of literature on fraud detection approaches.

Reference	Data Set	Main Methods	Performance Measures
Paula et al. (2016)	Federal Revenue of Brazil	AE	MSE
Heryadi and Warnars (2017)	Credit Card Transactions	LSTM	AUROC
Gomes et al. (2017)	Federal Revenue of Brazil	Deep AE	MSE, RMSE
Li et al. (2017b)	Payment Transactions	DNN	AUROC
Roy et al. (2018)	Credit Card Transactions	LSTM	Accuracy
Jurgovsky et al. (2018)	Credit Card Transactions	LSTM	AUROC
Wang and Xu (2018)	Automobile Insurance Company	LDA	F1-Score
Goumagias et al. (2018)	Greek Firms	DQL	Revenue

2.1.4 Portfolio Management

This financial application, which aims to identify the best possible course of action for selecting the best performing assets for a given period, is closely related to areas such as portfolio optimization, selection, and allocation (Ozbayoglu et al., 2020). For example, Takeuchi and Lee (2013) proposed to classify the stocks as low and high momentum. Their implementation used an RBM encoder-classifier

network, achieving high returns on the considered stocks. Grace (2017) used a deep MLPs network to adjust the portfolio allocation weights for the considered stocks, while Fu et al. (2018) proposed a machine learning framework based on deep MLPs to address the stock selection problem.

More recently, Jiang and Liang (2017) implemented a portfolio strategy using CNN and DRL on selected cryptocurrencies. Similarly, Jiang et al. (2017b) implemented cryptocurrency portfolio management strategy using RNN, LSTM, and CNN models. Finally, Lee and Yoo (2018) compared RNN-based models to construct a threshold-based portfolio using stock prices, while Iwasaki and Chen (2018) used deep FNN models, sentiment analysis, text mining, and word embeddings to predict stock prices. Liang et al. (2018) used Deep Reinforcement Learning (DRL) for portfolio allocation, while Zhou (2019) proposed to predict the next month's return using MLPs, LSTM and deep learning models. Table 2.4 summarizes portfolio management approaches and their relevant methods.

Table 2.4: Summary of literature on portfolio management approaches.

Reference	Data Set	Main Methods	Performance Measures
Takeuchi and Lee (2013)	Stock Prices	RBM	Accuracy
Grace (2017)	Stock Prices	MLP	Accuracy
Jiang and Liang (2017)	Cryptocurrencies	CNN, RL	SR
Jiang et al. (2017b)	Cryptocurrencies	CNN, RNN	SR
Fu et al. (2018)	Stock Prices	RF, DNN	Accuracy, Recall
Lee and Yoo (2018)	Stock Prices	RNN	Accuracy
Iwasaki and Chen (2018)	Tokyo Stock Exchange	CNN, LSTM	Accuracy
Liang et al. (2018)	China Stock Data	DRL	SR
Zhou (2019)	Stock Prices	LSTM, MLP	SR

2.1.5 Asset Pricing and Derivatives Market

This application aims to provide an accurate pricing or valuation of an asset, which is a relevant study area in finance (Cochrane, 2009). The main focus of asset pricing studies in machine learning is the development of hedging strategies, options pricing, futures, and forward contracts (Ozbayoglu et al., 2020). In this sense, Iwasaki and Chen (2018) proposed to predict stock prices using reports for sentiment analysis and deep FNN models. Then, the predicted stock prices were

used to implement different portfolio selection strategies. Culkin and Das (2017) used deep FNN models to predict option prices and compared their results with Black & Scholes option pricing formula. Hsu et al. (2018) proposed to predict Taiwan Capitalization Weighted Stock Index (TAIEX) option prices using bid-ask spreads and Black & Scholes option price model. Feng et al. (2018) proposed to use characteristic features, such as Industry momentum and asset growth, as the inputs of a deep learning model to predict US equity returns in American Stock Exchange (AMEX), National Association of Securities Dealers Automated Quotation (NASDAQ), and NYSE indices. Table 2.5 summarizes asset pricing approaches, showing relevant information of each study, such as the employed data set, main methods and performance measures.

Table 2.5: Summary of literature on asset pricing approaches.

Reference	Data Set	Main Methods	Performance Measures
Culkin and Das (2017)	Options	DNN	RMSE
Iwasaki and Chen (2018)	Stock Prices	LSTM, CNN	Accuracy
Hsu et al. (2018)	Options	MLP	RMSE, MAE, MAPE
Feng et al. (2018)	Equity Returns	DL	RMSE

2.1.6 Cryptocurrency and Blockchain Studies

The price prediction and trading systems are some of the main applications in cryptocurrency studies. Chen (2018) proposed a blockchain transaction traceability algorithm using Bitcoin data such as Hash value, public/private key, and digital signature. Nan and Tao (2018) proposed a three-stage bitcoin mixing detection method within an auto-encoder approach. Lopes (2018) proposed to combine technical indicators, sentiment analysis, CNN, and LSTM-based models for cryptocurrency trading. Similarly, Jiang et al. (2017b) proposed a framework based on RNN, LSTM, and CNN for Cryptocurrency portfolio management. Jiang and Liang (2017) implemented a portfolio management using DRL and CNN on Ethereum, Bitcoin, and Digital Cash. Spilak (2018) proposed a dynamic portfolio using MLP, RNN, and LSTM models. The proposal was tested in cryptocurrencies such as Namecoin, Litecoin, Dash, Monero, Dogecoin, Ripple, Nxt, and

Bitcoin. Finally, McNally et al. (2018) proposed to predict the Bitcoin price direction using Auto Regressive Integrated Moving Average (ARIMA), LSTM, and RNN. Their models were compared in terms of Root Mean Square Error (RMSE), precision, specificity, and sensitivity. Table 2.6 summarizes cryptocurrency approaches and their relevant data sets, methods, and performance measures.

Table 2.6: Summary of literature on cryptocurrencies and block chain studies.

Reference	Data Set	Main Methods	Performance Measures
Jiang et al. (2017b)	Cryptocurrencies	CNN, RNN	SR
Jiang and Liang (2017)	Crytptocurrencies	CNN	SR
Nan and Tao (2018)	Bitcoin	LE, Deep AE	F1-Score
Chen (2018)	Bitcoin	Fuzzy Cognitive Maps	Analytical Hierarchy Process
Lopes (2018)	Bitcoin, Litecoin	CNN, LSTM	MSE
Spilak (2018)	Cryptocurrencies	LSTM, RNN, MLP	Accuracy, F1-Score
McNally et al. (2018)	Bitcoin	LSTM, RNN	Accuracy, RMSE

2.1.7 Financial Sentiment Analysis

There is a growing interest in financial sentiment analysis applications such as algorithmic trading and trend prediction. For example, Wang et al. (2018b) combined technical strategies and sentiment analysis techniques to predict stock prices. Shi et al. (2018) proposed a text-based deep learning model to predict stock price movements using financial news from Bloomberg and Reuters. Similarly, Peng and Jiang (2015) used word embeddings, text mining, and financial news from Reuters and Bloomberg to predict stock price movements. Zhuge et al. (2017) proposed to predict the next day opening prices of stocks using emotional data from text posts and prices of index data. Das et al. (2018) developed a framework to predict Apple, Microsoft, and Google stock prices using Twitter sentiment data. Prosky et al. (2017) proposed to predict stock prices using news from Reuters and sentiment analysis. Li et al. (2017a) used three different sentiment classification (positive, neutral, and negative) to predict open and close

prices within and LSTM approach. Their results, when compared with SVM-based models, achieved higher overall performance. Iwasaki and Chen (2018) applied text mining and word embeddings on analyst reports to predict stock price using deep FNN models. In addition, they implemented a variety of portfolio selection strategies based on the projected stock returns. Huang et al. (2016) proposed to predict price directions using Hidden Markov Model (HMM) and CNN. Their models were tested on Twitter and financial price data, where CNN achieved the best performance. Table 2.7 summarizes financial sentiment analysis approaches and their relevant methods.

Table 2.7: Summary of literature on financial sentiment analysis approaches.

Reference	Data Set	Main Methods	Performance Measures
Peng and Jiang (2015)	Financial News	DNN	Accuracy
Huang et al. (2016)	Twitter Moods	DNN, CNN	Error Rate
Zhuge et al. (2017)	SSE Composite Index	LSTM	MSE
Prosky et al. (2017)	Financial News	LSTM, CNN	Accuracy
Li et al. (2017a)	Sentiment Posts	LSTM	Accuracy, F1-Score
Shi et al. (2018)	Financial News	DeepClue	Accuracy
Das et al. (2018)	Twitter Sentiment	RNN	Correlation
Wang et al. (2018b)	Stocks	DRSE	F1-Score
Iwasaki and Chen (2018)	Analyst Reports	LSTM, CNN	Accuracy

2.1.8 Other Financial Applications

Finally, some studies did not fit into any of the previously covered financial applications, e.g., success prediction, bank telemarketing, and payment classification. Dixon et al. (2015) used deep FNN to speed-up the price movement direction prediction problem, which is their main contribution. Kim et al. (2015) proposed to predict the success of bank telemarketing using CNN-based models. Their study considered finance-related attributes and phone calls of the bank marketing data. Lee et al. (2017) proposed to estimate the revenue and profit for the corporates using technical indicators, RBM, FNN, and DBN models. Ying et al. (2017) predicted social security payment types (such as transferred, unpaid, paid, and

repaid) using SVM, HMM, and LSTM. Lastly, Jeong and Kim (2019) combined deep learning strategies to increase profit in the market, predict the number of shares to trade, and prevent over-fitting with insufficient data. Table 2.8 summarizes financial application approaches and their relevant data sets, methods, and performance measures.

Table 2.8: Summary of literature on financial application approaches.

Reference	Data Set	Main Methods	Performance Measures
Dixon et al. (2015)	Foreign Exchange	DNN	Convergence
Kim et al. (2015)	Bank Telemarketing	CNN	Accuracy
Lee et al. (2017)	Stock Prices	RBM, DBN	RMSE, Profit
Ying et al. (2017)	Payment Records	DNN, RNN	Accuracy
Jeong and Kim (2019)	S&P500 Index	DNN	Profit, Correlation

2.2 Financial Data Analysis Approaches

A variety of widely accepted empirical studies have shown that financial variables such as stock prices and market index values are predictable (Zhong and Enke, 2017; Chong et al., 2017). These empirical studies combine technical analysis indicators with approaches based on econometrics, statistics, data mining, and artificial intelligence (Arévalo et al., 2017). In this section, we highlight the advances in stock market analysis and prediction, classifying them into four main categories, i.e., statistical techniques, pattern recognition, machine learning, and hybrid approaches.

2.2.1 Statistical Techniques

Statistical techniques, when analysing stock markets, usually assume linearity and stationarity of financial time-series. The statistical approaches are mainly represented by ARIMA-based models, which are well-known techniques to analyse stock markets. They combine auto regressive models and mean reversion of trading markets to capture the shock effects observed in financial time-series (Hiransha et al., 2018). For example, De Faria et al. (2009) predicted Brazilian stock indices by comparing NNs and exponential smoothing models. Their results

show that the used NN slightly outperformed the adaptive exponential smoothing model in terms of the RMSE. Dutta et al. (2012) used financial ratios in a logistic regression model to analyse the relationship between these ratios and the stock performance. Their results show that the companies are classified into good and poor classes with a 74.6% accuracy, which highlights the importance of company health in stock analysis and prediction.

Later, Devi et al. (2013) trained an ARIMA-based model using historical data of Indian companies. Their results suggest that, because of low error and volatility, the Nifty Index is more suitable for naïve investors. Ariyo et al. (2014) used standard error of regression, adjusted R-square, and Bayesian information criteria to choose an optimal ARIMA-based model. The chosen ARIMA model successfully predicted stock prices of Nokia and Zenith Bank. Bhuriya et al. (2017) proposed to predict stock prices of Tata Consultancy Services using variants of regression models such as linear, polynomial, and radial basis function. Their results suggest that the linear regression model outperformed compared techniques by achieving a confidence value of 0.97. In general, statistical techniques assume a linear correlation in time-series data, which limits their application in real-life scenarios. These limitations are addressed by emerging techniques such as machine learning and hybrid approaches (Zhu et al., 2019).

2.2.2 Pattern Recognition

Pattern Recognition techniques try to identify future trends using pattern matching and historical templates. Their main focus is the detection of patterns and trends in data (Parracho et al., 2010). These patterns are identified by using charts that aim to capture variations in derived indicators such as price, volume, and price momentum (Nesbitt and Barrass, 2004). In general, pattern recognition techniques try to predict future market behaviour based on the degree of charts matching (Leigh et al., 2002). Traditional chart patterns are gaps, flags, wedges, spikes, pennants, triangles, saucers, and head-and-shoulders, among others (Park and Irwin, 2007). These patterns, according to some studies (Parracho et al., 2010), provide useful information about the future behaviour of stock prices.

For example, Leigh et al. (2007) used a bull flag pattern to capture price movements. Then, this information was incorporated into a template matching. The technique was tested in 9000 trading days of NYSE, beating the average market profit most of the times. Parracho et al. (2010) proposed an algorithmic

trading system using genetic algorithms and template matching. The proposed strategy is trained on S&P 500 stock data between 1998 and 2004, while the testing is carried out from 2005 to 2010. The authors claim that their results outperform the buy and hold strategy on an index. Phetchanchai et al. (2010) analysed financial time-series using the zigzag movement in the data, showing a performance improvement when compared with binary trees.

Later, Cervelló-Royo et al. (2015) proposed a chart pattern based trading rule using a flag pattern to model the closing operations. That is, they considered both the opening and closing prices to decide whether or not to start an operation. The authors claim to achieve a positive performance on the Dow Jones Industrial Average (DJIA) index. Their proposal was also tested on European indexes such as Deutscher Aktienindex (DAX) and Financial Times Stock Exchange (FTSE). Chen and Chen (2016) proposed an approach to identify patterns on TAIEX and NASDAQ indices. Their proposal obtains higher index returns when compared with rough set theory, genetic algorithms, and hybrid approaches. Arévalo et al. (2017) proposed to trade DJIA based on filtered flag pattern recognition using template matching, obtaining higher profit and lower risk when compared with the approach proposed in Cervelló-Royo et al. (2015). Recently, Kim et al. (2018) proposed a pattern matching trading system using dynamic time warping (DTW) to trade index futures, resulting in higher annualized returns and showing that most patterns tend to be more profitable near to the clearing time.

2.2.3 Machine Learning

A variety of machine learning techniques have been used to analyse financial markets (Shen et al., 2012). They can be broadly classified into unsupervised and supervised learning approaches (Ballings et al., 2015).

In unsupervised learning, the task is to transform and reduce the data such that their specific characteristics are highlighted. This learning strategy, unlike supervised learning, tries to find previously undetected patterns in a data set without a desired output value or label, e.g., clustering and dimensionality reduction (Van Der Maaten et al., 2009; Xu and Wunsch, 2008; Lei et al., 2017).

In supervised learning, the task is to learn a function that maps an input to an output based on example input-output pairs (Lei et al., 2017). The function is inferred from labelled training examples, where each example is a pair of an input and a desired output value. The training stops when the model achieves

a desired level of accuracy on the training data. Traditional examples of this learning strategy are classification and regression problems (Ahmed et al., 2010).

Unsupervised Learning

Clustering Algorithms. Clustering is a common technique with applications in fields such as pattern recognition and image analysis, which involves grouping data points that share similar characteristics (Xu and Wunsch, 2008). That is, data points that are in the same group (cluster) should have similar properties (features), while data points in different groups should have highly dissimilar properties. Traditional clustering methods include k-means (Yu et al., 2018b) and hierarchical clustering (Liu et al., 2016).

Powell et al. (2008) compared the performance of supervised and unsupervised machine learning techniques in financial markets. Their results show similar performance when tested on S&P 500 data, i.e., SVM achieved 89.1%, while k-means achieves 85.6%. Babu et al. (2012) proposed a clustering method to predict short-term impact on stocks after the release of financial reports. Firstly, they use text analysis to convert each financial report into a feature vector. Then, for each cluster, a clustering method was applied to identify the centroids. Lastly, the stock price movements were predicted using the centroids information. Their results suggest that the proposed method outperforms SVM-based models in terms of accuracy.

Wu et al. (2014) proposed a model based on k-means to predict stock trends by identifying chart patterns from data. Their model outperforms previous works such as Wang and Chan (2007) and Chen (2011) in terms of mutual funds and average returns. Peachavanish (2016) proposed a clustering method to identify stocks by their trends and momentum characteristics. Their proposal was tested using historical price data of stocks listed on the Stock Exchange of Thailand (SET), outperforming compared methods in long-term prediction tasks.

Topic Models. These techniques aim to explain the behavioural drivers of different market participants. They can be classified into natural language processing and textual analysis (Buchanan, 2019). For example, Antweiler and Frank (2004) found that internet stock message boards help to predict market volatility. Although the effect on stock returns is statistically significant, the economic significance is small. Tetlock (2007) used a daily Wall Street Journal column to

examine interactions between stock prices and the media. The results show that there is a correlation between downward pressure on market prices and high media pessimism. Tetlock et al. (2008) proposed a dictionary-based approach to predict accounting earnings of individual firms and stock returns. Their results show that media content captures hard-to-quantify aspects of firms' fundamentals, which is incorporated into stock prices by investors.

Guo et al. (2016); Loughran and McDonald (2016) provided surveys of textual analysis applications in accounting and finance. They found that the main challenges of tonal classification are sarcasm, slang, and the changing vocabulary on social media. Manela and Moreira (2017) constructed a text-based measure of uncertainty using front-page articles of Wall Street Journal. They measured news implied volatility peaks during financial crises, stock market crashes, and world wars. Their results suggest that high news implied volatility predicts high future returns during normal times. Renault (2017) provided empirical evidence that online investor sentiment helps to predict intra-day stock index returns.

Supervised Learning

Random Forest. The Random Forest approach is based on decision tree models, which is useful for forecasting and classification tasks (Breiman, 2001). This ensemble method constructs many decision trees that are used to classify a new instance by the majority vote. In Random Forest, each decision tree node uses a subset of attributes randomly selected from the whole original set of attributes (Oshiro et al., 2012).

Ballings et al. (2015) predicted long term stock price directions in 5,767 European companies, finding that random forest provided the best performance in terms of area under the curve. Milosevic (2016) proposed a long term prediction approach for stock market prices. The study performed a feature selection strategy in several machine learning algorithms. Results show that, when compared with Naïve Bayes and SVM, Random Forest achieved the best performance with 0.751 F-Score.

Zhang et al. (2018a) proposed a stock price trend prediction strategy based on Random Forest. Their model was trained using the Shenzhen Growth Enterprise Market to classify multiple clips of stocks into four main classes, i.e., up, down, flat, and unknown. Their results show that the proposed strategy is robust to market volatility and achieves competitive performance in terms of return per

trade. In Medeiros et al. (2019), sixteen machine learning methods were compared with statistical models, finding that Random Forest was the most suitable model to indicate a degree of non-linearity in the inflation dynamics.

Neural Networks (NNs) NNs are computational structures that aim to recognize underlying relationships in a set of data through a process inspired by the biological central nervous system (Yegnanarayana, 2009). NNs have proven useful in a wide variety of real-world problems related to sequence prediction and finance (Bahrammirzaee, 2010; Mohammed et al., 2017; Bouallegue, 2017), where classification, decision support, financial analysis, and credit scoring are the main fields of applications (Tkáč and Verner, 2016). Jasic and Wood (2004) proposed to predict daily stock market index returns using an NN model. Their study uses daily closing values of DAX, S&P 500, TOPIX, and FTSE indices between 1965 and 1999. Their results show an improvement in prediction when compared with linear autoregressive models.

Enke and Thawornwong (2005) used NN-based model to predict future values on S&P 500 from 1976 to 1999. Their results show that trading strategies guided by classification models achieve higher risk-adjusted profits when compared with the buy-and-hold strategy. Yümlü et al. (2005) provided a comparative study to predict financial time-series using NN-based models. The compared methods were tested on the Istanbul Stock Exchange National 100 Index (XU100) over 12 years. Their results suggest that the smoothed-piecewise model outperforms the other ones in terms of hit rate, positive hit rate, negative hit rate, MSE, Mean Absolute Error (MAE), and correlation. Bernal et al. (2012) implemented a RNN to predict S&P 500 stock prices, outperforming a Kalman filter with a 0.0027 test error. Their approach was tested in 50 stocks, reporting competitive performance when compared with state of the art techniques. Chen et al. (2015) proposed a model based on LSTM to predict the return of Chinese shares. The authors claim that LSTM outperforms random forecasting when predicting stock returns.

Hafezi et al. (2015) proposed a NN-based model to predict eight years of DAX index. The authors claim to achieve competitive performance in the long-term when compared with genetic algorithms and NN models. Di Persio and Honchar (2017) used three different RNN models to predict Google stock prices. Their results show that LSTM outperforms compared methods with 72% accuracy using

a five-day prediction horizon. Selvin et al. (2017) predicted stock prices of the National Stock Exchange of India (NSE) using LSTM, RNN, and CNN. Their proposal was tested using sliding windows on equally spaced data. Their results show that CNN captures dynamical changes on the data, outperforming compared techniques in terms of prediction accuracy.

Samarawickrama and Fernando (2017) predicted future prices of companies listed in the Colombo Stock Exchange (CSE) using RNN-based models. Their results show that LSTM outperforms FNN in terms of prediction accuracy. Hossain et al. (2018) proposed a hybrid model based on deep learning to predict S&P 500 time-series over 66 years. Their proposal achieves 0.00098 MSE, outperforming similar NN approaches. Fischer and Krauss (2018) proposed an LSTM model to predict financial time-series using stock records of S&P 500 between 1992 and 2015, finding that common patterns of securities show high volatility and a short-term inversion return profile. Siami-Namini et al. (2018) conducted a comparative study of machine learning methods using historical monthly financial time-series from 1985 to 2018. The authors reported that LSTM, when compared with ARIMA-based models, provided the best overall performance in terms of RMSE. Rundo et al. (2019) proposed an LSTM-based framework to predict stock prices and trends in financial time-series. Their results show that the proposed framework outperforms statistical models in terms of prediction accuracy. Recently, Lv et al. (2019) observed the daily trading performance of stocks using a variety of machine learning algorithms such as SVM, Random Forest, Logistic Regression, and deep NN architectures, among others. Their results suggest that deep NN algorithms have better performance when transaction costs are considered.

Support Vector Machines (SVMs). SVMs are a supervised machine learning technique mainly used for data classification. This technique represents the samples as points in a space, aiming to create gap between different categories. Then, the new samples are classified considering the category in which they most likely belong (Scardapane et al., 2016), e.g., classifying a future stock price direction into rise or drop (Chiu and Chen, 2009).

Fan and Palaniswami (2001) proposed an SVM-based model to increase profits in stock markets. The authors report a total return of 208% over a five years period in the selected stocks. Cao and Tay (2003) compared the performance of models based on SVM, multilayer NN, and regularized radial base function in

financial time-series prediction. The models were evaluated using data from the Chicago Mercantile Market. Their results show that SVM achieves competitive performance considering the convergence speed and number of hyper parameters. Lee (2009) proposed to predict the trend of stock markets using an SVM model. The task was to predict the NASDAQ index direction over 6 years, i.e., from 2001 to 2007. Their results show that the proposed SVM-based model has the highest level of prediction accuracy and generalization when compared with information gain, symmetrical uncertainty, correlation-based feature selection, and back-propagation NNs.

Later, Schumaker and Chen (2009) combined textual analysis with SVM to analyse the impact of news articles on stock prices. They analysed a large number of news articles and stocks between October 26, 2005 and November 28, 2005. Their proposal estimated the closest value to the actual future stock price, direction of price movement, and the highest return using a simulated trading engine. Yeh et al. (2011) proposed an SVM-based model with kernel functions to predict stock market values. They developed a multi-kernel learning algorithm using sequential minimal optimization and a gradient projection approach. Their proposal, when trying to predict TAIEX index, outperforms similar methods between October 2002 and March 2005. Nayak et al. (2015) proposed to predict Indian stock indices (Bombay Stock Exchange and CNX Nifty) using SVM-based models. The authors claim that their proposal achieves relatively competitive performance in terms of MSE. Zhang et al. (2016) proposed a classification method that combines SVM, AdaBoost, and genetic algorithms. Their proposal was tested in shares of the Shenzhen Stock Exchange and NASDAQ, achieving competitive results when compared with NN models.

2.2.4 Hybrid Analysis

Hybrid Analysis methods combine different techniques such as statistical, pattern recognition, and machine learning. For example, Tiwari et al. (2010) proposed to predict trends of the Bombay Stock Exchange (BSE) Sensex using Hierarchical Hidden Markov Model (HHMM) and decision trees. Their proposal used a classifier to perform predictions, while the HHMM evaluated prediction performance, yielding an accuracy of 92.1%. Guresen et al. (2011) used a hybrid NN model within an Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

approach to extract input variables. Their proposal was tested on NASDAQ index values between October 7, 2008 and June 26, 2009, finding that MLP provides a simple and efficient alternative when compared with NN-based models.

Shen et al. (2012) proposed to predict daily stock prices combining statistical and SVM approaches. Their proposal achieves 77.6% prediction accuracy on the DJIA and around 85% for long term predictions. Dai et al. (2012) proposed to combine Non-linear Independent Component Analysis (NLICA) and NN to predict Asian stock markets. Firstly, with the aim of representing the underlying data information, they used NLICA to transform the input space into a feature space of independent components. Then, the model is built by feeding the NN with those independent components. The authors claim to improve prediction accuracy of NN-based models when using data from the Nikkei 225 and Shanghai B-share closing index. Wang et al. (2012) proposed to predict weekly stock prices by combining ARIMA and Back-propagation NN (BPNN), outperforming compared approaches when tested on the Shenzhen Integrated Index and DJIA with a 70.16% directional accuracy.

Yoshihara et al. (2014) considered long term effects of news events using a combination of Restricted Boltzmann Machine (RBM) and DBN to predict binary stock trends, i.e., up or down. The authors claim that their proposal achieved the lowest error rates when compared with SVM and DBN. Ding et al. (2015) proposed to predict S&P 500 index combining sentiment analysis and NN-based models. Their study considered more than 10 million events over seven years, achieving of 64.21% on the S&P 500 index. Rather et al. (2015) proposed to predict stock prices using ARIMA, genetic algorithms, ESN, and RNN models, achieving the lowest MAE and MSE when compared with RNN. Patel et al. (2015) compared four prediction models (NN, SVM, Random Forest, and Naive Bayes) using two different approaches. The first approach computes technical parameters on open, high, low, and closing prices, while the second one represents these technical parameters as trend deterministic data. Their approaches were tested on the NSE and S&P BSE Sensex, covering the period between January 2003 and December 2012. Their results suggest that Random Forest outperforms compared methods using the first approach. In addition, they found that prediction performance improves when the technical parameters are represented as trend deterministic data.

More recently, Dash and Dash (2016) proposed a decision support system using NN and trading strategies, which are seen as a classification problem with three possible values, i.e., buy, hold, and sell. Their model is compared with SVM, k-nearest neighbour, and decision tree algorithms using stock prices from BSE SENSEX and S&P 500. The authors claim that combining technical indicators with computational intelligence tools produce more profitable trading decisions. Li et al. (2016) proposed to predict stock prices using Extreme Learning Machine (ELM) and trading strategies. Their results show that ELM and SVM achieve higher prediction accuracy and convergence speed when compared with BPNN models. Creighton and Zulkernine (2017) extended the work of Wang et al. (2012) by applying a hybrid approach to daily stock price on the S&P 400 and S&P 500. Their results show that the proposed hybrid approach did not outperform its constituent models. Pierdzioch and Risse (2018) implemented an orthogonality test of the rationality of aggregate stock market forecasts using Boosted Regression Trees (BRT). The proposed implementation was tested on S&P 500 index from 1992 to 2014. Their results suggest that the rational expectations hypothesis cannot be rejected for short term forecasts. Zhong and Enke (2019) proposed to predict stock return directions using a NN-based model and Principal Component Analysis (PCA). The method was tested using closing prices of the SPDR S&P 500 ETF over 2518 trading days between 2003 and 2013. Their proposal achieves significantly higher classification accuracy than other hybrid machine learning algorithms. Finally, Alhnaity and Abbod (2020) proposed to predict financial time-series using intelligent model techniques and a feature extraction algorithm. Their study considered the FTSE 100, S&P 500 and Nikkei 225 next day closing prices, outperforming compared methods in terms of MSE, RMSE, and MAE.

Figure 2.1 summarizes the main financial data analysis approaches and their financial applications. This figure shows that algorithmic trading (see Section 2.1.1) and risk assessment (see Section 2.1.2) are popular financial applications and their models are mainly based on NNs, e.g., RNN, LSTM, and CNN.

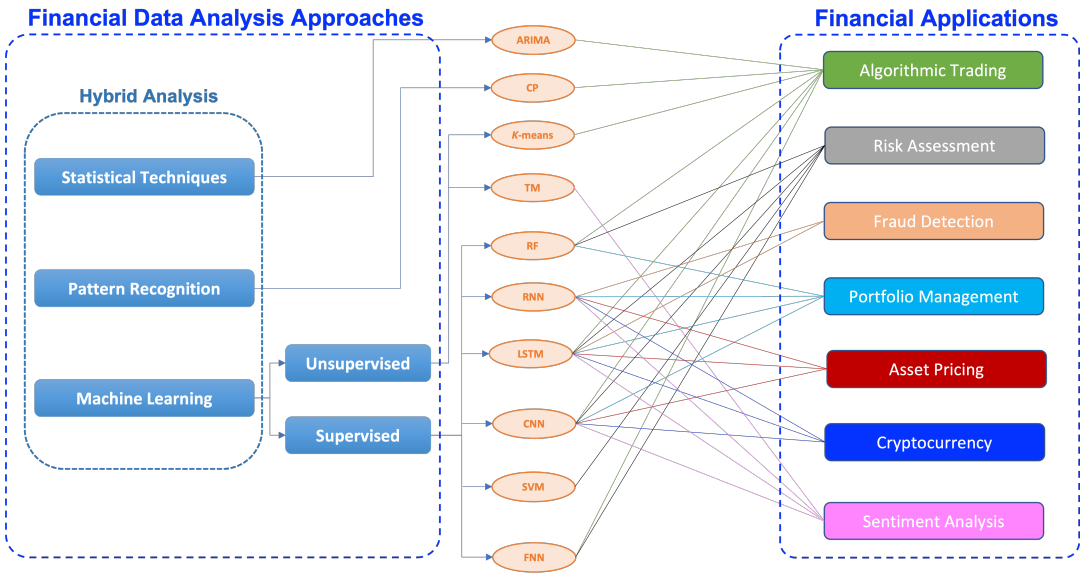


Figure 2.1: Financial data analysis approaches and their financial applications.

2.3 Research Gaps and Challenges

A variety of statistical and machine learning approaches for analysing financial data have been summarized in Sections 2.1 and 2.2. Although the existing studies have made significant contributions to the development of regression models within the context of financial markets, there still exist some limitations for those approaches. Thus, in order to highlight the research motivation of this thesis, the following summarises the identified research limitations:

1. **Statistical Models**. Several prediction methods that consider the sequential nature of financial time-series have been proposed. Statistical models, such as ARIMA (Liang and Schienle, 2019; Durbin and Koopman, 2012; Ramos et al., 2015), vector autoregression approaches (Jang, 2020), vector error correction models (Liang and Schienle, 2019), and HMM (Hassan, 2009; Dias et al., 2015; Zhang et al., 2019b) were developed for prediction and temporal pattern recognition tasks. However, their models may not be well suited for characterizing the stochastic nature, uncertain dynamics, and non-stationary conditions of real-world applications (Khashei and Hajirahimi, 2019; Suhermi et al., 2018).

2. **Feed-forward Neural Networks (FNN)**. These are powerful prediction tools compared to more conventional models, e.g., MLP (Ravi et al.,

2017; Abedin et al., 2019), time-delay neural networks (Tai et al., 2019), differential evolution training (Ilonen et al., 2003), hybrid particle swarm optimization (Mirjalili et al., 2012), multi verse optimizer (Faris et al., 2016), logistic regression with a single hidden layer (Belciug, 2020), fitting based early stopping (Chi et al., 2019), modular networks (Eshaghzadeh and Hajian, 2018), biogeography based optimization (Rodan et al., 2016), among others (Ojha et al., 2017). However, the processes occurring during their training are difficult to interpret, meaning that is not clear how learning from input data is done or how performance can be consistently ensured (Svozil et al., 1997; Olden and Jackson, 2002; Ghasemi et al., 2018). This limits their utility and acceptance in many real-world applications, as it is desirable to use techniques based on analytical functions that can be understood and validated (Zhang et al., 2018b). In addition, even when performing similar tasks, the proper choice of network parameters can vary widely and are often chosen using a trial-and-error process (Erkaymaz et al., 2017; Dutta et al., 2018).

3. **Recurrent Neural Networks (RNN)**. These methods model sequence structure with recurrent lateral connections and process the data sequentially one record at a time (Pascanu et al., 2013), e.g., LSTM (Li et al., 2017d; Fischer and Krauss, 2018; Liu et al., 2018), Bayesian scheme (Fortunato et al., 2017), hierarchical (Chung et al., 2016; Yu et al., 2016), convolutional approaches (Choi et al., 2017; Cakır et al., 2017), unitary evolution (Arjovsky et al., 2016), dilated recurrent skip connections (Chang et al., 2017), bidirectional networks (Schuster and Paliwal, 1997; Fan et al., 2014b; Ogawa and Hori, 2017), gated feedback (Chung et al., 2015), multimodal (Mao et al., 2014), Lyapunov components (Güler et al., 2005), complex valued networks (Hu and Wang, 2012), among others (Qin et al., 2020). However, these strategies may fail when predicting financial time-series, as they can become stuck in local minima during the training stage (Principe, 2010). In addition, as NN-based models are retrained at regular intervals in sequence prediction tasks, they require both significant computing and storage resources (Gu et al., 2014; Cui et al., 2016).

4. **Kernel Adaptive Filtering (KAF)**. These are data-driven approaches that have proven useful in identifying non-linear systems (Liu et al., 2010;

Li and Príncipe, 2017). Such non-parametric methods combine the convex optimization of linear adaptive filters and the universal approximation property of NNs (Chen et al., 2016a). Their training scheme uses a combination of error-correction and memory-based learning, meaning that the whole training set is not required to start learning the model. Thus, as the samples arrive, the model is sequentially updated while predictions are being made, which is useful in online applications. That is, the model is learned using a single pass through the entire training set in KAF algorithms. The main advantage of these methods, when compared with NNs, is their ability to represent non-linear functions linearly and universally in Reproducing Kernel Hilbert Space (RKHS), which is isometric-isomorphic to a high dimensional feature space (Schölkopf et al., 2018). This results in a simple convex optimization problem with a unique global optimum (Schölkopf and Smola, 2001). Traditional KAF methods include the kernel least mean square (KLMS) (Liu et al., 2008), kernel affine projection (KAPA) (Liu and Príncipe, 2008), and kernel recursive least squares (Engel et al., 2004), among others (Chen et al., 2013b; Zhang et al., 2019a). However, there remain two main challenges in KAF algorithms: determination of kernel-size and tuning of the step-size parameters (Chen et al., 2016a).

5. **Market Interdependence Approach**. In addition to the sequential nature of financial time-series, stock markets are themselves highly complex systems depending on various factors such as financial policies, national economic growth and sector performance (Zhang et al., 2019b). It has became clear that all agents involved in a given stock market may exhibit interconnections and correlations, representing important internal forces of the market (Collins and Biekpe, 2003; Jizba et al., 2012). That is, the movement of a stock market in a country is likely to be affected by movement of other stocks in both that country and in other regions (Masih and Masih, 2001). The following strategies have been proposed to identify and quantify interactions on this type of complex system (Greenblatt et al., 2012): (1) *Space-time*, such as covariance that measures the joint variability of two random variables (Wang and Ye, 2016; De Ketelaere et al., 2018), correlation that measures statistical relationships between two random variables (Kenett et al., 2015), Granger causality that determines whether one time-series is useful in predicting another one (Papana et al., 2017), and

Shannon entropy that measures the uncertainty of random variables (Sulthan et al., 2016); (2) *Space-frequency and Space-time-frequency*, such as Fourier transform that decomposes a function into its constituent frequencies (Fang and Chang, 2017; Saia et al., 2017), phase synchronization which is the process where cyclic signals tend to oscillate with a repeating sequence of relative phase angles (Radhakrishnan et al., 2016), directed transfer function that determines the directional influences between any given pair of channels in a multivariate data set (Kamiński et al., 2001), wavelet transform where basis functions are scaled and shifted versions of one function called a mother wavelet (Joo and Kim, 2015; Saia, 2017). The previous work studies how the price of one stock is influenced by the economic factors of other markets. However, their models do not consider changes in network structure over time, meaning that the conditions for which the models were optimized may disappear (Olsen et al., 2018). Hence, how to incorporate these interdependencies into an analytical model, such as sequential learning, to predict financial time-series in real-time remains an open issue.

The next chapter, aiming to address the first objective of this thesis, introduces a kernel-based approach to support sequence prediction tasks in financial markets. The proposed approach learns from continuous sequence of data records and addresses two well-known problems of KAF, i.e., selection of kernel-size and step-size parameters.

Chapter 3

A Kernel-based Sequence Prediction Approach

In this chapter, with the aim to address the first objective of this thesis, we propose a kernel-based sequence prediction approach for financial time-series[1]. The approach addresses two main challenges of kernel adaptive filtering (KAF): (1) the lack of an effective method to determine kernel-sizes in an online learning context; (2) how to tune the step-size parameter. The kernel-sizes, unlike traditional KAF formulations, are both created and updated in an online sequential way. Further, to improve convergence time, we propose an adaptive step-size strategy that minimizes the mean square error (MSE) using a stochastic gradient algorithm. The proposed approach is validated on two real-world data sets; results show both faster convergence to relatively low values of MSE and better accuracy when compared with KAF-based methods, long short-term memory, and recurrent neural networks.

This chapter is structured as follows: Section 3.1 introduces the proposed approach; Section 3.2 presents the experiment settings of this work; Sections 3.3 and 3.4 provides simulation results and summarizes the chapter, respectively.

[1]The outcomes of this chapter were published in *Neurocomputing* under the title "*Learning from data streams using kernel least-mean-square with multiple kernel-sizes and adaptive step-size*" (Garcia-Vega et al., 2019).

3.1 An Approach for Sequence Prediction

We develop an approach for sequence prediction that addresses two main challenges of KAF algorithms: (1) an algorithm that uses multiple kernel-sizes in sequence learning to address the kernel-size problem (Section 3.1.1); (2) an online technique to optimize the step-size (Section 3.1.2).

3.1.1 Multiple Kernel-Sizes in Online Sequential Learning

Suppose the goal is to learn a continuous input-output mapping $f : \mathbb{U} \to \mathbb{R}$ based on a sequence of input-output examples $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, where $\mathbb{U} \subset \mathbb{R}^M$ is the input domain, $\boldsymbol{u}_t \in \mathbb{R}^M$ is an input vector, and $y_t \in \mathbb{R}$ is the desired output. The kernel least mean square (KLMS) algorithm (see Appendix B.3.2), the simplest in the KAF family, is a sequential estimator of f such that f_t is updated using the last estimate f_{t-1} and the current example $\{\boldsymbol{u}_t, y_t\}$, yielding the following sequential rule in the original input space (Liu et al., 2008):

$$\begin{cases} f_0 = 0 \\ e_t = y_t - f_{t-1}(\boldsymbol{u}_t) \\ f_t = f_{t-1} + \eta e_t \kappa_\sigma(\boldsymbol{u}_t, \cdot) \end{cases} \tag{3.1}$$

where $e_t \in \mathbb{R}$ is the prediction error, f_t denotes the learned mapping at iteration t, $\eta \in \mathbb{R}^+$ is the step-size parameter, and $\kappa_\sigma(\cdot, \cdot) \in \mathbb{R}^+$ is a Mercer kernel with a kernel-size $\sigma \in \mathbb{R}^+$ that controls the mapping smoothness (Schölkopf and Smola, 2001). This sequential rule produces a growing radial-basis function network (RBFN) by allocating a new kernel unit for each new example with $\boldsymbol{u}_t$ as the centre and ηe_t as its coefficient, which poses time-space complexity issues for continuous adaptation scenarios[2].

We propose an online algorithm to address the kernel-size problem in the KLMS algorithm. This proposal, unlike traditional multiple-kernel-learning formulations, does not need a predefined set of kernel-sizes. Here, the algorithm creates a pool of kernel-sizes in an online sequential way. In addition, the appropriate kernel-size is selected for every new input sample. This allows the algorithm to adjust to abrupt changes in the system, which is useful in non-stationary conditions. More formally, the kernel-size can be computed as follows (Chen et al.,

[2]The set of centres and coefficients are also known as dictionary and weights, respectively.

2016a):

$$\sigma_t = \sigma_{t-1} + \frac{\rho e_{t-1} e_t \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\|^2 \kappa_{\sigma_{t-1}}(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t)}{\sigma_{t-1}^3}, \tag{3.2}$$

where: (1) the kernel-size at iteration $t-1$ is σ_{t-1}; (2) the free parameter for the kernel-size adaptation is ρ; (3) the prediction errors at time $t-1$ and t are e_{t-1} and e_t, respectively; (4) the input samples at time $t-1$ and t are $\boldsymbol{u}_{t-1}$ and $\boldsymbol{u}_t$, respectively; (5) $\|\cdot\|$ denotes the ℓ_2 norm; (6) $\kappa_\sigma(\cdot,\cdot)$ is a Mercer kernel. Thus, the following observations can be made from Equation (3.2): (1) the gradient direction depends on the signs of the prediction errors e_{t-1} and e_t. If there is little sign change, the desired mapping is likely a "*smoothing function*"; in contrast, if the sign changes frequently, the desired mapping is likely a "*zig zag function*"; (2) the magnitude of the gradient depends on σ_{t-1} through $\kappa_{\sigma_{t-1}}(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t)/\sigma_{t-1}^3$. For the case $\boldsymbol{u}_t \neq \boldsymbol{u}_{t-1}$, this term will approach zero when σ_{t-1} is very small or very large. Although the gradient goes to zero, it does not imply that σ_t is bounded, however with a proper initial value, the kernel-size will be adjusted within a reasonable range; (3) if the initial kernel-size σ_1 is inappropriately chosen, initial convergence speed can be very slow. In this case, the suitable initial kernel-size can be selected using a method such as Silverman's rule of thumb (Kang and Noh, 2019).

Equation (3.2) allows the kernel-size to be computed in an online sequential way. However, when the environment changes back to a previous mode, the online technique to optimize the kernel-size has no inherent mechanism for recall and thus must relearn the structure from scratch. In other words, as the kernel-size is updated sequentially, the prediction f_t (see Equation (3.1)) for the input sample $\boldsymbol{u}_t$ is calculated using σ_t, but the appropriate kernel-size for $\boldsymbol{u}_t$ — viewed from multiple-kernel-learning — may have been learned previously, i.e., $\sigma_1, \ldots, \sigma_{t-2}$, σ_{t-1}. Rather than discarding previously learned kernel-sizes, we use them to build a pool of kernel-sizes in an online sequential way. Then, for each new input sample, the appropriate kernel-size is selected from this pool. In practice, once the algorithm selects the appropriate kernel-size for the current sample $\boldsymbol{u}_t$, the next step is to apply a KLMS-based algorithm to perform the prediction task. Note that the KLMS-based algorithm provides a dictionary and its weights, which are used in a linear combination to obtain the predictions (see Equation (3.1), Algorithm 1, and Remark 2). We provide a step-by-step description of how we create the kernel-sizes in an online sequential way (Lines refer to Algorithm 1):

1. Start with an initial kernel-size σ_1, which is provided in advance (Line 1).

2. The centroid of the initial kernel-size $\boldsymbol{c}_1 \in \mathbb{R}^M$ is created using the first input sample $\boldsymbol{u}_1$ (Line 4).

3. The first input sample is stored in the dictionary of the initial kernel-size $\mathcal{C}_1$ (Line 5).

4. The new sample $\boldsymbol{u}_t$ and the centroids are projected into a high-dimensional feature space via a reproducing kernel (Line 12). This is also known as coherence (a fundamental parameter that characterizes a dictionary in linear sparse approximation problems) within the context of adaptive filtering with kernels (Richard et al., 2009). The coherence criterion suggests inserting the candidate input $\boldsymbol{u}_t$ as a new centre if its coherence $\max\limits_{1\leq i\leq|\mathbf{K}|} \kappa_{\sigma_i}(\boldsymbol{u}_t, \boldsymbol{c}_i)$ remains below a given threshold $\delta \in [0, 1]$. The main rationale behind the previous strategy is to provide both the level of sparsity and the coherence of the kernel-sizes created during learning (Gao et al., 2014).

5. The centroid threshold δ determines whether or not a new kernel-size should be created:

   - $\max\limits_{1\leq i\leq|\mathbf{K}|} \kappa_{\sigma_i}(\boldsymbol{u}_t, \boldsymbol{c}_i) \geq \delta$: This means that one of the current kernel-sizes is appropriate for the new sample; thus, there is no need to create a new kernel-size and the most similar kernel-size i^* is selected for the new sample $\boldsymbol{u}_t$ (Line 11). Next, the centroid of the most similar kernel-size $\boldsymbol{c}_{i^*}$ is updated (Lines 13-14).
   - $\max\limits_{1\leq i\leq|\mathbf{K}|} \kappa_{\sigma_i}(\boldsymbol{u}_t, \boldsymbol{c}_i) < \delta$: A new kernel-size is created (Line 17). Then, the current sample $\boldsymbol{u}_t$ is used to create the centroid of the new kernel-size $\boldsymbol{c}_{|\mathbf{K}|+1}$ (Lines 18-19). After that, the dictionary and the pool of kernel-sizes are updated (Lines 21-26).

6. Following selection of the appropriate kernel-size for the current sample, the next step applies a KLMS-based algorithm to perform the prediction task (Line 29). Lastly, the kernel-size and the step-size are updated before the next sample $\boldsymbol{u}_{t+1}$ arrives (Lines 32-36).

Algorithm 1: Multiple kernel-sizes in online sequential learning.

input : $\mathcal{T}$– training data
Parameter setting: η_1-initial step size, σ_1-initial kernel size, δ-centroid threshold
$\mathbf{K} = \{\sigma_1\}$: Pool of kernel-sizes
$l_1 = 1$: Effective size of σ_1 for centroid update
$\boldsymbol{c}_1 = \boldsymbol{u}_1$: Centroid of σ_1
$\mathcal{C}_1 = \{\boldsymbol{u}_1\}$: Initial dictionary of σ_1
$\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1\}$: Set of dictionaries
$\omega_1 = \eta_1 y_1$: Initial weights of σ_1
$\boldsymbol{\mathcal{W}} = \{\omega_1\}$: Set of weights
Computation:
while $\{\boldsymbol{u}_t, y_t\}$ *available* **do**
    Select the most similar kernel-size: $i^* = \arg\max_{1 \le i \le |\mathbf{K}|} \kappa_{\sigma_i}(\boldsymbol{u}_t, \boldsymbol{c}_i)$
    if $\max_{1 \le i \le |\boldsymbol{K}|} \kappa_{\sigma_i}(\boldsymbol{u}_t, \boldsymbol{c}_i) \ge \delta$ **then**
        Update centroid of the most similar kernel-size: $\boldsymbol{c}_{i^*} = \frac{l_{i^*}\boldsymbol{c}_{i^*} + \boldsymbol{u}_t}{l_{i^*}+1}$
        Update effective size of the most similar kernel-size: $l_{i^*} = l_{i^*} + 1$
    else
        The kernel-size $\sigma_{|\mathbf{K}|}$ reaches its final value: $\sigma_{|\mathbf{K}|} = \sigma_{|\mathbf{K}|,t-1}$
        Start updating a new kernel-size: $\sigma_{|\mathbf{K}|+1} = \sigma_{i^*,t-1}$ (similar to Line **1**)
        Create centroid for new kernel-size: $\boldsymbol{c}_{|\mathbf{K}|+1} = \boldsymbol{u}_t$
        Effective size of new kernel-size: $l_{|\mathbf{K}|+1} = 1$
        Knowledge transfer
        Create dictionary of new kernel-size: $\mathcal{C}_{|\mathbf{K}|+1} = \mathcal{C}_{i^*}$ (see Line **11**)
        Update set of dictionaries: $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}, \mathcal{C}_{|\mathbf{K}|+1}\}$
        Create weights of new kernel-size: $\omega_{|\mathbf{K}|+1} = \omega_{i^*}$ (see Line **11**)
        Update set of weights: $\boldsymbol{\mathcal{W}} = \{\boldsymbol{\mathcal{W}}, \omega_{|\mathbf{K}|+1}\}$
        Update most similar kernel-size: $i^* = |\mathbf{K}| + 1$
        Update pool of kernel-sizes: $\mathbf{K} = \{\mathbf{K}, \sigma_{|\mathbf{K}|+1}\}$
    Kernel-based adaptive filter
    input : $\mathcal{C}_{i^*}$, ω_{i^*}, σ_{i^*}, $\boldsymbol{u}_t$, y_t, η_t
    → [**Here**: KLMS-based algorithm] ←
    output: $\mathcal{C}_{i^*}$, ω_{i^*} (updated)

    if $i^* = |\boldsymbol{K}|$ **then**
        Update kernel-size in i^* using Equation (3.2):
        $$\sigma_{i^*,t} = \sigma_{i^*,t-1} + \frac{\rho e_{t-1} e_t \|\boldsymbol{u}_{i^*,t-1} - \boldsymbol{u}_t\|^2 \kappa_{\sigma_{i^*,t-1}}(\boldsymbol{u}_{i^*,t-1}, \boldsymbol{u}_t)}{\sigma^3_{i^*,t-1}}$$
    else
        The kernel-size σ_{i^*} reaches its final value: $\sigma_{i^*,t} = \sigma_{i^*,t-1}$
    Update the step-size η_t [To be explained in Section 3.1.2]
output: $\mathbf{K}$, $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{W}}$, $\{\boldsymbol{c}_i : i \in [1, |\mathbf{K}|]\}$

Remark 1 (Centroid) *In Algorithm 1, unlike the original formulation (Li and Príncipe, 2017), we use the centroids to determine the appropriate kernel-size*

for each new input sample $\boldsymbol{u}_t$. The new input sample is projected in a high-dimensional feature space. This is done using the centroids $\mathbf{c}_i$ and their kernel-sizes σ_i, i.e., here, the centroid is a single representative of each kernel-size.

Remark 2 (Kernel-size optimization) *In Algorithm 1, optimization of the kernel-size is performed only on the most recently created kernel-size (Line 33), which means that all other kernel-sizes have a fixed value (Line 35). Once a new kernel-size is created $|\boldsymbol{K}|+1$, optimization of the kernel-size $|\boldsymbol{K}|$ stops (Line 16). Consequently, the pool of kernel-sizes $\boldsymbol{K}$ is created and updated in an online sequential way.*

Remark 3 (Knowledge-transfer) *When a new σ is added to the pool of kernel-sizes, its dictionary and weights start with the dictionary of the most similar kernel-size i^* at time t (Lines 21-24); thus, it is possible to have overlaps in the dictionaries of the kernel-sizes. This strategy can be viewed as a smoothing procedure (also known as knowledge-transfer). In the worse case, the dictionary of the new σ will retain a dictionary size equivalent to KLMS. Note, if we allow the dictionary of the new σ to be initialized from scratch, as in traditional adaptive filtering, this will result in a discontinuity of performance in time. In practice, the kernel-sizes are automatically created in the early stages of training, meaning that overlaps are small. Our proposal uses previously learned knowledge with the appropriate kernel-size to enhance prediction. The knowledge-transfer scheme has similarities to a previously proposed method (Li and Príncipe, 2017); the key difference in the two strategies is that our proposal copies not only the weights but also the dictionary of the most similar kernel-size. This better utilizes the kernel-sizes, avoids large discontinuities in learning time, provides more efficient training in non-stationary conditions, and in some circumstances may yield a more compact network (Chen et al., 2012).*

Remark 4 (Computational and memory issues) *In our proposal, the appropriate kernel-size is selected when a new sample arrives. Then, the prediction task is performed using this kernel-size together with its dictionary and weights. Therefore, as mentioned in Remark 3, the selected dictionary will retain a dictionary size equivalent to KLMS in the worst case (see Figure 3.2).*

The general scheme of the proposed approach for sequence prediction at iteration t is summarized in Figure 3.1.

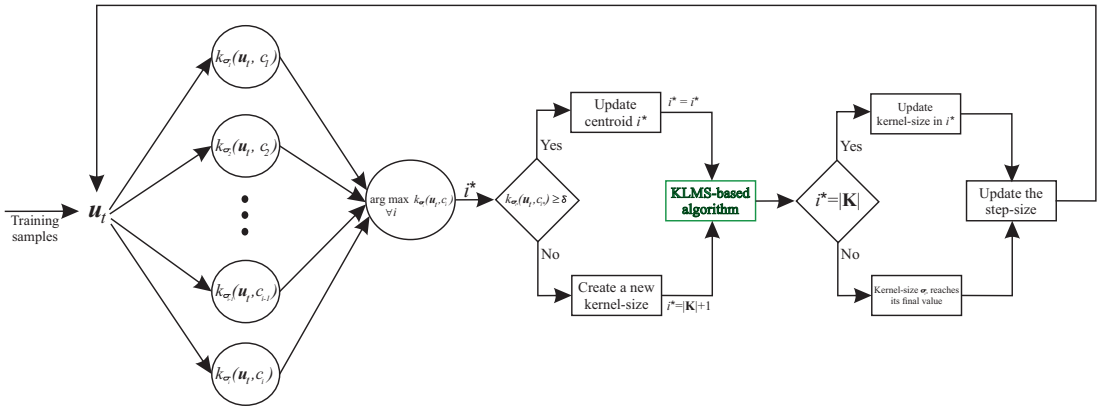
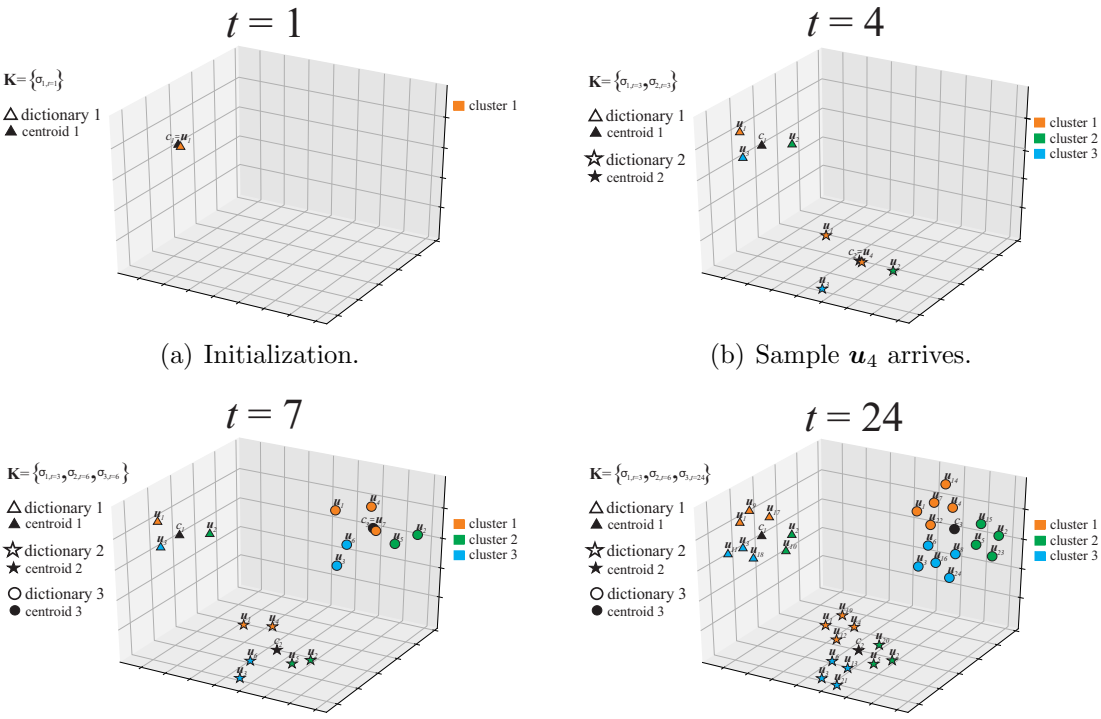


Figure 3.1: Proposed approach for sequence prediction at iteration t.

Figure 3.2 shows how Algorithm 1 works in practice when the input samples are $\boldsymbol{u}_t \in \mathbb{R}^3$. The first step is to start a kernel-size with its dictionary, where the syntax $\sigma_{1,t=1}$ stands for the first kernel-size at time $t = 1$ (see Figure 3.2(a)). At that moment, the update procedure (see Equation (3.2)) on the first kernel-size creates a new kernel-size. Then, at time $t = 4$, the similarity between $\boldsymbol{u}_4$ and the centroid of the first kernel-size c_1 is calculated. However, $\kappa_{\sigma_{1,t=3}}(\boldsymbol{u}_4, \boldsymbol{c}_1)$ is not greater than δ, indicating the presence of a new dynamic in the system. Consequently, a new kernel-size is created and $\boldsymbol{u}_4$ is its initial centroid. Note, when the second kernel-size is created at time $t = 4$: (1) optimization of the first kernel-size stops, and, from this point on, its kernel-size will be $\sigma_{1,t=3}$; (2) the second kernel-size starts with the last known σ; (3) optimization on the second kernel-size stops when a new kernel-size is again created; (4) to avoid discontinuities in learning, the samples in the dictionary of the first kernel-size ($\boldsymbol{u}_1$, $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$) are copied to the dictionary of the new kernel-size (knowledge transfer); (5) the three clusters inside the dictionaries of the kernel-sizes are generated by the KLMS-based algorithm (see Figure 3.2(b)). Finally, at time $t = 7$, a new kernel-size is again created and the process is repeated in a similar way as at time $t = 4$ (see Figure 3.2(c)). Note that, at time $t = 24$: (1) kernel-size optimization is performed only on the last created kernel-size; (2) the first and second kernel-size already have a fixed value, i.e., $\sigma_{1,t=3}$ and $\sigma_{2,t=6}$, respectively (see Figure 3.2(d)).

Lastly, a major advantage of Algorithm 1 is that the prediction is performed based on a relatively low number of samples. This is because for each new sample we select its best possible kernel-size as well as its closest dictionary (from the point of view of data distribution). For example, when $\boldsymbol{u}_{24}$ arrives, the prediction task is performed using only the **5 samples** of cluster 3 in the dictionary of the third kernel-size, i.e., $\boldsymbol{u}_3$, $\boldsymbol{u}_6$, $\boldsymbol{u}_8$, $\boldsymbol{u}_{16}$, and $\boldsymbol{u}_{24}$ (see Figure 3.2(d)). In comparison to the traditional KLMS algorithm, the same prediction is calculated using **24 samples**, i.e., $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\boldsymbol{u}_3$, $\cdots$, $\boldsymbol{u}_{24}$. As a result, Algorithm 1 runs much faster than KLMS and variants.



(a) Initialization.

(b) Sample $\boldsymbol{u}_4$ arrives.

(c) Sample $\boldsymbol{u}_7$ arrives.

(d) Sample $\boldsymbol{u}_{24}$ arrives.

Figure 3.2: Multiple kernel-sizes in online sequential learning.

3.1.2 An Online Technique to Optimize Step-Size

In this section, we propose an adaptive step-size strategy, another well-known challenge in KAFs. Thus, at each iteration, the step-size of the KLMS is optimized using the previous step-size and the current prediction error. More formally, at iteration t, when prediction error e_t is available, we propose to optimize the

step-size η_t by minimizing the instantaneous square error as follows:

$$\eta_t = \eta_{t-1} - \mu \frac{\partial}{\partial \eta_{t-1}} \left[e_t^2 \right]$$

where η_{t-1} is the step-size at iteration $t-1$ and $e_t = y_t - f_{t-1}(\boldsymbol{u}_t)$,

$$\begin{aligned}
\eta_t &= \eta_{t-1} - \mu \frac{\partial}{\partial \eta_{t-1}} \left[\left(y_t - f_{t-1}\Big(\boldsymbol{u}_t\Big) \right)^2 \right] \\
\eta_t &= \eta_{t-1} - \mu \frac{\partial}{\partial \eta_{t-1}} \left[y_t^2 - 2 y_t f_{t-1}\Big(\boldsymbol{u}_t\Big) + f_{t-1}^2\Big(\boldsymbol{u}_t\Big) \right]
\end{aligned}$$

from Equation (3.1), we have that $f_{t-1}(\boldsymbol{u}_t) = f_{t-2}(\boldsymbol{u}_t) + \eta_{t-1} e_{t-1} \kappa_\sigma(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t)$,

$$\begin{aligned}
\eta_t &= \eta_{t-1} - \mu \frac{\partial}{\partial \eta_{t-1}} \left[y_t^2 - 2 y_t \left(f_{t-2}\Big(\boldsymbol{u}_t\Big) + \eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right) \right. \\
&\quad \left. + \left(f_{t-2}\Big(\boldsymbol{u}_t\Big) + \eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right)^2 \right] \\
\eta_t &= \eta_{t-1} - \mu \left(\frac{\partial}{\partial \eta_{t-1}} \left[y_t^2 \right] - \frac{\partial}{\partial \eta_{t-1}} \left[2 y_t f_{t-2}\Big(\boldsymbol{u}_t\Big) \right] \right. \\
&\quad - \frac{\partial}{\partial \eta_{t-1}} \left[2 y_t \eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right] + \frac{\partial}{\partial \eta_{t-1}} \left[f_{t-2}^2\Big(\boldsymbol{u}_t\Big) \right] \\
&\quad + \frac{\partial}{\partial \eta_{t-1}} \left[2 f_{t-2}\Big(\boldsymbol{u}_t\Big) \eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right] \\
&\quad \left. + \frac{\partial}{\partial \eta_{t-1}} \left[\left(\eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right)^2 \right] \right)
\end{aligned}$$

since f_{t-2} does not depend on η_{t-1},

$$\eta_t = \eta_{t-1} - \mu \left(-2 \left[y_t - \left(f_{t-2}\Big(\boldsymbol{u}_t\Big) + \eta_{t-1} e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right) \right] e_{t-1} \kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \right)$$

note that, $f_{t-2}(\boldsymbol{u}_t) + \eta_{t-1} e_{t-1} \kappa_\sigma(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t)$ is actually $f_{t-1}(\boldsymbol{u}_t)$,

$$
\begin{aligned}
\eta_t &= \eta_{t-1} - \mu\Bigg(-2\Big[y_t - f_{t-1}\Big(\boldsymbol{u}_t\Big)\Big] e_{t-1}\kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big)\Bigg) \\
\eta_t &= \eta_{t-1} - \mu\Bigg(-2 e_t e_{t-1}\kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big)\Bigg) \\
\eta_t &= \eta_{t-1} + 2\mu e_t e_{t-1}\kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big) \\
\eta_t &= \eta_{t-1} + \beta e_t e_{t-1}\kappa_\sigma\Big(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t\Big)
\end{aligned}
$$

Thus, at iteration t, the step-size η can be calculated using the following sequential update algorithm,

$$
\eta_t = \eta_{t-1} + \beta e_t e_{t-1} \kappa_\sigma(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t) \tag{3.3}
$$

where $\beta = 2\mu$ is a free parameter for the step-size adaptation. Finally, combining Equation (3.1), Algorithm 1, and Equation (3.3), we propose the following sequential rule for sequence prediction:

$$
\begin{cases}
f_0 = 0 \\
e_t = y_t - f_{t-1}(\boldsymbol{u}_t) \\
f_t = f_{t-1} + \eta_{t-1} e_t \kappa_{\sigma_{i^*,t}}(\boldsymbol{u}_t, \cdot) \\
\eta_t = \eta_{t-1} + \beta e_t e_{t-1} \kappa_{\sigma_{i^*,t}}\left(\boldsymbol{u}_{|C_{i^*}|}, \boldsymbol{u}_t\right)
\end{cases} \tag{3.4}
$$

being $\sigma_{i^*,t}$ the appropriate kernel-size for the sample $\boldsymbol{u}_t$ and $\boldsymbol{u}_{|C_{i^*}|}$ the last sample stored in the dictionary i^* (see Algorithm 1).

3.2 Experimental Design

We validate the proposed approach for sequence prediction using MSE. The learned filter, at the final iteration, is used to compute the MSE values on each test set, as in Liu et al. (2010). The task is to predict the current value of the sample using the previous ten consecutive samples.

3.2.1 Data Sets

Testing is carried out on the following two publicly available data sets: (1) the mid-prices of two major currencies in the foreign exchange market[3]; (2) Tesla stock prices from its initial public offering[4].

Foreign exchange prices (FX). This collection has mid-prices (a reference price calculated by taking the average of the current quoted bid and ask prices) for currency pairs EUR/USD and GBP/USD with daily resolution over 12 years. In the simulations: (1) the data set is normalized for the computation convenience; (2) the training set covers January 3, 2005, to December 17, 2015, while the test set covers January 4, 2016, to May 18, 2017.

Tesla stock price (TSLA). This data set shows Tesla's stock price from its initial public offering. The data are normalized for the computation convenience, and predictions are carried out on the stock's closing price. The training set covers June 29, 2010, to June 13, 2016, while the test set covers July 5, 2016, to November 23, 2016.

3.2.2 Comparative Methods

For comparison purposes, the approach is contrasted with the following sequence prediction methods (see Appendix B):

1. Kernel least-mean-square (KLMS), which is the simplest and the starting point of many algorithms in the KAF family (Liu et al., 2010);

2. Kernel least-mean-square with variable step-size (KLMS-VSS-1), that uses a variable step-size algorithm in KLMS (Li and Hamamura, 2015);

[3]The data set is publicly available at https://www.dukascopy.com
[4]The data set is publicly available at https://www.kaggle.com/rpaguirre/tesla-stock-price

3. Kernel least-mean-square with variable step-size (KLMS-VSS-2), where the KLMS is tested with a recently proposed variable step-size strategy (Niu and Chen, 2018);

4. Nearest Instance Centroid-Estimation (NICE), which is a recently proposed method that out-performs traditional KAF-based algorithms in the prediction of chaotic time-series (Li and Príncipe, 2017);

5. Long short-term memory (LSTM), a type of recurrent neural network (RNN) used to learn sequences of observations. We use the implementation known as vanilla LSTM (Hochreiter and Schmidhuber, 1997), as it has shown stable performance when compared with other LSTM variants (Greff et al., 2017);

6. A regularized RNN proposed in Zaremba et al. (2015).

3.2.3 Parameter Settings

Tables 3.1 and 3.2 summarize the set-up of our proposal in the tested data sets. To ensure consistency in the results, both step-sizes η and η_1 remain the same for all KAF methods. Thus, as long as this condition is met, different step-size values from those shown in Tables 3.2 and 3.3 will not offer an advantage to any particular algorithm. In the simulations, the Mercer kernel is assumed to be the Gaussian kernel, i.e., $\kappa_\sigma(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t) = \exp\left(-\|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\|^2/2\sigma^2\right)$. The initial kernel-size σ_1 was adjusted using the strategy proposed in Liu et al. (2010), where the best kernel-size is the one with the lowest MSE value (see Table 3.1).

Performance is sensitive to the selection of ρ, β, and δ. However, the appropriate values for these parameters can be selected as follows: (1) ρ-kernel size adaptation, this parameter reflects a trade-off between mis-adjustment and speed of adaptation in Equation (3.2). An appropriate ρ value, based on our experimentation, is in the interval $[0, 1]$; (2) β-step size adaptation, where a value of 0.0001 has shown stable performance on all tested data sets; (3) δ-centroid threshold, which is a value between 0 and 1 that controls the number of kernel-sizes formed during training. Thus, the greater the value of δ, the more kernel-sizes will be formed.

Table 3.1: Testing MSE for proposal in considered data sets using different values of initial kernel-sizes σ_1. *FX*-foreign exchange; *TSLA*-Tesla stock price.

FX			**TSLA**	
σ_1	MSE *EUR/USD*	MSE *GBP/USD*	σ_1	MSE
0.005	0.00021	0.04530	*0.005*	0.00458
0.1	0.00022	0.00046	*0.1*	0.00409
0.25	0.00028	0.00047	*0.25*	0.00446
0.35	0.00036	0.00032	*0.35*	0.00073
0.45	0.00042	0.00041	*0.45*	0.00041
0.5	0.00045	0.00055	*0.5*	0.00048

Table 3.2: Parameter setting of proposed approach in considered data sets. *FX*-foreign exchange; *TSLA*-Tesla stock price; M-input vector size; η_1-initial step size; σ_1-initial kernel size; ρ-kernel size adaptation; β-step size adaptation; δ-centroid threshold.

Data Set		**Parameter**					
		M	η_1	σ_1	ρ	β	δ
FX	*EUR/USD*	10	0.05	0.005	0.25	0.0001	0.95
	GBP/USD	10	0.05	0.35	0.35	0.0001	0.95
TSLA		10	0.05	0.45	0.05	0.0001	0.05

The parameter settings of comparative methods is summarized in Table 3.3. The parameters η and η_1 were adjusted following the analysis described above. As previously, the kernel-sizes σ were selected using the strategy proposed in Liu et al. (2010) (see Table 3.4). The centroid distance λ of NICE is set at 2σ (Li and Príncipe, 2017). The number of neurons $\mathcal{N}$ and epochs $\mathcal{E}$ have been adjusted heuristically using a single hidden layer in LSTM (see Table 3.5). Note, for better performance, a common neural network (NN) approach is to add more layers to learn high-level features. However, when depth increases, errors between layers will be accumulated and gradients will vanish, meaning that the network degrades and becomes more difficult to train (Wang et al., 2018a). Additionally, there is no rule of thumb to select the number of hidden layers in LSTM networks (Bao et al., 2017; Palangi et al., 2016a,b). Thus, here we train the LSTM method using a single hidden layer with the following features: (1) the sigmoid activation

function is used for the LSTM blocks; (2) the Adam algorithm is employed for optimization (Kingma and Ba, 2014), as suggested in Tian et al. (2018); (3) the MSE is used as a loss function. Lastly, we implement the medium RNN proposed in Zaremba et al. (2015), which has 650 neurons per layer. The NN methods were implemented using TensorFlow (version 1.4.0) [5] and Keras (version 2.1.2) [6].

Table 3.3: Parameter setting of comparative methods in considered data sets. *FX*-foreign exchange. *TSLA*-Tesla stock price. M-input vector size, η-step size, η_1-initial step size, σ-kernel size, λ-centroid distance, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{E}$-epochs.

Data Set		**Method**	**Parameter**							
			M	η	η_1	σ	λ	$\mathcal{L}$	$\mathcal{N}$	$\mathcal{E}$
FX	EUR/USD	KLMS	10	0.05	-	0.1	-	-	-	-
		KLMS-VSS-1	10	-	0.05	0.1	-	-	-	-
		KLMS-VSS-2	10	-	0.05	0.1	-	-	-	-
		NICE	10	0.05	-	0.1	2σ	-	-	-
		LSTM	10	-	-	-	-	1	8	900
		RNN	10	-	-	-	-	2	650	200
	GBP/USD	KLMS	10	0.05	-	0.45	-	-	-	-
		KLMS-VSS-1	10	-	0.05	0.45	-	-	-	-
		KLMS-VSS-2	10	-	0.05	0.45	-	-	-	-
		NICE	10	0.05	-	0.45	2σ	-	-	-
		LSTM	10	-	-	-	-	1	8	4000
		RNN	10	-	-	-	-	2	650	4000
TSLA		KLMS	10	0.05	-	0.5	-	-	-	-
		KLMS-VSS-1	10	-	0.05	0.5	-	-	-	-
		KLMS-VSS-2	10	-	0.05	0.5	-	-	-	-
		NICE	10	0.05	-	0.5	2σ	-	-	-
		LSTM	10	-	-	-	-	1	8	400
		RNN	10	-	-	-	-	2	650	400

[5] https://www.tensorflow.org/
[6] https://keras.io/

Table 3.4: Testing MSE for KAF methods in considered data sets using different kernel-sizes σ. *FX*-foreign exchange. *TSLA*-Tesla stock price.

Data Set	σ	**Method**			
		KLMS		NICE	
		EUR/USD	*GBP/USD*	*EUR/USD*	*GBP/USD*
FX	*0.005*	0.01340	0.02621	0.01340	0.02621
	0.1	0.00025	0.00433	0.00034	0.00438
	0.25	0.00045	0.00192	0.00045	0.00192
	0.35	0.00045	0.00070	0.00045	0.00069
	0.45	0.00046	0.00035	0.00045	0.00035
	0.5	0.00048	0.00062	0.00048	0.00062
TSLA	*0.005*	0.49561		0.49561	
	0.1	0.00937		0.02213	
	0.25	0.00170		0.00181	
	0.35	0.00082		0.00131	
	0.45	0.00045		0.00043	
	0.5	0.00045		0.00042	

Table 3.5: Testing MSE for NN methods in considered data sets using different epochs $\mathcal{E}$. *FX*-foreign exchange prices. *TSLA*-Tesla stock price. $\mathcal{N}$-neurons.

Data Set		$\mathcal{E}$	LSTM $\mathcal{N}$				RNN
			1	2	4	8	
FX	*E*UR/USD	*100*	0.00671	0.00373	0.00441	0.00643	0.00029
		200	0.01770	0.01892	0.01969	0.01268	0.00028
		400	0.00130	0.00232	0.00893	0.00317	0.00116
		800	0.00635	0.00218	0.00768	0.00074	0.00475
		900	0.01455	0.00109	0.00146	0.00031	0.00508
		1000	0.01162	0.00057	0.00059	0.00137	0.00314
	*G*BP/USD	*1000*	0.00238	0.00624	0.00091	0.00071	0.00265
		2000	0.00105	0.00299	0.00395	0.00205	0.00380
		3000	0.00612	0.00649	0.00192	0.00438	0.00482
		4000	0.00373	0.00089	0.00304	0.00058	0.00069
		5000	0.00235	0.00084	0.00142	0.00088	0.03535
TSLA		*200*	0.00119	0.03434	0.00138	0.00062	0.04477
		400	0.00088	0.00063	0.00072	0.00046	0.00344
		600	0.00111	0.00104	0.00077	0.00058	0.00477
		800	0.00104	0.00090	0.00087	0.00047	0.02308
		1000	0.00074	0.00080	0.00055	0.00048	0.02391

3.3 Simulation Results and Analysis

We present quantitative and visual results of the approach for online prediction along with the comparative methods. The results give performance of the corresponding model on each test set (see Section 3.2.1). Table 3.6 summarizes the results found in Section 3.2.3. We see that our proposal out performs the other algorithms on all considered data sets, converging to smaller values of MSE, which suggests that the proposed adaptive step-size helps to improve convergence time. Note that NN models must be retrained regularly in sequence prediction tasks, which requires both significant computing and storage resources (Cui et al., 2016). That is, both algorithms (LSTM and RNN) need to pass the entire training set both forward and backward through the NN, which is also known as an epoch. This process is usually repeated several times during the learning stage. Thus, in online prediction tasks, the training set will be updated with every new sample that arrives in the system, meaning that several epochs have to be re-performed to find the best possible performance each time the training set is updated.

Table 3.6: Testing MSE in considered data sets. *MSE*-mean squared error. *Samples*-average number of samples used to predict test set. *FX*-foreign exchange prices. *TSLA*-Tesla stock price. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Data set	**Method**	**Testing MSE**		**Samples**	
		EUR/USD	*GBP/USD*	*EUR/USD*	*GBP/USD*
	KLMS	0.00025	0.00035	4000 ± 0	4000 ± 0
	KLMS-VSS-1	**0.00056**	0.00036	4000 ± 0	4000 ± 0
	KLMS-VSS-2	**0.00355**	0.00050	4000 ± 0	4000 ± 0
FX	NICE	0.00034	0.00035	607.9 ± 6.5	4000 ± 0
	LSTM	0.00031	**0.00058**	4000 ± 0	4000 ± 0
	RNN	0.00028	0.00069	4000 ± 0	4000 ± 0
	Proposal	0.00021	0.00032	347.2 ± 25.2	440.3 ± 5.4
	KLMS	0.00045		1500 ± 0	
	KLMS-VSS-1	**0.00223**		1500 ± 0	
	KLMS-VSS-2	**0.00652**		1500 ± 0	
TSLA	NICE	0.00042		1363.3 ± 216.7	
	LSTM	0.00046		1500 ± 0	
	RNN	**0.00343**		1500 ± 0	
	Proposal	0.00041		1369 ± 214	

The learning scheme of KAF-based methods, unlike LSTM and RNN, allows to make predictions while the model is updated sequentially at the same time, which is useful in online applications. Thus, although the concept of epoch is not used in KAF methods (Liu et al., 2010), this can be seen as an attempt to learn the network topology in a single epoch, i.e., the model is learned using a single pass through the entire training set in KAF algorithms.

The column "Samples" in Table 3.6 gives the average number of samples used to predict the test sets. These samples come from the learned filter at the final iteration. It is clear that KLMS, KLMS-VSS-1, KLMS-VSS-2, LSTM, and RNN use all the samples from the training sets to derive predictions on the test sets. These algorithms do not have a sparsification technique (selection of an important subset of data to train the model), which is a major drawback in online applications (Liu et al., 2010). For example, from the point of view of KLMS, the current dynamic of the system does not matter. Consequently, each sample in the test set of the FX data set is predicted using all 4000 samples that were learned during training. This results in high computational complexity in real-world applications, i.e., the KLMS dictionary grows linearly with each new sample. In contrast, our proposal uses a moderate number of samples to obtain each prediction in the test set, suggesting that the proposed multiple kernel-sizes strategy retains the most relevant samples to perform later prediction tasks effectively. Additionally, the knowledge-transfer strategy of our approach, compared to that used by the NICE algorithm, reduces the required number of samples without significant loss of accuracy (see Table 3.6).

Figures 3.3 and 3.4 show the predicted stock prices in the test sets. Figure 3.3 shows the predictions for Tesla stock prices, while Figures 3.4(a) and 3.4(b) displays the results for FX. The multiple kernel-sizes strategy and the adaptive step-size, incorporated in our approach, enable the kernel-based adaptive filter to converge more quickly while competitive performance is maintained on all tested data sets. However, if the initial kernel-size σ_1 is inappropriately chosen, the prediction accuracy may be adversely affected (see Table 3.1). In this case, the suitable initial value of σ_1 can be selected using a method such as Silverman's rule (Kang and Noh, 2019). The predictions of our proposal, as seen in Figure 3.4, are very close to the desired signals. This is a visual interpretation of the lowest MSE values obtained by the proposed approach in Table 3.6, proving its stable performance in real-world applications.

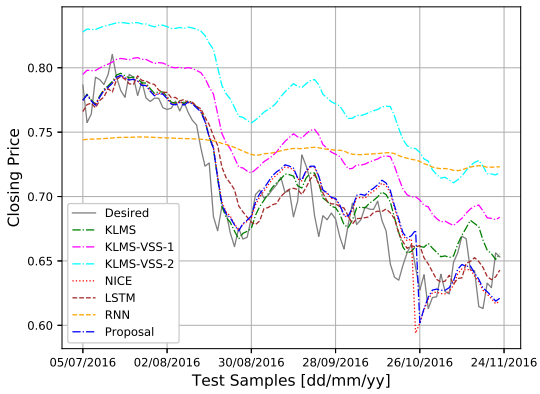
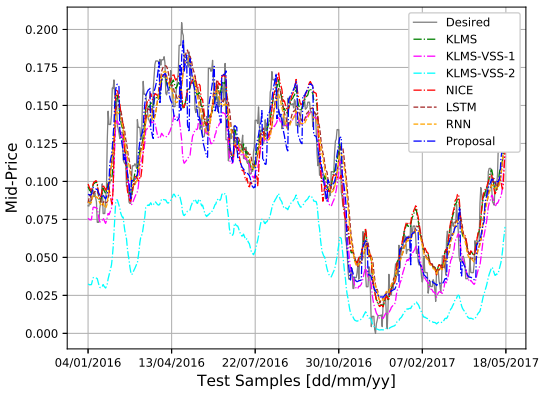
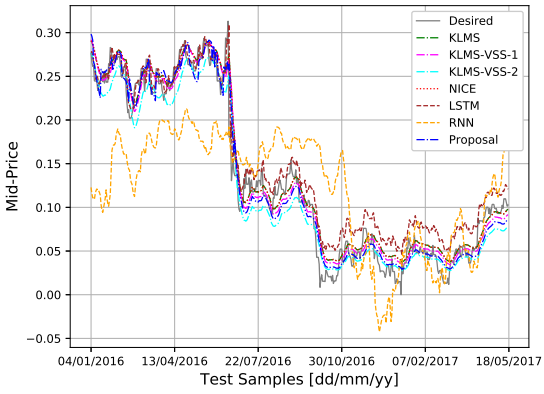


Figure 3.3: Performed predictions in Tesla stock prices.



(a) EUR/USD



(b) GBP/USD

Figure 3.4: Performed predictions in foreign exchange rates.

3.4 Chapter Summary

This chapter introduced a sequence prediction approach for financial time-series that addresses two main challenges of KAF-based methods: selection of appropriate kernel-size and step-size. The proposed multiple kernel-sizes and the adaptive step-size strategies are combined, improving convergence time while competitive performance is maintained. However, we must clarify that our approach may be adversely affected in the case of relatively few training samples, as it is more difficult to identify hidden patterns from data under this scenario. The minimum number of samples, based on our experiments, required to train highly non-stationary signals is 1500. Results demonstrate that our proposal learns from a continuous sequence of data records, adapts to changing statistics in the data, exhibits high tolerance to noisy conditions and provides stable performance in real-world applications.

The following chapter, with the aim to address the second objective of this thesis, will introduce an entropy-based cost function for KAF using higher order statistics of financial time-series. This will allow us to capture complex patterns behind the data and use that information within the context of sequence prediction. The cost function uses instantaneous entropy as the adaptation criterion, which is useful in online applications.

Chapter 4

An Entropy-based Cost Function for Sequence Prediction

Having looked at sequence prediction using a kernel-based approach, we now move to address the second objective of this thesis. The prediction of financial time-series such as stock returns (changes in price on an asset or investment over time) requires consideration of their stochastic nature and non-stationary conditions. Thus, higher order statistics and information content of financial time-series should be captured in addition to their energy. In this chapter, we propose a three-stage approach for stock returns prediction [1]: (1) an entropy-based adaptation criterion is developed; (2) the bandwidth for density estimation is optimized using the quadratic information potential criterion; (3) the architecture and functionality of neural networks (NNs) are extended to include kernel adaptive filtering (KAF). The main rationale behind the approach is to minimize the error entropy between the model output and the desired response using KAF. The approach has been tested on 10 different stocks over a 12 year interval; the results give relatively low values of mean square error (MSE) when compared with autoregressive-based methods, data-driven approaches based on NNs, and traditional kernel-based adaptive filters. In particular, unlike NNs, our proposal sequentially updates the model, in real-time, while predictions are being made, which is critical in streaming data applications.

This chapter is structured as follows: Section 4.1 introduces the proposed approach; Section 4.2 describes the experimental design of this work; Section 4.3 presents simulation results; and Section 4.4 summarizes the chapter.

[1] The outcomes of this chapter are under review.

4.1 An Entropy-based Prediction approach

We develop a three-stage approach for stock returns prediction: (1) adaptation criterion (Section 4.1.1); (2) bandwidth for density estimation (Section 4.1.2); (3) neural network architecture using kernel machines (Section 4.1.3).

4.1.1 Adaptation Criterion based on Entropy

Given a set of input-output samples $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, being $\boldsymbol{u}_t$ an M-dimensional input vector that belongs to the input set $\mathcal{U} \subset \mathbb{R}^M$, while $y_t \in \mathbb{R}$ is the output over time domain $t \in N$. The goal is to learn the underlying function $y = f(\boldsymbol{u})$ from the given input-output samples $\mathcal{T}$. In KAFs, the underlying function f will be a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ (see Appendix B.3.1). In addition, according to Mercer's theorem (Schölkopf and Smola, 2001), a Mercer kernel $\kappa_\sigma : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ induces a mapping $\varphi : \mathcal{U} \to \mathcal{F}$. This allows the inner products to be calculated in the feature space, using the kernel trick $\varphi(\boldsymbol{u}_t)^\top \varphi(\boldsymbol{u}^*) = \kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}^*)$, being $\boldsymbol{u}^*$ a new input vector. Thus, the function $f \in \mathcal{H}$ and a high-dimensional weight vector $\boldsymbol{\Omega} \in \mathcal{F}$ can be computed in the hypothesis and feature spaces as follows (Liu et al., 2008):

$$\min_{f \in \mathcal{H}} \sum_{t=1}^{N} \left(y_t - f(\boldsymbol{u}_t)\right)^2 + \lambda ||f||^2_{\mathcal{H}} \tag{4.1a}$$

$$\min_{\boldsymbol{\Omega} \in \mathcal{F}} \sum_{t=1}^{N} \left(y_t - \boldsymbol{\Omega}^\top \varphi(\boldsymbol{u}_t)\right)^2 + \lambda ||\boldsymbol{\Omega}||^2_{\mathcal{F}} \tag{4.1b}$$

where $\lambda \geq 0$ is the regularization factor that controls the solution smoothness and $||\cdot||_{\mathcal{H}}$ denotes the norm in $\mathcal{H}$. The solution of the above equations in a batch approach is computationally expensive, as the dimension of the Gram matrix equals the number of input patterns. In this sense, KAFs provide an efficient alternative that finds the solution in an online sequential way (Chen et al., 2016a).

We find the high-dimensional weight vector $\boldsymbol{\Omega}$ based on a KAF strategy, using stochastic gradient descent as the optimization criterion and instantaneous entropy as the adaptation cost,

$$\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} - \eta \frac{\partial}{\partial \boldsymbol{\Omega}_{t-1}} \left(\frac{\psi\left(\hat{\kappa}_{\hat{\sigma}_t}(e_{t,t} - e_{t,j})\right)}{\hat{\kappa}_{\hat{\sigma}_t}(e_{t,t} - e_{t,j})} \right) \tag{4.2}$$

where $\eta \in \mathbb{R}^+$ is the step-size parameter, ψ denotes the instantaneous entropy (Principe, 2010), $e_{t,j} = y_j - \boldsymbol{\Omega}_{t-1}^\top \varphi(\boldsymbol{u}_j)$ is the prediction error with hypothesis $\boldsymbol{\Omega}_{t-1}$ and input-output sample $(\boldsymbol{u}_j, y_j)$, while $\hat{\kappa}_{\hat{\sigma}_t}$ is the kernel function for density estimation and $\hat{\sigma}_t > 0$ its bandwidth at iteration t (Chen et al., 2018a). The main rationale behind the suggested strategy in Equation (4.2) is to minimize the error entropy between the model output and the desired response, meaning that higher order statistics and information content are captured rather than only the signal energy (Erdogmus and Principe, 2002b; Chen et al., 2007). Note that the definition of entropy can be very flexible (see Appendix C); for example, Shannon and information potential are special cases of ψ, which is a continuous concave real function over $[0, \infty]$ (Principe, 2010). Here the entropy is assumed to be the quadratic information potential criterion (see Equation (4.3)) due to its low computational complexity in practical applications (Chen et al., 2013a).

$$\psi\Big(\hat{\kappa}_{\hat{\sigma}_t}(e_{t,t} - e_{t,j})\Big) = -\left(\hat{\kappa}_{\hat{\sigma}_t}\Big(e_{t,t} - e_{t,j}\Big)\right)^2 \tag{4.3}$$

The error distribution, which is usually unknown in practice, can be approximated by using kernel density estimation on the L most recent errors as follows:

$$\hat{\kappa}_{\hat{\sigma}_t}\Big(e_{t,t} - e_{t,j}\Big) = \frac{1}{L\hat{\sigma}_t\sqrt{2\pi}} \sum_{j=t-L+1}^{t} \exp\Big(- \Big(e_{t,t} - e_{t,j}\Big)^2 / 2\hat{\sigma}_t^2\Big) \tag{4.4}$$

Thus, combining Equations (4.3) and (4.4), the following gradient update rule is obtained,

$$\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \frac{\eta}{L\hat{\sigma}_t\sqrt{2\pi}} \frac{\partial}{\partial \boldsymbol{\Omega}_{t-1}} \left[\sum_{j=t-L+1}^{t} \exp\Big(- (e_{t,t} - e_{t,j})^2 / 2\hat{\sigma}_t^2\Big)\right] \Big(- \frac{\partial e_{t,t}}{\partial \boldsymbol{\Omega}_{t-1}} + \frac{\partial e_{t,j}}{\partial \boldsymbol{\Omega}_{t-1}}\Big)$$

which can be unfolded as below,

$$\begin{aligned}\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \frac{\eta}{L\hat{\sigma}_t^3\sqrt{2\pi}} \Bigg[\sum_{j=t-L+1}^{t} \exp\Big(- (e_{t,t} - e_{t,j})^2 / 2\hat{\sigma}_t^2\Big) \\ \Big(e_{t,t} - e_{t,j}\Big)\Big(\varphi(\boldsymbol{u}_t) - \varphi(\boldsymbol{u}_j)\Big)\Bigg] \\ \Big(\varphi(\boldsymbol{u}_t) - \varphi(\boldsymbol{u}_j)\Big)\end{aligned}$$

The previous expression can be written as follows,

$$\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \eta \boldsymbol{\Phi}_t \boldsymbol{\Psi}(\boldsymbol{e}_t) \tag{4.5a}$$

$$f_t = f_{t-1} + \eta \boldsymbol{K}_t \boldsymbol{\Psi}(\boldsymbol{e}_t) \tag{4.5b}$$

where:

- $\boldsymbol{\Phi}_t = [\varphi(\boldsymbol{u}_{t-L+1}), \varphi(\boldsymbol{u}_{t-L+2}), \ldots, \varphi(\boldsymbol{u}_t)]$;
- $\boldsymbol{K}_t = [\kappa_\sigma(\boldsymbol{u}_{t-L+1}, \cdot), \kappa_\sigma(\boldsymbol{u}_{t-L+2}, \cdot), \ldots, \kappa_\sigma(\boldsymbol{u}_t, \cdot)]$;
- $\boldsymbol{\Psi}(\boldsymbol{e}_t) = \beta \sum_{j=t-L+1}^{t} \exp(-(e_{t,t} - e_{t,j})^2 / 2\hat{\sigma}_t^2)(e_{t,t} - e_{t,j})(\varphi(\boldsymbol{u}_t) - \varphi(\boldsymbol{u}_j))$,

being $\beta = 1/(L\hat{\sigma}_t^3\sqrt{2\pi})$. Lastly, the coefficients $\boldsymbol{\alpha}$ will be updated as:

$$\boldsymbol{\alpha}_{t,j} = \begin{cases} \eta \boldsymbol{\Psi}(\boldsymbol{e}_t), & j = t \\ \boldsymbol{\alpha}_{t-1,j} - \eta \boldsymbol{\Psi}(\boldsymbol{e}_t), & t - L < j < t \\ \boldsymbol{\alpha}_{t-1,j}, & 1 \leq j \leq t - L \end{cases} \tag{4.6}$$

4.1.2 Entropy-based Bandwidth for Density Estimation

We propose to optimize the bandwidth for density estimation $\hat{\sigma}_t$ using the quadratic information potential criterion expressed over time as follows:

$$\hat{\sigma}_t = \hat{\sigma}_{t-1} - \hat{\eta} \frac{\partial}{\partial \hat{\sigma}_{t-1}} \left(\frac{\psi\left(\hat{\kappa}_{\hat{\sigma}_{t-1}}(e_{t,t} - e_{t,j})\right)}{\hat{\kappa}_{\hat{\sigma}_{t-1}}(e_{t,t} - e_{t,j})} \right) \tag{4.7}$$

where $\hat{\eta} \in \mathbb{R}^+$ is the step-size parameter for density estimation. Therefore, considering Equations (4.3) and (4.4), the bandwidth can be computed through the gradient descent method as,

$$\hat{\sigma}_t = \hat{\sigma}_{t-1} + \hat{\eta} \frac{\partial}{\partial \hat{\sigma}_{t-1}} \left(\frac{1}{L\hat{\sigma}_{t-1}\sqrt{2\pi}} \sum_{j=t-L+1}^{t} \exp\Big(-(e_{t,t} - e_{t,j})^2 / 2\hat{\sigma}_{t-1} \Big) \right) \tag{4.8}$$

which results in the following update rule:

$$\hat{\sigma}_t = \hat{\sigma}_{t-1} + \hat{\beta} \sum_{j=t-L+1}^{t} \exp\Big(-(e_{t,t} - e_{t,j})^2 / 2\hat{\sigma}_{t-1} \Big)(e_{t,t} - e_{t,j})^2 \tag{4.9}$$

where $\hat{\beta} = \hat{\eta}/(L\hat{\sigma}_{t-1}^4\sqrt{2\pi})$.

4.1.3 Neural Network Architecture using Kernel Machines

With the aim of improving stock returns prediction, we extend the architecture of NNs to include KAF. This is mathematically viable as in a NN, any neuron can be directly replaced by a kernel machine without altering the architecture and functionality of the network (Vapnik, 2013; Suykens and Vandewalle, 1999; Cho and Saul, 2009; Duan et al., 2019). We propose to obtain a new representation of every input vector $\boldsymbol{u}_t$ using kernel machines instead of neurons under a NN architecture, which may be useful to capture complex patterns behind the data (Suykens, 2017). Here, as seen in Figure 4.1, we use a single hidden layer with m kernel machines.

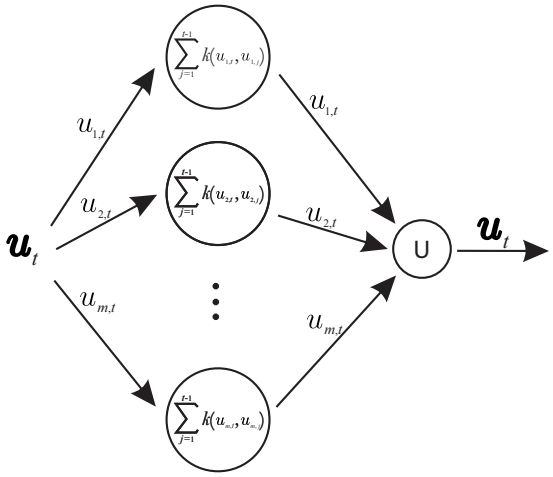


Figure 4.1: Proposed neural network architecture using kernel machines.

Thus, when a new input vector $\boldsymbol{u}_t$ arrives, its elements $u_{1,t}, u_{2,t}, \ldots, u_{1,m}$ are represented as a linear combination of kernel machines. The new representation of $\boldsymbol{u}_t$ is simply the union of all kernel machine outputs, where the mappings are obtained by using the Mercer kernel κ_σ. Finally, the proposed entropy-based prediction approach for stock returns is shown in Algorithm 2.

Algorithm 2: Proposed stock returns prediction approach.

input : $\mathcal{T}$– training data

Parameter setting:

- κ_σ- kernel function;
- σ- kernel parameter;
- η- step size;
- $\hat{\sigma}_1$- initial bandwidth density;
- $\hat{\eta}$- step size density;
- L- error samples length
- $\boldsymbol{\alpha}_1 = [\eta y_1]$- initialize coefficient vector

Computation:

while $\{\boldsymbol{u}_t, y_t\}$ *available* **do**

1. Get a new representation of $\boldsymbol{u}_t$ (see Figure 4.1)
2. Allocate a new unit
   $\boldsymbol{\alpha}_t = \left[\boldsymbol{\alpha}_{t-1}^\top, 0\right]^\top$
3. Compute the errors
   For $\max\{1, t-L+1\} \leq j \leq t$, **do**
   $e_{t,j} = y_t - \sum_{l=1}^{t-1} \boldsymbol{\alpha}_{t-1,l} \kappa_\sigma\left(\boldsymbol{u}_l, \boldsymbol{u}_j\right)$
4. Update the coefficient vector $\boldsymbol{\alpha}_t$ (see Equation (4.6))
5. Update the bandwidth $\hat{\sigma}_t$ (see Equation (4.9))

output: $\boldsymbol{\alpha}$

4.2 Experimental Design

The proposed approach is validated based on prediction of stock returns using MSE (Hacine-Gharbi and Ravier, 2018) and MAE (Wang and Lu, 2018). The learned filter, as suggested in Chen et al. (2016a), is used to compute the MSE and MAE values on each test set. The task is to predict the current day price change using the last ten stock returns.

4.2.1 Data Sets

We collected daily closing prices data from Yahoo Finance to calculate the log returns (see Appendix A.3). Then, as suggested in Siikanen et al. (2018), these returns are calculated by using the adjusted closing prices of each day. The testing has been carried out on 10 stocks from different economies over 12 years (see Table 4.1). The training set covers the period *17 January 2006* to *30 November 2016*, while the test set ranges from *3 January 2017* to *28 February 2018*. The data set, as suggested in Chen (2019), is standardized so all stocks have a mean estimation of zero and a standard deviation of one.

Table 4.1: Stocks in the experimental design.

Ticker	Stock
ADS	Adidas AG
DPW	Deutsche Post AG
HEI	Heidelberg Cement AG
MRK	Merck KGaA
ADM	Admiral Group PLC
CCL	Carnival PLC
IAG	International Consolidated Airlines Group
SKY	SKY PLC
VOD	Vodafone Group PLC
C	Citigroup Inc

4.2.2 Comparative Methods

The following online prediction methods are used for comparison purposes[2]:

1. Kernel Affine Projection (KAPA), a stochastic gradient methodology to solve the least squares problem in RKHS (Liu and Príncipe, 2008);

2. Kernel Least Mean Square (KLMS), which is the simplest and the starting point of many kernel adaptive filtering algorithms (Liu et al., 2008);

3. Kernel Minimum Error Entropy (KMEE), a generalized stochastic information gradient algorithm in RKHS (Chen et al., 2013a);

4. Long Short-Term Memory (LSTM), representing the state-of-the-art recurrent neural network model for sequence learning tasks (Araya et al., 2019);

[2] https://github.com/segarciave/PRL-2020

5. Vector Error Correction Model (VECM), which is considered the standard tool to handle multivariate time-series under non-stationary conditions (Liang and Schienle, 2019).

4.2.3 Parameter Settings

The parameters of compared methods, as seen in Table 4.2, were heuristically adjusted to achieve the best MSE on tested stocks. In particular, to ensure consistency in the results, L, η, and σ are the same for all KAF-based methods. The implementation known as vanilla LSTM, as in the previous chapter, is used due to its competitive performance on prediction tasks (Greff et al., 2017). The LSTM, as suggested in Cui et al. (2016); Tian et al. (2018); Kingma and Ba (2014), is trained using a single hidden layer with 20 neurons, the sigmoid activation function, and Adam optimization algorithm. The free parameters of VECM were selected using heuristic approaches and following strategies as proposed in the literature (Kuo, 2016).

Table 4.2: Parameter setting of compared methods. m-input vector size, L-error samples length, η-step size, $\hat{\eta}$-step size density, σ-kernel parameter, $\hat{\sigma}_1$-initial bandwidth density, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{A}$-number of lagged differences in the model, $\mathcal{I}$-cointegration rank.

Parameter	**Method**					
	KAPA	*KLMS*	*KMEE*	*LSTM*	*Proposal*	*VECM*
m	10	10	10	10	10	10
L	15	-	15	-	15	-
η	0.05	0.05	0.05	-	0.05	-
$\hat{\eta}$	-	-	-	-	1.81	-
σ	0.5	0.5	0.5	-	0.5	-
$\hat{\sigma}_1$	-	-	-	-	0.1	-
$\mathcal{L}$	-	-	-	1	-	-
$\mathcal{N}$	-	-	-	20	-	-
$\mathcal{A}$	-	-	-	-	-	3
$\mathcal{I}$	-	-	-	-	-	0

The step size for density estimation $\hat{\eta}$ was adjusted using the strategy proposed in Liu et al. (2010), where the value with the lowest MSE is chosen (see Table 4.3). In addition, based on our experimentation, an initial bandwidth for density estimation $\hat{\sigma}_1 = 0.1$ has shown stable performance on all tested stocks.

Table 4.3: Testing MSE using different values of step-size for density estimation.

$\hat{\eta}$	*0.8*	*1.0*	*1.5*	*1.81*	*2.0*
MSE	0.00215	0.00212	0.00186	0.00182	0.00183

4.3 Simulation Results and Analysis

The simulation results for MSE and MAE are shown in Tables 4.4 and 4.5, respectively. The last row on each table displays the average performance per algorithm, while the bold notation annotate by an asterisk indicates the best overall method. In the simulations, the Mercer kernel is assumed to be the Gaussian kernel due to its universal approximating capability, desirable smoothness and numeric stability (Chen et al., 2012). The MSE measures the average of the squares of the errors, where values closer to zero indicates better prediction performance.

The LSTM and our proposal, as seen in Table 4.4, outperform other algorithms on all considered stocks. However, in practice, these two methods use different learning strategies, that is, the former requires the entire training set in advance to start learning the model, while the later sequentially updates the model and performs predictions simultaneously (see Section 4.1). The proposed method, has the ability to represent non-linear functions linearly and universally in RKHS. This results in a simple convex optimization problem with a unique global optimum, which prevents the algorithm to get stuck in local minima during the training stage. In addition, the process occurring during the training of LSTM is difficult to interpret, meaning that is not clear how learning from input data is done. This is also known as the black box problem in NNs (Azodi et al., 2020). The MAE values of each method in the considered stocks are summarized in Table 4.5. This metric is a negatively-oriented score that measures the average of absolute errors. We see that our proposal, as on the previous metric, outperforms the KAF algorithms used for comparison. In particular, the first two compared methods (KAPA and KLMS) use MSE as the adaptation criterion, which may be a poor descriptor of optimality for non-linear and non-Gaussian conditions (Liu et al., 2010). In contrast, the proposed method uses information theoretic criteria to obtain a sample-based methodology that learns arbitrary non-linear systems. This allows the algorithm to capture higher order statistics of stock returns rather than simply their energy, i.e., the entropy-based cost function used by our approach helps to capture complex patterns behind the data, resulting in better accuracy.

Table 4.4: Testing MSE at final iteration on stock returns prediction. SD– Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Stock	**Method**					
	KAPA	*KLMS*	*KMEE*	*LSTM*	*Proposal*	*VECM*
ADS	**0.00027**	**0.00026**	**0.00027**	0.00027	0.00026	0.00036
DPW	**0.01441**	**0.0135**	**0.028**	0.01339	0.01342	0.01716
HEI	**0.00022**	**0.00018**	**0.00019**	**0.00018**	0.00018	**0.00024**
MRK	**0.00014**	0.00012	**0.00015**	0.00012	0.00012	**0.00016**
ADM	**0.00018**	**0.00015**	**0.00015**	**0.00015**	0.00015	**0.00021**
CCL	**0.00017**	**0.00012**	**0.00012**	**0.00012**	0.00012	**0.00019**
IAG	**0.0009**	**0.00088**	**0.00267**	**0.00087**	0.00088	**0.00108**
SKY	**0.00314**	**0.00283**	**0.00292**	**0.00281**	0.00281	**0.00368**
VOD	**0.00021**	**0.00014**	**0.00012**	**0.00012**	0.00012	**0.00015**
C	**0.0002**	**0.00015**	**0.00525**	**0.00014**	0.00014	**0.00024**
Mean	0.00198	0.00183	0.00398	0.00182	0.00182	0.00235
SD	0.00423	0.00397	0.00818	0.00394	0.00395	0.00504

Table 4.5: Testing MAE at final iteration in stock returns prediction. SD– Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Stock	**Method**					
	KAPA	*KLMS*	*KMEE*	*LSTM*	*Proposal*	*VECM*
ADS	**0.01126**	**0.01091**	**0.01151**	0.01116	0.01095	0.01356
DPW	**0.06575**	**0.06083**	**0.14325**	0.06071	0.06058	0.07111
HEI	**0.01139**	**0.00974**	**0.01024**	**0.00969**	0.00954	**0.01147**
MRK	**0.0078**	0.00709	**0.00881**	0.00716	0.00729	**0.00873**
ADM	**0.00968**	**0.00831**	**0.00828**	**0.00841**	0.00845	**0.0105**
CCL	**0.00985**	**0.00817**	**0.00798**	**0.0081**	0.00819	**0.01057**
IAG	**0.02178**	**0.02128**	**0.04483**	**0.02105**	0.02133	**0.02462**
SKY	**0.03539**	**0.03229**	**0.03375**	**0.0321**	0.03236	**0.03979**
VOD	**0.01133**	**0.00828**	**0.00711**	**0.00711**	0.00716	**0.00878**
C	**0.01094**	**0.00941**	**0.07152**	**0.00865**	0.00854	**0.01228**
Mean	0.0195	0.0176	0.0347	0.0174	0.0174	0.0211
SD	0.0173	0.0163	0.0415	0.0163	0.0163	0.0190

Finally, Figures 4.2 and 4.3 show the true and predicted stock returns for two representative stocks. The proposed approach, unlike the KAF algorithms used for comparison, does not have second-order statistics as the adaptation criterion. This avoids the Gaussian assumption limitation of similar methods and enhances prediction performance in realistic scenarios.

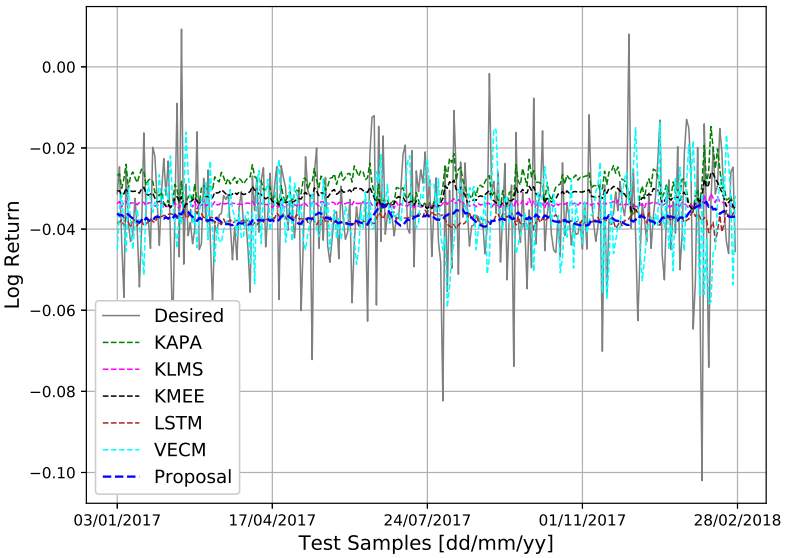


Figure 4.2: Stock returns prediction in the test set of HEI.

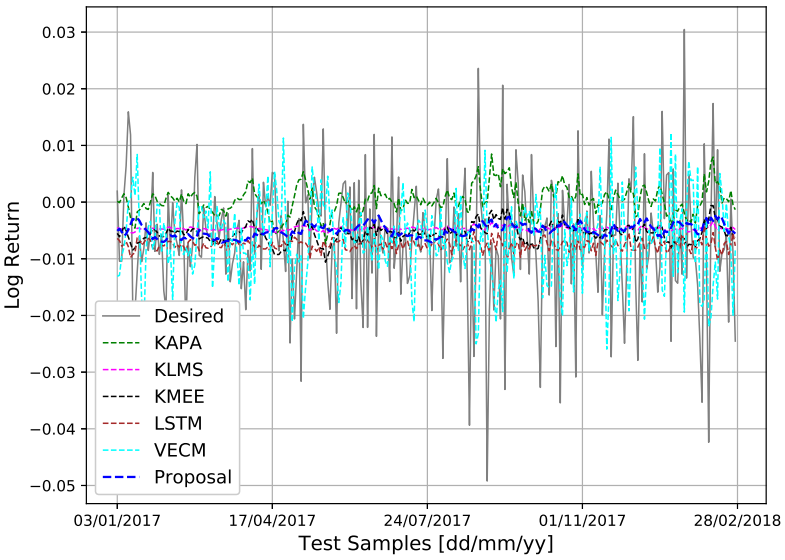


Figure 4.3: Stock returns prediction in the test set of CCL.

4.4 Chapter Summary

This chapter introduced an approach for predicting stock returns that minimizes the error entropy using KAF. The entropy-based adaptation criterion within a stochastic gradient descent optimization approach allows the capture of higher order statistics of financial time-series. This avoids the limitation of Gaussian assumption, improving performance in streaming data applications related to financial markets. Additionally, our proposal has $\mathcal{O}(t + L^2)$ computational complexity, which is still competitive when compared to conventional algorithms. However, we must clarify that when the parameter L (which indicates the window length for the most recent errors) is higher than 15, convergence time is adversely affected. The approach has been tested on 10 stocks over a 12 year interval. Simulations demonstrate that, when compared with KAF methods, the entropy-based cost function enhances prediction accuracy[3].

In the next chapter we address the last objective of this thesis by proposing a sequence prediction approach within a distributed learning paradigm. This incorporates the inter-dependences between financial markets into an analytical model, aiming to improve prediction accuracy and profitability.

[3]The Python implementation of the previous methods can be downloaded at https://github.com/segarciave/PRL-2020.

Chapter 5

A Kernel-based Stock Market Interdependence Approach

The last two chapters addressed two weaknesses of machine learning methods in the context of financial markets, i.e., sequence learning and higher order statistics. In this chapter, with the aim to address the last objective of this thesis, we propose a sequence prediction approach within a distributed learning paradigm.

The prediction of financial time-series, unlike traditional regression, requires consideration of both the sequential and interdependent nature of financial markets. Thus, we introduce a two-phase approach for stock returns prediction using sequential learning within a stock market interdependence approach [1]. The underlying models of each stock are learned separately using a kernel-based adaptive filter that encodes different patterns of the input space. Further, stock returns are predicted using not only their local models, but also the individual local models learned from other stocks, providing a natural way to incorporate these interdependencies. The approach is a distributed learning paradigm rather than a centralized one in the sense that individual prediction models are learned based solely on a local data store, thus avoiding expensive and time-consuming data transportation into an integrated, central data store. Such a distributed learning paradigm is critical for big data analysis and real-time learning. The proposal is validated on 24 stocks from three major economies; results show higher Sharpe ratio when compared with KAFs, LSTM, and autoregressive-based models.

[1] The outcomes of the work described in this chapter have been published in *Expert Systems with Applications* under the title "*Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach*" (Garcia-Vega et al., 2020).

5.1 Sequential and Interdependent Nature of Financial Time-Series

We develop a two-stage approach for stock returns prediction: (1) sequential learning, where the underlying models of each stock are learned separately using KAF (Section 5.1.1); (2) interdependence between stocks, where local models learned from different stock markets are used to improve prediction (Section 5.1.2).

5.1.1 Sequential Learning based on Adaptive Filtering

Given a set of training data $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, where $\boldsymbol{u}_t \in \mathbb{R}^M$ is an input vector and $y_t \in \mathbb{R}$ is the desired output (see Figure 5.1). The task is to infer the underlying function $y = f(\boldsymbol{u})$ from the given data $\mathcal{T}$ and, for a new input vector $\boldsymbol{u}^* \in \mathbb{R}^M$, to predict the value of a new observation $y^* \in \mathbb{R}$.

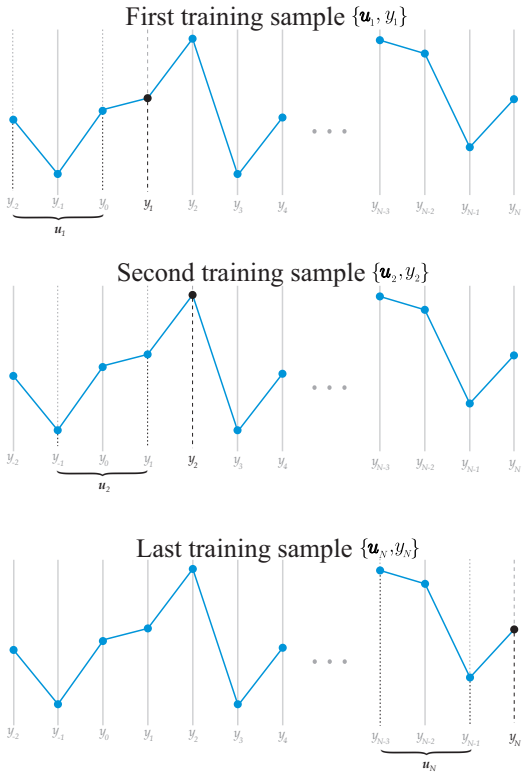


Figure 5.1: Three representative samples of the training set $\mathcal{T}$ when $M = 3$. The blue line represents stock returns of a given stock from which the training samples are selected. The upper, middle and lower graphs show the first, second, and last training samples, respectively.

In practice, KAF sequentially estimates f by using the current input-output pair $\{\boldsymbol{u}_t, y_t\}$ and updating the previous estimate f_{t-1} as follows (see Appendix B.3):

$$\begin{cases} f_0 = 0 \\ e_t = y_t - f_{t-1}(\boldsymbol{u}_t) \\ f_t = f_{t-1} + \eta e_t \kappa_\sigma(\boldsymbol{u}_t, \cdot) \end{cases} \tag{5.1}$$

being $\eta \in \mathbb{R}^+$ the step-size, f_t the learned mapping, $e_t \in \mathbb{R}$ the prediction error, while $\kappa_\sigma(\cdot,\cdot) \in \mathbb{R}^+$ is a Mercer kernel with a bandwidth $\sigma \in \mathbb{R}^+$ that controls the mapping smoothness (Schölkopf et al., 2018). Note that Equation (5.1) creates a kernel unit for every new sample, where $\boldsymbol{u}_t$ is the center and ηe_t its coefficient, posing additional issues for continuous adaptation scenarios. A challenge is to curb the network growth by either eliminating redundant information or minimizing information loss, i.e., only using input data with high information content as the new centers.

We propose to reduce the network size by partitioning the centers into distinct regions that encode different patterns of the input space. Thus, these patterns are identified using the change point detection method proposed by Yamanishi and Takeuchi (2002), as it has shown stable performance in non-stationary environments. This method, at each iteration t, determines whether a change in the distribution has occurred within the sequence $y_1, \ldots, y_t$. This is done by measuring how large the probability density function p_t has moved from p_{t-1} after learning from y_t. Particularly, it is stated that a change point has taken place at iteration t when the following inequality holds:

$$\epsilon(y_t, p_{t-1}) \geq \delta \tag{5.2}$$

where $\epsilon(y_t, p_{t-1}) = -\log p_{t-1}(y_t)$ denotes a prediction loss for y_t relative to a probability density function p_{t-1}, while $\delta \in \mathbb{R}^+$ is a predefined threshold. Thus, when a change-point is detected within the sequence $y_1, \ldots, y_t$, we form a new set of centres or dictionary. Additionally, to avoid large discontinuities in learning (Li and Príncipe, 2017), all the centres and coefficients of the closest dictionary are copied or transferred to the newly formed dictionaries as follows:

1. **Change in data distribution** $\epsilon(y_t, p_{t-1}) \geq \delta$: This indicates a change in data distribution; hence, the following three dictionaries are formed,

   - $\mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1} = \{\mathcal{C}_{i^*}, \boldsymbol{u}_t\}$, with $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_i : i \in [1, |\boldsymbol{\mathcal{C}}|]\}$;
   - $\mathcal{Y}_{|\boldsymbol{\mathcal{Y}}|+1} = \{\mathcal{Y}_{i^*}, y_{t-1}\}$, with $\boldsymbol{\mathcal{Y}} = \{\mathcal{Y}_i : i \in [1, |\boldsymbol{\mathcal{Y}}|]\}$;
   - $\mathcal{W}_{|\boldsymbol{\mathcal{W}}|+1} = \{\mathcal{W}_{i^*}, \eta e_t\}$, with $\boldsymbol{\mathcal{W}} = \{\mathcal{W}_i : i \in [1, |\boldsymbol{\mathcal{W}}|]\}$;

   where $|\boldsymbol{\mathcal{C}}| = |\boldsymbol{\mathcal{Y}}| = |\boldsymbol{\mathcal{W}}|$ denotes the number of elements in each dictionary. In addition, $\mathcal{C}_{i^*} = \{\boldsymbol{u}_j : j \in [1, L]\}$, $\mathcal{Y}_{i^*} = \{y_j : j \in [1, L]\}$, and $\mathcal{W}_{i^*} = \{\eta e_j : j \in [1, L]\}$ are the closest dictionaries to y_{t-1}. Note, $\mathcal{C}_{i^*}$, $\mathcal{Y}_{i^*}$, and $\mathcal{W}_{i^*}$ are found using y_{t-1} rather than y_t. This is because, during real-time prediction tasks, only the input vectors $\boldsymbol{u}_t$ are known and the desired output y_t is the value to be predicted. As seen in Figure 5.1, y_{t-1} is always the last element of the input vector $\boldsymbol{u}_t$. We find the closest dictionary i^* using the Kullback-Leibler divergence as:

$$i^* = \arg\min_{\forall i} p_{t-1}^{\mathcal{Y}_i} \ln\left(p_{t-1}^{\mathcal{Y}_i} / p_t^{\mathcal{Y}_i}\right) \tag{5.3}$$

   where $p_{t-1}^{\mathcal{Y}_i}$ and $p_t^{\mathcal{Y}_i}$ are the probability density functions of $\mathcal{Y}_i$ before and after learning y_{t-1}, respectively. The primary rationale behind the suggested strategy in Equation (5.3) is to quantify the information content that y_{t-1} will provide to each dictionary $\mathcal{Y}_i$. That is, when y_{t-1} does not provides high information content to a dictionary $\mathcal{Y}_i$, the Kullback-Leibler divergence will tend to zero, meaning that the two distributions $p_{t-1}^{\mathcal{Y}_i}$ and $p_t^{\mathcal{Y}_i}$ are identical. Thus, the dictionary $\mathcal{Y}_i$ that minimizes Equation (5.3) will be the closest dictionary to y_{t-1}.

2. **No change in data distribution** $\epsilon(y_t, p_{t-1}) < \delta$: This means that the data distribution has not changed and, therefore, it is unnecessary to divide the centres into a new region. In addition, to curb the growth of the radial-basis-function structure, we incorporate an online vector quantization technique as follows,

- $\min_{\forall i} p_{t-1}^{\mathcal{Y}_i} \ln\left(p_{t-1}^{\mathcal{Y}_i}/p_t^{\mathcal{Y}_i}\right) \leq \varepsilon$: The dictionary sizes remain the same and only the closest coefficient to $\boldsymbol{u}_t$ is updated using the following expression,

$$\mathcal{W}_{i^*}^{(j^*)} = \mathcal{W}_{i^*}^{(j^*)} + \eta e_t, \tag{5.4}$$

  where the closest coefficient j^* is computed as follows,

$$j^* = \arg\min_{\forall j} \left\| \boldsymbol{u}_t - \mathcal{C}_{i^*}^{(j)} \right\|_2, \tag{5.5}$$

  being $\|\cdot\|_2$ the ℓ_2 norm and $\varepsilon \in \mathbb{R}^+$ a predefined threshold. The previously imposed restraint assigns a new center $\boldsymbol{u}_t$ into the dictionary $\mathcal{C}_{i^*}$ only when y_{t-1} provides high information content to the dictionary $\mathcal{Y}_{i^*}$.

- $\min_{\forall i} p_{t-1}^{\mathcal{Y}_i} \ln\left(p_{t-1}^{\mathcal{Y}_i}/p_t^{\mathcal{Y}_i}\right) > \varepsilon$: The sample $\boldsymbol{u}_t$ is assigned as a new center to the closest dictionary $\mathcal{C}_{i^*}$, the previous desired output y_{t-1} is stored in the dictionary $\mathcal{Y}_{i^*}$, while the set of coefficients $\mathcal{W}_{i^*}$ is updated using e_t, i.e., $\mathcal{C}_{i^*} = \{\mathcal{C}_{i^*}, \boldsymbol{u}_t\}$, $\mathcal{Y}_{i^*} = \{\mathcal{Y}_{i^*}, y_{t-1}\}$, $\mathcal{W}_{i^*} = \{\mathcal{W}_{i^*}, \eta e_t\}$.

The above quantization technique has similarities to the method in Chen et al. (2012). The key difference between the two strategies is that our proposal is not based on the distance measure in the input space. Rather, we use the data distribution as the criterion to update the network. This enhances utilization efficiency of the closest centre, which may yield better prediction accuracy and produce a more compact network. The proposed strategy for sequential learning, when applied to a single stock, is summarized in Algorithm 3.

Algorithm 3: Proposed sequential learning strategy.

input : $\mathcal{T}$– training data

Parameter setting:

- η- step-size;
- σ- kernel parameter;
- δ- change point threshold;
- ε- quantization threshold
- Initial dictionaries: $\mathcal{C}_1 = \{\boldsymbol{u}_1\}$, $\mathcal{Y}_1 = y_0$, $\mathcal{W}_1 = \eta y_1$
- Sets of initial dictionaries: $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1\}$, $\boldsymbol{\mathcal{Y}} = \{\mathcal{Y}_1\}$, $\boldsymbol{\mathcal{W}} = \{\mathcal{W}_1\}$

Computation:

while $\{\boldsymbol{u}_t, y_t\}$ *available* **do**

- Select the closest dictionary: $i^* = \arg\min_{\forall i} p_{t-1}^{\mathcal{Y}_i} \ln\left(p_{t-1}^{\mathcal{Y}_i}/p_t^{\mathcal{Y}_i}\right)$
- Compute the filter output: $\hat{y}_t = \sum_{j=1}^{L} \mathcal{W}_{i^*}^{(j)} \kappa_\sigma\left(\mathcal{C}_{i^*}^{(j)}, \boldsymbol{u}_t\right)$
- Compute the error: $e_t = y_t - \hat{y}_t$
- **if** $\epsilon(y_t, p_{t-1}) \geq \delta$ **then**
  - Form new dictionaries:
  - $\mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1} = \{\mathcal{C}_{i^*}, \boldsymbol{u}_t\}$, $\mathcal{Y}_{|\boldsymbol{\mathcal{Y}}|+1} = \{\mathcal{Y}_{i^*}, y_{t-1}\}$, $\mathcal{W}_{|\boldsymbol{\mathcal{W}}|+1} = \{\mathcal{W}_{i^*}, \eta e_t\}$
  - Update set of dictionaries:
  - $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}, \mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1}\}$, $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{\mathcal{Y}}, \mathcal{Y}_{|\boldsymbol{\mathcal{Y}}|+1}\}$, $\boldsymbol{\mathcal{W}} = \{\boldsymbol{\mathcal{W}}, \mathcal{W}_{|\boldsymbol{\mathcal{W}}|+1}\}$
- **else**
  - **if** $\min_{\forall i} p_{t-1}^{\mathcal{Y}_i} \ln\left(p_{t-1}^{\mathcal{Y}_i}/p_t^{\mathcal{Y}_i}\right) \leq \varepsilon$ **then**
    - Select the closest center: $j^* = \arg\min_{\forall j} \left\|\boldsymbol{u}_t - \mathcal{C}_{i^*}^{(j)}\right\|$
    - Update coefficient of closest center: $\mathcal{W}_{i^*}^{(j^*)} = \mathcal{W}_{i^*}^{(j^*)} + \eta e_t$
  - **else**
    - Assign a new center: $\mathcal{C}_{i^*} = \{\mathcal{C}_{i^*}, \boldsymbol{u}_t\}$
    - Assign a new desired output: $\mathcal{Y}_{i^*} = \{\mathcal{Y}_{i^*}, y_{t-1}\}$
    - Assign a new coefficient: $\mathcal{W}_{i^*} = \{\mathcal{W}_{i^*}, \eta e_t\}$

output: $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{Y}}$, $\boldsymbol{\mathcal{W}}$

5.1.2 A Stock Market Interdependence Approach

With the aim of enhancing stock returns prediction, we consider interdependencies between stock markets. More formally, let $\mathcal{D} = \{\mathcal{T}_r : r \in [1, S]\}$ be the set of training samples of S stocks, where $\mathcal{T}_r = \{\boldsymbol{u}_{t,r}, y_{t,r} : t \in [1, N]\}$. The underlying models of each $\mathcal{T}_r$, as seen in Figure 5.2, are learned separately using Algorithm 3, giving three sets of dictionaries per stock, i.e., $\boldsymbol{\mathcal{C}}_r = \{\mathcal{C}_{i,r} : i \in [1, |\boldsymbol{\mathcal{C}}_r|]\}$, $\boldsymbol{\mathcal{Y}}_r = \{\mathcal{Y}_{i,r} : i \in [1, |\boldsymbol{\mathcal{Y}}_r|]\}$, and $\boldsymbol{\mathcal{W}}_r = \{\mathcal{W}_{i,r} : i \in [1, |\boldsymbol{\mathcal{W}}_r|]\}$.

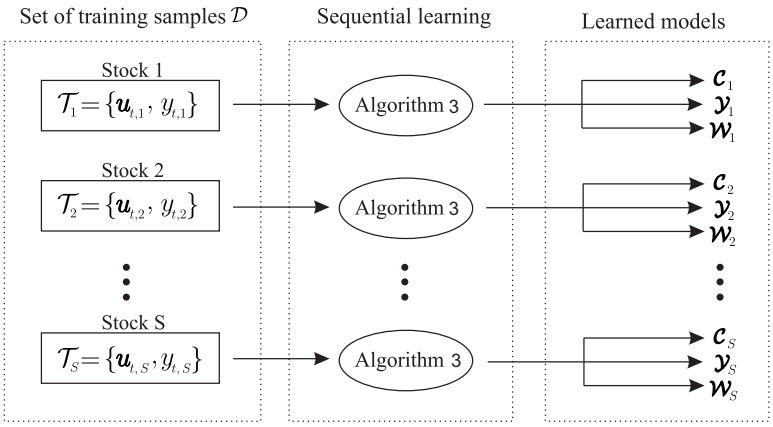


Figure 5.2: Sequential learning within a stock market interdependence approach.

Then, when a new input vector $\boldsymbol{u}^* \in \mathbb{R}^M$ arrives, the task is to predict a value $y^* \in \mathbb{R}$. Thus, the first step is to find the closest dictionary i^*, as in Section 5.1.1, using the following expression:

$$i^* = \arg\min_{i,r} p_{t-1}^{\mathcal{Y}_{i,r}} \ln \left(p_{t-1}^{\mathcal{Y}_{i,r}} / p_t^{\mathcal{Y}_{i,r}} \right), \tag{5.6}$$

from Equation (5.6), it can be seen that several stocks are considered in the selection of the closest dictionaries $\mathcal{C}_{i^*} = \{\boldsymbol{u}_j : j \in [1, L]\}$ and $\mathcal{W}_{i^*} = \{\eta e_j : j \in [1, L]\}$. In practice, these dictionaries are used to predict y^* as follows:

$$y^* = \sum_{j=1}^{L} \mathcal{W}_{i^*}^{(j)} \kappa_\sigma \left(\mathcal{C}_{i^*}^{(j)}, \boldsymbol{u}_t \right) \tag{5.7}$$

Note that, when a new sample $\boldsymbol{u}^*$ comes from the r-th stock, its prediction is usually calculated using the model learned on that stock. Here, we predict $\boldsymbol{u}^*$ using both the local model *and* individual local models learned from other stocks (see Equations (5.6) and (5.7)). This strategy has similarities to previously proposed methods such as ensemble learning (Dietterich et al., 2002; Krawczyk et al., 2017) and forecast combination (Newbold and Harvey, 2002; Baumeister and Kilian, 2015). The ensemble learning framework is constructed in two steps (Zhou, 2015): (1) a number of base learners are produced, which can be generated in a parallel or sequentially; (2) the base learners are combined using majority voting for classification or weighted averaging for regression.

However, the combination of multiple classifiers does not always outperform the best individual classifier (Polikar, 2009). In addition, better results may be obtained when some base learners are selected instead of an ensemble of them (Zhou et al., 2002). In contrast to ensemble learning methods, our approach does not combine the base learners; rather, here, the prediction tasks are performed only by the *best* learner. This better utilizes the internal forces of the market, providing a natural way to incorporate interdependencies between stock markets, and may enhance accuracy in real-time prediction tasks (Zhou and Tang, 2003). The proposed sequence prediction approach for stock returns is shown in Algorithm 4.

Algorithm 4: Predicting stock returns within a stock market interdependence approach.

input : $\boldsymbol{u}^*$; $\boldsymbol{\mathcal{C}}_1,\ldots,\boldsymbol{\mathcal{C}}_S$; $\boldsymbol{\mathcal{Y}}_1,\ldots,\boldsymbol{\mathcal{Y}}_S$; $\boldsymbol{\mathcal{W}}_1,\ldots,\boldsymbol{\mathcal{W}}_S$

Parameter setting:

  σ- kernel parameter

Computation:

Select the closest dictionary

$i^* = \arg\min_{\forall i,r} p_{t-1}^{\mathcal{Y}_{i,r}} \ln\left(p_{t-1}^{\mathcal{Y}_{i,r}}/p_t^{\mathcal{Y}_{i,r}}\right)$

Compute the output of the filter

$y^* = \sum_{j=1}^{L} \mathcal{W}_{i^*}^{(j)} \kappa_\sigma\left(\mathcal{C}_{i^*}^{(j)}, \boldsymbol{u}_t\right)$

output: y^*

5.2 Experimental Design

The aim is to use the last ten stock returns to predict the current day price change. The learned filter, as in Liu et al. (2010), is used to compute the performance values on each test set. We validate the proposed approach for stock returns prediction using MAE, MSE, and Sharpe Ratio (SR). The SR quantifies the average return earned in excess of the risk-free rate per unit of volatility or total risk (Wang et al., 2020). Here, as suggested in Almahdi and Yang (2019), the SR does not consider any risk-free rate. The previous metrics have been widely used to measure models' predictive power and their trading performance (France and Ghose, 2019; Portugal et al., 2018; Kalayci et al., 2019). The first two performance measures are regression-oriented metrics, while the last one is considered the industry standard for measuring risk-adjusted return (Jalota et al., 2017).

5.2.1 Data Sets

The daily closing prices data used to calculate the returns have been collected from Yahoo Finance[2]. Here, as suggested in Siikanen et al. (2018), we calculate daily log returns using the adjusted closing prices of each day. The publicly available data set can be downloaded using Python libraries such as yfinance[3]. Testing has been carried out on 24 different stocks from three major economies over 12 years (see Table 5.1). The considered training set ranges from January 17, 2006, to November 30, 2016, while the test set covers January 3, 2017, to February 28, 2018. In the simulations, as suggested in Chen (2019), the data set is standardized so all stocks have a mean estimation of zero and a standard deviation of one.

Table 5.1: Stocks in the experimental design.

Market	Ticker	Stock
DE	**ADS**	Adidas AG
	ALV	Allianz SE
	DPW	Deutsche Post AG
	DTE	Deutsche Telekom AG
	HEI	Heidelberg Cement AG
	LIN	Linde AG
	MRK	Merck KGaA
	SAP	SAP AG
UK	**ADM**	Admiral Group PLC
	AHT	Ashtead Group PLC
	BA	BAE Systems PLC
	BP	BP PLC
	CCL	Carnival PLC
	IAG	International Consolidated Airlines Group
	SKY	SKY PLC
	VOD	Vodafone Group PLC
US	**AAL**	American Airlines Group Inc
	AAPL	Apple Inc
	AMZN	Amazon Inc
	C	Citigroup Inc
	GOOGL	Alphabet In-CL A
	MSFT	Microsoft Corp
	SPY	SPDR S&P 500 Etf
	T	AT&T

[2]https://finance.yahoo.com/
[3]https://segarciave.github.io/stock_returns_prediction

5.2.2 Comparative Methods

For comparison purposes, the approach is contrasted with the following sequence prediction methods [4] (see Appendix B):

1. Long Short-Term Memory (LSTM), representing the state-of-the-art RNN model for sequence learning tasks (Nweke et al., 2018). Here, as it has shown competitive performance, the implementation known as vanilla LSTM is used (Greff et al., 2017);

2. Nearest Instance Centroid-Estimation (NICE), a recently proposed method that outperforms traditional KAF-based algorithms in prediction of chaotic time-series (Li and Príncipe, 2017);

3. Quantized Kernel Least-Mean-Square (QKLMS), a well-known method that uses an online vector quantization strategy (Chen et al., 2012);

4. Vector Autoregression (VAR), a forecasting method used to identify relationships among multiple time-series, being widely used in finance and econometrics. This method is a generalisation of the univariate autoregressive model (see Section 2.2.1) for forecasting a vector of time-series (Jang, 2020);

5. Vector Error Correction Model (VECM), the standard tool to handle multivariate non-stationary time-series. This method can be seen as a restricted VAR designed for use with non-stationary series that are known to be co-integrated (Liang and Schienle, 2019).

5.2.3 Parameter Settings

Table 5.2 summarizes the set-up of compared methods in the tested stocks. The parameters were heuristically adjusted to provide the best possible accuracy on this dataset. In particular, to ensure consistency in the results, both η and σ remain the same for all KAF methods. Thus, as long as this condition is met, different η and σ values from those shown in Table 5.2 will not offer an advantage to any particular algorithm.

We train the LSTM method using a single hidden layer with 20 neurons, as suggested in Cui et al. (2016). The LSTM activation function, as suggested

[4] The Python codes can be downloaded from https://github.com/segarciave/ESwA-2020

in Tian et al. (2018), is the sigmoid, while the optimization is performed by the Adam algorithm with MSE as the loss function (Kingma and Ba, 2014). The LSTM method was implemented using TensorFlow (version 1.4.0) [5] and Keras (version 2.1.2) [6]. The free parameters of autoregressive-based methods (VAR and VECM) were selected using heuristic approaches and following strategies proposed in the literature (Lütkepohl, 2013; Kuo, 2016). These parameters were chosen to achieve the best MSE on each tested data set. The performance of our proposal is sensitive to the selection of δ and ε; however, values for these parameters are selected as follows: (1) δ-threshold, based on our experimentation, an appropriate value is in the interval $[5, 15]$; (2) ε-quantization value, where a value of 0.0001 has shown stable performance on all tested stocks.

Table 5.2: Parameter setting of compared methods. M-input vector size, η-step size, σ-bandwidth, λ-quantization value, β-centroid distance, δ-threshold, ε-quantization value, $\mathcal{L}$-layers, $\mathcal{N}$-neurons per layer, $\mathcal{G}$-maximum number of lags, $\mathcal{A}$-number of lagged differences in the model, $\mathcal{I}$-cointegration rank.

Method	**Parameter**											
	M	η	σ	λ	β	δ	ε	$\mathcal{L}$	$\mathcal{N}$	$\mathcal{G}$	$\mathcal{A}$	$\mathcal{I}$
LSTM	10	-	-	-	-	-	-	1	20	-	-	-
NICE	10	0.05	0.5	0.06	2σ	-	-	-	-	-	-	-
Proposal	10	0.05	0.5	-	-	10	0.0001	-	-	-	-	-
QKLMS	10	0.05	0.5	0.4	-	-	-	-	-	-	-	-
VAR	10	-	-	-	-	-	-	-	-	15	-	-
VECM	10	-	-	-	-	-	-	-	-	-	3	0

5.3 Simulation Results and Analysis

The simulation results for the compared methods are shown in Tables 5.3 to 5.5, where the last row displays the average performance of each algorithm. The best overall method is in bold notation and marked with an asterisk. Table 5.3 shows the MAE values of each method in the considered stocks. The LSTM method outperforms the other algorithms, converging to the lowest average MAE value. This means that an error no greater than 0.012 can be expected during the prediction task on average. Although LSTM shows the best performance, the compared methods also converge to relatively low values of MAE. Note, a method

[5] https://www.tensorflow.org/
[6] https://keras.io/

that minimizes MAE will lead to forecasts of the median (Chai and Draxler, 2014), meaning that this scale-dependent metric may be unable to quantify the prediction of abrupt changes in stock returns.

Table 5.3: Testing MAE at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Stock		**Method**					
		LSTM	*NICE*	*Proposal*	*QKLMS*	*VAR*	*VECM*
DE	**ADS**	0.0109	0.0109	0.0137	0.0109	0.0115	0.0136
	ALV	**0.01**	0.0101	0.0122	0.01	0.0101	**0.0122**
	DPW	**0.0611**	**0.0608**	0.0645	**0.0609**	**0.0601**	**0.0711**
	DTE	**0.0063**	**0.0063**	0.0142	**0.0063**	**0.0065**	**0.0073**
	HEI	**0.0099**	**0.0097**	0.0161	**0.0098**	**0.0102**	**0.0115**
	LIN	**0.0077**	**0.0075**	0.01	**0.0075**	**0.0077**	**0.0093**
	MRK	**0.0071**	**0.0071**	0.0112	**0.0071**	**0.0073**	**0.0087**
	SAP	**0.007**	**0.0078**	0.013	**0.0078**	**0.0072**	**0.0086**
UK	**ADM**	**0.0084**	**0.0083**	0.0154	**0.0083**	**0.0087**	**0.0105**
	AHT	**0.0153**	**0.0153**	0.0215	**0.0152**	**0.0162**	**0.0196**
	BA	**0.0089**	**0.0087**	0.0134	**0.0088**	**0.0093**	**0.0106**
	BP	**0.0076**	**0.0077**	0.0194	**0.0077**	**0.0076**	**0.009**
	CCL	**0.0081**	**0.0082**	0.0156	**0.0082**	**0.0085**	**0.0106**
	IAG	**0.021**	**0.0213**	0.0322	**0.0212**	**0.0214**	**0.0246**
	SKY	**0.0323**	**0.0323**	0.039	**0.0322**	**0.0325**	**0.0398**
	VOD	**0.0072**	**0.0083**	0.0166	**0.0081**	**0.0074**	**0.0088**
US	**AAL**	**0.0139**	**0.0149**	0.0228	**0.0148**	**0.0155**	**0.0181**
	AAPL	**0.0086**	**0.0086**	0.0224	**0.0085**	**0.0088**	**0.0095**
	AMZN	**0.0095**	**0.0102**	0.0232	**0.01**	**0.0101**	**0.0115**
	C	**0.0087**	**0.0094**	0.0417	**0.0094**	**0.0097**	**0.0123**
	GOOGL	**0.008**	**0.0082**	0.0098	**0.0081**	**0.0082**	0.0091
	MSFT	**0.0077**	**0.0074**	0.01	**0.0075**	**0.0078**	**0.009**
	SPY	**0.0039**	**0.0038**	0.0057	**0.0038**	**0.0041**	**0.005**
	T	0.0083	0.0081	0.0105	0.0082	0.0081	0.0096
Mean		0.012	0.013	0.02	0.013	0.013	0.015
SD		0.012	0.012	0.013	0.012	0.011	0.014

Table 5.4 summarizes the MSE prediction performance, where lower values are better. The compared methods show similar MSE values, suggesting a relatively good prediction performance on all considered stocks under this metric. However, LSTM requires significant computing resources, as it needs to be retrained regularly on sequence prediction tasks (Cui et al., 2016). This means that several epochs have to be re-performed each time a new sample arrives in the system, which allows the best possible performance to be found when the training set is updated. The learning scheme of our proposal, unlike LSTM and autoregressive-based methods, does not require the entire training set in advance to begin learning of the model. In contrast, the model is updated sequentially while predictions

are obtained at the same time, providing an alternative to sequence prediction tasks. Additionally, the proposed interdependence strategy allows prediction of each stock using not only the local model but also the models learned from other stock markets, supporting the learning of long-term dependencies.

Table 5.4: Testing MSE at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Stock		Method					
		LSTM	*NICE*	*Proposal*	*QKLMS*	*VAR*	*VECM*
DE	**ADS**	0.0003	0.0003	0.0004	0.0003	0.0003	0.0004
	ALV	**0.0003**	0.0003	0.0003	0.0003	0.0003	**0.0003**
	DPW	**0.0135**	**0.0135**	0.014	**0.0134**	**0.0132**	**0.0172**
	DTE	**0.0001**	**0.0001**	0.0003	**0.0001**	**0.0001**	**0.0001**
	HEI	**0.0002**	**0.0002**	0.0004	**0.0002**	**0.0002**	**0.0002**
	LIN	**0.0001**	**0.0001**	0.0002	**0.0001**	**0.0001**	**0.0001**
	MRK	**0.0001**	**0.0001**	0.0002	**0.0001**	**0.0001**	**0.0002**
	SAP	**0.0001**	**0.0001**	0.0003	**0.0001**	**0.0001**	**0.0001**
UK	**ADM**	**0.0002**	**0.0002**	0.0004	**0.0002**	**0.0002**	**0.0002**
	AHT	**0.0005**	**0.0005**	0.0008	**0.0005**	**0.0005**	**0.0007**
	BA	**0.0002**	**0.0002**	0.0003	**0.0002**	**0.0002**	**0.0002**
	BP	**0.0001**	**0.0001**	0.0005	**0.0001**	**0.0001**	**0.0001**
	CCL	**0.0001**	**0.0001**	0.0004	**0.0001**	**0.0001**	**0.0002**
	IAG	**0.0009**	**0.0009**	0.0016	**0.0009**	**0.0009**	**0.0011**
	SKY	**0.0028**	**0.0028**	0.0034	**0.0028**	**0.0028**	**0.0037**
	VOD	**0.0001**	**0.0001**	0.0004	**0.0001**	**0.0001**	**0.0002**
US	**AAL**	**0.0004**	**0.0004**	0.0008	**0.0004**	**0.0004**	**0.0005**
	AAPL	**0.0001**	**0.0002**	0.0006	**0.0002**	**0.0002**	**0.0002**
	AMZN	**0.0002**	**0.0002**	0.0007	**0.0002**	**0.0002**	**0.0003**
	C	**0.0001**	**0.0002**	0.0019	**0.0002**	**0.0002**	**0.0002**
	GOOGL	**0.0001**	**0.0001**	0.0002	**0.0001**	**0.0001**	0.0002
	MSFT	**0.0001**	**0.0001**	0.0002	**0.0001**	**0.0001**	**0.0002**
	SPY	**0.000**	**0.000**	0.0001	**0.000**	**0.000**	**0.0001**
	T	0.0001	0.0001	0.0002	0.0001	0.0001	0.0002
Mean		0.001	0.001	0.001	0.001	0.001	0.001
SD		0.003	0.003	0.003	0.003	0.003	0.003

The previous metrics attempt to measure the predictive power of models. However, they do not provide insights into the performance of the compared methods from a trading point of view. Table 5.5 shows the SR values of each method in the considered stocks, where a higher value means greater returns for the portfolio relative to the inherent risk (Kaplanski et al., 2016). We see that our proposal outperforms the other algorithms, converging to higher values of SR, which suggests that the proposed interdependence strategy helps maximize returns while reducing volatility. This strategy selects a model based on each new data point that arrives. For example, it may be possible to obtain

some predictions on ADS (a stock traded in Germany) using the model learned from VOD (a stock traded in the UK). However, the convergence speed may be adversely affected if the kernel bandwidth parameter is inappropriately chosen. Consequently, a suitable value can be selected using a method such as Silverman's rule of thumb (Liu et al., 2010).

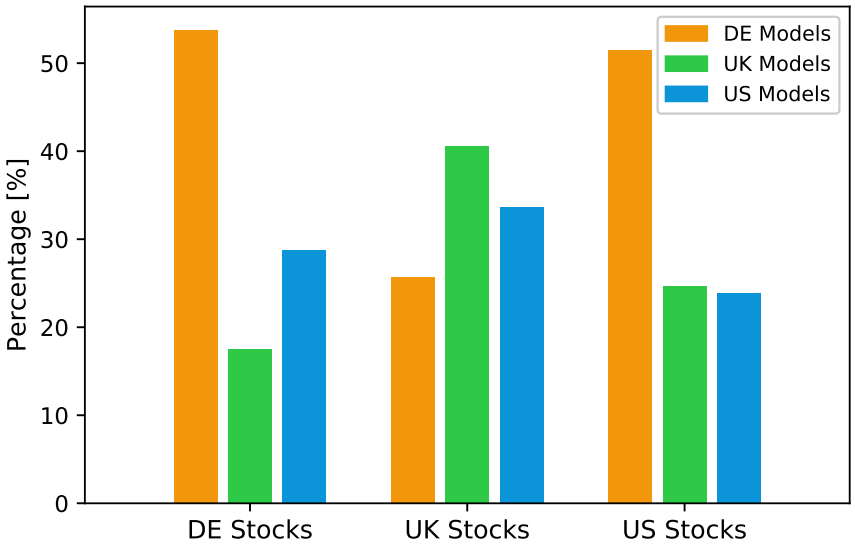
Table 5.5: Testing SR at final iteration in stock returns prediction. SD–Standard Deviation. For every compared method we conducted a paired t-test against our proposal. Highlighted values indicate statistical significance at 5%.

Stock		**Method**					
		LSTM	*NICE*	*Proposal*	*QKLMS*	*VAR*	*VECM*
DE	**ADS**	0.9017	0.2162	3.6583	0.0051	2.2682	0.9369
	ALV	**1.2965**	0.2089	3.3982	0.0729	1.0817	**0.0021**
	DPW	**-0.7522**	**-1.0027**	2.13	**-1.1187**	**-0.8717**	**0.7948**
	DTE	**0.2688**	**-0.0372**	1.9498	**-0.1236**	**0.7665**	**1.2571**
	HEI	**0.8574**	**0.2089**	2.0087	**-0.0287**	**1.3908**	**2.0077**
	LIN	**0.5044**	**0.1126**	3.1285	**0.0646**	**1.2623**	**2.5074**
	MRK	**0.7296**	**0.0983**	3.627	**-0.0257**	**1.4655**	**0.6756**
	SAP	**0.4705**	**0.0497**	4.39	**-0.0371**	**1.8604**	**2.4806**
UK	**ADM**	**0.4663**	**0.334**	5.9617	**0.0065**	**2.4326**	**1.0164**
	AHT	**2.2794**	**0.936**	5.3711	**-1.2048**	**-1.3246**	**-0.3778**
	BA	**0.7246**	**0.0998**	2.1991	**-0.0579**	**1.2097**	**1.2656**
	BP	**1.0361**	**1.5645**	6.549	**0.1094**	**-0.0275**	**-1.3034**
	CCL	**1.2433**	**0.5456**	5.9871	**0.0216**	**-0.7435**	**-0.5154**
	IAG	**1.9193**	**-1.123**	4.5652	**1.5662**	**0.3662**	**-1.3497**
	SKY	**-0.2469**	**0.3637**	1.4274	**0.318**	**-1.2898**	**-1.3334**
	VOD	**0.6955**	**0.222**	5.2585	**0.1688**	**0.3891**	**1.4227**
US	**AAL**	**1.1319**	**-0.5774**	5.8926	**0.4946**	**-1.144**	**-1.1616**
	AAPL	**0.6432**	**0.1279**	2.6692	**-0.1274**	**1.0422**	**1.648**
	AMZN	**0.5934**	**0.1681**	1.5527	**-0.0205**	**1.0327**	**1.7307**
	C	**0.975**	**0.2737**	3.6621	**0.0879**	**1.1656**	**-0.3031**
	GOOGL	**0.5118**	**0.1185**	2.9763	**0.0459**	**1.3334**	1.2835
	MSFT	**0.5291**	**0.1301**	2.8933	**0.0355**	**1.2268**	**2.1107**
	SPY	**0.4121**	**0.0827**	1.7878	**-0.0977**	**1.0711**	**1.7259**
	T	1.2947	0.1438	3.3456	-0.0684	1.3681	1.1828
Mean		0.77	0.136	3.6	0.004	0.722	0.738
SD		0.606	0.517	1.532	0.484	1.061	1.207

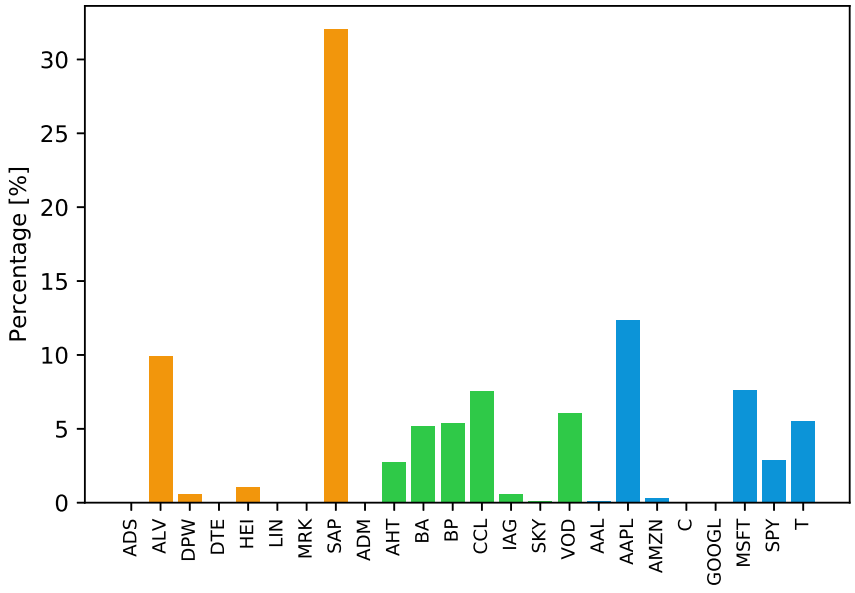
Figure 5.3 shows the models used by the approach to predict the considered stocks, where each color represents a different stock market. Figure 5.3(a) displays the percentage of models from Germany, the UK and the US that were used to predict stock returns. For example, 18% of the models used to predict the 8 stocks traded in Germany are from the UK, while 60% of the models used to predict the 8 stocks traded in the US are from Germany. The models from Germany, such as SAP AG (SAP) (see Figure 5.3(b)), are widely used to predict stock returns in the UK and the US (see Figure 5.3(a)), suggesting that the patterns

encoded by German stocks are more appropriate to predict other stock markets. Figure 5.3(b) also shows that there are some stocks with 0% such as ADS, DTE, LIN, SKY, AAL, etc. This means that their models were not used to obtain their own predictions, nor were they used to predict other stocks. However, their performance is still competitive (see Tables 5.3 to 5.5).

The true and predicted stock returns are shown in Figures 5.4 to 5.6 for three representative stocks, namely Allianz SE (ALV), Ashtead Group PLC (AHT), and Alphabet In-CL A (GOOGL), respectively. The top graphs (Figures 5.4(a), 5.5(a) and 5.6(a)) display predictions for all tested methods. The bottom graphs (Figures 5.4(b), 5.5(b) and 5.6(b)) show the predictions of our proposal, where each color represents a different stock market. Note that, unlike the other methods, the approach obtains more accurate predictions when predicting abrupt stock return changes. Figure 5.4(b) provides an explanation for this behavior, where it can be seen that predictions were calculated using models learned from stocks in the US and UK. Thus, the interdependence strategy of our proposal provides an advantage over competing algorithms, as they do not consider the interconnections between stock markets. Figure 5.5(b) shows a similar situation, where predictions are improved using models learned from the US. Figure 5.6(b) further suggests that the predictions of a particular stock can be enhanced by incorporating models learned from other stock markets.

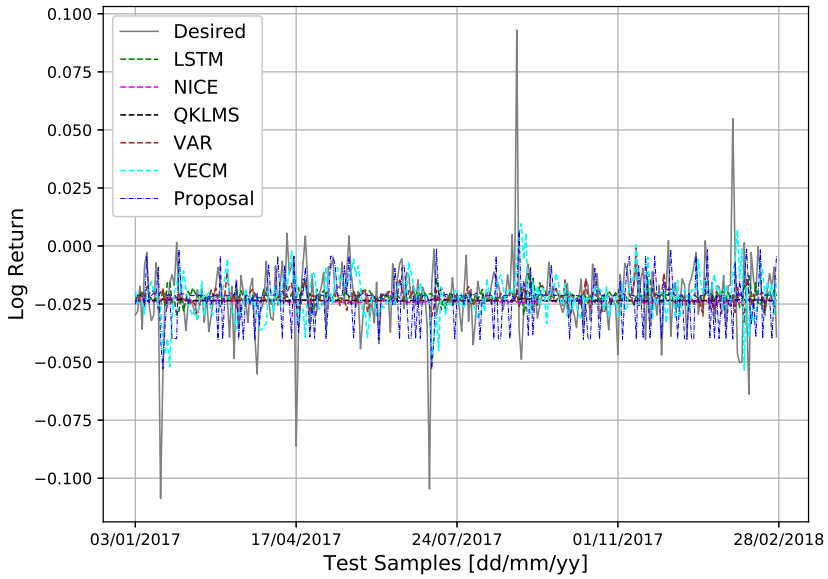


(a) Markets

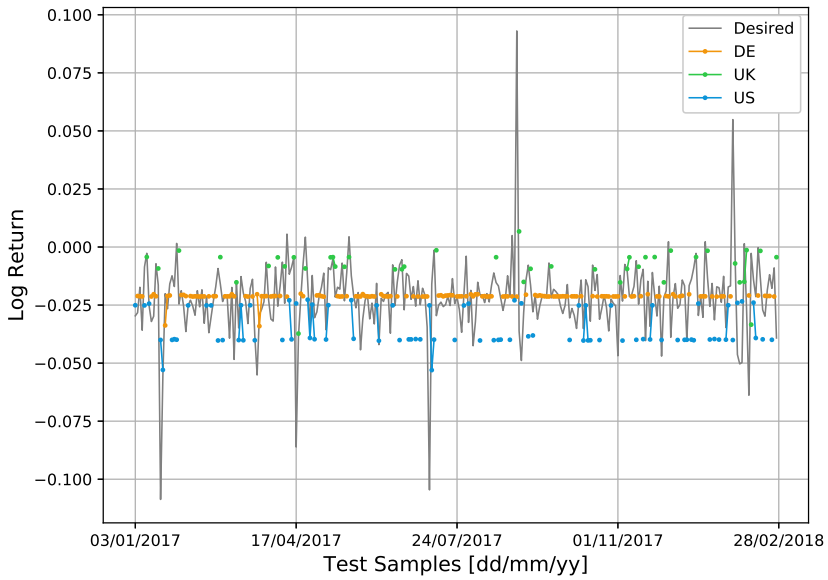


(b) Stocks

Figure 5.3: Models used by the proposed approach to predict the 24 stocks.

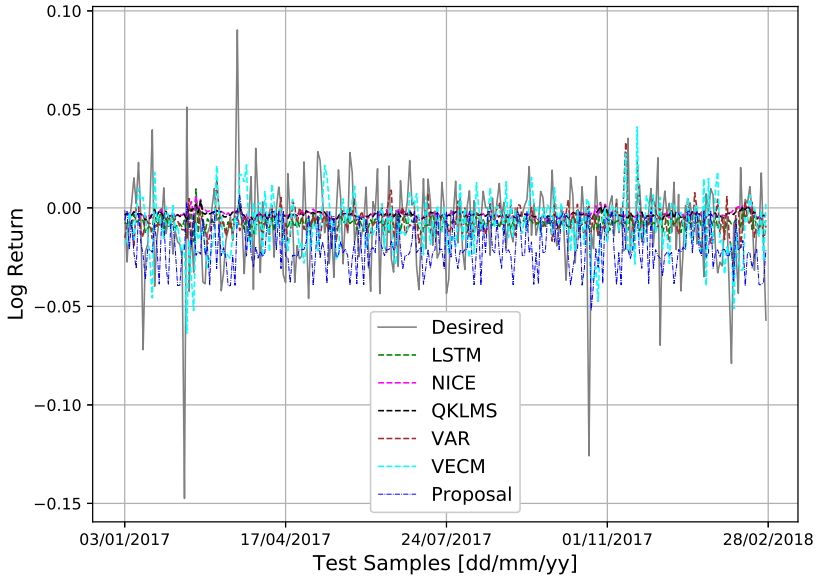


(a) ALV (compared methods)

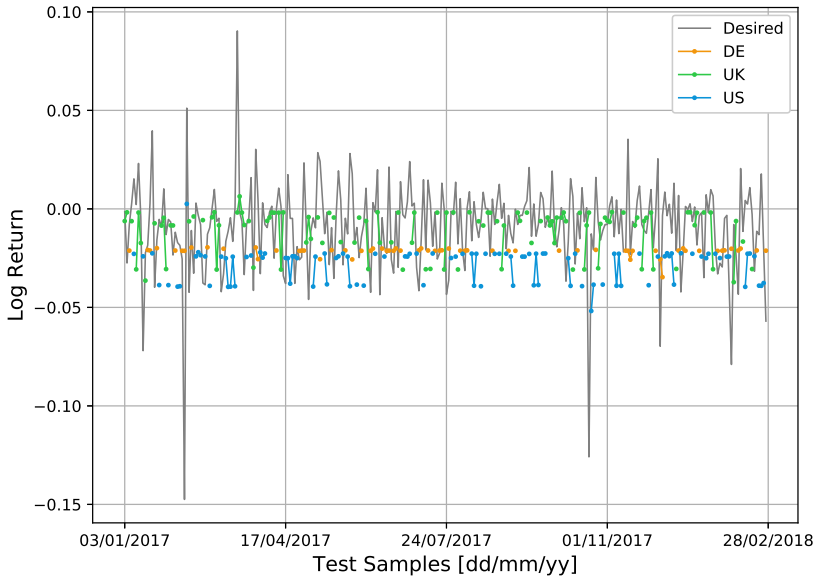


(b) ALV (only proposal)

Figure 5.4: Stock return predictions in the test sets of Allianz SE.

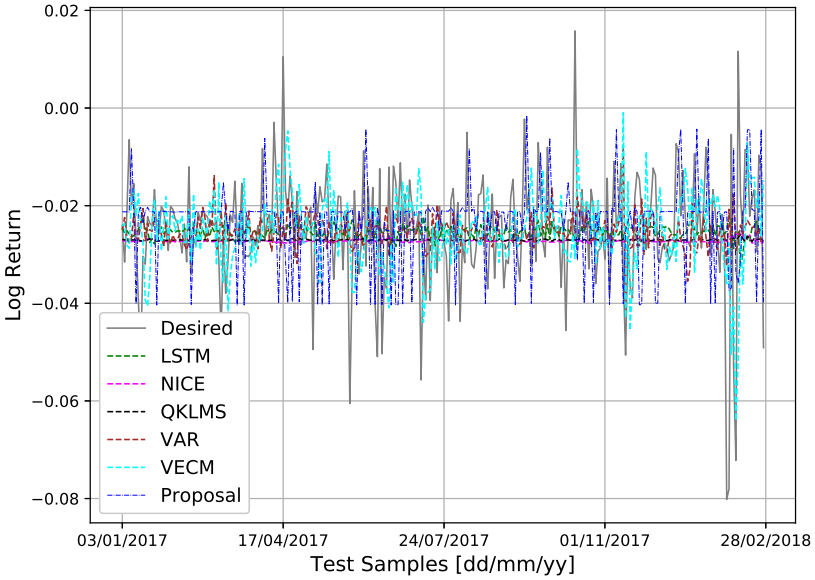


(a) AHT (compared methods)

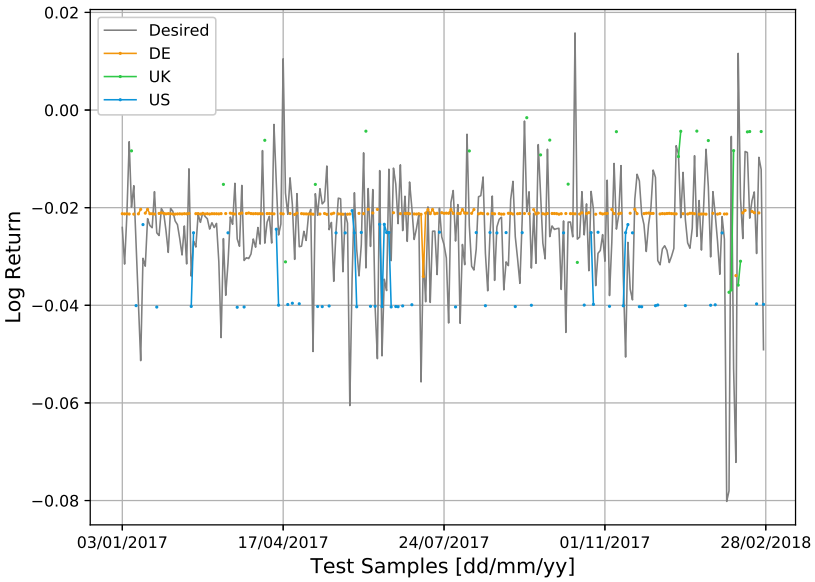


(b) AHT (only proposal)

Figure 5.5: Stock return predictions in the test sets of Ashtead Group PLC.



(a) GOOGL (compared methods)



(b) GOOGL (only proposal)

Figure 5.6: Stock return predictions in the test sets of Alphabet In-CL A.

5.4 Chapter Summary

This chapter introduced an approach for stock returns prediction using kernel adaptive filtering within a stock market interdependence approach. The approach sequentially predicts stock returns by considering interconnections between stock markets. The approach has been tested on 24 different stocks from three major economies, i.e., United States, United Kingdom, and Germany. Simulation results demonstrate that the interdependence strategy used by our proposal enhances prediction accuracy, representing an advantage over compared methods. Finally, in the following chapter, the three proposed approaches will be critically evaluated and suggestions for future work will be included.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, a variety of kernel-based approaches to support sequence prediction tasks in financial time-series have been developed. The approaches learn from continuous sequence of data records and exhibit high tolerance to noisy and non-stationary conditions. This thesis aims to address the weakness of data-centralized and off-line machine learning methods, which fail to consider fast time-varying characteristics from data resources in other regions, sectors, and markets. Thus, when designing the proposed kernel-based approaches, both the sequential and interdependent nature of financial time-series have been considered. The learning scheme of the proposed methods, unlike that of neural networks, is a combination of error-correction and memory-based learning, meaning that the whole training set is not required to start learning the model. In contrast, as the samples arrive, predictions are obtained while the model is updated sequentially at the same time, which is useful in online applications. Results show faster convergence to low MSE values and higher Sharpe ratio when compared with auto-regressive based models, recurrent neural networks, and KAF methods. In addition, regarding the distributed learning approach, results suggests that the United States market is more influenced by the European and not vice versa, which is in line with previous empirical findings.

The approach in Chapter 3, using stochastic gradient algorithms that minimize the mean square error, sequentially updates the kernel-size and step-size parameters of a kernel adaptive filtering approach. The kernel-size problem is addressed from a practical application focus; to the best of our knowledge, such

an approach has not been previously proposed. Further, the results show a general improvement in prediction accuracy in online sequential learning environments. In comparison to similar methods, our proposal converges more quickly and achieve better accuracy. Overall, relatively lower values of MSE are obtained in all tested data sets. Neural network methods such as LSTM give competitive performance; however, the LSTM model must be retrained regularly in sequence prediction tasks. The proposed approach, unlike LSTM, does not need the whole training set to start learning the model. In contrast, as samples arrive, predictions are generated and the model is updated sequentially at the same time.

The approach in Chapter 4 provides an alternative adaptation criterion when training kernel-based adaptive systems. The proposal uses instantaneous entropy as the adaptation criterion within a stochastic gradient descent optimization approach. This aims to minimize the error entropy between the model output and the desired response, capturing higher order statistics and information content of financial time-series rather than simply their energy. Further, with the aim of enhancing stock returns prediction, the architecture of neural networks is extended to incorporate kernel adaptive filtering by allowing any neuron to be replaced by a kernel machine, capturing complex patterns behind the data. The proposed approach has $\mathcal{O}(t + L^2)$ computational complexity, which is competitive with similar prediction methods. In particular, the average performance of KLMS is improved by 0.5%, while there is an improvement of 54% compared with KMEE. This demonstrate that approach exhibits high tolerance to noisy conditions and provides stable performance in real-world applications.

The approach in Chapter 5 sequentially predicts stock returns within a stock market interdependence approach. This means that predictions were performed using not only local models, but also from the individual local models learned from other stocks, providing a natural way to incorporate inter-dependencies between financial markets. The approach uses the data distribution as the criterion to encode different patterns of the input space, producing a more compact network. This strategy helped maximize returns while maintaining the robustness and simplicity of kernel-based adaptive filters. In addition, as simulation results show, the models learned from the German market are more suitable for making predictions in other stock markets. This suggests that the United States market is more influenced by the European and not vice versa, which is in line with previous empirical findings (Jizba et al., 2012; Rezayat and Yavas, 2006).

6.2 Future Work

We have proposed kernel-based approaches to sequentially predict financial time-series. However, considering the theoretical and experimental results, there remain a number of areas of interest to improve learning performance and enhance data interpretability. The following proposes ideas for future work plans.

6.2.1 Predicting Several Steps Ahead

Proposed methodologies in Chapters 3 to 5 were effective when sequentially predicting financial time-series. They were designed within a one step ahead strategy using ordered sequences of data records. The proposed methodologies aimed to predict the next value, rather than predicting values of several steps ahead. The development of methodologies to sequentially predict financial time-series several steps ahead may enhance long-term investment strategies, practitioners decision-making, and portfolio allocation (Chen et al., 2018b).

Predicting several steps ahead may incur additional challenges such as accumulation of errors, model complexity, and increased uncertainty (Du et al., 2018). There are three main strategies for addressing these challenges (Taieb and Atiya, 2015): (1) *recursive*, where previously predicted values are taken as inputs by a one step ahead predictor; (2) *direct*, where different prediction models are trained to directly predict each step ahead; (3) *joint*, where a multi-output model is designed to return a vector of future values in a single step.

An interesting line of work involves the extension of this research to sequentially predict financial time-series several steps ahead, i.e., weekly or monthly predictions using historical data of stock markets. However, the following limitations must be considered: (1) *recursive* strategy methods suffer from the accumulated problem, meaning that errors are propagated through iterations, which affects prediction accuracy; (2) the cost of training *direct* strategy methods is high when the prediction horizon increases, restricting their flexibility in real-world applications; (3) *joint* strategy methods have to train new models from scratch once the prediction horizon is changed.

Thus, a possible direction to follow is extending the sequence prediction methodologies developed in Chapters 3 to 5 to several steps ahead using a *joint* strategy. This may avoid the error accumulation and conditional independent assumption of *recursive* and *direct* methods, respectively. In addition, the *joint*

strategy opens the door to formulate kernel adaptive filter models within a multiple input approach, which may enhance prediction accuracy in streaming data applications. In this sense, with the aim to deal with multi-modal analysis, kernel tensor representations may be incorporated into the sequence prediction approaches.

6.2.2 Automated Machine Learning

The kernel-size and adaptive step-size strategies proposed in Chapter 3, when predicting financial time-series, proved to improve convergence time while maintaining competitive performance. A limitation of our proposal is the need to choose initial values for those parameters. In this thesis, this limitation was addressed using a grid-search strategy minimizing the mean squared error. However, when working with larger databases, finding the optimal initial parameters may be too complex in terms of memory constraints or huge computational complexity.

In large scale applications, defining a default search space is critical to achieve good predictive performance. Thus, potential directions for future work include applying automated machine learning approaches to our methodologies. In particular, one direction to follow is the implementation of Bayesian optimization strategies, as they have proven useful to search for the best parameter configurations in large scale applications (Salinas et al., 2020). Another alternative is to use maximum marginal likelihood (Liu et al., 2010), which has been shown to be effective in determining the parameters in kernel-based methods such as kernel adaptive filters.

The automated selection of a Mercer kernel for the methodologies proposed in Chapters 4 and 5 is another interesting line of research. This parameter defines similarity between data points in any kernel method. Thus, techniques such as multiple kernel learning may provide different notions of similarity to address the kernel selection problem. The study of additional Mercer kernels with universal approximation capability may provide alternatives to better capture complex patterns in noisy and non-stationary conditions, maximizing returns and improving prediction performance.

Then, with the aim of enhancing data representation, dimensional reduction techniques and additional information theoretic quantities (such as mutual information) may be extended to incorporate the methodologies proposed in Chapters 3 to 5. The dimensional reduction approaches aim to preserve the

significant structure of high-dimensional data in a low-dimensional space. This may provide a way to measure and visualize complex non-linear dynamic relationship between stock markets.

6.2.3 Transfer Learning

The knowledge transfer strategy used in Chapter 3 proved to be an efficient alternative when training in non-stationary conditions. That is, the transfer learning strategy aimed to use previously learned knowledge to enhance sequence prediction tasks in univariate financial time-series. Thus, another direction for future work is to extend the transfer learning strategy to different data sets, rather than simply using stock or foreign exchange prices (see Chapters 3 to 5). These sources may include heterogeneous and unstructured data from textual news reports, well-structured high frequency limit order book data, tick-by-tick quotes on different asset classes, commodity prices, economic indicators, and inflation rates. This may help financial institutions to prevent losses on portfolios of financial assets that are traded in the financial markets. Then, such a learning strategy may be combined with the automated machine learning work described in the previous section. This would allow an extension of our research to big data applications, and perform transfer learning across different regions, markets, or sectors. For example, in addition to analysing stock returns, the models may consider topics that appear in the media, giving them additional information that may be useful to perform prediction tasks.

Another interesting line of work is to integrate sentiment analysis (news data information) into the distributed learning paradigm proposed in Chapter 5. The extraction of information from news data may be done using Natural Language Processing techniques such as *word2vec* (Mikolov et al., 2013), which provides word associations from a large structured set of texts (also known as corpus). The news data may be obtained from the *Global Database of Events, Language, and Tone (GDELT)*, which provides free access to news media reports in more than 100 languages around the world (Leetaru and Schrodt, 2013). This information may provide insights to understand future trends in stock markets. In this sense, hybrid models based on deep learning and kernel principles may encode better temporal and spatial relationships, providing robust models under highly non-stationary conditions.

Appendix A

Financial Data

Financial data are continuously generated by different and separate data sources, such as banks, corporations, and individuals (Rejeb and Arfaoui, 2016). These data usually provide information on stock prices, transaction costs, quoted rates, reported earning, among others (Giudici, 2018). In this section, we introduce the definitions of financial time-series, assets, markets, returns, and stock prices.

A.1 Financial Time-Series

Financial time-series are ordered sequence of data records that become available over time (Taylor, 2008). The sequence imposes an order on the samples that must be preserved when training models and making predictions (Gueniche et al., 2015). Financial institutions aim to predict these time-series, such that investors could hedge their assets or take appropriate actions given their investment objectives and risk tolerance (Laitinen and Suvas, 2016). In practice, large numbers of orders are placed to buy or sell stocks within weeks, days, minutes, and seconds. This may be done by using algorithmic trading, which is a method that executes orders using automated trading instructions (see Section 2.1.1). High-frequency trading executes orders thousands of times a day without holding positions at the end of the day (Aitken et al., 2015). Thus, from a high-frequency point of view, the time intervals are the lengths of time between successive trades of the same stock. In this thesis, we focus on equally spaced data rather than high-frequency and its micro-structure characteristics.

A.2 Financial Assets and Markets

An asset is any resource, owned by a business or an economic entity, that can be traded (Diehl, 2016). The following are the most common types of financial assets: *cash* (Yilmaz, 2020), representing paper currency; *bonds* (Dey and Gibbon, 2018), which are contracts representing a loan; *stock shares* (Ameriks et al., 2020), these are documents representing ownership and the rights of that ownership; *master limited partnerships* (Massey, 2016), which is a publicly traded entity taxed as a partnership; *real estate investment trusts* (Waldron, 2018), which owns assets based on real estate.

Markets and Exchanges. The term market refers to an organization that allows trading financial instruments, where the most popular is the exchange market (Gentle, 2020), e.g., *London Stock Exchange* (LSE), *New York Stock Exchange* (NYSE), and *Tokyo Stock Exchange* (TSE).

Foreign Exchange. The foreign exchange (FX) market is the largest and most liquid of financial markets (Gau and Wu, 2017). The FX is a non-stop cash market where it is possible to speculate on changes in exchange rates of foreign currencies (Wang and Xie, 2016). This market operates through a global network of banks, corporations and individuals trading one currency for another but has no physical location and no central exchange (Stosic et al., 2016).

Types of Owners. The owners of financial assets are individuals or institutions. The institutional investors are hedge funds, commercial banks, and insurance companies. The financial decisions are made by institutional investors, while the assets may be owned by individuals (Kidwell et al., 2016).

Returns. Financial returns are the incomes produced by the changes in value of the asset itself over a given period of time (Wacker et al., 2016). The total return of an asset is the relative change in price plus the income in the form of interest or dividends (Gentle, 2020).

Risk. The variation in the value (or return) of an asset is known as risk. The following are the main types of risk: *market risk* (Dowd, 2007), which is the risk of losses because of movements in market prices; *model risk* (Tunaru, 2015),

this risk is associated with the use of inaccurate models to make decisions; *credit risk* (Bielecki and Rutkowski, 2013), describing the loss due to a failure to make payments; *liquidity risk* (Hassan et al., 2019), which is the risk associated with a security that cannot be traded quickly in the market; *operational risk* (Jarrow, 2008), this risk is related to inadequate or failed internal processes.

A.3 Returns on Assets

Given a financial time-series $\mathcal{T} = \{x_t : t \in [1, N]\}$, where $x_t \in \mathbb{R}$ is the value of an asset at time t, while $N \in \mathbb{R}$ indicates the number of samples in the time-series. The *simple return* and *log return* are computed as follows (Siikanen et al., 2018):

$$R = \frac{x_t - x_{t-1}}{x_{t-1}} \tag{A.1a}$$

$$\tilde{R} = \ln\left(\frac{x_t}{x_{t-1}}\right) \tag{A.1b}$$

where $\ln(\cdot)$ is the natural logarithm. *Log return* is normally used when analysing volatility (Chaim and Laurini, 2018), while *simple return* is preferred when analysing portfolios (Zhou and Xu, 2018). In addition, the *annualized log return* assumes a yearly resolution t (Beshears et al., 2009). Similarly, in the *daily log return*, x_{t-1} and x_t are the closing prices of consecutive days (Ardia et al., 2019). The same analysis can be done for *weekly log return* and *monthly log return* (Novak, 2007; Chan et al., 2008).

Aggregating Log Returns. Let $x_1, \ldots, x_5$ be a sequence of prices. The sequence of log returns are $\tilde{R}_2 = \log(x_2) - \log(x_1), \ldots, \tilde{R}_5 = \log(x_5) - \log(x_4)$. Then, the log return from time $t = 1$ to $t = 5$ is given by (Alexeev et al., 2014),

$$\log(x_5) - \log(x_1) = \tilde{R}_2 + \tilde{R}_3 + \tilde{R}_4 + \tilde{R}_5 \tag{A.2}$$

Thus, the log return from $t = 1$ to $t = N$ can be expressed as:

$$\tilde{R} = \sum_{i=2}^{N} \tilde{R}_i \tag{A.3}$$

Aggregating Simple Returns. Given two asset values $x_{1,1}$ and $x_{2,1}$ at time $t = 1$. The simple return, at time $t = 2$, can be computed as (see Equation (A.1a)):

$$R_1 = \frac{x_{1,2} - x_{1,1}}{x_{1,1}} \tag{A.4a}$$

$$R_2 = \frac{x_{2,2} - x_{2,1}}{x_{2,1}} \tag{A.4b}$$

The combined return considers the relative total values in the two time periods. Thus, the combined simple return can be computed as (Gentle, 2020):

$$\begin{aligned} R &= \frac{x_{1,1}}{x_{1,1} + x_{2,1}} \left(\frac{x_{1,2}}{x_{1,1}} - 1 \right) + \frac{x_{2,1}}{x_{1,1} + x_{2,1}} \left(\frac{x_{2,2}}{x_{2,1}} - 1 \right) \\ R &= \frac{x_{1,1}}{x_{1,1} + x_{2,1}} R_1 + \frac{x_{2,1}}{x_{1,1} + x_{2,1}} R_2 \end{aligned} \tag{A.5}$$

A.4 Stock Prices

A stock price represents the value, current situation, and future perspectives of a company in the market (Kumar, 2019). The price of a given asset can be measured using indexes that are associated with economic variables such as production, employment, and interest rates (Peiro, 2016). The most reliable indicator of the present value of a security (financial instrument that holds some monetary value) is called current price (or market value), which is the most recent price that is traded on an exchange (Hart and Zingales, 2017). In practice, the current price indicates how much a buyer (or a seller) would be willing to accept to make a transaction.

Market Value. The market value is the stock price times the number of shares that the company has issued, where these shares are not owned by the company itself (Kucharska-Stasiak, 2018). The enterprise value, which measures the total value of a company, considers the market value, debts, and cash on the balance sheet of the company (Liu and Zhang, 2017).

Stock Price Fluctuations and Trading Volume. The trading volume is the amount of shares traded during a given period of time (Oliveira et al., 2017). The price fluctuations are usually represented in a candlestick chart, which contains information about the open, high, low, and closing prices.

Fundamental and Technical Analysis. Fundamental analysis studies general considerations about the company such as overall value, earnings, and debts. Technical analysis aims to predict future price movements studying patterns of price changes and trading volumes (Gentle, 2020).

National Best Bid and Offer. This a regulation that requires brokers to trade at the best available ask price (lowest) and the best available bid price (highest) when buying and selling securities for customers. The mid-price is a reference price calculated by taking the average of the current quoted bid and ask prices (Battalio et al., 2004).

Pricing Gaps. The abrupt changes in the stock price depend on various factors such as the revelation of a large holding in the stock by a large institution, major shareholder selling, the propagation of statements through media sources, financial policies, among others (Maskawa, 2016).

Initial Public Offering. The initial public offering, also known as stock market launch, is the process of selling the stocks of a company on a public stock exchange for the first time (Khatri, 2017).

Appendix B

Machine Learning Models

Learning approaches are usually performed in three different ways (Alpaydin, 2020): (1) **supervised** learning, which requires a collection of desired or target responses; (2) **unsupervised** learning, where no desired outputs are given to the learning algorithm; (3) **reinforcement** learning, which learns using a sequence of state–action–reward without an explicit input–output available.

B.1 Neural Networks

The basic neural network (NN) model can be described as follows (Bishop, 2006):

$$a_j = \sum_{i=0}^{M} w_{ji}^{(1)} u_i \tag{B.1}$$

where the idea is to construct N linear combinations of the input variables $u_0, u_1, \ldots, u_M$, being $j = 0, 1, \ldots, D$ and the superscript indicates that the corresponding parameters are in the first layer of the network. The parameters a_j and $w_{ji}^{(1)}$ are usually known as activations and weights, respectively. Then, each parameter a_j is transformed using the following differentiable and non-linear activation function (Sibi et al., 2013):

$$z_j = h(a_j) \tag{B.2}$$

The non-linear function $h(\cdot)$ is usually a sigmoidal function such as the *logistic* and *tanh* functions. The previous values are linearly combined to provide the

following output unit activations in the second layer of the network

$$a_k = \sum_{j=0}^{D} w_{kj}^{(2)} z_j$$

where $k \in [1, K]$, being K the total number of outputs. Then, with the aim of providing a set of network outputs y_k, the outputs a_k are transformed using an appropriate activation function, which depends on the nature of the data and the assumed distribution of target variables (Sharma, 2017). Thus, each output unit activation is transformed using a logistic sigmoid function as

$$\hat{y}_k = \varsigma(a_k) \tag{B.3}$$

where $\varsigma(a) = 1/(1 + \exp(-a))$. Finally, the overall network function takes the following form

$$\hat{y}_k(\boldsymbol{u}, \boldsymbol{w}) = \varsigma\left(\sum_{j=0}^{D} w_{kj}^{(2)} h\left(\sum_{i=0}^{M} w_{ji}^{(1)} u_i \right) \right) \tag{B.4}$$

Thus, the neural network model is a non-linear function from a set of input variables $\{u_i\}$ to a set of output variables $\{\hat{y}_k\}$ controlled by a vector of adjustable parameters $\boldsymbol{w}$ (see Figure B.1). Note that the NN uses continuous sigmoidal non-linearities in the hidden units. This means that the NN is differentiable with respect to the network parameters, which is useful during the training stage (Da Silva et al., 2017).

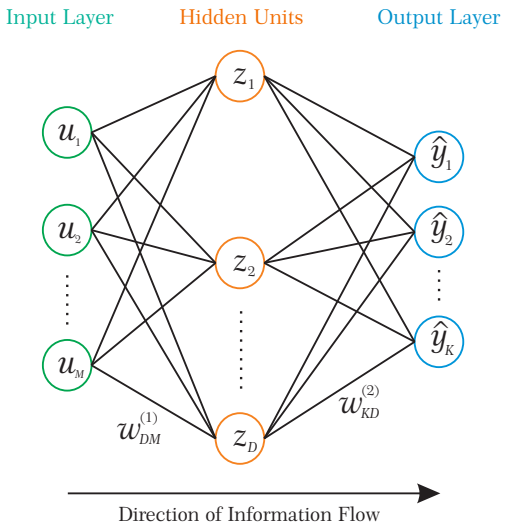


Figure B.1: Neural Network diagram with two layers.

The previous network architecture is the most commonly used in practice, which can be generalized by considering additional layers (Bishop, 2006). In addition, NNs are considered universal approximators, meaning that they can uniformly approximate any continuous function on a compact input domain to a desired accuracy (Sonoda and Murata, 2017). However, even when performing similar tasks, the proper choice of network parameters can vary widely and are often chosen in a trial-and-error process (Tzeng and Ma, 2005; Erkaymaz et al., 2017; Dutta et al., 2018).

Network Training. Given a sequence of input-output examples $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, where $\boldsymbol{u}_t \in \mathbb{R}^M$ is an input vector and $y_t \in \mathbb{R}$ is the desired output. The goal is to minimize the sum-of-squares error function given by

$$J(\boldsymbol{w}) = \frac{1}{2} \sum_{t=1}^{N} \|\hat{\boldsymbol{y}}(\boldsymbol{u}_t, \boldsymbol{w}) - \boldsymbol{y}_t\|^2 \tag{B.5}$$

where $\|\cdot\|$ denotes the ℓ_2 norm. In practice, the non-linearity of the network function $y(\boldsymbol{u}_t, \boldsymbol{w})$ causes the error $J(\boldsymbol{w})$ to be non-convex and local maxima of the likelihood may be found, corresponding to a local minima of the error function (Bishop, 2006). The choice of the output unit activation and matching error functions depends on the type of problem being solved as (Yegnanarayana, 2009): (1) Regression problems: linear outputs and sum-of-squares error; (2) Binary classification: logistic sigmoid outputs and cross-entropy error; (3) Multi-class classification: soft-max outputs with the corresponding multi-class cross-entropy error function. The Equation (B.5) can be solved by finding a weight vector $\boldsymbol{w}$ that minimizes the chosen function $J(\boldsymbol{w})$ as follows

$$\boldsymbol{w}_{\tau+1} = \boldsymbol{w}_\tau + \Delta \boldsymbol{w}_\tau \tag{B.6}$$

The previous expression can be solved by using gradient information, meaning that $\nabla J(\boldsymbol{w})$ is evaluated with a new weight vector $\boldsymbol{w}_{\tau+1}$ at each iteration step τ. The training of neural networks is usually done by using gradient information forms (Hughes et al., 2018). Thus, the weight vector $\boldsymbol{w}$ can be updated with the following rule

$$\boldsymbol{w}_{\tau+1} = \boldsymbol{w}_\tau - \eta \nabla J(\boldsymbol{w}_\tau) \tag{B.7}$$

where $\eta > 0$ is the learning rate or step-size parameter. Thus, at each step, the

weight vector is moved in the direction of the greatest rate of decrease of the error function, which is why this approach is also known as gradient or steepest descent. In addition, as the whole training set is required to evaluate ∇J, these techniques are called batch methods (Wilson and Martinez, 2003). In practice, with the aim of finding a solution, it may be necessary to run a gradient-based algorithm multiple times by using a different randomly chosen starting point (Nielsen, 2015). There is an online version of gradient descent that has proven useful when training neural networks on large data sets (Li and Liang, 2018). These methods, also known as sequential gradient descent or stochastic gradient descent, make an update to the weight vector based on one data point at a time (see Equation (B.8))

$$\boldsymbol{w}_{\tau+1} = \boldsymbol{w}_{\tau} - \eta \nabla J_t(\boldsymbol{w}_{\tau}) \tag{B.8}$$

The previous expression, unlike batch methods, handles redundancy in the data more efficiently (Duchi and Singer, 2009). In general, the training algorithms of neural networks involve an iterative procedure to minimize an error function, where the weights of the network are adjusted sequentially. There are two different stages at each iteration (Lee et al., 2016): (1) first stage, where the derivatives of the error function with respect to the weights are computed; (2) second stage, where the weights are adjusted using the derivatives of the previous stage.

B.1.1 Multilayer Feed-forward Neural Networks

The feed-forward neural network is an artificial NN where the connections between the nodes do not form a cycle (Fine, 2006). That is, the information moves in only one direction (forward) from the input layer to the output layer. This class of networks have multiple layers of neurons that are interconnected in a feed-forward way. For example, as seen in Figure B.2, there is an input layer of source neurons, multiple hidden layers, and an output layer of neurons. The hidden layers are useful to extract relevant features contained in the input data. The training of an multi-layer perceptron network is usually done by using a back-propagation algorithm that consists of two stages (Sandberg et al., 2001): (1) *Forward phase*. The free parameters of the network are fixed and the input signal is propagated through the network from one layer to the next one. This stage finishes with the computation of the following error signal $e_t = y_t - \hat{y}_t$, where y_t is the desired output and $\hat{y}_t$ is the actual output produced by the network in response to the

input vector $\boldsymbol{u}_t$; (2) *Backward phase*. Here, the error e_t is propagated through the network in a backward direction, which is used to adjust the free parameters of the network.

The back-propagation learning may be implemented as follows: (1) *Sequential mode*, where the parameter adjustments are made on an example-by-example basis (Jin et al., 2019). (2) *Batch mode*, where the parameter adjustments are made on an epoch-by-epoch basis, meaning that the parameters are updated once the entire set of training examples have been considered (Sim et al., 2019).

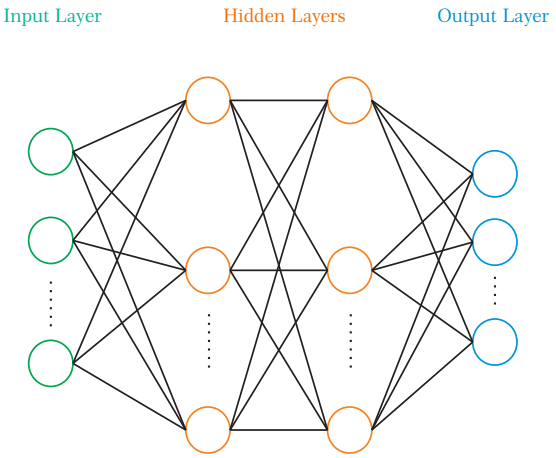


Figure B.2: Multilayer Feed-forward Neural Network architecture.

B.1.2 Convolutional Neural Networks

The convolutional neural network (CNN) is an artificial NN for image recognition (Krizhevsky et al., 2012), where each image is divided into topological portions to find particular patterns during learning (see Figure B.3). In practice, each image is represented as a three-dimensional matrix, containing information about its width, height, and color. Then, this information is putted on convolution with the filter set, meaning that the inner product of each filter and the input is computed along the image. This produces a set of activation maps for each filter that are superposed to get an output volume, which is also known as convolutional layer (Klein et al., 2015).

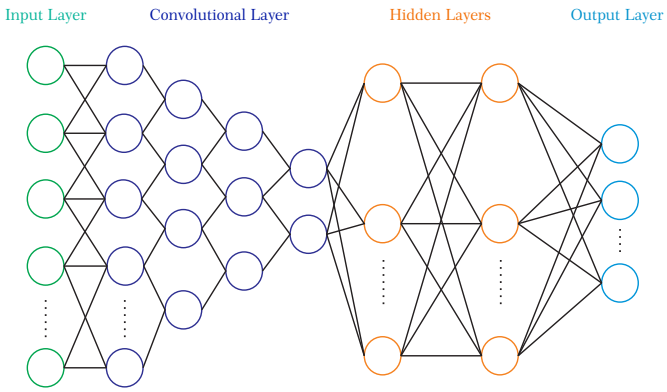


Figure B.3: Convolutional Neural Network architecture.

B.1.3 Recurrent Neural Networks

The recurrent neural network (RNN) is an artificial NN that contains at least one feedback connection (see Figure B.4), allowing information to flow in a loop (Gao et al., 2019). This enables the networks to do temporal processing and learn sequences, e.g., sequence recognition or temporal prediction. There are three main RNN architectures (Zaccone et al., 2017): (1) standard multilayer perceptron with added loops, which uses the non-linear mapping capabilities of multilayer perceptron and also have some form of memory; (2) uniform structures, where every neuron is connected to all the others and may also have stochastic activation functions; (3) simple architectures, where learning can be performed using similar gradient descent procedures to those used in the back-propagation algorithm for feed-forward neural networks.

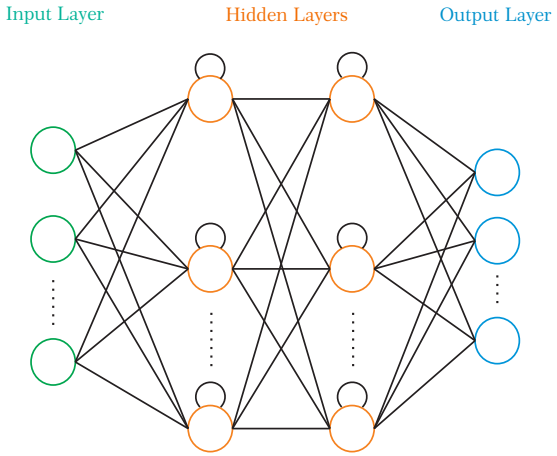


Figure B.4: Recurrent Neural Network architecture.

B.2 Linear Adaptive Filters

Suppose the goal is to learn a continuous input-output mapping $f : \mathcal{U} \rightarrow \mathbb{R}$ based on a sequence of input-output examples $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, where $\mathcal{U} \subset \mathbb{R}^M$ is the input domain, $\boldsymbol{u}_t \in \mathbb{R}^M$ is an input vector, and $y_t \in \mathbb{R}$ is the desired output. The linear adaptive filers adjust their free parameters automatically in response to statistical variations in the environment. These filters, as seen in Figure B.5, have an input signal vector $\boldsymbol{u}_t$ that is applied to the filter at time t. This produces the actual response $\hat{y}_t$, which is compared with a desired output y_t to compute the error signal e_t. Then, the error e_t is used to adjust the weights $\mathbf{w}_{t-1}$ by an incremental amount $\Delta\mathbf{w}_t$. The new weight of the filter can be computed as $\mathbf{w}_t = \mathbf{w}_{t-1} + \Delta\mathbf{w}_t$. The previous adaptive filtering process is continuously repeated until the filter reaches a condition.

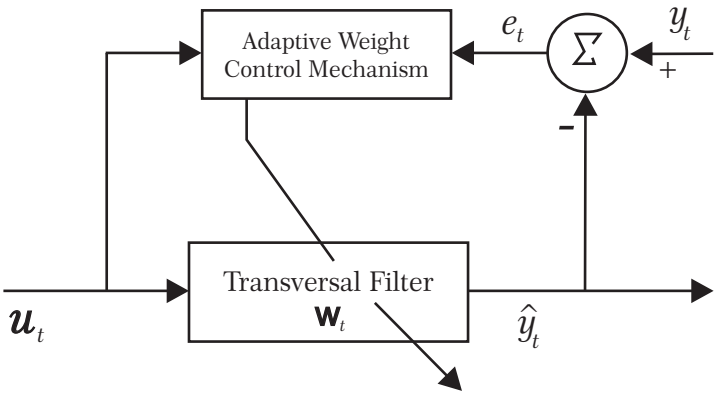


Figure B.5: Linear adaptive filter structure.

B.2.1 Least Mean Square Algorithm

The simplest and most commonly used form of an adaptive filtering algorithm is least mean square (LMS). The LMS algorithm operates by minimizing the instantaneous cost function $J_t = \frac{1}{2}e_t^2$, where the prediction error is $e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{u}_t$. Then, following the instantaneous version of gradient descent, the adjustment $\Delta\mathbf{w}_t$ is computed as follows

$$\Delta\mathbf{w}_t = \eta e_t \boldsymbol{u}_t \tag{B.9}$$

where $\eta \in \mathbb{R}^+$ is the step-size parameter and $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta e_t \boldsymbol{u}_t$. Thus, the LMS algorithm assumes a linear model and uses the following procedure:

$$\begin{cases} \mathbf{w}_0 = 0 \\ e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{u}_t \\ \mathbf{w}_t = \mathbf{w}_{t-1} + \eta e_t \boldsymbol{u}_t \end{cases} \tag{B.10}$$

where $\mathbf{w}_t$ is the estimate of the optimal weight at iteration t. However, if the mapping between y and $\boldsymbol{u}$ is highly nonlinear, then LMS is likely to perform poorly (Zhang et al., 2017b).

B.2.2 Recursive Least Squares Algorithm

The recursive least squares (RLS), like the least mean square (LMS) algorithm, uses an error-correction learning strategy; the key difference in the two algorithms is that RLS aims to minimize the sum of squared estimation errors up to and including the time t. Thus, given the sequence of training data $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$, the RLS algorithm estimates the weight $\mathbf{w}_{t-1}$ by minimizing the following cost function

$$J_t = \sum_{j=1}^{t} \left(y_j - \mathbf{w}^\top \boldsymbol{u}_j\right)^2 \tag{B.11}$$

The state of the adaptive filter, which is represented by the estimation of $\mathbf{w}_{t-1}$, provides a summary of all data processed by the RLS algorithm. The adjustment of $\mathbf{w}$ at time t can be computed as

$$\Delta \mathbf{w}_t = \mathbf{k}_t e_t \tag{B.12}$$

where $\mathbf{k}_t = \mathbf{P}_t \boldsymbol{u}_t$ is the gain vector of the RLS algorithm, $e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{u}_t$ is the prediction error, and $\mathbf{P}_t$ is defined as

$$\mathbf{P}_t = \mathbf{R}_t^{-1} \tag{B.13}$$

being $\mathbf{R}_t = \sum_{j=1}^{t} \boldsymbol{u}_j \boldsymbol{u}_j^\top$ the time-averaged correlation matrix. Thus, $\mathbf{w}_t$ is updated using the following sequential rule

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{k}_t e_t \tag{B.14}$$

Finally, the RLS algorithm uses the following procedure

$$\begin{cases} \mathbf{w}_0 = 0 \\ \mathbf{P}_0 = 0 \\ r_t = 1 + \boldsymbol{u}_t^\top \mathbf{P}_{t-1} \boldsymbol{u}_t \\ \mathbf{k}_t = \mathbf{P}_{t-1} \boldsymbol{u}_t / r_t \\ e_t = y_t - \boldsymbol{u}_t^\top \mathbf{w}_{t-1} \\ \mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{k}_t e_t \\ \mathbf{P}_t = \left[\mathbf{P}_{t-1} - \mathbf{k}_t \mathbf{k}_t^\top r_t \right] \end{cases} \tag{B.15}$$

where $\mathbf{P}_t \in \mathbb{R}^{M \times M}$ is the state-error-correlation matrix and M is the dimensionality of the input vector $\boldsymbol{u}$.

B.2.3 Affine Projection Algorithm

The affine projection algorithm (APA) uses the online nature of least mean square (LMS), while reduces the gradient noise by using multiple samples. The APA algorithm aims to minimize the following cost function:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \mathbb{E}\left\{ |y - \mathbf{w}^\top \boldsymbol{u}|^2 \right\} \tag{B.16}$$

being $\mathbb{E}\{\cdot\}$ the expectation operator. Then, when the gradient descent is applied to $\mathbf{w}$, the following sequential rule is obtained

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta \mathbf{U}_t \left[\boldsymbol{y}_t - \mathbf{U}_t^\top \mathbf{w}_{t-1} \right] \tag{B.17}$$

where $\mathbf{U}_t = [u_{t-K+1}, \ldots, \boldsymbol{u}_t]_{M \times K}$, $\boldsymbol{y}_t = [y_{t-K+1}, \ldots, y_t]^\top$, and K denotes the most recent inputs and observations.

B.3 Kernel Adaptive Filtering

These filters are non-linear approximators that combine the universal approximation property of neural networks and the convex optimization of linear adaptive filters (Liu et al., 2010). The goal, as seen in Figure B.6, is to learn the underlying function $y = f(\boldsymbol{u})$ from the given input-output samples $\mathcal{T} = \{\boldsymbol{u}_t, y_t : t \in [1, N]\}$.

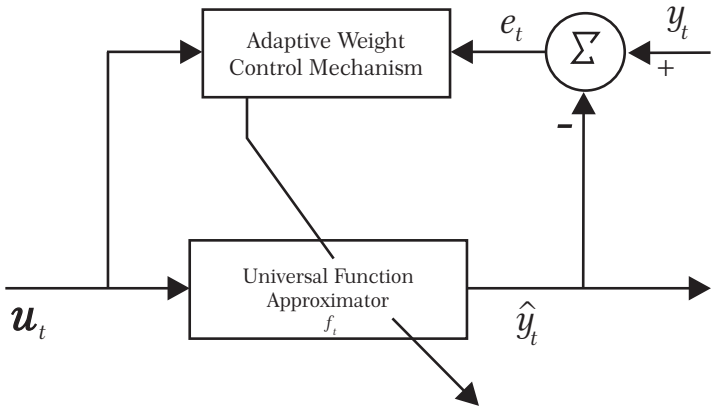


Figure B.6: Non-linear adaptive filter structure.

In kernel adaptive filtering (KAF), the underlying function f will be a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, which is isometric-isomorphic to a high dimensional feature space (Schölkopf et al., 2018). In addition, according to Mercer's theorem (Sun, 2005), a Mercer kernel $\kappa_\sigma : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ induces a mapping $\varphi : \mathcal{U} \to \mathcal{F}$ as follows:

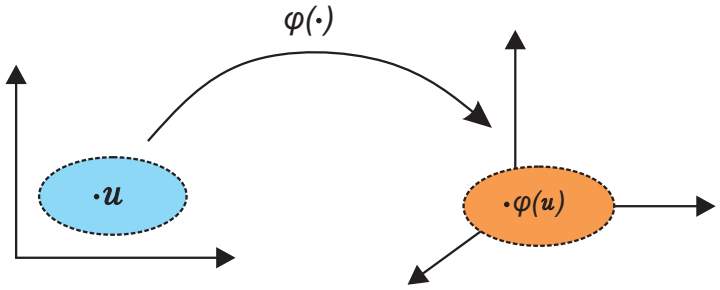


Figure B.7: Non-linear mapping from the input space to the feature space.

This allows the inner products to be calculated in the feature space, using the kernel trick $\varphi(\boldsymbol{u}_t)^\top \varphi(\boldsymbol{u}^*) = \kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}^*)$, being $\boldsymbol{u}^*$ a new input vector. The most

widely used Mercer kernel is the Gaussian function,

$$\kappa_\sigma\left(\boldsymbol{u}_t, \boldsymbol{u}^*\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left\|\boldsymbol{u}_t - \boldsymbol{u}^*\right\|^2}{2\sigma^2}\right) \tag{B.18}$$

where $\|\cdot\|$ denotes the ℓ_2 norm and $\sigma \in \mathbb{R}^+$ is the kernel-size (or bandwidth) that controls the mapping smoothness (Chen et al., 2013b). The function $f \in \mathcal{H}$ and a high-dimensional weight vector $\boldsymbol{\Omega} \in \mathcal{F}$ can be computed in the hypothesis and feature spaces as (Liu et al., 2008):

$$\min_{f \in \mathcal{H}} \sum_{t=1}^{N} \left(y_t - f(\boldsymbol{u}_t)\right)^2 + \lambda \left\|f\right\|_{\mathcal{H}}^2 \tag{B.19a}$$

$$\min_{\boldsymbol{\Omega} \in \mathcal{F}} \sum_{t=1}^{N} \left(y_t - \boldsymbol{\Omega}^\top \varphi(\boldsymbol{u}_t)\right)^2 + \lambda \left\|\boldsymbol{\Omega}\right\|_{\mathcal{F}}^2 \tag{B.19b}$$

where $\lambda \geq 0$ is the regularization factor that controls the solution smoothness and $\|\cdot\|_{\mathcal{H}}$ denotes the norm in $\mathcal{H}$. The solution of the above equations in a batch approach is computationally expensive, as the dimension of the Gram matrix equals the number of input patterns. In this sense, KAFs provide an efficient alternative that finds the solution in an online sequential way (Chen et al., 2016a).

B.3.1 Reproducing Kernel Hilbert Spaces

A Hilbert space is a linear, complete, and normed space endowed with an inner product (Halmos, 2012). This can be seen as a generalization of the two-dimensional (or three-dimensional) Euclidean plane to spaces with infinite number of dimensions. The reproducing kernel Hilbert space (RKHS) is a special Hilbert space associated with a kernel κ_σ (Berlinet and Thomas-Agnan, 2011). More formally, given the Hilbert space $\mathcal{H}$ of real-valued functions defined on a set $\mathcal{U}$ with an inner product $\langle \cdot, \cdot \rangle$ and a real-valued bivariate function $\kappa_\sigma\left(\boldsymbol{u}, \boldsymbol{u}^*\right)$ on $\mathcal{U} \times \mathcal{U}$. The function $\kappa_\sigma\left(\boldsymbol{u}, \boldsymbol{u}^*\right)$ is non-negative definite if for any $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N\} \subset \mathcal{U}$ and for any not all zero corresponding real numbers $\{\alpha_1, \alpha_2, \ldots, \alpha_N\} \subset \mathbb{R}$ the following condition is met,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \kappa_\sigma(\boldsymbol{u}_i, \boldsymbol{u}_j) \geq 0 \tag{B.20}$$

The previous non-negative function $\kappa_\sigma(\boldsymbol{u}, \boldsymbol{u}^*)$ is considered a reproducing kernel if there exists a uniquely determined (possibly infinite-dimensional) Hilbert space $\mathcal{H}$ consisting of functions on $\mathcal{U}$ such that (Principe, 2010): *i)* $\forall \boldsymbol{u} \in \mathcal{U}$, $\kappa_\sigma(\cdot, \boldsymbol{u}) \in \mathcal{H}$; *ii)* $\forall \boldsymbol{u} \in \mathcal{U}$, $\forall f \in \mathcal{H}$, $f(\boldsymbol{u}) = \langle f, \kappa_\sigma(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}}$. The first property maps each point of the input space into a function in the RKHS, while the second property is also known as the reproducing property of $\kappa_\sigma(\boldsymbol{u}, \boldsymbol{u}^*)$ in $\mathcal{H}$. Thus, when using kernel-based learning algorithms, the linear algorithm is implicitly executed in kernel feature space, while the data and operations are done in the input space.

B.3.2 Kernel Least Mean Square Algorithm

The kernel least mean square (KLMS) algorithm, with the aim of overcoming the limitation of linearity in least mean square (see Appendix B.2.1), employs a kernel-induced mapping $\varphi : \mathcal{U} \to \mathcal{F}$ to transform the input $\boldsymbol{u}_t$ into a high-dimensional feature space $\mathcal{F}$. Thus, when the LMS algorithm is applied to the new example sequence $\{\varphi(\boldsymbol{u}_t), y_t\}$, the following expression is obtained in the feature space $\mathcal{F}$,

$$\begin{cases} \boldsymbol{\Omega}_0 = 0 \\ e_t = y_t - \boldsymbol{\Omega}_{t-1}^\top \varphi(\boldsymbol{u}_t) \\ \boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \eta e_t \varphi(\boldsymbol{u}_t) \end{cases} \tag{B.21}$$

where e_t is the prediction error, $\boldsymbol{\Omega}_t$ denotes the estimation of the weight vector in $\mathcal{F}$, and $\eta \in \mathbb{R}^+$ is the step-size parameter. The weight $\boldsymbol{\Omega}_t$ can be expressed as a linear combination of all previous inputs weighted by the prediction errors as follows (Chen et al., 2012),

$$\begin{aligned} \boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-1} + \eta e_t \varphi(\boldsymbol{u}_t) \\ \boldsymbol{\Omega}_t &= [\boldsymbol{\Omega}_{t-2} + \eta e_{t-1} \varphi(\boldsymbol{u}_{t-1})] + \eta e_t \varphi(\boldsymbol{u}_t) \\ \boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-2} + \eta \left[e_{t-1} \varphi(\boldsymbol{u}_{t-1}) + e_t \varphi(\boldsymbol{u}_t) \right] \\ \boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_0 + \eta \sum_{j=1}^{t} e_j \varphi(\boldsymbol{u}_j) \\ \boldsymbol{\Omega}_t &= \eta \sum_{j=1}^{t} e_j \varphi(\boldsymbol{u}_j) \end{aligned}$$

Thus, when a new input vector $\boldsymbol{u}^* \in \mathbb{R}^M$ arrives in the system, the output can be computed as

$$\begin{aligned}
\boldsymbol{\Omega}_t^\top \varphi(\boldsymbol{u}^*) &= \left[\eta \sum_{j=1}^{t} e_j \varphi(\boldsymbol{u}_j)^\top\right] \varphi(\boldsymbol{u}^*) \\
\boldsymbol{\Omega}_t^\top \varphi(\boldsymbol{u}^*) &= \eta \sum_{j=1}^{t} e_j \left[\varphi(\boldsymbol{u}_j)^\top \varphi(\boldsymbol{u}^*)\right] \\
\boldsymbol{\Omega}_t^\top \varphi(\boldsymbol{u}^*) &= \eta \sum_{j=1}^{t} e_j \kappa_\sigma\left(\boldsymbol{u}_j, \boldsymbol{u}^*\right)
\end{aligned}$$

Finally, the KLMS algorithm estimates the input-output mapping f using the following sequential learning rule in the original space (Li and Príncipe, 2017):

$$\begin{cases} f_0 = 0 \\ e_t = y_t - f_{t-1}(\boldsymbol{u}_t) \\ f_t = f_{t-1} + \eta e_t \kappa_\sigma(\boldsymbol{u}_t, \cdot) \end{cases} \tag{B.22}$$

where f_t denotes the learned mapping at iteration t and $\kappa_\sigma(\cdot,\cdot) \in \mathbb{R}^+$ is a Mercer kernel with a kernel-size $\sigma \in \mathbb{R}^+$ that controls the mapping smoothness (Chen et al., 2013b). This sequential rule produces a growing radial-basis function network by allocating a new kernel unit for every new example with $\boldsymbol{u}_t$ as the center and ηe_t as its coefficient, which poses time-space complexity issues for continuous adaptation scenarios (Liu et al., 2010). In addition, there remain three main challenges in KAF algorithms: selection an appropriate Mercer kernel, determination of kernel parameter and tuning of the step-size (Chen et al., 2016a).

B.3.3 Kernel Recursive Least Squares Algorithm

The kernel recursive least squares (KRLS) is the recursive least squares (RLS) algorithm (see Appendix B.2.2) applied to the example sequence $\{\varphi(\boldsymbol{u}_t), y_t\}$, where the aim is to minimize the following expression (Engel et al., 2004)

$$\min_{\boldsymbol{\Omega} \in \mathcal{F}} \sum_{j=1}^{t} \left(y_t - \boldsymbol{\Omega}^\top \varphi(\boldsymbol{u}_j)\right)^2 + \lambda \left\|\boldsymbol{\Omega}\right\|_{\mathcal{F}}^2 \tag{B.23}$$

The KRLS algorithm, which assumes a radial basis function network structure at each iteration, sequentially estimates the input-output mapping f as follows (Zhou et al., 2017):

$$\begin{cases} \mathbf{Q}_1 = (\lambda + \kappa_\sigma(\boldsymbol{u}_1, \boldsymbol{u}_1))^{-1} \\ \boldsymbol{\alpha}_1 = \mathbf{Q}_1 y_1 \\ \mathbf{h}_t = [\kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}_1), \ldots, \kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}_{t-1})]^\top \\ \mathbf{z}_t = \mathbf{Q}_{t-1}\mathbf{h}_t \\ r_t = \lambda + \kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}_t) - \mathbf{z}_t^\top \mathbf{h}_t \\ \mathbf{Q}_t = r_t^{-1} \begin{bmatrix} \mathbf{Q}_{t-1} r_t + \mathbf{z}_t \mathbf{z}_t^\top & -\mathbf{z}_t \\ -\mathbf{z}_t^\top & 1 \end{bmatrix} \\ e_t = y_t - \mathbf{h}_t^\top \boldsymbol{\alpha}_{t-1} \\ \boldsymbol{\alpha}_t = \begin{bmatrix} \boldsymbol{\alpha}_{t-1} - \mathbf{z}_t r_t^{-1} e_t \\ r_t^{-1} e_t \end{bmatrix} \end{cases} \tag{B.24}$$

Thus, given a new input $\boldsymbol{u}^* \in \mathbb{R}^M$, the following expression is used to compute the output of the system

$$f(\boldsymbol{u}^*) = \sum_{j=1}^{t} \boldsymbol{\alpha}_{t,j} \kappa_\sigma(\boldsymbol{u}_j, \boldsymbol{u}^*) \tag{B.25}$$

B.3.4 Kernel Affine Projection Algorithm

The kernel affine projection (KAPA) is an affine projection (APA) algorithm (see Appendix B.2.3) in feature space. The KAPA algorithm, unlike KLMS, does not uses the instantaneous values for approximating the mean squared error (MSE) criterion. In particular, KAPA replaces the MSE by an approximation from the L most recent errors. More formally, let $\boldsymbol{e}_t = [e_{t,t-L+1}, e_{t,t-L+2}, \ldots, e_{t,t}]^\top$, where $e_{t,j}$ is the prediction error with hypothesis f_{t-1} and input-output pair $(\boldsymbol{u}_j, y_j)$, i.e., $e_{t,j} = y_j - f_{t-1}(\boldsymbol{u}_j)$. The KAPA algorithm estimates the weight

vector $\boldsymbol{\Omega}$ as follows:

$$\begin{aligned}
\boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-1} - \eta \frac{\partial}{\partial \boldsymbol{\Omega}_{t-1}} \left(\frac{1}{2} \left\| \boldsymbol{e}_t^2 \right\| \right) \\
\boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-1} - \eta \left[\frac{\partial}{\partial \boldsymbol{\Omega}_{t-1}} \boldsymbol{e}_t^\top \right] \boldsymbol{e}_t \\
\boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-1} - \eta \left[\frac{\partial}{\partial \boldsymbol{\Omega}_{t-1}} \left(\boldsymbol{y}_t - \boldsymbol{\Phi}_t^\top \boldsymbol{\Omega}_{t-1}^\top \right) \right] \boldsymbol{e}_t \\
\boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{t-1} + \eta \boldsymbol{\Phi}_t^\top \boldsymbol{e}_t
\end{aligned}$$

such that $\boldsymbol{y}_t = [y_{t-L+1}, y_{t-L+2}, \ldots, y_t]^\top$ and $\boldsymbol{\Phi}_t = [\varphi(\boldsymbol{u}_{t-L+1}), \varphi(\boldsymbol{u}_{t-L+2}), \ldots, \varphi(\boldsymbol{u}_t)]$. The KAPA algorithm estimates the input-output mapping f as

$$f_t = f_{t-1} + \eta \mathbf{K}_t \boldsymbol{e}_t \tag{B.26}$$

where $\mathbf{K}_t = [\kappa_\sigma(\boldsymbol{u}_{t-L+1}, \cdot), \kappa_\sigma(\boldsymbol{u}_{t-L+2}, \cdot), \ldots, \kappa_\sigma(\boldsymbol{u}_t, \cdot)]$. The learned mapping by KAPA is computed using the following expression

$$f_t(\cdot) = \sum_{j=1}^{t} \boldsymbol{\alpha}_{t,j} \kappa_\sigma(\boldsymbol{u}_j, \cdot) \tag{B.27}$$

being $\boldsymbol{\alpha}_t = [\boldsymbol{\alpha}_{t,1}, \boldsymbol{\alpha}_{t,2}, \ldots, \boldsymbol{\alpha}_{t,t}]^\top$ a coefficient vector, which is related to the errors at each sample. The KAPA algorithm, unlike KLMS, updates the most recent coefficients.

Appendix C

Information Theory

Information theory, which has proven useful when analyzing data of complex systems (Guo et al., 2018), studies how to optimally quantify and transmit messages through noisy channels (Pierce, 2012). This theory plays a central role in the design of communication systems, signal processing, and data compression (Csiszar and Körner, 2011). In a communication system, the information is exchanged between two stations called transmitter and receiver (Lin et al., 2010). First, the information is generated, processed, shaped, and encoded on the transmitter side. Then, the information is transmitted over a channel, which usually adds noise and distortion to the original message. Finally, the receiver gets a distorted or corrupted version of the information sent by the transmitter. The problem is how to recover the original information from the corrupted message, which is normally modeled as stationary additive Gaussian noise with a given variance (Zhang and Chan, 2015). Information theory addresses the previous problem by studying the statistical structure of messages and analyzing the noise levels in the channel. This is done by using statistical descriptors such as entropy and mutual information. However, knowledge of the data probability density function (PDF) is a necessary first step to estimate these statistical descriptors, which in practice means either assume an analytical model or to use a non-parametric estimator (Deng et al., 2016). In this section, we briefly introduce the definitions of entropy, mutual information, and divergence.

C.1 Entropy

Given a random variable x with a set of possible outcomes $S_X = \{s_1, \ldots, s_N\}$ having probabilities $p_X = \{p_1, \ldots, p_N\}$, where $p(x = s_k) = p_k$, $p_k \geq 0$, and $\sum_{x \in S_x} p(x) = 1$. The uncertainty of the random variable X, as seen in Equation (C.1), can be computed as the sum of the uncertainty in each message weighted by the probability of each message (Wu et al., 2013b),

$$H(X) = -\sum_k p(x_k) \log p(x_k) \tag{C.1}$$

The previous expression, which is also known as Shannon or information entropy, measures the uncertainty and assumes that for $p(x_k) = 0$ the condition $p(x_k) \log p(x_k) = 0$ is met. The information entropy, as seen in Equation (C.1), has the same form of physical entropy. However, the former is a property of the probability mass function (PMF) (Baratpour and Bami, 2012), while the latter is a property of the state of the physical system. The following are some of the properties of Shannon entropy (Principe, 2010): *i)* $H(p, 1-p)$ is a continuous function of p; *ii)* $H(p_1, p_2, \ldots, p_N)$ is a symmetric function of p_k; *iii)* $H(p_1 \cdot p_2) = H(p_1) + H(p_2)$ for independent events.

C.2 Mutual Information

Let us consider the transmitted message $X = \{x_k\}_{k=1}^N$ and the received message $Y = \{y_k\}_{k=1}^N$ with a probability distribution $p(X, Y)$ over the product space. The mutual information between X and Y, which can be seen as the total decrease in uncertainty in X by observing Y (Haghighat et al., 2011), is computed as follows:

$$I(X, Y) = \sum_i \sum_k p(x_k, y_i) \log \frac{p(x_k|y_i)}{p(x_k)} \tag{C.2a}$$

$$I(X, Y) = \sum_i \sum_k p(x_k, y_i) \log \frac{p(x_k, y_i)}{p(x_k)p(y_i)} \tag{C.2b}$$

where $p(X, Y)$ is the joint mass function of X and Y. In addition, the joint entropy $H(X, Y)$ of a pair of random variables (X, Y) and the conditional entropy

$H(Y|X)$ of Y given X are computed using the following expressions:

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y) \tag{C.3}$$

$$H(Y|X) = -\sum_x \sum_y p(x,y) \log p(x|y) \tag{C.4}$$

The mutual information, which is always greater than zero, quantifies the intersection of $H(X)$ and $H(Y)$. That is, this statistical descriptor quantifies the reduction of uncertainty in X after observing Y. Thus, combining Equations (C.2) to (C.4), the mutual information can also be expressed as,

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{C.5a}$$

$$I(X,Y) = H(X) - H(X|Y) \tag{C.5b}$$

$$I(X,Y) = H(Y) - H(Y|X) \tag{C.5c}$$

The mutual information $I(X,Y)$, as seen in Figure C.1, quantifies the intersection of the marginal entropies $H(X)$ and $H(Y)$,

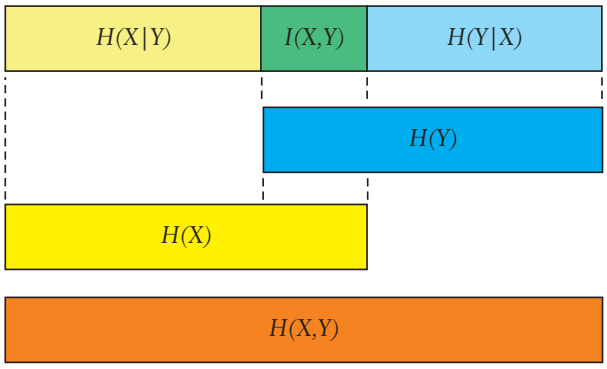


Figure C.1: Relationships between entropy and mutual information.

C.3 Divergence and Mutual Information

The Kullback–Leibler (KL) divergence, which measures the dissimilarity between two different probability densities $p(x)$ and $q(x)$, can be computed using the following expression,

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{C.6}$$

The KL divergence is not considered a distance measure, as it is not symmetric and does not meet the triangular inequality criteria (Moreno et al., 2004). Thus, the mutual information can be seen as a special case of the KL divergence. This is obtained when $p(x)$ is the joint probability density function of x and y, while $q(x)$ is the product of the marginals as follows:

$$D\Big(p(X,Y)||p(X)q(Y)\Big) = I(X,Y) \tag{C.7}$$

Appendix D

Information-Theoretic Learning

Information-theoretic learning (ITL) uses the general framework of information theory in the design of cost functions for adaptive systems (Giraldo and Principe, 2013). However, the main issue in ITL is how to estimate entropy and divergence directly from samples. In this section, using ITL approaches, we will introduce non-parametric estimators of entropy and divergence that can be derived directly from data.

D.1 Renyi Entropy

Let us assume a discrete probability distribution $P = \{p_1, p_2, \ldots, p_N\}$ fulfilling the conditions of $\sum_k p_k = 1$ and $p_k \geq 0$. The Renyi information of order α, which is also known as Renyi α entropy can be computed as

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \left(\sum_k p_k^\alpha \right) \tag{D.1}$$

where the information potential α allows to compute several measurements of uncertainty within a given distribution (Kapur, 1994). In Renyi entropy, unlike Shannon entropy, the logarithm appears outside the sum. This allows the use of ITL in machine learning problems where the probability density function of experimental data is not known (Principe, 2010). The following are some of the properties of Renyi entropy (Aczél and Daróczy, 1975; Xu, 1999):

- $H(p_1, \ldots, p_N)$ is a continuous function of all the probabilities p_k, meaning that small changes in the probability distribution produce small changes in

the entropy.

- $H(p_1, \ldots, p_N)$ is permutationally symmetric, indicating that the entropy will not change with the permutation of any p_k in the distribution.
- $H(1/n, \ldots, 1/n)$ is a monotonic increasing function of n, meaning that the uncertainty and entropy increase as the number of choices in an equiprobable distribution increases.
- Given two independent probability distributions $p = (p_1, p_2, \ldots, p_N)$ and $q = (q_1, q_2, \ldots, q_N)$, then $H(p \cdot q) = H(p) + H(q)$, where $p \cdot q$ the joint probability distribution.
- $H_\alpha(X)$ is non-negative.
- $H_\alpha(0, 1) = H_\alpha(1, 0) = 0$. The Renyi entropy is concave when $\alpha \leq 1$, while it is neither pure convex nor pure concave when $\alpha > 1$.
- $(\alpha - 1)H_\alpha(X)$ is a concave function of p_k.
- $H_\alpha(X)$ is a bounded, continuous and non-increasing function of α.
- $H_\alpha(A \cap B) = H_\alpha(A) - H_\alpha(B|A)$, where $\alpha \in \mathbb{R}$.

D.2 Quadratic Renyi Entropy

The quadratic Renyi entropy H_2 is obtained when $\alpha = 2$, which allows to implicitly use an Euclidean distance from the point $p(x)$ in the simplex to the origin of the space (Principe, 2010). This has proven useful to quantify diversity in econometric applications, as H_2 may be more suitable than Shannon entropy for entropy maximization (Hart, 1975; Huang and Kou, 2014). The quadratic Renyi entropy can be computed as follows:

$$H_2(X) = -\log\left(\sum_k p_k^2\right) \tag{D.2a}$$

$$H_2(X) = -\log\int p^2(x)dx \tag{D.2b}$$

where $p(x)$ is the continuous probability density function defined in $[0, 1]$. In practice, traditional approaches first estimate the probability density function

and then compute its entropy. In contrast, H_2 estimates entropy directly from samples rather than explicitly calculating the probability density function. The probability density function of N independent and identically distributed samples $\{x_1, x_2, \ldots, x_N\}$ can be computed using the kernel (Parzen) estimation as,

$$\hat{p}_X(x) = \frac{1}{N\sigma} \sum_{i=1}^{N} \kappa_\sigma \left(\frac{x - x_i}{\sigma} \right) \tag{D.3}$$

being $\kappa_\sigma(\cdot)$ a symmetric, continuous, and differentiable kernel function (Schölkopf et al., 2018). Thus, combining Equations (B.18), (D.2b) and (D.3), the following approximation of the quadratic Renyi entropy is obtained,

$$\begin{aligned}
\hat{H}_2(X) &= -\log \int_{-\infty}^{\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \kappa_\sigma(x - x_i) \right)^2 dx \\
\hat{H}_2(X) &= -\log \frac{1}{N^2} \int_{-\infty}^{\infty} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \kappa_\sigma(x - x_j) \cdot \kappa_\sigma(x - x_i) \right) dx \\
\hat{H}_2(X) &= -\log \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \int_{-\infty}^{\infty} \kappa_\sigma(x - x_j) \cdot \kappa_\sigma(x - x_i) dx \\
\hat{H}_2(X) &= -\log \left(\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \kappa_{\sigma\sqrt{2}}(x_j - x_i) \right)
\end{aligned} \tag{D.4}$$

Equation (D.4) is a major contribution of information-theoretic learning, as it estimates entropy directly from samples without explicitly computing the probability density function (Han et al., 2011).

D.3 Renyi Divergence and Mutual Information

Given the independent and identically distributed samples $\{x_p(1), \ldots, x_p(N)\}$ and $\{x_q(1), \ldots, x_q(N)\}$ drawn from $p(x)$ and $q(x)$, respectively. The Renyi divergence can be computed as follows (Principe, 2010):

$$\hat{D}_\alpha(p||q) = \frac{1}{\alpha - 1} \log \frac{1}{N} \sum_{j=1}^{N} \left(\frac{\sum_{i=1}^{N} \kappa_\sigma \Big(x_p(j) - x_p(i) \Big)}{\sum_{i=1}^{N} \kappa_\sigma \Big(x_q(j) - x_q(i) \Big)} \right)^{\alpha - 1} \tag{D.5}$$

The previous expression has the following properties: *i)* $\hat{D}_\alpha(p||q) \geq 0$, $\forall\ p$, q, $\alpha > 0$; *ii)* $\hat{D}_\alpha(p||q) = 0$ if $p(x) = q(x)\ \forall x \in \mathbb{R}$; *iii)* $\lim_{\alpha\to 1} \hat{D}_\alpha(p||q) = D(p||q)$, where $D(p||q)$ is the Kullback-Leibler divergence (see Equation (C.6)). Similarly, as proposed in Rényi (1976), the non-parametric estimator for Renyi mutual information is given as,

$$\hat{I}_\alpha(X) = \frac{1}{\alpha - 1} \log \frac{1}{N} \sum_{j=1}^{N} \left(\frac{\left(\frac{1}{N} \sum_{i=1}^{N} \prod_{o=1}^{n} \kappa_{\sigma o}\Big(x(j) - x(i) \Big) \right)}{\prod_{o=1}^{n} \left(\frac{1}{N} \sum_{i=1}^{N} \kappa_{\sigma o}\Big(x_o(j) - x_o(i) \Big) \right)} \right)^{\alpha - 1} \tag{D.6}$$

Bibliography

Abedin, M. Z., Guotai, C., Moula, F.-E., Azad, A. S., and Khan, M. S. U. (2019). Topological applications of multilayer perceptrons and support vector machines in financial decision support systems. *International Journal of Finance & Economics*, 24(1):474–507.

Aczél, J. and Daróczy, Z. (1975). On measures of information and their characterizations. *New York*, 168.

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147.

Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621.

Aitken, M., Cumming, D., and Zhan, F. (2015). High frequency trading and end-of-day price dislocation. *Journal of Banking & Finance*, 59:330–349.

Aldridge, I. (2013). *High-frequency trading: a practical guide to algorithmic strategies and trading systems*, volume 604. John Wiley & Sons.

Alexeev, V., Tapon, F., et al. (2014). The number of stocks in your portfolio should be larger than you think: Diversification evidence from five developed markets. *Journal of Investment Strategies*, 4:43–82.

Alhnaity, B. and Abbod, M. (2020). A new hybrid financial time series prediction model. *Engineering Applications of Artificial Intelligence*, 95:103873.

Almahdi, S. and Yang, S. Y. (2019). A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning. *Expert Systems with Applications*, 130:145–156.

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press.

Ameriks, J., Kézdi, G., Lee, M., and Shapiro, M. D. (2020). Heterogeneity in expectations, risk tolerance, and household stock shares: The attenuation puzzle. *Journal of Business & Economic Statistics*, 38(3):633–646.

An, S., Liu, W., and Venkatesh, S. (2007). Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154 – 2162. Part Special Issue on Visual Information Processing.

Antweiler, W. and Frank, M. Z. (2004). Is all that talk just noise? the information content of internet stock message boards. *The Journal of Finance*, 59(3):1259–1294.

Araya, I. A., Valle, C., and Allende, H. (2019). A multi-scale model based on the long short-term memory for day ahead hourly wind speed forecasting. *Pattern Recognition Letters*.

Ardia, D., Bluteau, K., and Rüede, M. (2019). Regime changes in bitcoin garch volatility dynamics. *Finance Research Letters*, 29:266–271.

Arévalo, R., García, J., Guijarro, F., and Peris, A. (2017). A dynamic trading rule based on filtered flag pattern recognition for stock market price forecasting. *Expert Systems with Applications*, 81:177–192.

Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE.

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128.

Azodi, C. B., Tang, J., and Shiu, S.-H. (2020). Opening the black box: Interpretable machine learning for geneticists. *Trends in Genetics*.

Babu, M. S., Geethanjali, N., and Satyanarayana, B. (2012). Clustering approach to stock market prediction. *International Journal of Advanced Networking and Applications*, 3(4):1281.

Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, 19(8):1165–1195.

Ballings, M., Van den Poel, D., Hespeels, N., and Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056.

Bao, W., Yue, J., and Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One*, 12(7):e0180944.

Baratpour, S. and Bami, Z. (2012). On the discrete cumulative residual entropy.

Battalio, R., Hatch, B., and Jennings, R. (2004). Toward a national market system for us exchange–listed equity options. *The Journal of Finance*, 59(2):933–962.

Baumeister, C. and Kilian, L. (2015). Forecasting the real price of oil in a changing world: a forecast combination approach. *Journal of Business & Economic Statistics*, 33(3):338–351.

Belciug, S. (2020). Logistic regression paradigm for training a single-hidden layer feedforward neural network. application to gene expression datasets for cancer research. *Journal of Biomedical Informatics*, 102:103373.

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

Bernal, A., Fok, S., and Pidaparthi, R. (2012). Financial market time series prediction with recurrent neural networks. *State College: Citeseer.[Google Scholar]*.

Beshears, J., Choi, J. J., Laibson, D., and Madrian, B. C. (2009). How does simplified disclosure affect individuals' mutual fund choices? Technical report, National Bureau of Economic Research.

Bezerra, P. C. S. and Albuquerque, P. H. M. (2017). Volatility forecasting via svr–garch with mixture of gaussian kernels. *Computational Management Science*, 14(2):179–196.

Bhuriya, D., Kaushal, G., Sharma, A., and Singh, U. (2017). Stock market predication using a linear regression. In *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 2, pages 510–513. IEEE.

Bielecki, T. R. and Rutkowski, M. (2013). *Credit risk: modeling, valuation and hedging*. Springer Science & Business Media.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bouallegue, K. (2017). A new class of neural networks and its applications. *Neurocomputing*, 249:28–47.

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

Buchanan, B. (2019). Artificial intelligence in finance. *London: The Alan Turing Institute*.

Cakır, E., Parascandolo, G., Heittola, T., Huttunen, H., and Virtanen, T. (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303.

Cao, J., Li, Z., and Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications*, 519:127–139.

Cao, L.-J. and Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518.

Caraiani, P. (2014). The predictive power of singular value decomposition entropy for stock market dynamics. *Physica A: Statistical Mechanics and its Applications*, 393:571–578.

Caraiani, P. (2018). Modeling the comovement of entropy between financial markets. *Entropy*, 20(6):417.

Cerchiello, P., Nicola, G., Ronnqvist, S., and Sarlin, P. (2017). Deep learning bank distress from news and numerical financial data. *ArXiv Preprint ArXiv:1706.09627*.

Cervelló-Royo, R., Guijarro, F., and Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Systems with Applications*, 42(14):5963–5975.

Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250.

Chaim, P. and Laurini, M. P. (2018). Volatility and return jumps in bitcoin. *Economics Letters*, 173:158–163.

Chan, W., Cheung, S. H., Zhang, L.-X., and Wu, K. (2008). Temporal aggregation of equity return time-series models. *Mathematics and Computers in Simulation*, 78(2-3):172–180.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. (2017). Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 77–87.

Chatzis, S. P., Siakoulis, V., Petropoulos, A., Stavroulakis, E., and Vlachogiannakis, N. (2018). Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112:353–371.

Chen, B., Hu, J., Pu, L., and Sun, Z. (2007). Stochastic gradient algorithm under (h, φ)-entropy criterion. *Circuits, Systems & Signal Processing*, 26(6):941–960.

Chen, B., Liang, J., Zheng, N., and Príncipe, J. C. (2016a). Kernel least mean square with adaptive kernel size. *Neurocomputing*, 191:95–106.

Chen, B., Xing, L., Zheng, N., and Príncipe, J. C. (2018a). Quantized minimum error entropy criterion. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1370–1380.

Chen, B., Yuan, Z., Zheng, N., and Príncipe, J. C. (2013a). Kernel minimum error entropy algorithm. *Neurocomputing*, 121:160–169.

Chen, B., Zhao, S., Zhu, P., and Príncipe, J. C. (2012). Quantized kernel least mean square algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 23(1):22–32.

Chen, B., Zhao, S., Zhu, P., and Principe, J. C. (2013b). Quantized kernel recursive least squares algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 24(9):1484–1491.

Chen, B., Zhu, P., and Principe, J. C. (2011). Survival information potential: A new criterion for adaptive system training. *IEEE Transactions on Signal Processing*, 60(3):1184–1194.

Chen, H., Xiao, K., Sun, J., and Wu, S. (2017). A double-layer neural network framework for high-frequency forecasting. *ACM Transactions on Management Information Systems (TMIS)*, 7(4):1–17.

Chen, J. (2019). *Text-classification methods and the mathematical theory of Principal Components*. PhD thesis, Georgia Institute of Technology.

Chen, K., Zhou, Y., and Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2823–2824. IEEE.

Chen, N., Ribeiro, B., and Chen, A. (2016b). Financial credit risk assessment: a recent review. *Artificial Intelligence Review*, 45(1):1–23.

Chen, R.-Y. (2018). A traceability chain algorithm for artificial neural networks using T–S Fuzzy cognitive maps in blockchain. *Future Generation Computer Systems*, 80:198–210.

Chen, T.-L. (2011). Forecasting the taiwan stock market with a stock trend recognition model based on the characteristic matrix of a bull market. *African Journal of Business Management*, 5(23):9947–9960.

Chen, T.-l. and Chen, F.-y. (2016). An intelligent pattern recognition model for supporting investment decisions in stock market. *Information Sciences*, 346:261–274.

Chen, Y., Zhang, C., He, K., and Zheng, A. (2018b). Multi-step-ahead crude oil price forecasting using a hybrid grey wave model. *Physica A: Statistical Mechanics and its Applications*, 501:98–110.

Cheng, W., Rolls, E. T., Qiu, J., Yang, D., Ruan, H., Wei, D., Zhao, L., Meng, J., Xie, P., and Feng, J. (2018). Functional connectivity of the precuneus in

unmedicated patients with depression. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, 3(12):1040–1049.

Chi, J., Li, X., Wang, H., Gao, D., and Gerstoft, P. (2019). Sound source ranging using a feed-forward neural network trained with fitting-based early stopping. *The Journal of the Acoustical Society of America*, 146(3):EL258–EL264.

Chiu, D.-Y. and Chen, P.-J. (2009). Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm. *Expert Systems with Applications*, 36(2):1240–1248.

Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350.

Choi, K., Fazekas, G., Sandler, M., and Cho, K. (2017). Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE.

Chong, E., Han, C., and Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205.

Chung, J., Ahn, S., and Bengio, Y. (2016). Hierarchical multiscale recurrent neural networks. *ArXiv Preprint ArXiv:1609.01704*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2015). Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075.

Cochrane, J. H. (2009). *Asset pricing: Revised edition*. Princeton University Press.

Collins, D. and Biekpe, N. (2003). Contagion and interdependence in african stock markets. *South African Journal of Economics*, 71(1):181–194.

Creighton, J. and Zulkernine, F. H. (2017). Towards building a hybrid model for predicting stock indexes. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4128–4133. IEEE.

Csiszar, I. and Körner, J. (2011). *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press.

Cui, Y., Ahmad, S., and Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504.

Culkin, R. and Das, S. R. (2017). Machine learning in finance: The case of deep learning for option pricing. *Journal of Investment Management*, 15(4):92–100.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., and dos Reis Alves, S. F. (2017). Artificial neural network architectures and training processes. In *Artificial Neural Networks*, pages 21–28. Springer.

Dai, W., Wu, J.-Y., and Lu, C.-J. (2012). Combining nonlinear independent component analysis and neural network for the prediction of asian stock market indexes. *Expert Systems with Applications*, 39(4):4444–4452.

Das, S., Behera, R. K., Rath, S. K., et al. (2018). Real-time sentiment analysis of twitter streaming data for stock prediction. *Procedia Computer Science*, 132:956–964.

Dash, R. and Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, 2(1):42–57.

De Faria, E., Albuquerque, M. P., Gonzalez, J., Cavalcante, J., and Albuquerque, M. P. (2009). Predicting the brazilian stock market through neural networks and adaptive exponential smoothing methods. *Expert Systems with Applications*, 36(10):12506–12509.

De Ketelaere, B., Hubert, M., Rousseeuw, P., and Vranckx, I. (2018). Real-time outlier detection based on DetMCD. In *Book of Abstracts*, page 99.

Deng, Y., Bao, F., Deng, X., Wang, R., Kong, Y., and Dai, Q. (2016). Deep and structured robust information theoretic learning for image analysis. *IEEE Transactions on Image Processing*, 25(9):4209–4221.

Devi, B. U., Sundar, D., and Alli, P. (2013). An effective time series analysis for stock trend prediction using ARIMA model for nifty midcap-50. *International Journal of Data Mining & Knowledge Management Process*, 3(1):65.

Dey, C. and Gibbon, J. (2018). New development: Private finance over public good? questioning the value of impact bonds. *Public Money & Management*, 38(5):375–378.

Di Persio, L. and Honchar, O. (2017). Recurrent neural networks approach to the financial forecast of Google assets. *International Journal of Mathematics and Computers in Simulation*, 11:7–13.

Dias, J. G., Vermunt, J. K., and Ramos, S. (2015). Clustering financial time series: New insights from an extended hidden Markov model. *European Journal of Operational Research*, 243(3):852–864.

Diehl, E. (2016). Law 2: Know the assets to protect. In *Ten Laws for Security*, pages 45–66. Springer.

Dietterich, T. G. et al. (2002). Ensemble learning. *The Handbook of Brain Theory and Neural Networks*, 2:110–125.

Dimpfl, T. and Peter, F. J. (2014). The impact of the financial crisis on transatlantic information flows: An intraday analysis. *Journal of International Financial Markets, Institutions and Money*, 31:1–13.

Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Dixon, M., Klabjan, D., and Bang, J. H. (2015). Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. In *Proceedings of the 8th Workshop on High Performance Computational Finance*, pages 1–6.

Dixon, M., Klabjan, D., and Bang, J. H. (2017). Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*, 6(3-4):67–77.

Doucoure, B., Agbossou, K., and Cardenas, A. (2016). Time series prediction using artificial wavelet neural network and multi-resolution analysis: Application to wind speed data. *Renewable Energy*, 92:202–211.

Dowd, K. (2007). *Measuring market risk*. John Wiley & Sons.

Du, Z., Qin, M., Zhang, F., and Liu, R. (2018). Multistep-ahead forecasting of chlorophyll a using a wavelet nonlinear autoregressive network. *Knowledge-Based Systems*, 160:61–70.

Duan, S., Yu, S., Chen, Y., and Principe, J. C. (2019). On kernel method–based connectionist models and supervised deep learning without backpropagation. *Neural Computation*, 32(1):97–135.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934.

Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*. Oxford University Press.

Dutta, A., Bandopadhyay, G., and Sengupta, S. (2012). Prediction of stock performance in indian stock market using logistic regression. *International Journal of Business and Information*, 7(1).

Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. (2018). Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pages 121–138. Springer.

Engel, Y., Mannor, S., and Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285.

Enke, D. and Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4):927–940.

Erdogmus, D. and Principe, J. C. (2002a). An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems. *IEEE Transactions on Signal Processing*, 50(7):1780–1786.

Erdogmus, D. and Principe, J. C. (2002b). Generalized information potential criterion for adaptive system training. *IEEE Transactions on Neural Networks*, 13(5):1035–1044.

Erkaymaz, O., Ozer, M., and Perc, M. (2017). Performance of small-world feedforward neural networks for the diagnosis of diabetes. *Applied Mathematics and Computation*, 311:22–28.

Eshaghzadeh, A. and Hajian, A. (2018). 2d inverse modeling of residual gravity anomalies from simple geometric shapes using modular feed-forward neural network. *Annals of Geophysics*, 61(1):115.

Fan, A. and Palaniswami, M. (2001). Stock selection using support vector machines. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1793–1798. IEEE.

Fan, J., Han, F., and Liu, H. (2014a). Challenges of big data analysis. *National Science Review*, 1(2):293–314.

Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. (2014b). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Fang, M. and Chang, C.-L. (2017). Options pricing efficiency with fractional fast fourier transform. *DEStech Transactions on Environment, Energy and Earth Sciences*, (apeesd).

Faris, H., Aljarah, I., and Mirjalili, S. (2016). Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Applied Intelligence*, 45(2):322–332.

Faust, O., Ang, P. C. A., Puthankattil, S. D., and Joseph, P. K. (2014). Depression diagnosis support system based on eeg signal entropies. *Journal of Mechanics in Medicine and Biology*, 14(03):1450035.

Feng, G., Polson, N. G., and Xu, J. (2018). Deep factor alpha. *ArXiv Preprint ArXiv:1805.01104*, 2.

Fethi, M. D. and Pasiouras, F. (2010). Assessing bank efficiency and performance with operational research and artificial intelligence techniques: A survey. *European Journal of Operational Research*, 204(2):189–198.

Fine, T. L. (2006). *Feedforward neural network methodology*. Springer Science & Business Media.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Fortunato, M., Blundell, C., and Vinyals, O. (2017). Bayesian recurrent neural networks. *ArXiv Preprint ArXiv:1704.02798*.

Fraiwan, L., Lweesy, K., Khasawneh, N., Wenz, H., and Dickhaus, H. (2012). Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier. *Computer Methods and Programs in Biomedicine*, 108(1):10–19.

France, S. L. and Ghose, S. (2019). Marketing analytics: Methods, practice, implementation, and links to other fields. *Expert Systems with Applications*, 119:456–475.

Fu, X., Du, J., Guo, Y., Liu, M., Dong, T., and Duan, X. (2018). A machine learning framework for stock selection. *ArXiv Preprint ArXiv:1806.01743*.

Gao, J. and Hu, J. (2013). Fast monitoring of epileptic seizures using recurrence time statistics of electroencephalography. *Frontiers in Computational Neuroscience*, 7:122.

Gao, W., Chen, J., Richard, C., and Huang, J. (2014). Online dictionary learning for kernel LMS. *IEEE Transactions on Signal Processing*, 62(11):2765–2777.

Gao, X., Shi, M., Song, X., Zhang, C., and Zhang, H. (2019). Recurrent neural networks for real-time prediction of TBM operating parameters. *Automation in Construction*, 98:225–235.

Garcia-Vega, S., Zeng, X.-J., and Keane, J. (2019). Learning from data streams using kernel least-mean-square with multiple kernel-sizes and adaptive step-size. *Neurocomputing*, 339:105–115.

Garcia-Vega, S., Zeng, X.-J., and Keane, J. (2020). Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach. *Expert Systems with Applications*, 160:113668.

Gau, Y.-F. and Wu, Z.-X. (2017). Macroeconomic announcements and price discovery in the foreign exchange market. *Journal of International Money and Finance*, 79:232–254.

Gentle, J. (2020). *Statistical Analysis of Financial Data: With Examples in R*. CRC Press.

Ghasemi, F., Mehridehnavi, A., Perez-Garrido, A., and Perez-Sanchez, H. (2018). Neural network and deep-learning algorithms used in QSAR studies: merits and drawbacks. *Drug Discovery Today*, 23(10):1784–1790.

Giraldo, L. G. S. and Principe, J. C. (2013). Information theoretic learning with infinitely divisible kernels. *ArXiv Preprint ArXiv:1301.3551*.

Giudici, P. (2018). Financial data science. *Statistics & Probability Letters*, 136:160–164.

Göçken, M., Özçalıcı, M., Boru, A., and Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications*, 44:320–331.

Gomes, T. A., Carvalho, R. N., and Carvalho, R. S. (2017). Identifying anomalies in parliamentary expenditures of brazilian chamber of deputies with deep autoencoders. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 940–943. IEEE.

Goumagias, N. D., Hristu-Varsakelis, D., and Assael, Y. M. (2018). Using deep Q-learning to understand the tax evasion behavior of risk-averse firms. *Expert Systems with Applications*, 101:258–270.

Grace, A. (2017). Can deep learning techniques improve the risk adjusted returns from enhanced indexing investment strategies.

Gray, R. M. (2011). *Entropy and information theory*. Springer Science & Business Media.

Greenblatt, R. E., Pflieger, M., and Ossadtchi, A. (2012). Connectivity measures applied to human brain electrophysiological data. *Journal of Neuroscience Methods*, 207(1):1–16.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Gu, R., Xiong, W., and Li, X. (2015). Does the singular value decomposition entropy have predictive power for stock market?—evidence from the shenzhen stock market. *Physica A: Statistical Mechanics and its Applications*, 439:103–113.

Gu, Y., Liu, J., Chen, Y., Jiang, X., and Yu, H. (2014). TOSELM: timeliness online sequential extreme learning machine. *Neurocomputing*, 128:119–127.

Gueniche, T., Fournier-Viger, P., Raman, R., and Tseng, V. S. (2015). CPT+: Decreasing the time/space complexity of the compact prediction tree. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 625–636. Springer.

Güler, N. F., Übeyli, E. D., and Güler, I. (2005). Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Systems with Applications*, 29(3):506–514.

Guo, L., Shi, F., and Tu, J. (2016). Textual analysis and machine leaning: Crack unstructured data in finance and accounting. *The Journal of Finance and Data Science*, 2(3):153–170.

Guo, X., Zhang, H., and Tian, T. (2018). Development of stock correlation networks using mutual information and financial big data. *PloS One*, 13(4):e0195941.

Guresen, E., Kayakutlu, G., and Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397.

Hacine-Gharbi, A. and Ravier, P. (2018). A binning formula of bi-histogram for joint entropy estimation using mean square error minimization. *Pattern Recognition Letters*, 101:21–28.

Hafezi, R., Shahrabi, J., and Hadavandi, E. (2015). A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Applied Soft Computing*, 29:196–210.

Haghighat, M. B. A., Aghagolzadeh, A., and Seyedarabi, H. (2011). A non-reference image fusion metric based on mutual information of image features. *Computers & Electrical Engineering*, 37(5):744–756.

Halmos, P. R. (2012). *A Hilbert space problem book*, volume 19. Springer Science & Business Media.

Han, M., Liang, Z., and Li, D. (2011). Sparse kernel density estimations and its application in variable selection based on quadratic Renyi entropy. *Neurocomputing*, 74(10):1664–1672.

Han, Y., Yang, K., and Zhou, G. (2013). A new anomaly: The cross-sectional profitability of technical analysis. *Journal of Financial and Quantitative Analysis*, pages 1433–1461.

Härdle, W. (1990). *Applied nonparametric regression*. Number 19. Cambridge University Press.

Hart, O. and Zingales, L. (2017). Companies should maximize shareholder welfare not market value. *ECGI-Finance Working Paper*, (521).

Hart, P. E. (1975). Moment distributions in economics: an exposition. *Journal of the Royal Statistical Society: Series A (General)*, 138(3):423–434.

Hassan, M. K., Khan, A., and Paltrinieri, A. (2019). Liquidity risk, credit risk and stability in Islamic and conventional banks. *Research in International Business and Finance*, 48:17–31.

Hassan, M. R. (2009). A combination of hidden Markov model and fuzzy model for stock market forecasting. *Neurocomputing*, 72(16-18):3439–3446.

Hauke, J. and Kossowski, T. (2011). Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*, 30(2):87–93.

Heryadi, Y. and Warnars, H. L. H. S. (2017). Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, stacked LSTM, and CNN-LSTM. In *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, pages 84–89. IEEE.

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. (2018). NSE stock market prediction using deep-learning models. *Procedia Computer Science*, 132:1351–1362.

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

Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. *Expert Systems with Applications*, 117:287–299.

Hossain, M. A., Karim, R., THulasiram, R., Bruce, N. D., and Wang, Y. (2018). Hybrid deep learning model for stock price prediction. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1837–1844. IEEE.

Hsu, P.-Y., Chou, C., Huang, S.-H., and Chen, A.-P. (2018). A market making quotation strategy based on dual deep learning agents for option pricing and bid-ask spread estimation. In *2018 IEEE International Conference on Agents (ICA)*, pages 99–104. IEEE.

Hu, J. and Wang, J. (2012). Global stability of complex-valued recurrent neural networks with time-delays. *IEEE Transactions on Neural Networks and Learning Systems*, 23(6):853–865.

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E., and Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, 36:534–551.

Huang, Y., Huang, K., Wang, Y., Zhang, H., Guan, J., and Zhou, S. (2016). Exploiting twitter moods to boost financial trend prediction based on deep network models. In *International Conference on Intelligent Computing*, pages 449–460. Springer.

Huang, Y. and Kou, G. (2014). A kernel entropy manifold learning approach for financial data analysis. *Decision Support Systems*, 64:31–42.

Hughes, T. W., Minkov, M., Shi, Y., and Fan, S. (2018). Training of photonic neural networks through in situ backpropagation and gradient measurement. *Optica*, 5(7):864–871.

Ilonen, J., Kamarainen, J.-K., and Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105.

Ince, H. and Trafalis, T. B. (2007). Kernel principal component analysis and support vector machines for stock price prediction. *IIE Transactions*, 39(6):629–637.

Iwasaki, H. and Chen, Y. (2018). Topic sentiment asset pricing with DNN supervised learning. *Available at SSRN 3228485*.

Jalota, H., Thakur, M., and Mittal, G. (2017). Modelling and constructing membership function for uncertain portfolio parameters: A credibilistic framework. *Expert Systems with Applications*, 71:40–56.

Jang, W. W. (2020). Risk aversion, uncertainty, and monetary policy: Structural vector autoregressions identified with high-frequency external instruments. *Economics Letters*, 186:108675.

Jarrow, R. A. (2008). Operational risk. *Journal of Banking & Finance*, 32(5):870–879.

Jasic, T. and Wood, D. (2004). The profitability of daily stock market indices trades based on neural network predictions: Case study for the S&P 500, the DAX, the TOPIX and the FTSE in the period 1965–1999. *Applied Financial Economics*, 14(4):285–297.

Jeong, G. and Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138.

Jiang, Y., Yu, M., and Hashmi, S. M. (2017a). The financial crisis and co-movement of global stock markets—a case of six major economies. *Sustainability*, 9(2):260.

Jiang, Z. and Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 905–913. IEEE.

Jiang, Z., Xu, D., and Liang, J. (2017b). A deep reinforcement learning framework for the financial portfolio management problem. *ArXiv Preprint ArXiv:1706.10059*.

Jin, L., Huang, Z., Chen, L., Liu, M., Li, Y., Chou, Y., and Yi, C. (2019). Modified single-output Chebyshev-polynomial feedforward neural network aided with subset method for classification of breast cancer. *Neurocomputing*, 350:128–135.

Jizba, P., Kleinert, H., and Shefaat, M. (2012). Rényi's information transfer between financial time series. *Physica A: Statistical Mechanics and its Applications*, 391(10):2971–2989.

Joo, T. W. and Kim, S. B. (2015). Time series forecasting based on wavelet filtering. *Expert Systems with Applications*, 42(8):3868–3874.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., and Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100:234–245.

Kalayci, C. B., Ertenlice, O., and Akbay, M. A. (2019). A comprehensive review of deterministic models and applications for mean-variance portfolio optimization. *Expert Systems with Applications*.

Kamiński, M., Ding, M., Truccolo, W. A., and Bressler, S. L. (2001). Evaluating causal relations in neural systems: Granger causality, directed transfer function and statistical assessment of significance. *Biological Cybernetics*, 85(2):145–157.

Kang, Y.-J. and Noh, Y. (2019). Comparison study of kernel density estimation according to various bandwidth selectors. *Journal of the Computational Structural Engineering Institute of Korea*, 32(3):173–181.

Kaplanski, G., Levy, H., Veld, C., and Veld-Merkoulova, Y. (2016). Past returns and the perceived Sharpe ratio. *Journal of Economic Behavior & Organization*, 123:149–167.

Kapur, J. N. (1994). *Measures of information and their applications*. Wiley-Interscience.

Karaoglu, S., Arpaci, U., and Ayvaz, S. (2017). A deep learning approach for optimization of systematic signal detection in financial trading systems with big data. *International Journal of Intelligent Systems and Applications in Engineering, SpecialIssue (SpecialIssue)*, pages 31–36.

Kee, C.-Y., Ponnambalam, S., and Loo, C.-K. (2017). Binary and multi-class motor imagery using renyi entropy for feature extraction. *Neural Computing and Applications*, 28(8):2051–2062.

Kenett, D. Y., Huang, X., Vodenska, I., Havlin, S., and Stanley, H. E. (2015). Partial correlation analysis: Applications for financial markets. *Quantitative Finance*, 15(4):569–578.

Khashei, M. and Hajirahimi, Z. (2019). A comparative study of series ARIMA/MLP hybrid models for stock price forecasting. *Communications in Statistics-Simulation and Computation*, 48(9):2625–2640.

Khatri, N. N. (2017). Factors influencing investors investment in initial public offering. *International Journal of Management and Applied Science*, 3(7).

Kidwell, D. S., Blackwell, D. W., Sias, R. W., and Whidbee, D. A. (2016). *Financial institutions, markets, and money*. John Wiley & Sons.

Kim, K.-H., Lee, C.-S., Jo, S.-M., and Cho, S.-B. (2015). Predicting the success of bank telemarketing using deep convolutional neural network. In *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pages 314–317. IEEE.

Kim, S. H., Lee, H. S., Ko, H. J., Jeong, S. H., Byun, H. W., and Oh, K. J. (2018). Pattern matching trading system based on the dynamic time warping algorithm. *Sustainability*, 10(12):4641.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Kirkos, E. and Manolopoulos, Y. (2004). Data mining in finance and accounting: a review of current research trends. In *Proceedings of the 1st International Conference on Enterprise Systems and Accounting (ICESAcc)*, pages 63–78.

Kissell, R. L. (2013). *The science of algorithmic trading and portfolio management*. Academic Press.

Klein, B., Wolf, L., and Afek, Y. (2015). A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4840–4848.

Korbel, J., Jiang, X., and Zheng, B. (2019). Transfer entropy between communities in complex financial networks. *Entropy*, 21(11):1124.

Korczak, J. and Hemes, M. (2017). Deep learning for financial time series forecasting in a-trader system. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 905–912. IEEE.

Kovács, G., Tóth, L., Van Compernolle, D., and Ganapathy, S. (2017). Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout. *Pattern Recognition Letters*, 100:44–50.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Kucharska-Stasiak, E. (2018). 15 myths about market value. *Real Estate Management and Valuation*, 26(3):113–121.

Kumar, M. and Thenmozhi, M. (2014). Forecasting stock index returns using ARIMA-SVM, ARIMA-ANN, and ARIMA-random forest hybrid models. *International Journal of Banking, Accounting and Finance*, 5(3):284–308.

Kumar, P. R. and Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques–a review. *European Journal of Operational Research*, 180(1):1–28.

Kumar, S. (2019). Asymmetric impact of oil prices on exchange rate and stock prices. *The Quarterly Review of Economics and Finance*, 72:41–51.

Kuo, C.-Y. (2016). Does the vector error correction model perform better than others in forecasting stock price? an application of residual income valuation theory. *Economic Modelling*, 52:772–789.

Lahsasna, A., Ainon, R. N., and Teh, Y. W. (2010). Credit scoring models using soft computing methods: A survey. *The International Arab Journal of Information Technology*, 7(2):115–123.

Laitinen, E. K. and Suvas, A. (2016). Financial distress prediction in an international context: Moderating effects of Hofstede's original cultural dimensions. *Journal of Behavioral and Experimental Finance*, 9:98–118.

Lanbouri, Z. and Achchab, S. (2015). A hybrid deep belief network approach for financial distress prediction. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6. IEEE.

Lee, J., Jang, D., and Park, S. (2017). Deep learning-based corporate performance prediction model considering technical capability. *Sustainability*, 9(6):899.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508.

Lee, M.-C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8):10896–10904.

Lee, S. I. and Yoo, S. J. (2018). Threshold-based portfolio: the role of the threshold and its applications. *The Journal of Supercomputing*, pages 1–18.

Leetaru, K. and Schrodt, P. A. (2013). GDELT: Global data on events, location, and tone, 1979–2012. In *International Studies Association (ISA) Annual Convention*, volume 2, pages 1–49. Citeseer.

Lei, B., Xu, G., Feng, M., Zou, Y., Van der Heijden, F., De Ridder, D., and Tax, D. (2017). *Classification, Parameter Estimation and State Estimation*. Wiley Online Library.

Leigh, W., Frohlich, C. J., Hornik, S., Purvis, R. L., and Roberts, T. L. (2007). Trading with a stock chart heuristic. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):93–104.

Leigh, W., Modani, N., Purvis, R., and Roberts, T. (2002). Stock market trading rule discovery using technical charting heuristics. *Expert Systems with Applications*, 23(2):155–159.

Li, J., Bu, H., and Wu, J. (2017a). Sentiment-aware stock market prediction: A deep learning method. In *2017 International Conference on Service Systems and Service Management*, pages 1–6. IEEE.

Li, K. and Príncipe, J. C. (2017). Transfer learning in adaptive filters: The nearest instance centroid-estimation kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 65(24):6520–6535.

Li, L., Zhou, J., Li, X., and Chen, T. (2017b). Poster: Practical fraud transaction prediction. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2535–2537.

Li, M., Liu, X., and Ding, F. (2017c). The maximum likelihood least squares based iterative estimation algorithm for bilinear systems with autoregressive moving average noise. *Journal of the Franklin Institute*, 354(12):4861–4881.

Li, X., Peng, L., Yao, X., Cui, S., Hu, Y., You, C., and Chi, T. (2017d). Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation. *Environmental Pollution*, 231:997–1004.

Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., Wang, F., Min, H., and Deng, X. (2016). Empirical analysis: stock market prediction via extreme learning machine. *Neural Computing and Applications*, 27(1):67–78.

Li, Y. and Hamamura, M. (2015). Zero-attracting variable-step-size least mean square algorithms for adaptive sparse channel estimation. *International Journal of Adaptive Control and Signal Processing*, 29(9):1189–1206.

Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166.

Li, Y., Lin, X., Wang, X., Shen, F., and Gong, Z. (2017e). Credit risk assessment algorithm using deep neural networks with clustering and merging. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 173–176. IEEE.

Liang, C. and Schienle, M. (2019). Determination of vector error correction models in high dimensions. *Journal of Econometrics*, 208(2):418–441.

Liang, Z., Chen, H., Zhu, J., Jiang, K., and Li, Y. (2018). Adversarial deep reinforcement learning in portfolio management. *ArXiv Preprint ArXiv:1808.09940*.

Lin, J.-S., Huang, C.-F., Liao, T.-L., and Yan, J.-J. (2010). Design and implementation of digital secure communication based on synchronized chaotic systems. *Digital Signal Processing*, 20(1):229–237.

Lipping, T., Erkintalo, N., Särkelä, M., Takala, R. S., Katila, A., Frantzén, J., Posti, J. P., Müller, M., and Tenovuo, O. (2017). Connectivity analysis of full montage EEG in traumatic brain injury patients in the ICU. In *European Medical and Biological Engineering Conference (EMBEC) and the Nordic-Baltic Conference on Biomedical Engineering and Medical Physics (NBC) 2017*, pages 97–100. Springer.

Liu, A.-A., Su, Y.-T., Nie, W.-Z., and Kankanhalli, M. (2016). Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):102–114.

Liu, H., Mi, X.-w., and Li, Y.-f. (2018). Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network. *Energy Conversion and Management*, 156:498–514.

Liu, W., Pokharel, P. P., and Principe, J. C. (2008). The kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554.

Liu, W. and Príncipe, J. C. (2008). Kernel affine projection algorithms. *EURASIP Journal on Advances in Signal Processing*, 2008(1):784292.

Liu, W., Principe, J. C., and Haykin, S. (2010). *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley Publishing, 1st edition.

Liu, X. and Zhang, C. (2017). Corporate governance, social responsibility information disclosure, and enterprise value in China. *Journal of Cleaner Production*, 142:1075–1084.

Liu, Y.-k., Xie, F., Xie, C.-l., Peng, M.-j., Wu, G.-h., and Xia, H. (2015). Prediction of time series of NPP operating parameters using dynamic model based on BP neural network. *Annals of Nuclear Energy*, 85:566–575.

Lopes, G. D. L. F. (2018). Deep learning for market forecasts.

Loughran, T. and McDonald, B. (2016). Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230.

Lu, D. W. (2017). Agent inspired trading using recurrent reinforcement learning and LSTM neural networks. *ArXiv Preprint ArXiv:1707.07338*.

Luo, C., Wu, D., and Wu, D. (2017). A deep learning approach for credit scoring using credit default swaps. *Engineering Applications of Artificial Intelligence*, 65:465–470.

Lütkepohl, H. (2013). Vector autoregressive models. In *Handbook of Research Methods and Applications in Empirical Macroeconomics*. Edward Elgar Publishing.

Lv, D., Yuan, S., Li, M., and Xiang, Y. (2019). An empirical study of machine learning algorithms for stock daily trading strategy. *Mathematical Problems in Engineering*, 2019.

Malik, N., Singh, P. V., and Khan, U. (2018). Can banks survive the next financial crisis? an adversarial deep learning model for bank stress testing. *An Adversarial Deep Learning Model for Bank Stress Testing (June 30, 2018)*.

Mammone, N., Inuso, G., La Foresta, F., Versaci, M., and Morabito, F. C. (2011). Clustering of entropy topography in epileptic electroencephalography. *Neural Computing and Applications*, 20(6):825–833.

Manela, A. and Moreira, A. (2017). News implied volatility and disaster concerns. *Journal of Financial Economics*, 123(1):137–162.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (M-RNN). *ArXiv Preprint ArXiv:1412.6632*.

Marques, A., García, V., and Sánchez, J. S. (2013). A literature review on the application of evolutionary computing to credit scoring. *Journal of the Operational Research Society*, 64(9):1384–1399.

Masih, A. M. and Masih, R. (2001). Dynamic modeling of stock market interdependencies: an empirical investigation of australia and the Asian NICs. *Review of Pacific Basin Financial Markets and Policies*, 4(02):235–264.

Maskawa, J.-i. (2016). Collective behavior of market participants during abrupt stock price changes. *PloS One*, 11(8):e0160152.

Massey, E. C. (2016). Master limited partnerships: A pipeline to renewable energy development. *University of Colorado*, 87:1009.

McNally, S., Roche, J., and Caton, S. (2018). Predicting the price of bitcoin using machine learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343. IEEE.

Medeiros, M. C., Vasconcelos, G. F., Veiga, A., and Zilberman, E. (2019). Forecasting inflation in a data-rich environment: the benefits of machine learning methods. *Journal of Business & Economic Statistics*, pages 1–22.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *ArXiv Preprint ArXiv:1301.3781*.

Milosevic, N. (2016). Equity forecast: Predicting long term stock price movement using machine learning. *ArXiv Preprint ArXiv:1603.00751*.

Mirjalili, S., Hashim, S. Z. M., and Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, 218(22):11125–11137.

Mohammed, M. A., Ghani, M. K. A., Hamed, R. I., Ibrahim, D. A., and Abdullah, M. K. (2017). Artificial neural networks for automatic segmentation and identification of nasopharyngeal carcinoma. *Journal of Computational Science*, 21:263 – 274.

Moreno, P. J., Ho, P. P., and Vasconcelos, N. (2004). A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1385–1392.

Mourelatos, M., Alexakos, C., Amorgianiotis, T., and Likothanassis, S. (2018). Financial indices modelling and trading utilizing deep learning techniques: The Athens SE FTSE/ASE large cap use case. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–7. IEEE.

Nan, L. and Tao, D. (2018). Bitcoin mixing detection using deep autoencoder. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 280–287. IEEE.

Navon, A. and Keller, Y. (2017). Financial time series prediction using deep learning. *ArXiv Preprint ArXiv:1711.04174*.

Nayak, R. K., Mishra, D., and Rath, A. K. (2015). A Naïve SVM-KNN based stock market trend reversal analysis for indian benchmark indices. *Applied Soft Computing*, 35:670–680.

Nazário, R. T. F., e Silva, J. L., Sobreiro, V. A., and Kimura, H. (2017). A literature review of technical analysis on stock markets. *The Quarterly Review of Economics and Finance*, 66:115–126.

Neagoe, V.-E., Ciotec, A.-D., and Cucu, G.-S. (2018). Deep convolutional neural networks versus multilayer perceptron for financial prediction. In *2018 International Conference on Communications (COMM)*, pages 201–206. IEEE.

Neri, C. and Schneider, L. (2012). Maximum entropy distributions inferred from option portfolios on an asset. *Finance and Stochastics*, 16(2):293–318.

Nesbitt, K. V. and Barrass, S. (2004). Finding trading patterns in stock market data. *IEEE Computer Graphics and Applications*, 24(5):45–55.

Newbold, P. and Harvey, D. I. (2002). Forecast combination and encompassing. *A Companion to Economic Forecasting*, pages 268–283.

Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., and Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 2018. Determination Press San Francisco, CA.

Niimi, A. (2015). Deep learning for credit card data analysis. In *2015 World Congress on Internet Security (WorldCIS)*, pages 73–77. IEEE.

Niu, Q. and Chen, T. (2018). A new variable step size LMS adaptive algorithm. In *2018 Chinese Control And Decision Conference (CCDC)*. IEEE.

Nobre, R. H., Rodrigues, F. A., Marques, R. C., Nobre, J. S., Neto, J. F., and Medeiros, F. N. (2016). SAR image segmentation with Renyi's entropy. *IEEE Signal Processing Letters*, 23(11):1551–1555.

Novak, S. Y. (2007). Measures of financial risks and market crashes.

Nweke, H. F., Teh, Y. W., Al-Garadi, M. A., and Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261.

Ogawa, A. and Hori, T. (2017). Error detection and accuracy estimation in automatic speech recognition using deep bidirectional recurrent neural networks. *Speech Communication*, 89:70–83.

Ojha, V. K., Abraham, A., and Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116.

Olden, J. D. and Jackson, D. A. (2002). Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1-2):135–150.

Oliveira, N., Cortez, P., and Areal, N. (2017). The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert Systems with Applications*, 73:125–144.

Olsen, R. B., Glattfelder, J. B., and Golub, A. (2018). The alpha engine: Designing an automated trading algorithm. In *High-Performance Computing in Finance*, pages 49–76. Chapman and Hall/CRC.

Ormos, M. and Zibriczky, D. (2014). Entropy-based financial asset pricing. *PloS One*, 9(12):e115742.

Oshiro, T. M., Perez, P. S., and Baranauskas, J. A. (2012). How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer.

Ozbayoglu, A. M., Gudelek, M. U., and Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*, page 106384.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., and Ward, R. (2016a). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Palangi, H., Ward, R. K., and Deng, L. (2016b). Distributed compressive sensing: A deep learning approach. *IEEE Transactions on Signal Processing*, 64(17):4504–4518.

Papana, A., Kyrtsou, C., Kugiumtzis, D., and Diks, C. (2017). Financial networks based on Granger causality: A case study. *Physica A: Statistical Mechanics and its Applications*, 482:65–73.

Park, C.-H. and Irwin, S. H. (2007). What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826.

Park, H.-J., Friston, K. J., Pae, C., Park, B., and Razi, A. (2018). Dynamic effective connectivity in resting state FMRI. *Neuroimage*, 180:594–608.

Parker, C. S., Clayden, J. D., Cardoso, M. J., Rodionov, R., Duncan, J. S., Scott, C., Diehl, B., and Ourselin, S. (2018). Structural and effective connectivity in focal epilepsy. *NeuroImage: Clinical*, 17:943–952.

Parracho, P., Neves, R., and Horta, N. (2010). Trading in financial markets using pattern recognition optimized by genetic algorithms. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 2105–2106.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268.

Paula, E. L., Ladeira, M., Carvalho, R. N., and Marzagao, T. (2016). Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 954–960. IEEE.

Peachavanish, R. (2016). Stock selection and trading based on cluster analysis of trend and momentum indicators. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 317–321.

Peiro, A. (2016). Stock prices and macroeconomic factors: Some european evidence. *International Review of Economics & Finance*, 41:287–294.

Peng, Y. and Jiang, H. (2015). Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *ArXiv Preprint ArXiv:1506.07220*.

Phetchanchai, C., Selamat, A., Rehman, A., and Saba, T. (2010). Index financial time series based on zigzag-perceptually important points. In *Journal of Computer Science*. Citeseer.

Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. *ArXiv Preprint ArXiv:1009.6119*.

Pierce, J. R. (2012). *An introduction to information theory: symbols, signals and noise*. Courier Corporation.

Pierdzioch, C. and Risse, M. (2018). A machine-learning analysis of the rationality of aggregate stock market forecasts. *International Journal of Finance & Economics*, 23(4):642–654.

Pokharel, R., Seth, S., and Principe, J. C. (2013). Mixture kernel least mean square. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Polikar, R. (2009). Ensemble learning. *Scholarpedia*, 4(1):2776. revision #186077.

Portugal, I., Alencar, P., and Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227.

Powell, N., Foo, S. Y., and Weatherspoon, M. (2008). Supervised and unsupervised methods for stock trend forecasting. In *2008 40th Southeastern Symposium on System Theory (SSST)*, pages 203–205. IEEE.

Principe, J. C. (2010). *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media.

Principe, J. C., Xu, D., Zhao, Q., and Fisher, J. W. (2000). Learning from examples with information theoretic criteria. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 26(1-2):61–77.

Prosky, J., Song, X., Tan, A., and Zhao, M. (2017). Sentiment predictability for stocks. *ArXiv Preprint ArXiv:1712.05785*.

Qin, H., Gong, R., Liu, X., Bai, X., Song, J., and Sebe, N. (2020). Binary neural networks: A survey. *Pattern Recognition*, page 107281.

Radhakrishnan, S., Duvvuru, A., Sultornsanee, S., and Kamarthi, S. (2016). Phase synchronization based minimum spanning trees for analysis of financial time series with nonlinear correlations. *Physica A: Statistical Mechanics and its Applications*, 444:259–270.

Ramos, P., Santos, N., and Rebelo, R. (2015). Performance of state space and arima models for consumer retail sales forecasting. *Robotics and Computer-Integrated Manufacturing*, 34:151–163.

Rather, A. M., Agarwal, A., and Sastry, V. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241.

Ravi, V., Kurniawan, H., Thai, P. N. K., and Kumar, P. R. (2008). Soft computing system for bank performance prediction. *Applied Soft Computing*, 8(1):305–315.

Ravi, V., Pradeepkumar, D., and Deb, K. (2017). Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms. *Swarm and Evolutionary Computation*, 36:136–149.

Rawte, V., Gupta, A., and Zaki, M. J. (2018). Analysis of year-over-year changes in risk factors disclosure in 10-K filings. In *Proceedings of the Fourth International Workshop on Data Science for Macro-Modeling with Financial and Economic Datasets*, pages 1–4.

Reboredo, J. C. (2018). Green bond and financial markets: Co-movement, diversification and price spillover effects. *Energy Economics*, 74:38–50.

Rejeb, A. B. and Arfaoui, M. (2016). Financial market interdependencies: A quantile regression analysis of volatility spillover. *Research in International Business and Finance*, 36:140–157.

Renault, T. (2017). Intraday online investor sentiment and return patterns in the us stock market. *Journal of Banking & Finance*, 84:25–40.

Rényi, A. (1976). Some fundamental questions of information theory. *Selected Papers of Alfred Renyi*, 2(174):526–552.

Rezayat, F. and Yavas, B. F. (2006). International portfolio diversification: A study of linkages among the US, European and Japanese equity markets. *Journal of Multinational Financial Management*, 16(4):440–458.

Richard, C., Bermudez, J. C. M., and Honeine, P. (2009). Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57(3):1058–1067.

Rodan, A., Faris, H., Alqatawna, J., et al. (2016). Optimizing feedforward neural networks using biogeography based optimization for e-mail spam identification. *International Journal of Communications, Network and System Sciences*, 9(01):19.

Rödder, W., Gartner, I. R., and Rudolph, S. (2010). An entropy-driven expert system shell applied to portfolio selection. *Expert Systems with Applications*, 37(12):7509–7520.

Rönnqvist, S. and Sarlin, P. (2015). Detect & describe: Deep learning of bank stress in the news. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 890–897. IEEE.

Rönnqvist, S. and Sarlin, P. (2017). Bank distress in the news: Describing events through deep learning. *Neurocomputing*, 264:57–70.

Roy, A., Sun, J., Mahoney, R., Alonzi, L., Adams, S., and Beling, P. (2018). Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134. IEEE.

Rundo, F., Trenta, F., Di Stallo, A. L., and Battiato, S. (2019). Advanced markov-based machine learning framework for making adaptive trading system. *Computation*, 7(1):4.

Saia, R. (2017). A discrete wavelet transform approach to fraud detection. In *International Conference on Network and System Security*, pages 464–474. Springer.

Saia, R., Carta, S., et al. (2017). A frequency-domain-based pattern mining for credit card fraud detection. In *IoTBDS*, pages 386–391.

Salinas, D., Shen, H., and Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning*, pages 8438–8448. PMLR.

Samarawickrama, A. and Fernando, T. (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6. IEEE.

Samuel, M. and Okey, L. E. (2015). The relevance and significance of correlation in social science research. *International Journal of Sociology and Anthropology Research*, 1(3):22–28.

Sandberg, I. W., Lo, J. T., Fancourt, C. L., Principe, J. C., Katagiri, S., and Haykin, S. (2001). *Nonlinear dynamical systems: feedforward neural network perspectives*, volume 21. John Wiley & Sons.

Scardapane, S., Fierimonte, R., Di Lorenzo, P., Panella, M., and Uncini, A. (2016). Distributed semi-supervised support vector machines. *Neural Networks*, 80:43–52.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Schölkopf, B., Smola, A. J., and Bach, F. (2018). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. The MIT Press.

Schumaker, R. P. and Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., and Soman, K. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647. IEEE.

Sezer, O. B., Ozbayoglu, M., and Dogdu, E. (2017). A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters. *Procedia Computer Science*, 114:473–480.

Sharma, A. and Panigrahi, P. K. (2013). A review of financial accounting fraud detection based on data mining techniques. *ArXiv Preprint ArXiv:1309.3944*.

Sharma, S. (2017). Activation functions in neural networks. *Towards Data Science*, 6.

Shen, S., Jiang, H., and Zhang, T. (2012). Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, Stanford, CA*, pages 1–5.

Shi, L., Teng, Z., Wang, L., Zhang, Y., and Binder, A. (2018). DEEPCLUE: Visual interpretation of text-based deep stock prediction. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1094–1108.

Si, W., Li, J., Ding, P., and Rao, R. (2017). A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 431–436. IEEE.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401. IEEE.

Sibi, P., Jones, S. A., and Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, 47(3):1264–1268.

Siikanen, M., Baltakys, K., Kanniainen, J., Vatrapu, R., Mukkamala, R., and Hussain, A. (2018). Facebook drives behavior of passive households in stock markets. *Finance Research Letters*, 27:208–213.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC Press.

Sim, H. S., Kim, H. I., and Ahn, J. J. (2019). Is deep learning for image recognition applicable to stock market prediction? *Complexity*, 2019.

Singh, A. and Príncipe, J. C. (2011). Information theoretic learning with adaptive kernels. *Signal Processing*, 91(2):203–213.

Sirignano, J. and Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459.

Sirignano, J., Sadhwani, A., and Giesecke, K. (2016). Deep learning for mortgage risk. *ArXiv Preprint ArXiv:1607.02470*.

Sonoda, S. and Murata, N. (2017). Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268.

Spilak, B. (2018). Deep neural networks for cryptocurrencies price prediction. Master's thesis, Humboldt-Universität zu Berlin.

Stosic, D., Stosic, D., Ludermir, T., de Oliveira, W., and Stosic, T. (2016). Foreign exchange rate entropy evolution during financial crises. *Physica A: Statistical Mechanics and its Applications*, 449:233–239.

Suhermi, N., Prastyo, D. D., Ali, B., et al. (2018). Roll motion prediction using a hybrid deep learning and arima model. *Procedia Computer Science*, 144:251–258.

Sulthan, A., Jayakumar, S., and David, G. (2016). On the review and application of entropy in finance. *International Journal of Business Insights & Transformation*, 10(1).

Sun, H. (2005). Mercer theorem for RKHS on noncompact sets. *Journal of Complexity*, 21(3):337–349.

Sun, J., Li, H., Huang, Q.-H., and He, K.-Y. (2014). Predicting financial distress and corporate failure: A review from the state-of-the-art definitions, modeling, sampling, and featuring approaches. *Knowledge-Based Systems*, 57:41–56.

Suykens, J. A. (2017). Deep restricted kernel machines using conjugate feature duality. *Neural Computation*, 29(8):2123–2163.

Suykens, J. A. and Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, 10(4):907–911.

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62.

Tai, W., Teng, Q., Zhou, Y., Zhou, J., and Wang, Z. (2019). Chaos synchronization of stochastic reaction-diffusion time-delay neural networks via non-fragile output-feedback control. *Applied Mathematics and Computation*, 354:115–127.

Taieb, S. B. and Atiya, A. F. (2015). A bias and variance analysis for multistep-ahead time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1):62–76.

Takeuchi, L. and Lee, Y.-Y. A. (2013). Applying deep learning to enhance momentum trading strategies in stocks. In *Technical Report*. Stanford University.

Taylor, S. J. (2008). *Modelling financial time series*. World Scientific.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168.

Tetlock, P. C., Saar-Tsechansky, M., and Macskassy, S. (2008). More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467.

Tian, Y., Zhang, K., Li, J., Lin, X., and Yang, B. (2018). LSTM-based traffic flow prediction with missing data. *Neurocomputing*, 318:297–305.

Tiwari, S., Pandit, R., and Richhariya, V. (2010). Predicting future trends in stock market by decision tree rough-set based hybrid system with HHMM. *International Journal of Electronics and Computer Science Engineering*, 1(3).

Tkáč, M. and Verner, R. (2016). Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, 38:788–804.

Tran, D. T., Magris, M., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2017). Tensor representation in high-frequency financial data for price change prediction. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE.

Tran, K., Duong, T., and Ho, Q. (2016). Credit scoring model: A combination of genetic programming and deep learning. In *2016 Future Technologies Conference (FTC)*, pages 145–149. IEEE.

Troiano, L., Villa, E. M., and Loia, V. (2018). Replicating a trading strategy by means of LSTM for financial industry applications. *IEEE Transactions on Industrial Informatics*, 14(7):3226–3234.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2017). Using deep learning to detect price change indications in financial markets. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2511–2515. IEEE.

Tunaru, R. S. (2015). *Model risk in financial markets: From financial engineering to risk management*. World Scientific.

Tzeng, F.-Y. and Ma, K.-L. (2005). *Opening the black box-data driven visualization of neural networks*. IEEE.

Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: a comparative. *Journal of Machine Learning Research*, 10(66-71):13.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.

Verikas, A., Kalsyte, Z., Bacauskiene, M., and Gelzinis, A. (2010). Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: a survey. *Soft Computing*, 14(9):995–1010.

Wacker, J. G., Yang, C., and Sheu, C. (2016). A transaction cost economics model for estimating performance effectiveness of relational and contractual governance. *International Journal of Operations & Production Management*.

Waldron, R. (2018). Capitalizing on the state: The political economy of real estate investment trusts and the 'resolution'of the crisis. *Geoforum*, 90:206–218.

Wang, G.-J. and Xie, C. (2016). Tail dependence structure of the foreign exchange market: A network view. *Expert Systems with Applications*, 46:164–179.

Wang, J., Peng, B., and Zhang, X. (2018a). Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.

Wang, J.-J., Wang, J.-Z., Zhang, Z.-G., and Guo, S.-P. (2012). Stock index forecasting based on a hybrid model. *Omega*, 40(6):758–766.

Wang, J.-L. and Chan, S.-H. (2007). Stock market trading rule discovery using pattern recognition and technical analysis. *Expert Systems with Applications*, 33(2):304–315.

Wang, M.-C. and Ye, J.-K. (2016). The relationship between covariance risk and size effects in emerging equity markets. *Managerial Finance*, 42(3):174–190.

Wang, Q., Xu, W., and Zheng, H. (2018b). Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing*, 299:51–61.

Wang, S. (2010). A comprehensive survey of data mining-based accounting-fraud detection research. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 50–53. IEEE.

Wang, W., Li, W., Zhang, N., and Liu, K. (2020). Portfolio formation with preselection using deep learning from long-term financial data. *Expert Systems with Applications*, 143:113042.

Wang, W. and Lu, Y. (2018). Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In *IOP Conference Series: Materials Science and Engineering*, volume 324, page 012049. IOP Publishing.

Wang, Y. and Xu, W. (2018). Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud. *Decision Support Systems*, 105:87–95.

West, J. and Bhattacharya, M. (2016). Intelligent financial fraud detection: a comprehensive review. *Computers & Security*, 57:47–66.

Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451.

Wu, K., Bethel, E., Gu, M., Leinweber, D., and Rübel, O. (2013a). A big data approach to analyzing market volatility. *Algorithmic Finance*, 2(3-4):241–267.

Wu, K.-P., Wu, Y. P., and Lee, H.-M. (2014). Stock trend prediction by using a-means and aprioriall algorithm for sequential chart pattern mining. *Journal of Information Science & Engineering*, 30(3).

Wu, Y., Zhou, Y., Saveriades, G., Agaian, S., Noonan, J. P., and Natarajan, P. (2013b). Local Shannon entropy measure with statistical tests for image randomness. *Information Sciences*, 222:323–342.

Xu, D. (1999). *Energy, entropy and information potential for neural computation.* PhD thesis, Citeseer.

Xu, J., Zhou, X., and Wu, D. D. (2011). Portfolio selection using λ mean and hybrid entropy. *Annals of Operations Research*, 185(1):213–229.

Xu, R. and Wunsch, D. (2008). *Clustering*, volume 10. John Wiley & Sons.

Yamanishi, K. and Takeuchi, J.-i. (2002). A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 676–681. ACM.

Yegnanarayana, B. (2009). *Artificial neural networks.* PHI Learning Pvt. Ltd.

Yeh, C.-Y., Huang, C.-W., and Lee, S.-J. (2011). A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3):2177–2186.

Yeh, S.-H., Wang, C.-J., and Tsai, M.-F. (2015). Deep belief networks for predicting corporate defaults. In *2015 24th Wireless and Optical Communication Conference (WOCC)*, pages 159–163. IEEE.

Yilmaz, H. (2020). WAS "yilmaz cash management model" accepted by the theory of finance. *American Journal of Finance*, 5(1):16–23.

Ying, J. J.-C., Huang, P.-Y., Chang, C.-K., and Yang, D.-L. (2017). A preliminary study on deep learning for predicting social insurance payment behavior. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1866–1875. IEEE.

Yong, B. X., Rahim, M. R. A., and Abdullah, A. S. (2017). A stock market trading system using deep neural network. In *Asian Simulation Conference*, pages 356–364. Springer.

Yoshihara, A., Fujikawa, K., Seki, K., and Uehara, K. (2014). Predicting stock market trends by recurrent deep neural networks. In *Pacific rim International Conference on Artificial Intelligence*, pages 759–769. Springer.

Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. (2016). Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593.

Yu, L., Zhou, R., Tang, L., and Chen, R. (2018a). A DBN-based resampling SVM ensemble learning paradigm for credit classification with imbalanced data. *Applied Soft Computing*, 69:192–202.

Yu, S.-S., Chu, S.-W., Wang, C.-M., Chan, Y.-K., and Chang, T.-C. (2018b). Two improved k-means algorithms. *Applied Soft Computing*, 68:747–755.

Yümlü, S., Gürgen, F. S., and Okay, N. (2005). A comparison of global, recurrent and smoothed-piecewise neural models for Istanbul stock exchange (ISE) prediction. *Pattern Recognition Letters*, 26(13):2093–2103.

Zaccone, G., Karim, M. R., and Menshawy, A. (2017). *Deep Learning with TensorFlow*. Packt Publishing Ltd.

Zaremba, W., Sutskever, I., and Vinyals, O. (2015). Recurrent neural network regularization. In *International Conference on Learning Representations (ICLR)*.

Zhang, H., Wang, L., Zhang, T., and Wang, S. (2019a). The nearest-instance-centroid-estimation kernel recursive least squares algorithms. *IEEE Transactions on Circuits and Systems II: Express Briefs*.

Zhang, J., Cui, S., Xu, Y., Li, Q., and Li, T. (2018a). A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69.

Zhang, L., Aggarwal, C., and Qi, G.-J. (2017a). Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2141–2149.

Zhang, M., Jiang, X., Fang, Z., Zeng, Y., and Xu, K. (2019b). High-order hidden Markov model for trend prediction in financial time series. *Physica A: Statistical Mechanics and its Applications*, 517:1–12.

Zhang, Q. and Chan, T. H. (2015). A Gaussian noise model of spectral amplitudes in soliton communication systems. In *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 455–459. IEEE.

Zhang, S., Zheng, W. X., and Zhang, J. (2017b). A new combined-step-size normalized least mean square algorithm for cyclostationary inputs. *Signal Processing*, 141:261–272.

Zhang, X.-d., Li, A., and Pan, R. (2016). Stock trend prediction based on a new status box method and AdaBoost probabilistic support vector machine. *Applied Soft Computing*, 49:385–398.

Zhang, Z., Beck, M. W., Winkler, D. A., Huang, B., Sibanda, W., Goyal, H., et al. (2018b). Opening the black box of neural networks: methods for interpreting neural network models in clinical applications. *Annals of Translational Medicine*, 6(11).

Zhao, J., Zhang, H., and Zhang, J. A. (2020). Gaussian kernel adaptive filters with adaptive kernel bandwidth. *Signal Processing*, 166:107270.

Zhao, S., Chen, B., and Principe, J. C. (2011). Kernel adaptive filtering with maximum correntropy criterion. In *The 2011 International Joint Conference on Neural Networks*, pages 2012–2017. IEEE.

Zhao, S., Chen, B., Zhu, P., and Príncipe, J. C. (2013). Fixed budget quantized kernel least-mean-square algorithm. *Signal Processing*, 93(9):2759–2770.

Zheng, Y., Wang, S., Feng, J., and Chi, K. T. (2016). A modified quantized kernel least mean square algorithm for prediction of chaotic time series. *Digital Signal Processing*, 48:130–136.

Zhong, X. and Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139.

Zhong, X. and Enke, D. (2019). Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, 5(1):4.

Zhou, B. (2019). Deep learning and the cross-section of stock returns: Neural networks combining price and fundamental information. *Available at SSRN 3179281*.

Zhou, H., Huang, J., and Lu, F. (2017). Reduced kernel recursive least squares algorithm for aero-engine degradation prediction. *Mechanical Systems and Signal Processing*, 95:446–467.

Zhou, R., Wang, X., Dong, X., and Zong, Z. (2013). Portfolio selection model with the measures of information entropy-incremental entropy-skewness. *Advances in Information Sciences and Service Sciences*, 5(8):833.

Zhou, W. and Xu, Z. (2018). Portfolio selection and risk investment under the hesitant fuzzy environment. *Knowledge-Based Systems*, 144:21–31.

Zhou, Z.-H. (2015). Ensemble learning. *Encyclopedia of Biometrics*, pages 411–416.

Zhou, Z.-H. and Tang, W. (2003). Selective ensemble of decision trees. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 476–483. Springer.

Zhou, Z.-H., Wu, J., and Tang, W. (2002). Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263.

Zhu, Y., Zhou, L., Xie, C., Wang, G.-J., and Nguyen, T. V. (2019). Forecasting SMEs' credit risk in supply chain finance with an enhanced hybrid ensemble machine learning approach. *International Journal of Production Economics*, 211:22–33.

Zhuge, Q., Xu, L., and Zhang, G. (2017). LSTM neural network with emotional analysis for prediction of stock price. *Engineering Letters*, 25(2).

Zio, E. (2018). The future of risk assessment. *Reliability Engineering & System Safety*, 177:176–190.