University of Southampton

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

On Quantifying the Role of Exogenous Macro-Economic Information in Machine Learning for Modelling Financial Data

by

Luis Jairo Montesdeoca Bermúdez

February 2020

Thesis for the degree of

Doctor of Philosophy in Computer Science

University of Southampton

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Luis Jairo Montesdeoca Bermúdez

Data from the financial markets are a source of challenging inference problems. Machine learning tools are increasingly used for the analysis of financial data. They are observed to provide more accurate models than classical analytical models that depend on specific assumptions. In this work, we ask if the inclusion of external (exogenous) macro-economic information into a model fitting procedure may be useful to improve the quality of analysis and predictions of financial time series. This dissertation explores this case by addressing several problems in empirical finance which are tackled by using a range of machine learning methods with exogenous macro-economic data.

First, we study a non-parametric approach to mapping the price of traded option contracts to the value of the underlying asset and the time to maturity. We explore if additional information would be helpful in improving this mapping. We show that this is the case, and further we show that there is a relationship between volume traded and volatility of an asset that is not apparent in the raw data, but it is seen through their influence on the prices of options. Then, we consider the non-negative matrix factorization (NMF) method and extend it with eXogenous information to specify a new model (XNMF). We present a learning algorithm for it and illustrate its better performance than NMF using equity prices and underlying macroeconomic variables. We show how residual signals arising in time series analysis can be explained by a sparse regression taken over related macroeconomic variables (the Kalman LagLasso model) to help in financial analysis. A comparison between stock index values and Bitcoin using this model illustrates clear underlying differences between them.

Finally, we study a powerful representation learning framework popular in machine learning (VAE) and extend it with inductive exogenous variable. Thus, we created a probabilistic XNMF (PAE-XNMF) that is able to generate financial data, with lower reconstruction error than a probabilistic NMF; and Recurrent Neural Networks, specifically, the Long Short Term Memory model (LSTM). We show that LSTM captures time series dynamics. Then we combined LSTM with attention mechanism to gain more interpretability of the influence of macro-economic data on predicting financial time series.

Contents

Nomenclature

MSE	Mean Square Error
RBF	Radial Basis Function
VAE	Variational Autoencoder
LSTM	Long Short Term Memory
ANN	Artificial Neural Networks
RNN	Recurrent Neural Networks
FTSE	Financial Times Stock Exchange
NMF	Non-negative Matrix Factorization
XNMF	Exogenous Non-negative Matrix Factorization

Declaration of Authorship

I, Luis Jairo Montesdeoca Bermudez, declare that the thesis entitled *Machine Learning for Modelling Financial Data* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

  [a] Montesdeoca and Niranjan (2016)

  [b] Squires et al. (2017a)

  [c] Montesdeoca and Niranjan (2019)

  [d] Montesdeoca et al. (2019)

Signed: ________________________

Date: ________________________

Chapter 1

Introduction

1.1 Finance and Financial Markets

Financial markets are places where assets are traded. Over the recent decades, they have become highly complex systems with very complex products as instruments. From a computational perspective, what is of interest are the prices at which assets are traded and the underlying determinants of these prices. Complexity in financial markets arises from a range of factors such as fundamental knowledge about an asset (e.g. the profitability of a company), macroeconomic policy set by governments and how that may affect a particular asset (e.g. central bank interest rates), past values of an asset and trader sentiments and how different players in a market react to new information and uncertainty.

Financial instruments come in three broad classes: Bonds, Stocks and Derivatives. Bonds are issued by governments and large companies to raise capital from the markets, that may be used for investment, such as constructing roads in the case of governments, or purchase of new equipment, in the case of a manufacturing company. A bond is a relatively safe asset in which the issuer guarantees a fixed rate of payment periodically to the holder (purchaser). Stocks are shares issued by a company on its ownership. The holder of a share in a company owns a fraction of that company and this ownership is a tradable asset. Additionally, should the company periodically pay dividends or its profits to shareholders, the holder of this class of assets benefits proportionally.

Derivatives are instruments designed as contracts on the predicted future performance of an asset. For example, a futures contract is a legal agreement between two parties that mandates the sale or purchase of a financial product at a predetermined value on a future date. The other type, options, are contracts that will permit the holder to buy or sell an asset at a particular price at, or during, a fixed time in the future. Over the decades, very sophisticated instruments such as exotic options and interest rate swaps have been created, driving up the complexity of markets.

Despite the bad publicity they attract, financial markets and the instruments and the instruments traded in them have an important role in the capitalist system. Firms that operate in such a system need the financial markets to raise capital against some promise of future reward for the investor. Firms trading across different currencies, and whose business is characterised by a time lag between delivery of goods or services and the settlement of payments, may want to guard against adverse exchange rate movements by securing forward contracts. Large pension funds and similar fund managers routinely purchase options contracts to mitigate adverse price movements by hedging. Such pursuits of stability and raising capital is balanced by the profit making desire of those investors, small and big, who wish to reap good rewards from the capital they hold by taking a risk in the financial markets. From the perspective of these investors a high return is expected, for example by a company in which they hold shares becoming highly profitable and paying out large dividends, comes at a risk, for example that company making a loss. But the holder of capital is willing to take that risk, hoping for a high reward. Thus the financial markets may be seen as offering a field for these two types of players, one driven by the need for capital and stability and the other driven by profiteering and risk averseness.

1.2 Research Objective

Computational problems in the financial markets arise primarily from sources of uncertainty in how prices are modelled. Our attention in this dissertation is in data-driven models that largely fall under the topic of machine learning. Machine learning methods are based on statistical approaches, using linear and non-linear functions approximations as opposed to mechanistic models that start from assumptions about the underlying variables and relationships among them, leading to deterministic or stochastic differential equations.

Machine learning tools are considered to be more accurate models than classical analytical models that are relying on assumptions and weak correlations. Many researchers are finding ways for improving the accuracy of machine learning. One way for this aim is getting more data. However, in many circumstances it is difficult to get more data that is directly involved. For that reason, we bring the hypothesis that the inclusion of exogenous financial data may help to improve the performance of machine learning methods and to extract useful information on financial time series.

We explore this idea by addressing three types of problems that arise in financial data analysis: (a) analysing time series; (b) pricing derivatives; and (c) extracting the influence of exogenous information. Amongst other things, the research will seek to answer the following questions:

- Can external (exogenous) macro-economic information improve the quality of machine learning techniques in financial time series analysis and predictions?
- Can machine learning techniques gain more interpretability by the inclusion of exogenous data?

1.3 Computational Problems in Finance

Time series analysis and forecasting is the most widely studied data analysis problem in finance. The objective here is to forecast the future value of a time series based on regularity that can be captured from its past behaviour. We will give a brief overview of different linear and non-linear time series models later in the Section 2.1. We will also specify models of Kalman filtering and Recurrent Neural Networks in this work (Section 2.1.2 and Section 2.3.2.2).

A particular focus of attention in our work is about understanding the residual signal arising from fitting a time series model. Starting from the work of Mahler (2009), we ask if exogenous macroeconomic variables may explain, or to what extent they might explain, the residual signal. An example of this is in Section 5.3.2, a difference in behaviour between the value of Bitcoin (cryptocurrencies) and stock index is observed when we ask.

Recurrent neural networks, particularly a class of them known as Long Short Term Memory (LSTM), are popular in the literature because they can potentially capture information over long time windows. This is of particular interest for financial time series modelling because markets are known to be non-stationary; i.e. the statistical properties of the variables of interest changed over time. This in part follows from the efficient market hypothesis that arbitrage opportunities arising from any regularity in the market will be spotted by participants and will be cancelled out, and partly from interventions and policy decision arising from the political environment in which markets operate. We explore the LSTM, in which a balance between the effects of distance and recent past can be automatically learnt via its hidden state as a method for capturing such non-stationary behaviour. A review of LSTM is given in Section 2.3.2.3 and Section 6.3.1 presents empirical work on options and stock index prices.

Derivative pricing, arising as the price of contracts drawn for future execution, basically depends on how one might model the uncertainty between the present and the time of execution. There is literature on using stochastic differential equation-based approaches to pricing derivatives. Under very restrictive assumptions, the celebrated Black-Scholes model leads to a closed form solution. Where these assumptions are not applicable, simulation-based models are used to arrive at a probability distribution over the payoff of the contract at its execution. We review the approaches in Section 2.2.1.

A question for the data-driven researcher is what is the mapping between the traded price of an option contracted and the underlying parameters that relate to it. Pioneering work by Hutchinson et al. (1994) showed that an efficient approximation between them can be learned from data using a class of non-parametric models called radial basis functions. We devote a chapter in this thesis to a follow-up work building on this approach and ask if additional variables could be included to make this mapping more accurate. An intriguing outcome of this study is the empirical relationship between volume traded of an asset and its volatility. Are these correlated? Our empirical results show that even when explicit correlation between these two is absent, their effect on pricing of derivatives shows significant correlations.

Market prices of assets are determined by several factors. Past behaviour of a time series, taken in isolation, is insufficient to predict its future value. A question for the data analyst is what might be the influence of exogenous information such as macroeconomic variables on our ability to predict time series, or explain the prediction residual when fitting a time series model?

In this thesis, we work with frameworks to address this issue, devoting a substantial part of the thesis to this aspect of modelling financial data. First, we develop a model of matrix factorization that includes known exogenous variables. Sparsity constraints arising from non-negativity automatically suppress irrelevant variables in this setting (Chapter 4). We also include this use of exogenous variables into a more modern machine learning method of learning subspace representation, the Variational Autoencoder (Chapter 6). Thirdly, we use a sparse linear regression (Lasso) to approximate the residual signal of an AR process. We compare the resulting variable selection for modelling on stock index time series and a cryptocurrency time series.

1.4 Organization of the Dissertation and Contributions

We make the following contributions:

- We expand the feature set of Hutchinson et al. (1994) to improve the accuracy, including additional variables relating to the underlying asset. We go beyond state-of-the-art work on the subject and demonstrate how additional variables can be used in more accurate estimations in the work on approximating option prices. A surprising contribution from this work is a statement we are able to make about the relationship between volume traded and volatility, a topic that has attracted controversy in the literature; while the correlation between them is modest, in terms of their contribution to pricing derivatives they are highly related.
- We propose a matrix factorization method that includes known exogenous variables as additional components of subspace modelling. We expect such factorizations

to potentially uncover sector-specific drivers from among a very wide range of macroeconomic variables available. Specifically, our model represents the variation in any asset as consisting of having contributions from sector-specific components and selected macroeconomic variables.

- We construct a simple linear time series model that attempts to explain the variation in the residual signal by means of exogenous variables. We use Kalman filter to estimate the autoregressive model and a sparsity inducing linear regression with LagLasso to select relevant subsets of influencing variables to compare. We also illustrate that the influencing variables are vastly different for cryptocurrencies from stock indices (S&P 500).
- We extend the probabilistic NMF using a Variational Autoencoder (PAE-NMF) model to include exogenous variables (PAE-XNMF), we illustrate its effect on financial data of FTSE100 constituents and a set of relevant macroeconomic variables. Besides we attempted to relate the gating signals to regime switching in time series. Some encouraging preliminary results have been obtained and current work is ongoing through applying attention mechanism.

1.4.1 Publications

The work so far has resulted in publications in four peer-reviewed conference proceedings:

[1] Luis Montesdeoca and Mahesan Niranjan. Extending the feature set of a data-driven artificial neural network model of pricing financial options. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6, Dec 2016

[2] Steven Squires, Luis Montesdeoca, Adam Prügel-Bennett, and Mahesan Niranjan. Non-negative matrix factorization with exogenous inputs for modeling financial data. In *International Conference on Neural Information Processing*, pages 873–881. Springer, 2017a

[3] Luis Montesdeoca and Mahesan Niranjan. On comparing the influences of exogenous information on Bitcoin prices and stock index values. In *1st International Conference on Mathematical Research for Blockchain Economy.* Springer, 2019

[4] Luis Montesdeoca, Steven Squires, and Mahesan Niranjan. Variational autoencoder for non-negative matrix factorization with exogenous inputs applied to financial data modelling. In *11th International Symposium on Image and Signal Processing and Analysis.* IEEE, 2019

A journal manuscript building on [2] is under review, submitted to Journal of Expert Systems with Applications. Also a NMF review paper is under review, submitted to Journal of of IEEE Transactions on Neural Networks and Learning Systems.

1.4.2 Report Organization

The structure of this report is:

[a] In Chapter 2 we review the key concepts from where our techniques were applied and other related techniques.

[b] In Chapter 3 we expand the model of Hutchinson et al. (1994) and we explain the correlations between features founded.

[c] In Chapter 4 we introduce our matrix factorization method with exogenous information as additional components for modelling financial data.

[d] In Chapter 5 we compare two approaches to quantifying the role of exogenous variables.

[e] In Chapter 6 we apply two deep learning models. First, we explain the method that we introduce for performing a probabilistic NMF by Variational Autoencoders with exogenous variables. And second, we analyse if the use of a Recurrent Neural Network can respond to changes in regime, and also is performance by applying dual attention mechanism.

[f] In Chapter 7 we state conclusions and directions for future works.

Chapter 2

Review of Analytical Tools

This chapter is a review of financial concepts and various related work that use machine learning models for modelling financial data. It covers characterization of markets, financial derivatives and time series. Dimensionality reduction is also reviewed

2.1 Time Series Analytical Tools

2.1.1 Time Series Models

A signal measured in regular time steps is a 'time series' which is a sequence of time-ordered values in uniform intervals. It is useful to see changes over time on underlying assets, security or economic variables, or for comparing how it changes with other variables over the same time interval. There are various linear models for fitting time series with different stochastic processes. In the level of process, there are models that capture the correlation structure, such as: the autoregressive models (AR), the moving average models (MA), the integrated models (I), ARMA and ARIMA. They depend linearly on past data points.

2.1.1.1 ARMA Models

Autoregressive moving Average (ARMA) model is applied for predictions of future movements (Choi, 2012). It gives a description of a stationary stochastic process. Its first term is auto-regression (AR) and the second is moving average (MA). The AR(p) (autoregressive of order p) can be written as Equation 2.1.

$$\hat{x}_t = \sum_{j=1}^{p} \phi_{jt} y_{t-j}. \tag{2.1}$$

We can write it as vector notation: $\hat{x}_t = \boldsymbol{\phi}^T \boldsymbol{y}_t$. Where the past values of the time series are in $\boldsymbol{y}_t$ and the regression coefficients in $\boldsymbol{\phi}$.

The second part MA, in which observations of a random variable are modelled by the shock at and also before time t. The MA(p) (moving average of order q) model can be written as Equation 2.2.

$$\hat{x}_t = \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{2.2}$$

It means that if we observe a negative shock to the economy. It not only affects the time that it happens, it may also affect the near future. With the combination of the two models, we get the ARMA(p,q) model in Equation 2.3

$$\hat{x}_t = \epsilon_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{2.3}$$

Therefore we can think that, ARMA tells how dependent a signal is on the previous values and errors. This model is used for predictions of future movements and gives a description of a stationary stochastic process. The ARMA models have some limitations like they can be applied only for stationary time series data, and this makes it unusable for financial data that has non-stationary behaviour. Also time series with trend and seasonality are non-stationary (Faraway and Chatfield, 1998).

2.1.1.2 Autoregressive Integrated Moving Average (ARIMA)

A solution for non-stationarity cases is the ARIMA model (Box et al., 1970) which is a generalization of an ARMA model to add non-stationarity. Likewise ARMA, its parts are the auto-regression (AR) and the moving average (MA), but it has included (I) that refers to Integrated. In ARIMA a non-stationary time series is made stationary by differencing of raw observations. The ARIMA(p,d,q) is defined as follows: (p), are the number of lags, (d) is the degree of differencing, and (q) the size of moving average.

$$\big(1 - \sum_{i=1}^{p} \alpha_i L^i\big)(1 - L)^d y_t = \big(1 + \sum_{j=1}^{q} \theta_j L^j\big)\epsilon_t, \tag{2.4}$$

where d is the differencing level in which most of the cases is equal to 1 and L is the lag operator. When d is equal to 0, the ARIMA model is reduced to an ARMA (p,q) model. Similarly with ARIMA(p,0,0) is just a (AR)model and ARIMA(0,0,q) is the MA model. But when it is ARIMA(0,1,0), it becomes a Random Walk.

2.1.2 Estimating AR - Least Squares, Kalman Filter

2.1.2.1 Least Squares Estimation

Least squares considers that the curve with minimal sum of deviations is the best fit curve, e.g. least square error from a given set of data. A mathematical relationship is found between the time factor and the data points, $(x_1, y_1), \dots, (x_n, y_n)$, where x as independent variable and y_t the dependent variable, the expected values. The least squares estimator has the form $f(x, \beta)$ where the overall solution is held in the vector $\boldsymbol{\beta}$ that minimizes the fitting curve $f(x)$ with deviation (error) $r_i = y_i - f(x_i, \boldsymbol{\beta})$:

$$\sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - f(x_i, \boldsymbol{\beta}))^2 \tag{2.5}$$

Least squares problems have two types: linear or ordinary least squares (OLS) (Equation 2.6) and non-linear least squares. It depends on the residuals if they are linear for each unknown. The OLS occurs in regression analysis with a closed-form solution. The non-linear type is by interaction, in which each iteration the system is approximated by a linear one.

$$f(x, \beta) = \sum_{i=1}^{n} \beta_i \alpha_i(x), \tag{2.6}$$

where α_i is a function of x allowing to $X_{ij} = \alpha_i(x_j)$. The $\boldsymbol{\beta}$ is given by:

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T y. \tag{2.7}$$

2.1.2.2 The Simple Kalman Filter (KF)

Kalman Filter is used for filtering out noise from data and as predictors. They are applied to noisy signals like in finance. The Kalman filter, introduced by Kalman (1960), is a variance minimizing algorithm which updates the state estimate when information arrives, which enables processing in real-time. KF is widely applicable to applications that make linearity assumptions.

Let $\hat{x}_t$ be the observations of the model at time t and let $\boldsymbol{\theta}$ be a hidden random vector (unobserved variables). Thus we can use the equation 5.2 and 5.3:

$$\hat{x}_t = \boldsymbol{y}_t^T \boldsymbol{\theta}_t + \mathrm{v}_t, \tag{2.8}$$

where $\boldsymbol{y}_t$ is known and v_t is zero-mean Gaussian white noise with covariance R. $\boldsymbol{\theta}_t$ is given by:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{w}_t \tag{2.9}$$

where $\mathbf{w}_t$ is zero-mean Gaussian white noise with covariance $\boldsymbol{Q}$, $\mathbf{w}_t \sim \mathcal{N}(0, \boldsymbol{Q})$.

The Kalman filter is a process that recursively estimates the coefficients of the model represented by a vector $\boldsymbol{\theta}$ based on the values of the previous days coefficients, an uncertainty matrix adjusted every time step, and some tuning parameters that model the error.

The Kalman filter equations are given by:
Prediction step:

$$\boldsymbol{\theta}_{t|t-1} = \boldsymbol{\theta}_{t-1|t-1} \tag{2.10}$$

$$\boldsymbol{P}_{t|t-1} = \boldsymbol{P}_{t-1|t-1} + \boldsymbol{Q} \tag{2.11}$$

Correction step:

$$r_t = \hat{x}_t - \boldsymbol{y}_t^T \boldsymbol{\theta}_{t|t-1} \tag{2.12}$$

$$\boldsymbol{k}_t = \boldsymbol{P}_{t|t-1} \boldsymbol{y}_t (\boldsymbol{y}_t^T \boldsymbol{P}_{t|t-1} \boldsymbol{y}_t + \boldsymbol{R})_t^{-1} \tag{2.13}$$

$$\boldsymbol{\theta}_{t|t} = \boldsymbol{\theta}_{t|t-1} + \boldsymbol{k}_t r_t \tag{2.14}$$

$$\mathbf{P}_{t|t} = \big(\mathbf{I} - \boldsymbol{k}_t \boldsymbol{y}_t^T\big) \mathbf{P}_{t|t-1} \tag{2.15}$$

where the signal modelled is r_t, from the vector of past values $\boldsymbol{y}_t$. $\boldsymbol{\theta}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$ are predictions of the parameters and error covariances in them respectively. $\mathbf{I}$ is the identity matrix. $\boldsymbol{R}$ and $\boldsymbol{Q}$ tune the entire Kalman filter.

The Kalman algorithm makes a prediction of the signal and calculates its residual error r_t by predicting $\boldsymbol{\theta}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$ from $\boldsymbol{\theta}_{t-1|t-1}$ and $\boldsymbol{P}_{t-1|t-1}$. Then we use this residual with the term *Kalman gain* ($\mathbf{k}_t$) in the posterior updates. We perform some empirical studies, applying the Kalman filter in Section 5.4

2.1.3 Non-linear Models

2.1.3.1 Extended Kalman Filters (EKF)

The extended Kalman filter is a non-linear version of Kalman filter. The state transition and observations models are not required to be linear functions of the state. However, they need to be differentiable functions.

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \mathbf{w}_t, \tag{2.16}$$

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \mathbf{v}_t, \tag{2.17}$$

where $\boldsymbol{u}_t$ is the control vector, $\mathbf{w}_t$ and $\mathbf{v}_t$ are zero-mean multivariate Gaussian noises with covariance matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ respectively. The function f predicts the state from the previous estimate and h predicts the measurement from the predicted state. The functions f and h compute a matrix of partial derivatives.

EKF is a set of equations that takes the underlying process model for making the estimation on the current state and after that corrects this estimation. It linearises a model about a working point by a first-order Taylor Series. In other words, the EKF transforms the non-linear models at each time step into linearized systems of equations. Then the linearized equations apply the standard Kalman filter. Therefore the update equations are the same but with the difference of Equation 2.18:

Correction step:

$$r_t = \hat{x}_t - h(\boldsymbol{\theta}_{t|t-1}) \tag{2.18}$$

The performances of a modified KF depend on the considered system. Poor state estimation is a result of high non-linearity. Particle filter (PF) is more suitable for non-linear system or with non-Gaussian noise because PF neither requires the system to be linear nor assumes that the noise is Gaussian (De Bernardis et al., 2016).

2.1.3.2 Particle Filters (PF)

Particle filters are a Monte Carlo technique for the solution of the state estimating problem (Arulampalam et al., 2002). PF is an useful tool for a variety of problems. Its aim is to represent the required posterior density function by a set of random samples (particles) and its associated weights. These samples and its weights are used to compute the estimates of internal states in dynamical system. When the number of samples increases, this Monte Carlo representation becomes an equivalent representation of the posterior probability function, and the solution approaches the optimal Bayesian estimate (De Bernardis et al., 2016).

2.2 Derivatives Pricing Methods

A derivative in finance is a contract or security between two parties, in which its price is a result of the performance or fluctuations of the underlying assets. The most popular financial derivative is an option, a contract between two parties to sell or buy a financial instrument such as stock, index, futures or currency at an agreed price. The prices accorded between the parties is called the *strike price* and it can be exercised at specified time (*time to maturity*). There are two agreements in option contracts, one is *Call* and another *Put*. Those give the right but not an obligation of buy or sell stocks, during the time in which their values increase or decrease. In the market, there are several kind of options such as the American options, the European options and Exotic options. European options are different from American options because they can be exercised at their time of maturity (the time when they expire). These options tend to be sold for less than their nominal or part value comparing with American options.

There several options pricing techniques but the world's most well-know options pricing model is the Black–Scholes.

2.2.1 Black–Scholes Model (BS)

Black and Scholes (1973) published a formula for calculating the theoretical price of option contracts. Their conceptual idea focused on creating a portfolio without risk, taking positions in bonds, underlying stock and options.

This model makes the following assumptions:

- There are no transaction costs for purchasing or selling.
- It can only be exercised at the time of maturity.
- Effects of dividends are not considered when is paid out at the time of the life of the option.
- Volatility of the underlying assets and its risk-free rate are constant and known
- No restriction on short selling and no taxes.

The option contract in this model depends on the following variables:

$$\frac{\delta V}{\delta t} + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2} + rS\frac{\delta V}{\delta S} - rV, \tag{2.19}$$

where the asset price is S. The volatility of underlying asset is σ and the risk-free interest rate is r.

This formula for pricing *Call* C_t and for *Put* options P_t is (Hull, 2011):

$$C_t = S_t N\big(d_1\big) - N\big(d_2\big)Ke^{-rt_m} \tag{2.20}$$

$$P_t = -S_t N\big(-d_1\big) - N\big(-d_2\big)Ke^{-rt_m}, \tag{2.21}$$

where the strike price desired is K and the cumulative normal distribution function is $N()$. The time to maturity is t_m, when it expires it is expressed as percent of a year. C

and P are the *Call* and *Put* price of the option respectively. The values of d_1 and d_2 are given by Equation 2.22 and 2.23 respectively:

$$d_1 = \frac{\ln(S_t/X) + (r + \sigma^2/2)t_m}{\sigma\sqrt{t_m}} \tag{2.22}$$

$$d_2 = \frac{\ln(S_t/X) + (r - \sigma^2/2)t_m}{\sigma\sqrt{t_m}} \tag{2.23}$$

where σ is the volatility.

So far, we have seen the most basic type of options which are European *Call* and *Put* options. They have simple features and payoffs. However, Korn et al. (2001) describe that in the financial markets we can find more complicated payoff, such as: Barrier, American and Asian options (know as exotic options). Thus for these cases, the only way to price these options are by numerical methods.

2.2.2 Binomial Tree Method

This method introduced by Cox et al. (1979) is widely used by market professionals as an options valuation method. It assumes a high number of small binomial movements in the stock price movements. This model separates the time of life of an option to subintervals of time with a specified magnitude and stock price at the beginning. Then this model will assume with a probability value that the movements of the price will go up or it will decrease at the end of each period with a probability value. The value of the risk-free interest rate will be the expected return for all traded options due to the principle of risk-neutral validation. Thus discounting the expected values of interest rate with risk-neutral, we can value future cash flows.

2.3 Machine Learning Models

Machine learning methods usually fall into two groups: supervised and unsupervised learning. Most problems we consider in this work, fall under the category of supervised learning, where a target signal is available (e.g. linear/non-linear regression). Hence we review several supervised learning methods in this section and briefly mention the unsupervised learning method of K-means clustering at the end. We also consider matrix methods for dimensionality reduction which form the major part of this work. More elaborate methods such as semi-supervised learning, where part of the data has targets associated with it and parts do not, are not considered here.

2.3.1 Basic Machine Learning Models

We first mention some of the basic machine learning model that are widely used from the financial practitioners (SVR, Decision Trees and Random Forest).

2.3.1.1 Support Vector Machine for Regression (SVR)

Support Vector Machine is a powerful non-linear model that is known for its high performance in several studies. Not only it can be used for classification, but also for regression. It projects the data in higher dimension where data can be linearly separable. Some studies have applied SVR to financial data, such as Cecchini et al. (2010) who use SVRs to detect corporate management fraud. Besides Société Générale, a French bank, indicate that SVMs help to make better equity investment decisions in the long and short term. Also, Härdle et al. (2007) employ SVR to get the default risk of some companies. A last example is Härdle et al. (2011) that use SVMs extract the necessary information from financial balance sheets for predicting the company solvency.

2.3.1.2 Decision Trees and Random Forests

Decision Trees have a logic that is simpler to understand due to their binary tree structure. It divides the data into two nodes to ask a binary if/else question. It is used for example to analyse how financially healthy is the company, in which after training, a single node contains the healthy companies and the another the high risk ones. Thus it helps to trace many question of the company features.

Breiman (2001) introduced the Random Forest technique. It is composed of decision trees. Random Forest has been applied for forecasting. It also helps in the importance classification of a variable when a decision tree is been constructed. Some studies state that Random forest is a robust method that permits noise and outliers in the training set (Yeh et al., 2014). Several financial studies have used it such as (Medeiros et al., 2019), who established that Random Forest is better to indicate a degree of non-linearity in the dynamic inflation. Another area that Random Forest has been applied is in banking, for prediction of bankruptcy (Barboza et al., 2017).

2.3.1.3 K-Means Clustering

Clustering has been applied in finance, which helps to define which kind of returns and trends are in a time series. K-means, an example that partitions a dataset into an user specified number of clusters (Wu et al., 2008). Given a set of data points and an integer k, the k-means algorithm randomly selects k centroids and seeks to minimize an objective function between each data point and its nearest centroid producing k clusters. The advantages of k-means include its simplicity, ease of implementation and applicability to a wide range of problems (Xu and Wunsch, 2005). The disadvantages include: its inability to detect outliers or the appropriate number of clusters, and inefficiently selecting initial centroids.

2.3.2 Neural Networks Models

Neural Networks are very useful tools to find the relationship between inputs and outputs. Most of the Neural Networks are trained by **Backpropagation** which tunes the weights for better prediction accuracy. It is a form of learning where you learn the correct answer by getting feedback every time. In other words, a recursive method for calculating the weights that trains the network until it is capable of performing the task with better accuracy.

2.3.2.1 Radial Basis Function Networks

Radial basis function is a learning network which evaluates the length of the data with their centers c that are also called *cluster*.

The formulation for Radial Basis Functions is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} c_i \times h_i(\|\boldsymbol{x} - \boldsymbol{z}\|) + p(\boldsymbol{x}) \tag{2.24}$$

where:

- $\boldsymbol{x}$ is the vector with d inputs $x_1, x_2, \ldots, x_d$
- $\boldsymbol{z}$ are the d-centers.
- $\|\cdot\|$ is the vector norm
- $p()$ is a polynomial function
- c_i are the coefficients and
- h_i are weights to be determined

RBF Networks have a different approach from other neural networks like Multi-Layer Perceptron, which learns to approximate functions with a hidden layer of sigmoid units.

The characteristics of the RBF networks are:

- They are feed-forward networks with two layers.
- In the hidden nodes, the radial basis functions are implemented, such as Gaussian functions
- In the output, linear summation functions are implemented
- The training is fast because it first determines the weights that connect input layer to the hidden layer and after that determines the weights from hidden to output layer.

2.3.2.2 Recurrent Neural Network (RNN)

A RNN is a normal neural network that includes a feedback again into the input (self-connected hidden layer which spans time points). It is also considered as deep learning where its depth on time steps.

RNN keeps the signals of the previous input and feeds it into the current calculations unlike feedforward networks. They were introduced by Hopfield (1982) and has been applied to financial data (e.g. (Hsieh et al., 2011) and (Giles et al., 1997)).

RNNs are trained by *backpropagation through time* (Werbos, 1990), with some limitations for long time steps because the gradient tends to exploit or vanish (Bengio et al., 1994). Another popular type of RNN that was introduced to overcome that limitation is the Long Short Term Memory (LSTM) network.

2.3.2.3 Long Short Term Memory (LSTM)

LSTM was introduced by Hochreiter and Schmidhuber (1997). They can learn long-term dependencies, overcoming the problem of vanishing or exploiting the gradient (Sak et al., 2014). LSTM networks are composed of a hidden layer called a memory cell. Each memory cell has three gates which maintain and adjust its cell state $\boldsymbol{C}_t$, and feeds into itself across time steps. These gates are: a forget gate $\boldsymbol{f}_t$, an input gate $\boldsymbol{i}_t$, and an output gate $\boldsymbol{o}_t$. This structure is shown in Figure 2.1.

The main part of the LSTM is the cell state $\boldsymbol{c}$, which is the red line in Figure 2.1. It passes information through all the chain with minor linear interactions and without changes. The LSTM is formed by three gates that are composed out of a sigmoid neural net layer.

The first gate in the LSTM is the forget gate which decides by a sigmoid layer what information will not be taken account from the cell state. It receives $\boldsymbol{y}_{t-1}$ and $\boldsymbol{x}_t$, the Equation 2.25 correspond to this gate.

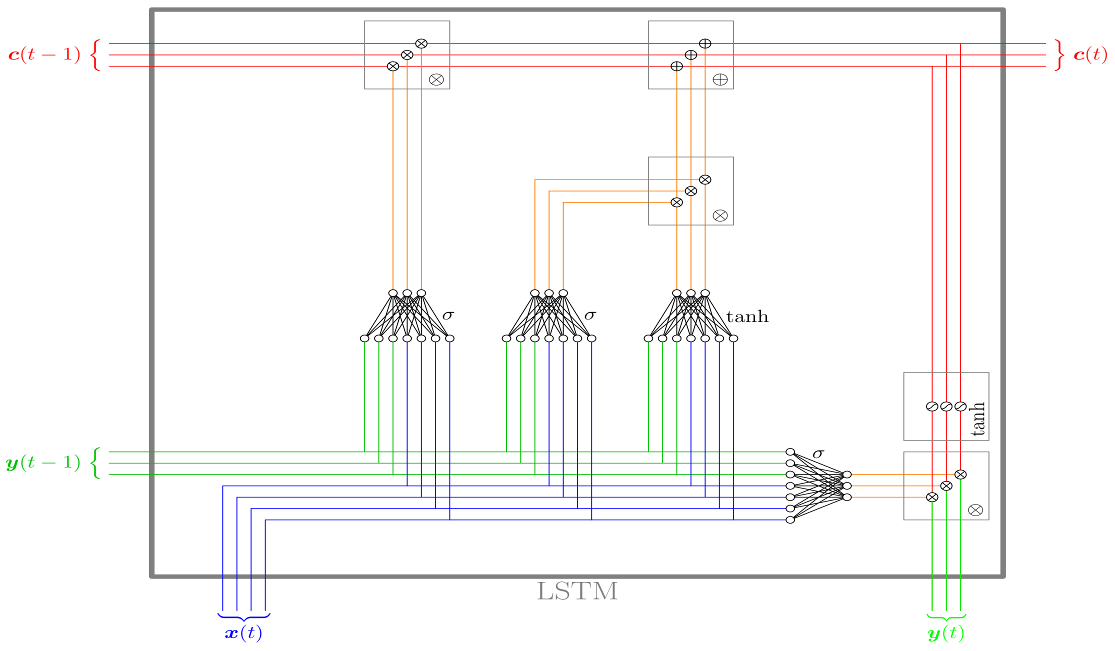


FIGURE 2.1: The LSTM structure. Here is illustrated the gating mechanism that LSTM has to get longer memory. The vector $\boldsymbol{c}$ represents the memory cell that is fed into itself across time steps t. The vector $\boldsymbol{x}$ is the input and $\boldsymbol{y}$ the output of the unit. This graph was made by Prof. Adam Prugel-Bennett from our machine learning journal club at University of Southampton

$$\boldsymbol{f}_t = \sigma\big(W_f \cdot \big[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\big] + b_f\big) \tag{2.25}$$

The next part is the input gate, which is composed of two parts. First, it determines what information will be updated. The next part is a hyperbolic tangent layer that creates a vector that has the new candidate values (Equation 2.27).

$$\boldsymbol{i}_t = \sigma\big(W_i \cdot \big[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\big] + b_i\big) \tag{2.26}$$

$$\tilde{c}_t = \tanh\big(W_c \cdot \big[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\big] + b_c\big) \tag{2.27}$$

The new memory cell state $\boldsymbol{c}_t$ is calculated by multiplying $\boldsymbol{c}_{t-1}$ with the value from the forget gate, $\boldsymbol{f}_t$, and it creates the new candidate values.

$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t \tag{2.28}$$

In the last part indicates which part of the cell state will be shared and multiplied with the values from the sigmoid layer to create the hidden state values.

$$\boldsymbol{o}_t = \sigma\big(W_o \cdot \big[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\big] + b_o\big) \tag{2.29}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh\left(\boldsymbol{c}_t\right) \tag{2.30}$$

LSTM has various successful applications such as: handwriting recognition, in which it generates sequences with long-range structure by predicting one data point at a time (Graves et al., 2013). A further example is in translation, in which LSTM achieves a high Bilingual Evaluation Understudy (BLEU) score task translating from English to French (Sutskever et al., 2014).

2.3.2.4 Gated Recurrent Unit (GRU)

The idea of gated recurrent unit (GRU) layer was proposed by Cho et al. (2014), and it is quite similar to the long short term memory (LSTM) layer which also has gating units but without including a separate memory cell. GRU is composed by two gates, one called "reset gate r" which determines the combination between the new input and the previous memory. And the "update gate $\boldsymbol{z}$" which defines the quantity of previous memory to be shared. As previously mentioned, it does not include an internal memory $\boldsymbol{c}_t$ that LSTM has in its structure.

The update gate is:

$$\boldsymbol{z} = \sigma\left(\boldsymbol{x}_t U + \boldsymbol{s}_{t-1} X\right) \tag{2.31}$$

$$\boldsymbol{r} = \sigma\left(\boldsymbol{x}_t U^r + \boldsymbol{s}_{t-1} X^r\right) \tag{2.32}$$

$$\boldsymbol{h} = \tanh\left(\boldsymbol{x}_t U + (\boldsymbol{s}_{t-1} * r) W\right) \tag{2.33}$$

$$\boldsymbol{s}_t = (1 - \boldsymbol{z}) * \boldsymbol{h} + \boldsymbol{z} * \boldsymbol{s}_{t-1} \tag{2.34}$$

Unlike LSTM, the GRU unit does not control the flow of information by using a memory unit. It uses all hidden states directly without any control. GRUs are slightly faster to train for having fewer parameters than LSTM. But, with large data, the LSTMs with more expressiveness may get better performance. Regularization technique penalize the loss function of the weights vector w by adding a L_1 or a L_2 norm.

A common problem for machine learning models is overfitting, which means that the model performs well in training data but not in test data. Hence regularization is a strategy useful for reducing the test error in which many studies have been working on, see e.g. Goodfellow et al. (2016).

2.4 Dimensionality Reduction

Dimension reduction helps to project the whole data into a lower dimensional space, however it gets more difficult when there are more features.

2.4.1 Principal Component Analysis (PCA) and Independent Component Analysis (ICA)

A popular method for dimensionality reduction is Principal Component Analysis. PCA is sensitive to the scaling of each variable. Initially, PCA was introduced as an analogue of the principal axis theorem in mechanics by Pearson (1901) and then know as PCA by Hotelling (1933). PCA uses an orthogonal transformation in which a set of observations of possibly correlated variables are converted into a set that is not linearly correlated. The new features obtained by PCA are linear combinations of the original features, through the projection of the data to the first few principal directions. The first principal component gives the direction in which the variability is the highest in the observations. Then the second components are found by the linear combination of predictors uncorrelated with the first principal component. One noticeable drawback of PCA is that a clear financial interpretation of the feature space might be lost; each new feature after PCA is a linear projection of many features (e.g. financial ratios). Despite of this disadvantage, PCA has been applied in many studies using financial data. For example, Yu et al. (2014) introduced a method that brings PCA into the support vector machine (SVM), which improves the training accuracy for construction of a stock selection model, by extracting the low-dimensional and efficient feature information. Li and Khashanah (2015) used PCA to confirm the volatility day pattern, and to use eigenvectors as weight in distance metric in the clustering step, for generating and forecasting volatility.

Another method that has been used is Independent Component Analysis (ICA), in which a multivariate signal is separated into independent components under the assumption that the components are statistically independent in non-Gaussian distribution. Initially, with financial data, ICA was applied in tick-by-tick foreign exchange (FX) price series to get the true price by separating the observational noise (Moody and Wu, 1997). Also it has been applied for bankruptcy prediction (Chen and Vieira, 2009), time series analysis (Coli et al., 2005) and clustering (Guo et al., 2008).

2.4.2 Non-Negative Matrix Factorization (NMF)

NMF was invented by Paatero and Tapper (1994) under the name positive matrix factorization (PMF), who states that PMF produces better fit to the data than the customary factor analysis (principal component analysis (PCA)). However it started to be popular and called Non-negative matrix factorization (NMF) when Lee and Seung (1999) published useful algorithms and demonstrated that NMF is able to learn parts of faces from an image and semantic features of text. NMF is a model able to learn a parts-based representation by imposing non-negativity constraints that allow only non-subtractive combinations.

A range of areas have applied NMF, for example, bioinformatics (Tjioe et al., 2008), pattern discovery (Brunet et al., 2004), chemometrics (Gao et al., 2005), computer vision (Shashua and Hazan, 2005), music (Smaragdis and Brown, 2003), text mining (Pauca et al., 2004), pattern recognition (Liu et al., 2006), document clustering (Xu et al., 2003), etc. An important area that NMF has been applied is financial market where it is used for understanding the hidden components which drive the system. Stock market and individual stock prices are determined by fluctuations in underlying factors (signals), which are unknowns. Academics and practitioners of assets management have studied NMF in financial data to learn the components which drive the stock market. The aim is to avoid huge investment losses, due to the obstacle in stock modelling and forecasting lies in our inability to discover the true data generating process (Allen and Morzuch, 2006).

In standard NMF the aim is to find a lower-dimensional representation of the data. Given $\mathbf{V}$ with non-negative elements, with m observations and dimension n, let data be $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n] \in \mathbb{R}_{\geq 0}^{m \times n}$, NMF seeks decompose $\mathbf{V}$ into a non-negative $m \times r$ basis matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_r] \in \mathbb{R}_{\geq 0}^{m \times r}$ and non-negative $r \times n$ coefficient matrix $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_r] \in \mathbb{R}_{\geq 0}^{r \times n}$, such that

$$\mathbf{V} \approx \mathbf{WH} \tag{2.35}$$

Where $\mathbb{R}_{\geq 0}^{m \times n}$ stands for the set of $m \times n$ element-wise non-negative matrices. Thus NMF creates a new representation of the data in a significantly reduced subspace (r).

The classic and more practical approach for finding efficient and effective solutions to NMF for non-negativity constraint is to perform alternating minimization of a suitable cost function as the similarity measures between $\mathbf{V}$ and the product $\mathbf{WH}$. Where a similarity measure of $D(\mathbf{V}||\mathbf{WH})$ can be defined. We can use distances or divergences to quantify the difference between the original data $\mathbf{V}$ and the approximation $\mathbf{WH}$.

NMF is not convex, and its objective function can be a sole cost function or a set of cost functions with same global minima to be minimized sequentially or simultaneously. The most used objective functions are Square of Euclidean Distance (SED) and Generalized Kullback–Leibler Divergence (GKLD). Most of the NMF algorithms use a two-block coordinate descent scheme with different block sizes and various optimization approaches for each block. They optimize $\mathbf{W}$ or $\mathbf{H}$, while keeping the other fixed. This is mainly because the subproblem in one factor is convex. In other words it is a non-negative least squares problem (NNLS). Given the Square Euclidean Distance function of two factor matrices, these procedures perform a constrained minimization with respect to one matrix while holding the other matrix fixed; and then the minimization is performed again with the roles of the matrices reversed as follows:

$$\min_{\boldsymbol{W}\geq 0} D(\boldsymbol{V}||\boldsymbol{W}\boldsymbol{H}) = \min_{\boldsymbol{W}\geq 0} \quad \frac{1}{2}||\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}||^2_{Fro}, \tag{2.36}$$

$$\min_{\boldsymbol{H}\geq 0} D(\boldsymbol{V}||\boldsymbol{W}\boldsymbol{H}) = \min_{\boldsymbol{H}\geq 0} \quad \frac{1}{2}||\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}||^2_{Fro}, \tag{2.37}$$

The Frobenius norm where $||\mathbf{V} - \mathbf{WH}||^2_{\text{Fro}}$ is minimised. There are many of algorithms for finding the matrices $\mathbf{W}$ and $\mathbf{H}$. An early version called multiplicative updates was introduced by Lee and Seung (1999), which effectively minimises an objective function. It is described in Equation 2.38:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{[\mathbf{H}\mathbf{V}^T]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]}, \tag{2.38}$$

where $\odot$ and $\frac{[]}{[]}$ are the element-wise multiplication and element-wise division respectively.

Lee and Seung (2001) claimed that the multiplicative update rules converge to a local minimum and to accelerate the convergence rate, one popular method is to apply gradient descent algorithms with additive update rules.

In NMF the selection of rank r is tricky. Many approaches try different values of r and pick the one performing best for the application at hand; more can be found in Kanagal and Sindhwani (2010). This r is the size of the new subspace, which selects how many features will be extracted from the data. A mechanism for selecting the subspace size by using a minimum description length technique is proposed by Squires et al. (2017b). They also demonstrate that this technique provides plausible estimates of the FTSE 100 dataset r-value, which gives a value of 8, which can be consider the underlying trends, and this value is close to the 10 sectors of which the FTSE 100 is composed.

2.4.3 Variational Autoencoder (VAE)

The Variational Autoencoder (VAE) (Kingma and Welling, 2013; Reed et al., 2014) learns the marginal likelihood of the data in a generative process, for a dataset $\mathbf{x}$ of samples from a distribution. This can be compressed as Equation 2.39 in order to understand the underlying causal relations.

$$\max_{\phi,\theta} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \tag{2.39}$$

where $\mathbf{z}$ are the ground truth generative factors, ϕ and θ parametrise the distributions of the VAE encoder and the decoder respectively. VAE models the conditional $p_\theta(\mathbf{x}|\mathbf{z})$ as a function approximator and uses a variational approximation $q(\mathbf{z}|\mathbf{x})$ of the posterior.

$$\log p_\theta(\mathbf{x}|\mathbf{z}) = D_{KL}\big(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\big) + \mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}) \tag{2.40}$$

where $D_{KL}(\|)$ stands for the non-negative Kullback–Leibler divergence between the true and the approximate posterior. Hence, maximising $\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z})$ is equivalent to maximising the lower bound to the true objective in Equation 2.41

$$\log p_\theta(\mathbf{x}|\mathbf{z}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\big) \tag{2.41}$$

The prior $p(\mathbf{z})$ and posterior $q_\phi(\mathbf{z}|\mathbf{x})$ distributions are parametrised as Gaussians with diagonal covariance matrices. And the prior is set to the isotropic unit Gaussian $\mathcal{N}(0, 1)$. This way of parametrising the distributions allows to use the *reparametrisation trick* to estimate gradients of the lower bound with respect to the parameters ϕ, where each random variable $z_i \sim q_\phi(z_i|\mathbf{x}) = \mathcal{N}(\mu_i, \sigma_i)$ is parametrised as a differentiable transformation of a noise variable $\epsilon \sim \mathcal{N}(0, 1)$.

2.5 Review of NMF Applied to Financial Data Problems

2.5.1 Banking

Currently, in the banking industry loans have a fundamental role, where the financial institutions need to validate the credit rate before granting loans. If customer validation is not properly monitored and controlled, resources will be wasted (loans will not be paid). Thus, with better validation, the number of bad loans will drop, granting loans will be faster and bank costs will be significantly reduced. There are many data mining techniques to predict and validate bank customers.

2.5.1.1 Credit Scoring

Credit Risk Analysis aims to identify the risk level when a customer is granted a loan (Wah and Ibrahim, 2010). Currently, practitioners are applying models based on statistical or operation research methods. These models are built with historical payments information from thousands of current clients. These models have the objective to assign to applicants either *good credit* class or *bad credit* class. The first class (good credit) corresponds to those that are likely to pay back their financial obligations and the second class are those that have high probability of defaulting, hence their credits should be denied.

Nevertheless, credit assessments have high dimensional data, which brings a problem for classification training. The models for credit scoring take the relationship between the historical information and future credit performance. Hence it brings the need of using a dimensionality reduction model to solve this problem. For example, NMF was used via a projected gradient method for credit risk assessment of financial intermediaries before a Support Vector Machine (SVM) classifier is trained (Chen et al., 2013). Thus the SVM classifier then predicts on the new data. The output is the decision value of each instance. They used a dataset of private credit card operation from Brazilian retail chain with 50 000 instances. Each credit card is labelled as good and bad, and contains 132 features. They showed that this method, successfully transformed the data and it improved the SVM classification accuracy (85.27%). It was compared to several benchmark model on dimensionality reduction, such as PCA (74.440%), ICA (77.44%) and LDA (81.44%). It also found direction where the classes are better separated. However, they did not give details of the NMF structure to identify the most relevant features, neither how many of them were.

Two years after, Sun et al. (2015) found similar results by adding sparsity. Their aim is to increase the selection of relevant features that contribute to the performance of credit risk classification. They labelled the data in positive and negative, which consist of private label credit card from the American retail chain with 560 000 instances that have 300 features each one. They used accuracy, precision, and Type I error rate as measures. This method successfully transformed this data and improved the SVM classification accuracy (88.789%), compared with some benchmark methods, such as PCA (74.240%), ICA (77.430%), LDA (82.44%). However, they did not explain if the applied sparsity in both matrices or only in $\mathbf{H}$, neither the amount of sparseness applied.

2.5.1.2 Financial Distress Data

Corporate bankruptcy prediction models have been growing on interest due to important cases of banks defaulting from the last financial crisis. Ribeiro et al. (2009) showed that NMF learnt local features from firms that were label in distress and was able to encode factors that can be used for bankruptcy prediction. It extracted discriminative features and then combined them with learning classifiers for distress or bankruptcy detection, to encode the most useful factors. They highlighted the fact that NMF as feature extraction includes interactions and correlations between features which are omitted in others selection methods. Their data contains financial statements of 60 000 industrial French companies and financial ratios spanning from 2002 to 2006. In this database, around 3 000 were distressed companies (declared bankrupted), and it included information about 30 financial ratios, with 90 features. They made the initialization by K-means because it gives slightly better results compared to random, with an optimal rank of 25. They used a KNN classifier in which the performance improves by around 5% with *K-Means* initialization of W and with *SVM* with slightly better by around 1%. Furthermore, Ribeiro and Chen (2011) extended this work by introducing a weighted graph subspace learning method. They aimed to get compact representation of data and to make subsequent processing. The Graph regularized Non-Negative Matrix Factorization (GNMF) model (Cai et al., 2011) was used to create a weight graph matrix and for learning the subspace models that ease bankruptcy prediction. Their dataset consists of financial statements of 12 000 French companies, in which 600 declared bankruptcy and the remainder are healthy. Thus, this model predicts if the company is healthy or bankrupt with higher accuracy.

2.5.2 Stock Market

In the stock market, underlying forces determine stock prices, but they are unknown factors. NMF has been used to help to find those underlying factors, such as Liu, Tang (2009). They decomposed a matrix formed of 40 daily closing prices of stocks from Shenzhen component index with a length of 19 years. They applied the multiplicative update (MU) rules algorithm and created clusters with the center of the underlying forces by Kmeans. Those values considered the trends of the stock market and drove the classification of all the stocks in different clusters. Additionally, they analysed how k (dimensions) affects the classification and found that in every k, there is a number of stocks that are allocated in the same cluster which have similar performance. These findings are similar to an early work in de Fréin et al. (2008b) that will be explained in Section 2.5.2.1.

Finding correlations between the stock market and news is another interest. Some studies consider that returns have correlation with newspaper content (Campbell and Hentschel, 1992; Yermack, 1997; Atkins et al., 2018; Gidofalvi and Elkan, 2001; Chan, 2003; Tetlock, 2007). Matrix factorization methods have been applied to help to find this correlation. For instance, Ming et al. (2014) states that an unified latent factor model can characterize a joint correlation between stock prices and newspaper content. This model uses a sparse matrix factorization (SMF) based on news stories from the Wall Street Journal (WSJ) for making predictions on the direction of individual stock price movements. To solve the model learning that was formulated as a SMF problem, Alternating Direction Method of Multipliers (ADMM) was applied. ADMM is an algorithm that breaks convex optimization problems into smaller pieces which are then easier to handle for solving them (Boyd et al., 2011). Their technique is defined by a non-negative feature vector, which is a latent factor model that describes a stock (for example, it belongs to energy sector), with the daily average investor mood and the stock price. They used daily trading data from 1/1/2008 to 9/30/2013 that is fixed to 10 latent factors for predicting about 553 stocks from the S&P 500, DJIA and Nasdaq indexes. They quantify the stock news from a document by counting daily mentions of the top 1000 words with highest frequency and the name of the companies from the all 553 stock. They worked with 1354 words. Their method is $\mathbf{R} = \mathbf{UWY}$, where $\mathbf{R}$ is the return, $\mathbf{U}$ are the latent variables and **WY** are the latent variables for the daily news. They got an accuracy of 55.7%, which performs better than all baselines (Previous price 46.9%, previous return 49.1%, Autoregressive (AR) models with 10 lags of previous price 49.5%, AR with lags 10 of returns 50.9% and regression on the previous day's price 51.4 and regression on return 50.8%) in terms of directional accuracy.

Furthermore, Sun et al. (2016) extended Ming et al. (2014) work to use with textual information from user-generated microblogs, with the aim of finding correlations of the stock price direction and social media content. They did not consider news articles, instead they adopted the market information that is contained in high volume social media from Stock-Twits streams, which are high quantity but low quality sources of text information. This model was tested on 4 years of information with 40 features content of 420 stocks from S&P 500. As a result, it outperforms by a percentage point in all years a baseline regression (Previous return/price, Autoregressive (AR) models and Random). It supports the idea of Ming et al. (2014), that SMF methods can be used for predicting the stock market using text mining. They were able to get market indicators from StockTwits streams to predict the stock price movements and showed a negative correlation between the word count and price.

2.5.2.1 Underlying trends and Clustering

Researchers working with financial data are trying to identify the underlying trends from the stock market. The stock prices have fluctuations that no behave independently of each other, they are driven by latent factors. For instance, de Fréin et al. (2008b) applied NMF to identify underlying trends in the stock market data. By their equation $\mathbf{y} = \mathbf{AX}+\mathbf{V}$, They determined the clustering by examining the rows of $\mathbf{A}$ and deciding which rows have the dominant coefficients at the same positions in the columns, and then group them by K-means. However, the clusters did not depend on the NMF parameters and they have not compared the effect of clustering with other models. The main statement in their work is that diversified portfolios are recommended to not be based on the sectors where the stocks belongs, instead it should be considered by their underlying trends. Their experiments were on 30 stocks of the Dow Jones Industrial Index from the past 20 years. As a result, the clusters created with the NMF did not consist of stocks that belong to the same sector. It can be potentially used to guide on portfolio diversification.

Creating and managing portfolios of financial assets that can match investments to objectives is a complicated work. A suggested solution for this is by the development of low risk portfolios, which can be led by diversification theory. Markowitz introduced the principle of attempting to maximize expected return, subject to a risk value or for minimizing the risk depending on the return value given (Markowitz, 1952). The objective is to group data into subgroups with similar behaviour. It could be considered as a clustering problem, and it obtains ratios of low risk-reward. NMF has been analysed on how it may help for better diversification. For instance, Wang in 2012 organised the stocks in groups which were based on their association with the underlying trends that were extracted by applying NMF (Wang, 2012). They added two properties to derive a variant of NMF, which are: the smoothness of trend matrix **W** and the rowwise sum-to-one of weight matrix **H**. With the aim of finding patterns between parameters and trends. This work was run on US blue chip historical stocks prices. Furthermore, they extracted two trends which displayed opposite oscillations. Thus, it may be possible to create portfolios of stocks that behave differently to market changes. Another observation is that the smoothness of the trend matrix had weaker influence than the sum-to-one constrain enforcer on the weight matrix. From the factorization was gotten a clustering in which can be allocated assets to create a diversified portfolio with a well balanced risk.

K-means can not establish the effectiveness and coherence of cluster when is applied with stock data (Cai et al., 2016). It does not handle the noise because tends to find spherical clusters. A way to overcome this issue is matrix decomposition strategies, that produces more clean data to get better interpretation of the results. In 2016, Pazienza et al. studied how clustering approaches are applied for finding a trend-based portfolio diversification which is more consistent than sector-based portfolio (Pazienza et al., 2016). Their model was applied on the past 10 years closing prices of 28 stocks from NASDAQ Stock index and it considered only 8 market sectors overall, even thought they worked with rank values of 3 to 8. NMF was applied to find the clusters of stocks (subgroups) that have similar trends. Here the **W** represents clusters centroids, and **H** represent clusters membership (the largest values in the row). They compared NMF with Convex NMR (C-NMF) and Convex-Hull NMF (CH-HNM) which are two of its variants, which had convexity constraints for getting a better identification of similar stock trends. As a result, CH-NMF improved the interpretation of cluster properties, it converged in only one iteration, and it was fast and scalable for reconstruction quality. Additionally, they showed that NMF was better at identifying clustering properties, with yield low frobenius norm error (13.54) and more efficient in time of convergence, comparing to C-NMF (32.17) and CH-NMF (46.25). However, this work had a lack of clarity on how the trends were conformed.

2.5.2.2 Anomalies detection

Finding potential risk in the stock market is another important area for financial practitioners, because the fluctuations of the stock market affect the operations of the financial market. Some works have tried to detect anomaly fluctuations by analysing the data of the stock market. In 2016, Chen et al. proposed a method by using NMF to acquire the weight coefficient set that denotes the characteristics of date perfectly, to then decompose them by wavelet transform, to obtain the abnormal fluctuations from all of decomposition hierarchy - where the anomaly fluctuations are positioned out by means of weighted fusion (Chen et al., 2016). They used the weighted approach to judge the abnormal waveforms which are identified as those that are beyond the interval range, and are considered as normal fluctuations. Their data set consists of 3851 daily values of 42 stocks from the Shanghai stock market, between January 2000 to December 2015. As a result, NMF was effective for extracting a weight coefficient matrix, which contains the most characteristic expression from a range of stock index data. They found 107 abnormal fluctuations but there are not further details and comparisons about this approach which can be applicable for fraud detection.

2.5.2.3 Portfolio Optimization

Investing in a variety of assets to create a diversified portfolio for reducing the exposure to risk is a complicated practice. Dimensionality reduction methods are useful for enabling the implementation of more robust models on smaller datasets. These methods achieve an interpretable description of complex data, and help to tackle the complexity on construction of large-scale portfolios. The optimal allocation rate of a weighted combination of assets in a portfolio is enabled by the mean-variance (MV) model. This optimization process minimizes the portfolio variance given an expected rate of return (Markowitz, 1952). It uses standard deviation of asset returns to quantify uncertainty. Thus a portfolio is considered optimal when its yield led to minimum risk at an expected return.

The set where each optimal portfolio lies at a defined risk and expected return is the efficient portfolio frontier. Some works have analysed the use of positive dimensionality reduction on portfolio optimization. For example, Tayalı and Tolun (2018) examined mean-variance (MV) portfolio optimization model. Their dataset consists of daily observations on closing prices of the three major Istanbul Stock market indices (XU030, XU050, and XU100). It captures the long-term market pattern by reducing on a lower time dimension their dataset. They computed 20 optimal portfolios for each dataset. Their results show that it improved the efficiency with a higher performance (0.48%) more than benchmark back-testing results (0.28%) which were constructed from unreduced data.

In addition, the Sparse-semi-NMF technique was introduced by de Fréin et al. (2008a), to cluster stocks based on latent trends. Their dataset consists of 353 synthetic de-trended stock market returns. Their aim was to cluster the stocks by the underlying trends. They compared with the benchmark model, which is the de-trended diffusion model that was derived from the Black-Scholes model. They found that Sparse-semi-NMF outperforms with 25% the benchmark, and it was able to identify clusters in the diversification of a portfolio. Also, they show that sparse-semi-NMF can decompose stock data into a meaningful assignment matrix and latent trends matrix by combining intuitive factorization advantages with sparse assignments.

2.5.3 Term Structure of Interest Rates

The term structure of Interest rates or yield curve is the relationship between interest rates and different maturities. It is very important in economics and finances for pointing out what the market participants are expecting about coming changes in the interest rates. Level, steepness and curvature are the three factors for the structure of interest rates that were defined by Litterman and Scheinkman (1991). The yield curve data is non-negative by nature; for that reason NMF was applied. Models for yield curve aim to obtain insights of the current expectation of the future analysis on economy and finance, hence it is desired to ease the interpretation. Some studies have been working on it such as Takada and Stern (2015), that considered three factors for the models under comparison with PCA which does not minimize the approximation error. Their dataset consists of the Brazilian term structure of interest rates obtained from future contracts traded at BM&F Bovespa (stock exchange located in Sao Paulo), from 05/13/2003 to 10/09/2013, and with 3, 6, 12, 24 and 60 months vertices time spans. They also compared with the Nelson-Siegel (NS) model which presents some fitting problems. NMF was applied to find factors for yield curves. As a result, the data adjustment with NMF (0.018% error) is far better than that obtained from usual techniques such as PCA (5.973% error) or NS (16.46% error). Hence, NMF was found to be a suitable factorization model for yield curves and useful for investment strategies to analyse future behaviours of yield curves.

2.5.4 Intraday Trading Volume of a Security

The total amount of traded contracts of a security over the trading period of a day is called intraday trading volume. The intraday trading volume captures part of the intraday trading activity and represents a proxy for the intraday liquidity of a security. This has been reported to possess an intraday U-shaped pattern, e.g. heavy trading volume when the trading day begins and at the end; light volume traded in the middle of the trading day (Jain and Joh, 1988). In Takada and Stern (2016), NMF was for the first time applied to capture the intraday trading volume patterns. They investigated the statistical factors behind the intraday trading volume. Their dataset contains securities selected from the Brazilian stock exchange (BM&F Bovespa) for the period from April 2013 until September 2013. Statistical factors are unobserved variables used to describe observed data. NMF was applied with only one factor to identify the well-known U-shaped intraday trading volume pattern. The U-shaped pattern is very important for execution strategies, it is based on an important benchmark for execution strategies. Additionally, they also identified interpretable factors when considering NMF with two factors. One factor represents the level of volume for the beginning of a trading day and another factor when the trading day finishes. They also compared that NMF has a higher percentage of explained variance (72.35%) than 34.09% of PCA and lower residual sum of squares (RSS) $3.15e^{15}$ than PCA that has $2.04e^{16}$ for the joint estimation of selected equities.

2.6 Research Design

We aim to quantify the role of exogenous macro-economic information in machine learning for modelling financial data. We use quantitative approach on historical financial data.

2.6.1 Data Collection

To start, we searched for financial data providers that we have available. One is DataStream Thomson Reuters that provides historical data with a minimum of daily frequency values. DataStream contains a range of financial instruments worldwide. The second information supplier is the Bloomberg terminal. It allows to get financial data at real-time. The advantage of Bloomberg terminal compared with DataStream is the time frequency. In Bloomberg you can get hourly, minutely and tick data. Both financial information suppliers were located at University of Southampton.

From DataStream, we collected hundreds of historical call and put options that already expired with same time-scale (daily values). To explore the mapping of the price of traded option contracts and the underlying asset. We also collected the FTSE100 constituents and several macroeconomic variables with different time-scale. And finally, from DataStream, we collected the more macro-economic variables from Japan, UK and USA for comparison of models; and from Bloomberg terminal we collected several derivative options with higher time frequency (every minute).

2.6.2 Research Tools

We run several of our experiments in Iridis, the University's High Performance Computing Facility, and associated support services at the University of Southampton. Iridis has 464 compute nodes with dual 2.0 GHz Intel Skylake processors. Each compute node has 40 CPUs per node with 192 GB of DDR4 memory. A drawback of those financial sources when you want to compare global information is the low number of shared variables between countries.

Our code was compiled and run using Matlab and Python. Matlab is a proprietary programming language with multi-paradigm numerical computing environment. It eases the manipulation of matrices, plots the data and incorporates programs of others programming languages. Matlab has a machine learning built-in function that simplifies implementations.

Python is an interpreted high-level programming language. It provides an automatic memory management with multiple programming paradigms. It has many comprehensive standard libraries. In python can be used OpenMP for shared-memory multi-core systems and CUDA for many-core graphics processors. Two popular libraries were applied in python: Keras (Chollet, 2015a) and Pytorch (Paszke et al., 2017). Keras enables faster experimentation with deep neural networks, and run on top of TensorFlow (Abadi et al., 2016). Keras focuses on being user-friendly with simplicity for fast development. On the other hand, Pytorch is lower-level that works on array expressions. It is more complex but with better debugging capabilities compared with Keras.

2.6.3 Metrics

In the evaluation of machine learning models is very important to choose the appropriate metric for comparing the performance of models.

2.6.3.1 Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE)

Mean squared error is widely used in regression. It is applied to find the average squared error between the true values and the values from the prediction. Equation 2.43 describe how MSE is calculated.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.42}$$

The weakness of using MSE is that a single very bad prediction when is squared, it may skew the metric, by making the error worse or the opposite effect when the error is small. It could led to overestimate or underestimate the model performance.

Root mean squared error is the square root of MSE, which allows scaling the errors to the scale of the targets. Unlike MSE, RMSE penalizes large errors that can be suitable in several cases.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.43}$$

On the other hand, mean absolute error is determined by averaging the absolute differences between true values and the predicted values. Unlike RMSE, the single predictions are weighted equally in the average when MAE is used.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid \tag{2.44}$$

Another difference with the metrics mentioned previously it that MAE is not differentiable. However, MAE is considered to be less sensitive to outliers than MSE.

2.6.3.2 Pearson Correlation and R-Squared (R^2)

Pearson Correlation (Benesty et al., 2009) is a correlation measurement that indicates the relationship between two continues variables. Its value is between +1 and -1, with +1 means a strong positive correlation, -1 a strong negative correlation and zero mean no correlation. The coefficient is calculated by placing the two variables on a scatter plot (as X and Y), if some linearity of the scatter plot is found, the higher is the relationship.

$$\rho = \frac{cov(X,Y)}{\sigma_X, \sigma_Y} \tag{2.45}$$

Where the Person coefficient ρ is the covariance *cov* of a pair of variables X and Y, divided by the multiplication of their standard deviation σ_X and σ_Y.

The Person coefficient has been used in finance to measure the diversity of a portfolio. It is good to mention that correlation is not causation.

On the other hand, R-Squared, also known as the coefficient of determination, measures the strength of the relationship between independent and dependent variables.

$$R^2 = 1 - \frac{ExplainedVariation}{TotalVariation} \tag{2.46}$$

When R-squared values are close to 1, it means that movements of a dependent variable can be explained from the movements of the independent variable. Values below 0.6 indicate a weak correlation.

R-squared needs to be adjusted when it works with several independent variables because every predictor increments the value and never reduces it. Thus, it can compensate for this effect and only add then when they improve the model.

2.6.3.3 Euclidean Distance and Frobenius Norm

The Euclidean Norm (Danielsson, 1980) is the distance between a couple of samples p and q applied to an n-dimensional feature space. It is the square root of the sum of squares distances in each dimension that measures the length of the vector.

$$d(q,p) = \sqrt{\sum_{i=1}^{n}(q_1 - p_1)^2} \tag{2.47}$$

The Frobenius norm is an extension of the Euclidean norm to matrices instead of vectors. It comes from the Frobenius inner product $\langle A, B\rangle_{Fro}$ that is a component-wise inner product of two matrices and returns a number

$$\|A\|_{Fro} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{2.48}$$

2.6.3.4 Akaike Information Criterion (AIC)

Akaike Information Criterion (Akaike, 1974) is an estimator based on training data that provides an estimation of information lost on future values by the given model. The AIC is very popular for statistical inference. Its name came from its creator Hirotugu Akaike. The AIC selects the model that provides the lowest disparity with the true distribution. It helps to deal with the risk of overfitting and underfitting. When AIC value is lower, it means a better fit.

$$\text{AIC} = -2 * ln(L) + 2 * k, \tag{2.49}$$

where the value of the likelihood is L and the quantity of estimated parameters is k.

2.6.3.5 Kullback–Leibler divergence (KL)

The Kullback–Leibler divergence (Kullback and Leibler, 1951) measures the difference between two probability distributions of data, original and the approximated. It sometimes is known as relative entropy. When the KL divergence value is zero, it means that the two distributions are identical. Thus, we obtain the amount of information that is lost when we approximate one distribution with the original.

$$D_{KL}(p\|q) = \sum p(x) log \frac{p(x)}{q(x)}, \tag{2.50}$$

where p is the posterior probability distribution of data, q is the prior probability distribution. KL divergence is combined with neural networks to learn complex approximation distribution. For example VAE, described in Section 2.4.3, can generate optimal approximation distribution by using KL divergence. Despite the fact that KL divergence gives the distance between distributions, it is not considered as a distance measure.

2.7 Summary

In this chapter, we described several classical analytical models that depend on specific assumptions. Then, we outline a range of machine learning models that are considered to provide more accuracy than classical analytical models. We reviewed one of them (NMF) for a better understanding of how it enhances the performance and the analysis of financial data. By this review, we can get an idea of the impact of improving the performance of dimensionality reduction by inclusion of exogenous information in the financial sector. Starting with stock market analysis, where NMF achieved the stock index fluctuation detection, NMF was substantially better than a number of portfolio building strategies, also for clustering stocks based on latent trends and not by sectors in which some stocks have performed differently as their sectors that they behave. NMF was applied for portfolio optimization in which NMF is worthy of consideration in the diversification for increasing portfolio efficiency. Also, to address the credit risk analysis problem, which is a crucial task in finance and management. In the review of NMF, we found that exogenous macro-economic variables have not being considered for improving this dimensionality reduction. Therefore, we can notice that a lack of experiments with respect of this effect can support the need of our research. In this chapter, we also described our research experimentation in which we mentioned the data source, the tools that we used and the metrics that we apply.

Chapter 3

Extending The Feature Set Of A Data-Driven Artificial Neural Network Model Of Pricing Financial Options

In this chapter, we explore if additional information may be helpful to enhance a non-parametric approach that maps the price of trade option contracts to the value of the underlying asset and the time to maturity. We consider volume traded, historic volatility, observed interest rates and combinations of these as additional features. In addition to giving empirical results on how the inclusion of these variables improve performance, we show that there is an interesting correlation between volume traded and volatility through their influence on the price of a derivative option, which is not evident in the raw data.

3.1 Introduction

Pricing derivative contracts is a challenging problem in financial engineering because contracts mature at some point in the future and there are multiple sources of uncertainty between the current time at which a fair price for the contract needs to be determined and the point at which it may be exercised. The celebrated Black–Scholes model of options pricing makes specific assumptions about a stochastic process model of the underlying asset price and other factors relating to it (Hull, 2011). These assumptions lead to a solvable differential equation and result in a closed-form pricing formula for certain simple derivative instruments. For more complex contracts, where analytical solutions are not possible, Monte Carlo simulation and numerical analysis based methods have been developed. There is significant interest in the research literature and wide practical applications of these topics (Brandimarte, 2013; Trønnes, 2018).

In this context, the work of Hutchinson et al. (1994), could be seen as an elegant development from a machine learning or data-driven modelling point of view. The authors showed that a non-parametric artificial neural network, specifically a Radial Basis Functions (RBF) model, can be trained to approximate the complex relationship between the prices of an options contract and the underlying asset. In particular, they used only the normalized asset price and the time to maturity of the contract as inputs to the network and further showed that the derivatives of the mapping learned by the network faithfully reproduced the hedge ratio (Δ, Delta), a widely used parameter in balancing portfolio risk. Neural networks are powerful non-linear approximators, and the RBF architecture itself has found a wide range of applications including speech classification (Niranjan and Fallside, 1990), time series prediction (Kadirkamanathan and Niranjan, 1993) and financial engineering (Golbabai et al., 2012) among others. RBF is easily deployed in problems that require sequential learning and adaptive model complexity as demonstrated in the resource allocating network (Platt, 1991; Kadirkamanathan et al., 1991; Molina and Niranjan, 1996), and their generalization properties have been analysed in (Holden and Niranjan, 1995, 1997).

In addition to the demonstration that options prices may be well-approximated, the work of Hutchinson et al. (1994), which forms the basis of the present study, is notable for a second reason that is of interest in financial engineering. Their work showed that the derivatives of the learned model, which is easily computed for the RBF model analytically, turned out to be good approximations to the hedge ratio. This ratio determines the construction of a risk-neutral portfolio in which the uncertainty induced by a stochastic process model of asset price changes may be cancelled out. In later development, based on the work of Hutchinson et al. (1994), Niranjan (1997) showed that the RBF model has been used in this context and the Black–Scholes model itself, may be cast as dynamical systems, and the unknowns in the model inferred in a sequential setting using the Extended Kalman Filter (EKF) algorithm. A broader review of the uses of neural networks, with currency options as example, is given in Chen and Sutcliffe (2012).

A further topic of interest in empirical finance is the relationship between the traded volume of an asset and its volatility. Do assets that have more trading volume show greater price fluctuations, and hence greater uncertainty? While it is tempting to expect such a relationship, there may be no theoretical grounds to reach such a conclusion. An underlying theoretical premise, known as the Mixture of Distribution Hypothesis (MDH), introduced by Clark (1973), suggests that daily trading volume and price changes are driven by the same flow of information. Starting from this, there are empirical studies that have attempted to explore this relationship. For example, a study of the Istanbul Stock Exchange data using Granger causality suggests a bidirectional relationship between the two (Celik, 2013). However, Karpoff (1987) reports an asymmetry relationship, which is also supported by other studies (Jain and Joh, 1988; Andersen, 1996). Similar explorations have been carried out on the Korean and New York Stock Exchange data (Choi et al., 2012; Darrat et al., 2007). Similarly, on intra-day trading data of the S&P 500 index, a negative correlation is reported (Amatyakul, 2010).

In this work, we extend the work of Hutchinson et al. (1994) by asking the question if expanding the feature set to include additional variables of interest can help in improving their data-driven model of options pricing. Specifically, we include combinations of historic volatility, volume traded and interest rates as inputs in addition to the normalized asset price and time to maturity, as illustrated in Figure 3.1. Empirical work we carried out, on a range of data with a much wider scope than the original work, shows this to be the case. We compare the performance by using the mean square error (MSE). We follow this up with an analysis of how much the volume traded and volatility of asset help in predicting options prices and demonstrate an intriguing correlation between their relative contributions.

This chapter is structured as follows: in Section 3.2 we present our model including the data that we used; in Section 3.4 we display our results; and in Section 3.5 we conclude and discuss future research directions.

3.2 Model

We implement our Radial Basis Functions (RBF) following the work of Hutchinson et al. (1994), in which they restricted their analysis on options drawn on the S&P 500 Index only. We use more input data, and the inputs and outputs are shown in Figure 3.1. In our own work, we tested the effect of the choice of the number of basis functions, hence the model complexity, and found the value of four. We also confirmed this by running the RBF model with the Akaike information criterion (AIC) (Akaike, 1974), an estimator of information loss, as the method of model selection (Figure 3.2).

Let the input data vector to the RBF model be given by the vector

$$\boldsymbol{x} = \begin{bmatrix} \frac{S}{X} & T - t \end{bmatrix}^T, \tag{3.1}$$

where S denotes the underlying asset price, X the strike price of the contract and $T - t$ is the time to maturity (the difference between the time of maturity, T, of the contract and the present time t). With this vector of features as input, each of the basis functions in the RBF model for predicting options prices is written as:

$$\boldsymbol{\phi}_k = \sqrt{\left[(\boldsymbol{x} - \boldsymbol{m}_k)^T \quad \sum\nolimits_k^{-1} \quad (\boldsymbol{x} - \boldsymbol{m}_k) + b_k \right]}, \tag{3.2}$$

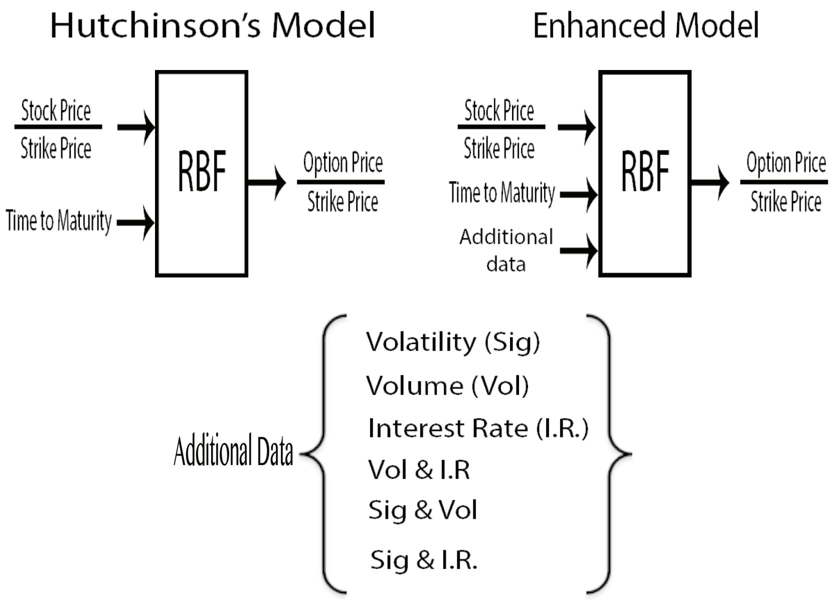


FIGURE 3.1: The data-driven neural network scheme of approximating options prices. Left: Hutchinson et al. (1994) model with two inputs; Right: Our proposed extension with the two initial inputs and the inclusion of additional variables. The symbol (&) indicates that both data inputs are included.

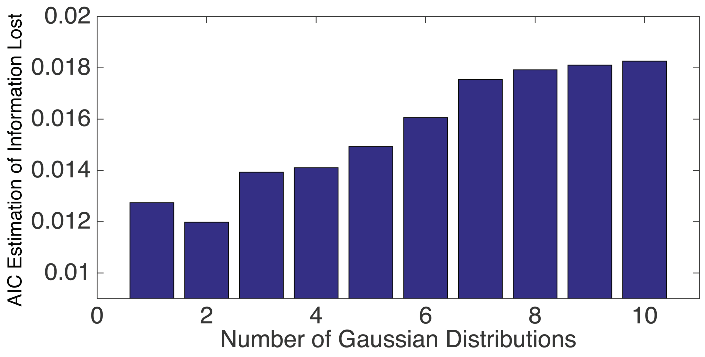


FIGURE 3.2: AIC estimation of information lost of the number of Gaussian distributions used on a small subset of the data. Consistent with Hutchinson et al. (1994), a small number of basis functions was considered sufficient.

where ϕ_k denotes the response of a nonlinear basis function which is parameterized by local mean $\boldsymbol{m}_k$ and covariance matrix Σ_k and b_k, a local bias term.

Basis function locations and local covariance matrices are estimated by fitting a Gaussian Mixture Model (GMM) to the distribution of input data, making the model sensitive to its local density. For simplicity, in our implementations, we set the bias terms b_k to zero. In addition to these nonlinear terms, the model includes a linear part as well. Thus, the least squares problem to solve is shown by the simultaneous equations:

$$\begin{pmatrix} \phi_{11} & \dots & \phi_{14} & x_{11} & x_{12} & 1 \\ \phi_{21} & \dots & \phi_{24} & x_{21} & x_{22} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \phi_{n1} & \dots & \phi_{n4} & x_{n1} & x_{n2} & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_4 \\ w_1 \\ w_2 \\ w_0 \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \tag{3.3}$$

The number of observations is n and $c_j, j = 1, \dots, n$ are the output *call* option prices normalized by their strike prices. The vector of unknowns $\begin{bmatrix} \lambda_1 & ... & \lambda_4 & w_1 & w_2 & w_0 \end{bmatrix}^T$ is estimated by linear least squares. A pseudo-inverse solution to the problem is often used:

$$\boldsymbol{w} = \left(\boldsymbol{P}^T\boldsymbol{P}\right)^{-1}\boldsymbol{P}^T\boldsymbol{\Phi}, \tag{3.4}$$

where $\boldsymbol{P}$ is the so called design matrix and $\boldsymbol{\Phi}$, the vector of outputs. Often, for reasons of numerical ill-conditioning and to avoid over-fitting by the model, a regularization term in the form of a diagonal matrix is added before the inversion of $\boldsymbol{P}^T\boldsymbol{P}$:

$$\boldsymbol{w} = \left(\boldsymbol{P}^T\boldsymbol{P} + \gamma\,\boldsymbol{I}\right)^{-1}\boldsymbol{P}^T\boldsymbol{\Phi}, \tag{3.5}$$

where γ controls how much regularization is applied. Regularisation is one way to improve generalisation performance.

Once the model is trained, the resulting output is given by,

$$\widehat{c} = \boldsymbol{\Phi\lambda} + \boldsymbol{w^T x} + w_0 \tag{3.6}$$

3.3 Data

For empirical evaluation, we collected daily prices of 21 *call* and 11 *put* options written on the Financial Times Stock Exchange (FTSE) 100 index and nine *call* and seven *put* options on the popular software companies Apple and Microsoft. Additionally, we considered minute-by-minute intra-day *call* and *put* options prices on a contract on Apple with a strike price of 95 and 100, maturing in September 2015 and June 2016 respectively. All those derivative options were collected from DataStream Thomson Reuters. Table 3.1 lists the range of options considered and includes their strike prices (the number in the name of the *call*(C) or the *put* (P) option), dates of maturity and the number of days that the options were issued until they expired.

In Figure 3.3, we normalised the *call* options by strike price and plotted versus stock price and time to expiration. The black points represent daily prices. This graph shows the density of the majority of those values is close to the time to maturity because the

Opt.	**Exp.**	**Days**	**Opt.**	**Exp.**	**Days**
C6600	09-13	171	C6600	12-14	397
C6700	09-13	290	C6700	12-14	359
C6800	09-13	171	C6800	12-14	397
C6700	06-14	352	C6900	12-14	299
C6750	06-14	85	C7000	12-14	387
C6800	06-14	525	P6000	09-13	462
C6900	06-14	291	P6300	09-13	525
C7000	06-14	288	P6400	09-13	525
C7100	06-14	171	P6500	09-13	250
C6600	09-14	462	P6600	09-13	513
C6400	09-14	462	P6700	09-13	171
C6800	09-14	462	P6800	09-13	525
C6900	09-14	299	P6600	06-14	513
C7000	09-14	387	P6700	06-14	352
C7200	09-14	462	P6750	06-14	85
C7400	09-14	342	P6800	06-14	525
CAPL50	07-15	239	CMSF39	07-15	236
CAPL85	07-15	239	PAPL120	07-15	239
CAPL90	07-15	239	PAPL130	07-15	238
CAPL95	07-15	239	PAPL135	07-15	238
CAPL105	07-15	236	PAPL140	07-15	238
CAPL110	07-15	236	PAPL150	07-15	236
CAPL115	07-15	236	PMSF45	07-15	236
CAPL120	07-15	236	PMSF47	07-15	236

TABLE 3.1: *Call* and *put* options with daily prices, used in this study with different time to maturity and numbers of trading days. The strike prices and an indication whether the option is a *call* or *put* is in the name of the contract (e.g. C6600 - Call option with a strike price of 6600). This data was collected from DataStream Thomson Reauter.

market exchange strategy keeps having options that are expiring in the current time or near future.

For the selection of training data, for any of the options, we took a temporal window of 40% of the data as the training set and evaluated the model performance on the next point in time, as it is illustrated in Figure 3.4. We moved this window one sample at a time and repeated the training and testing, in order to base our model on single sample unseen data. The reason for this choice of window, rather than a randomized training/test partition as often used in machine learning is that the financial data is expected to have temporal structure and in any practical application, one is likely to apply a trained model in the next point in time. We computed volatility over a window of 25% of the data immediately preceding a point of analysis, following Hull (2011).

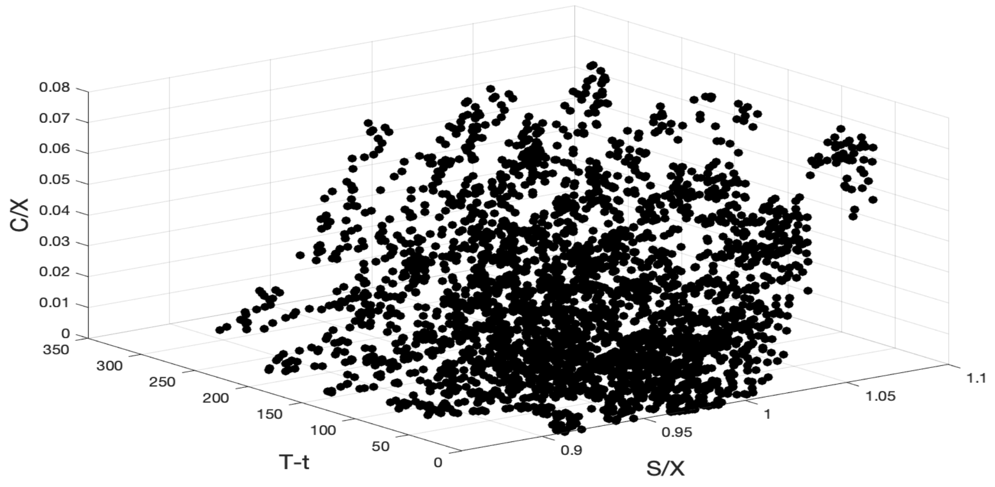


FIGURE 3.3: Call option prices normalized by strike price and plotted versus stock price and time to expiration. The black points represent daily prices. Here we can see that these values are more dense in the time that is close time to maturity. It is a fact that the market exchange strategy is to have options that are expiring in the current time or next month.

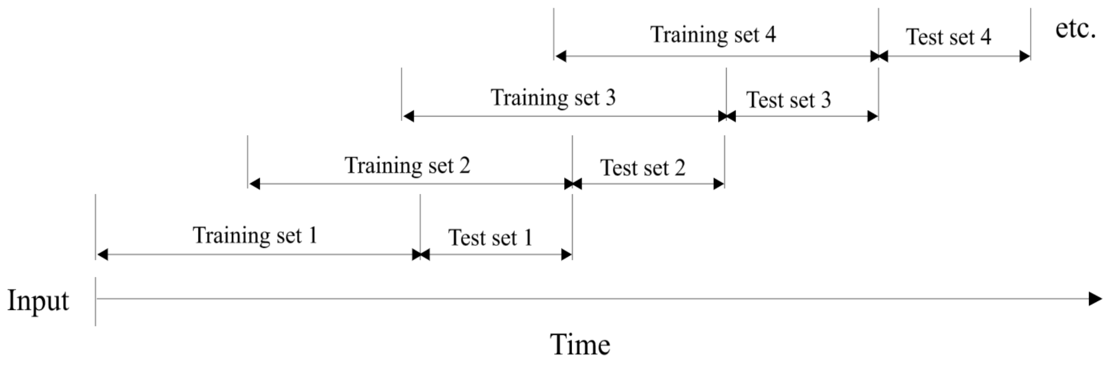


FIGURE 3.4: Sliding windows for dividing data in the training and testing set. We take the first set for training and we test over the next value. Then moving the training set and testing set to their next values and continue successively.

3.4 Results

3.4.1 Pricing Derivative options

After evaluating the derivative options listed in Table 3.1, we demonstrate that the model of Hutchinson et al. (1994) with additional inputs relevant to the pricing of options, enhance the accuracy of approximation.

3.4.1.1 Equity Derivative Options

In Figure 3.5, we show an example of the improvement from adding additional variables to the RBF model, where we can appreciate the values with dot points are much closer to the target values. Evaluation with all the call options on the FTSE-100 index comparing by the mean square error is shown in Figure 3.6.

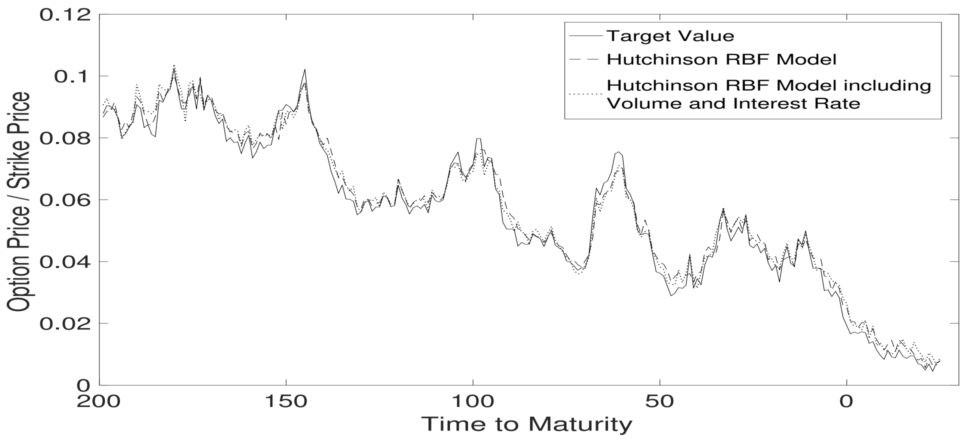


Figure 3.5: Price of an option on the FTSE-100 index (strike price 6800, expiring Sept. 2013) and its approximations from Hutchinson *et al.*'s model and a model that includes interest rate and trading volume as additional inputs.

Furthermore, in Table 3.2 we describe the averages of mean squared modelling errors for each for the additional variables that we consider in our experiment for approximating the *call* and *put* options. This table probes that enhancement by additional variables was the case.

-	**Hutch.**	**Vol.**	**Sig**	**Sig & Vol**	**I.R.**	**IR & Vol**
C	5.19	4.27	5.51	4.79	4.70	4.21
P	9.13	8.78	9.31	9.74	8.38	8.93

Table 3.2: Averages of mean squared modelling errors of the various models on the different option contracts. The distribution of mean squared errors is shown in Figure 3.6. Here we can see the effect of additional inputs enhancing RBF performance. Each value in the table should be scaled by 10^{-6}.

3.4.1.2 Single Derivative Options

We explored the effect of additional variables in the RBF model applied to single derivative options that correspond to the companies Apple and Microsoft, listed in Table 3.1. We illustrate in Figure 3.7 that the performance of the models with all the additional data cases was better than the original Hutchinson model. Here, with volatility as additional input, we see only a marginal improvement, whereas all others gave a significant reduction of error when compared with the model of Hutchinson.

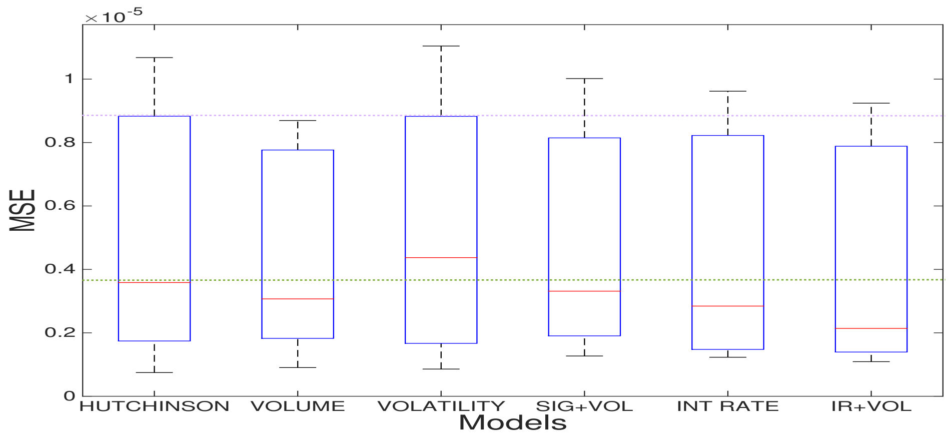


FIGURE 3.6: Prediction performance on *call* options on the FTSE-100 index for the various models considered. Overall, the largest improvement in prediction accuracies is obtained when volume traded and interest rates are included as additional covariates.

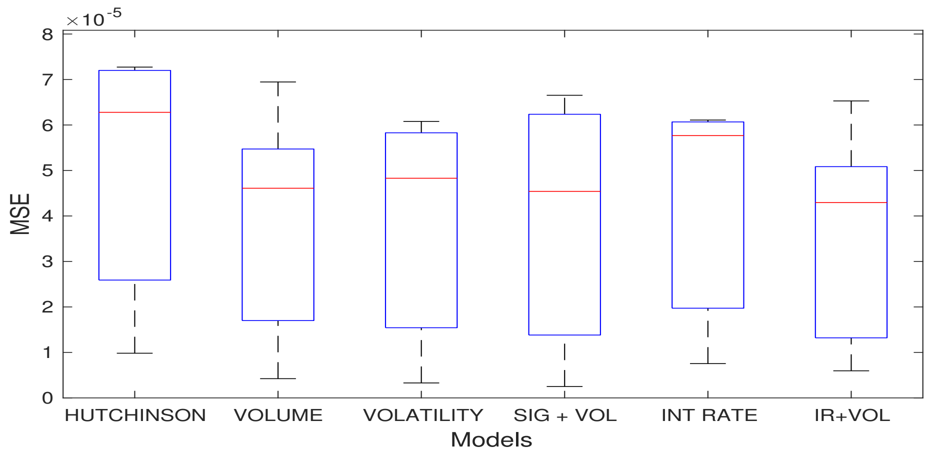


FIGURE 3.7: Prediction performance on single equity options that correspond to Apple and Microsoft. Here, we can see the improvements approximating option price with the additional variables considered.

An interesting evidence here, despite additional inputs enhance the performance of the model, they were not in similar level from both cases (equity options and single options). For example, in equity options, interest rate as an additional variable was the highest contributor for reducing the mean square error. However, in single options, it was not the case.

3.4.1.3 Derivative Options with higher frequency Data

With the aim of analysing the effect of additional variables not only with daily observations. We went further and we explore it on minute-by-minute intra-day *call* and *put* equity options. Table 3.3 shows the performance of RBF predictors, in which important improvement when volume and volatility with volume is seen. The additional variables help to enhance the model with lower error MSE than Hutchinson model.

-	Hutchinson	Volume	Sigma	Sigma & Volume
C	2.97	2.19	2.64	2.76
P	2.72	2.57	2.83	2.29

Table 3.3: Average prediction performances of RBF model with volume and volatility as additional inputs, compared with Hutchinson model on minute-by-minute intra-day *call* and *put* equity options. Each value in the table should be scaled by 10^{-5}.

3.4.2 Volume Traded and Volatility

After finding evidence of model enhanced by the inclusion of additional variables. We decided to explore the relationship between volatility and trading volume. The measurement of this correlation is made by the mean squared errors of the predictive models with volume traded and volatility as additional inputs.

Figure 3.8 (left boxplot) shows the distribution of correlations in mean squared errors of model fitting with volume traded and volatility as additional inputs on 48 options which is 0.675. The right boxplot in Figure 3.8 shows the correlation between the raw values of volume trading and volatility.

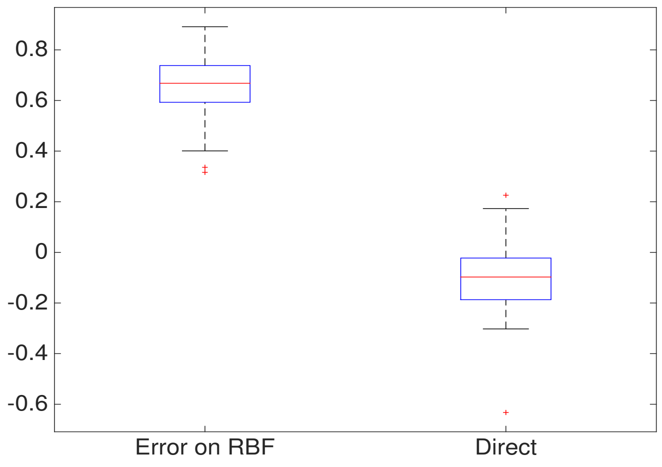


Figure 3.8: The comparison of the Pearson correlation between trading volume and volatility. The left side boxplot shows the correlation on both inputs data from the mean square errors of the RBF models in all the options tested. Here we can see that the mean on the correlation of each option analysed is over *0.67* giving a substantial correlation. However, the right side boxplot shows the actual values of the trading volume and volatility do not have a correlation.

In Figure 3.9, we show a scattered plot of the errors of the RBF model with volume as a additional variable and the errors with volatility as other input. With the aim of appreciate a correlation between them on predicting the single options prices.

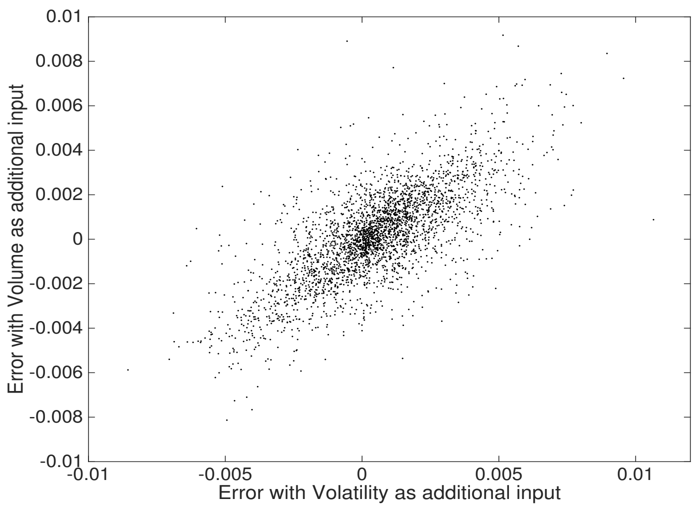
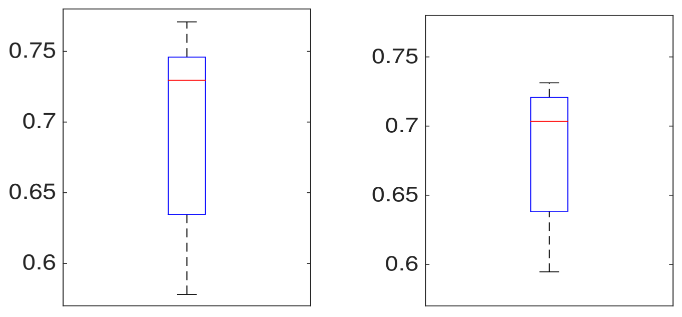


FIGURE 3.9: The errors of RBF model with volume with additional variable, with the errors of the model with volatility as additional input.

Then, we evaluate this appreciation by finding the Person coefficient value between the Mean Square Errors on the RBF with volume and volatility as additional inputs on all the *call* single options. We found that in effect, there is a considerable correlation between those variables with a Pearson coefficient value of *0.6818.* Similarly, we explored the Pearson coefficient value for *put* single options, and we got the value of *0.6985.* This is shown in figure 3.10



(a) *Put* single options (b) Call single options

FIGURE 3.10: Correlation between Mean Square Error values of the RBF models that include trading volume and volatility on *put*(a) and *call*(b) equity options.

On the other hand, we also reviewed this evidence on the minute-by-minute intra-day *call* equity option. And we found an important Pearson correlation of *0.7199.* It is clearly shown in Figure 3.11.

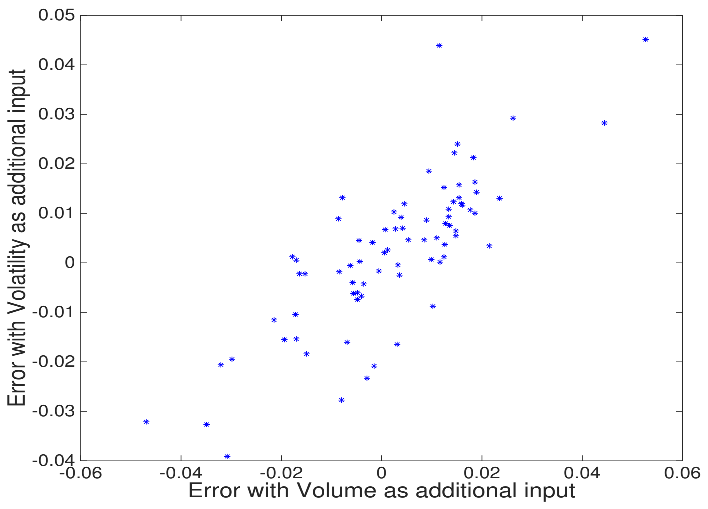


Figure 3.11: Contribution to prediction error from volatility and volume traded on minute-by-minute intra-day data on equity *call* option, showing a high correlation of 0.72.

The empirical work reported here is on static data, mapping instantaneous values of covariates to the response variable. We have found an important improvement in the RBF performance by the inclusion of additional variables. Besides, an interesting correlation between volume and volatility which is not visible in their raw values.

Apart from the selection of training and test data, we have not explicitly modelled the temporal dynamics. However, with financial time series, these dynamics are important. For this particular model, a state-space formulation and inference using Extended Kalman filter (Section 2.1.3.1) has been shown to be possible Niranjan (1997). However, the performance of this model depends on the considered system. We encourage to do further work on applying other state-space models including exogenous variables that are seen enhancing the predicting model.

3.5 Summary

In this chapter, we study a non-parametric neural network model that quantifies the complex relationship between a class of financial instruments known as options and that of the underlying asset on which the contract is drawn. Whereas previous work introducing this model uses the asset price and the time to maturity of the contract as its only inputs, we have demonstrated that the inclusion of additional features relating to the contract, namely the volatility, volume of the underlying asset traded and the risk-free interest rate help in improving the accuracy with which the market value of the contract may be predicted. Thus, our empirical results carried out on index options, equity options and an intra-day contract helped to probe our initial hypothesis.

The results led to the exploration of an intriguing relationship between the volatility of an asset and the amount of volume traded, a topic that has attracted healthy discussion in empirical finance. Our model makes an interesting contribution to this discussion in that we demonstrate that while the volume and volatility of an asset do not correlate significantly, their contributions to modelling the price of an options contract drawn on the asset do show significant correlation. This has also been demonstrated on the FTSE index, individual equities and intra-day datasets.

Chapter 4

Non-Negative Matrix Factorization with Exogenous Inputs for Modeling Financial Data

In this chapter we propose a model suitable for the analysis of multivariate financial time series data in which the variation in data is explained by latent subspace factors and contributions from a set of observed macro-economic variables. The macroeconomic variables being external inputs, the model is termed XNMF (eXogenous inputs NMF). We illustrate that the proposed model gives a lower reconstruction error than the NMF by the inclusion of the exogenous inputs. We show that XNMF can be effective in clustering stocks in similar trading sectors together via the latent representations learned.

4.1 Introduction

Non-negative matrix factorization (NMF) has been shown to be an useful decomposition for multivariate data. Unsupervised learning algorithms such as principal components analysis can be understood as factorizing a data matrix subject to different constraints. Non-negativity is an useful constraint for matrix factorization that can learn parts-based representation of the data (Lee and Seung, 2001). The non-negative basis vectors that are learned are used in distributed, sparse combinations to generate expressiveness in the reconstructions (Turk and Pentland, 1991).

In standard NMF we consider an input matrix with m dimensions and n samples: $\mathbf{V} \in \mathbb{R}^{m \times n}$. The aim is to find a lower-dimensional representation of the data by factorising $\mathbf{V}$ into two matrices $\mathbf{W}$ and $\mathbf{H}$ such that $\mathbf{V} \approx \mathbf{WH}$, where $\mathbf{W} \in \mathbb{R}^{m \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times n}$. Generally $r \ll m$ and $r \ll n$ so that NMF creates a new representation of the data in a significantly reduced subspace.

An example where NMF methods have been applied to financial data is de Fréin et al. (2008b). They apply it to identify underlying trends in stock market data. Also, de Fréin et al. (2008a) introduced an approach to portfolio diversification with a sparse-semi-NMF technique to minimize the risk when selecting a portfolio of holdings. Wang (2012) applied matrix factorization to extract underlying trends and group stocks into families based on their association with these trends, to diversify stock portfolio for reducing investment losses in the stock market. The appeal of NMF in this context is that returns on assets, expressed as ratios of their market prices, are positive. Factorizing multivariate asset return data into low rank factors can potentially discover low dimensional representations that are determined by sectors of assets (e.g. banking, blue chip companies, etc.) that are likely to show similar responses.

Financial time series data arise from a highly complex system driven by trader behaviour, a range of macro-economic variables and the arrival of new information. Pure time series analysis, univariate and multivariate, have been applied extensively to asset returns (Weigend et al., 1990; Tamiz et al., 1996; Omran, 1997), exchange rates (Babu and Reddy, 2015) and derivatives (Niranjan, 1997; Montesdeoca and Niranjan, 2016). However, statistical signal analysis methods usually do not take into account exogenous information from macroeconomic variables that may have significant contributions to market movements.

In this chapter, we propose a matrix factorization method that includes known exogenous variables as additional components of subspace modelling. We expect such factorizations to potentially have a lower reconstruction error, in Frobenius norm distance, compared with the NMF. We empirically demonstrate the effective performance of our approach on share price data from FTSE100 companies that also helps to answer our research question.

This chapter is structured as follows: in Section 4.2 we present our model including the underlying mathematics; in Section 4.4 we display our results; and in Section 4.5 we conclude and discuss future research directions.

4.2 Model and learning algorithm

Our aim is to find a combined representation of the share price data using the share price itself with the inclusion of exogenous information. Standard NMF method finds a representation such that $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1$, where $\mathbf{W}_1$ and $\mathbf{H}_1$ are matrices to be found. Considering an additional matrix $\mathbf{W}_2$ which contains the macro-variables and applying standard NMF we will have a separate representation $\mathbf{V} \approx \mathbf{W}_2\mathbf{H}_2$, where $\mathbf{H}_2$ will be a matrix to be found. XNMF is a combination of those 2 separated representation into one model such that:

$$\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2 \tag{4.1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{m \times r_1}$, $\mathbf{H}_1 \in \mathbb{R}^{r_1 \times n}$ and $\mathbf{H}_2 \in \mathbb{R}^{r_2 \times n}$ are the matrices that NMF will find. The matrix $\mathbf{W}_2 \in \mathbb{R}^{m \times r_2}$ has the macro-economic variables fixed. The time points are represented by m, the subspace to select is r_1, the number of exogenous variables is r_2 and the number of stocks is n.

Lee and Seung (1999) introduced an effective method to find those matrices separately and they called the multiplicative update technique. This solution which comes from a coordinate gradient descent gives updates to $\mathbf{W}$ and $\mathbf{H}$

$$\mathbf{W} \leftarrow \mathbf{W} \otimes \frac{\left[\mathbf{V}\mathbf{H}^T\right]}{\left[\mathbf{W}\mathbf{H}\mathbf{H}^T\right]}, \tag{4.2}$$

$$\mathbf{H} \leftarrow \mathbf{H} \otimes \frac{\left[\mathbf{W}^T\mathbf{V}\right]}{\left[\mathbf{W}^T\mathbf{W}\mathbf{H}\right]}, \tag{4.3}$$

where $\otimes$ and $\frac{[]}{[]}$ are the element-wise multiplication and element-wise division respectively. These updates push the matrices towards a minimization of the objective function $||\mathbf{V} - \mathbf{W}\mathbf{H}||^2_{\text{Fro}}$.

To find a solution to Equation 4.1 we seek a set of matrices which minimise

$$f = ||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}. \tag{4.4}$$

As minimising Equation 4.4 with respect to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_1$ together is non-convex, we hold two of the matrices constant whilst updating the third using multiplicative updates, following Gillis (2014). Each individual problem is then convex, although the overall problem remains non-convex and there is no guarantee of reaching an optimal solution. As multiplicative updates are a type of scaled gradient descent therefore we multiply out Equation (4.4) and get:

$$\begin{aligned} f =& \operatorname{tr}\left[(\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2)^T(\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2)\right] \\ =& \operatorname{tr}\Big[\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{W}_1\mathbf{H}_1 - \mathbf{V}^T\mathbf{W}_2\mathbf{H}_2 - \\ & \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{V} + \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2 - \\ & \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{V} + \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2\Big]. \end{aligned} \tag{4.5}$$

We then differentiate Equation 4.5 with respect to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively to give three equations:

$$\nabla_{\mathbf{W}_1} f = 2(\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T - \mathbf{V}\mathbf{H}_1^T), \tag{4.6}$$

$$\nabla_{\mathbf{H}_1} f = 2(\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2 - \mathbf{W}_1^T\mathbf{V}) \tag{4.7}$$

and

$$\nabla_{\mathbf{H}_2} f = 2(\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2 + \mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2^T\mathbf{V}). \tag{4.8}$$

Rather than two sets of multiplicative updates our solution has three, but the principles are the same. We do not have four because $\mathbf{W}_2$ is a fixed matrix with the macro-variables. We apply multiplicative updates to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ by:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 \otimes \frac{[\mathbf{V}\mathbf{H}_1^T]}{[\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T]}, \tag{4.9}$$

$$\mathbf{H}_1 \leftarrow \mathbf{H}_1 \otimes \frac{[\mathbf{W}_1^T\mathbf{V}]}{[\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2]} \tag{4.10}$$

and

$$\mathbf{H}_2 \leftarrow \mathbf{H}_2 \otimes \frac{[\mathbf{W}_2^T\mathbf{V}]}{[\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2 + \mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1]} \tag{4.11}$$

where $\otimes$ is the Hadamard product and $\frac{[]}{[]}$ indicates element-wise division. We will discuss how changes to $\mathbf{W}_1$ reduce the objective function noting that the same argument also applies to changes in $\mathbf{H}_1$ and $\mathbf{H}_2$. As we want to follow the gradient down towards a minimum, if $\nabla_{\mathbf{W}_1} f < 0$ then we want to increase $\mathbf{W}_1$. This is equivalent to $\mathbf{V}\mathbf{H}_1^T > \mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T$, and, as shown in Equation (4.9), $\mathbf{W}_1$ is increased. Conversely if $\nabla_{\mathbf{W}_1} f > 0$ then we need $\mathbf{W}_1$ to decrease, which the multiplicative update does because $\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T > \mathbf{V}\mathbf{H}_1^T$. The final eventuality, that $\nabla_{\mathbf{W}_1} f = 0$, implies we have found a minimum of $\mathbf{W}_1$ and so do not want to change $\mathbf{W}_1$, which the multiplicative update does. We should note that if $\nabla_{\mathbf{W}_1} f = 0$ we are not necessarily at a minimum of the objective function as the other two matrices may still change which might change the situation of $\mathbf{W}_1$ such that $\nabla_{\mathbf{W}_1} f$ is no longer zero. The same arguments apply to $\mathbf{H}_1$ and $\mathbf{H}_2$. The objective function should be moved monotonically towards a minimum by the multiplicative updates.

To deal with non-stationarity that may exist over such a long period in time, we split the data into four equal sections in time and show results on all four separately. The stocks returns are estimated by daily positions and investors hold or trade according to the circumstances (positive or negative) each following day. The daily returns are calculated arithmetically and are estimated through Equation (4.12), in which the price level is $p_1, p_2, \ldots, p_t$. The return at time t is formed by:

$$\mathbf{r}_t = \frac{p_t - p_{t-1}}{p_{t-1}}. \tag{4.12}$$

4.3 Data

We explain the effectiveness of our XNMF model and learning algorithm empirically using daily data from FTSE100 constituents taken over twenty years period (August 1996 to May 2017) from DataStream Thomson Reuters. Those values are the constituents stock market prices at each available point, basically week days except for public holidays. The choice of which macro-variables to use is somewhat arbitrary, there are many potential macro-variables, and they can be changed. The list of macro-variables in Table 4.1 is not in appropriate format as it is a mixture of frequencies. The best way of tally up these different time-scales is not straightforward. We compensate for the differences in frequency between the share data (recorded on work days) and the macro-variable data we have linearly interpolated between all the macro-variable data so that the dimensionality (the number of time points) are equal. The dates are converted into MatLab's standard dates to achieve the interpolation. We did not consider macro-variables that have negative values, such as inflation rate and balance of payments. We split the dataset into four equal sections in time to deal with non-stationarity that my exist over such a long period in time.

TABLE 4.1: Macro-economic variables used in this study

Macro-variables	**Frequency**	**Units**
GDP (Market Prices)	Quarterly	10^9 GBP
Unemployment	Monthly	%
Interest Rate	Monthly	%
Imports Goods&Services	Quarterly	10^3 GBP
Exports	Monthly	10^6 GBP
Oil Imports	Monthly	10^3 metric tons
Gross National Income	Quarterly	10^6 GBP
M1 Money Supply	Monthly	10^9 GBP
Productivity	Quarterly	%
British Pounds/ US Dolar	Daily	index
Contribution to CPI	Monthly	%
Oil price	daily	GBP/barrel
Oil Invest	Daily	10^9 GBP
Government Gross Reserve	Monthly	10^6 GBP

4.4 Results

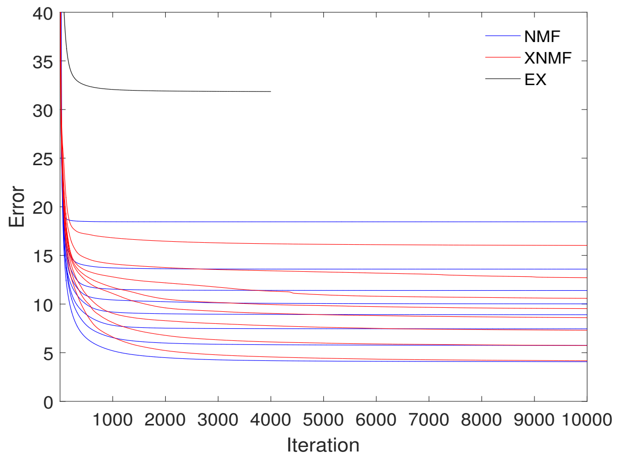
We first confirm empirically that our algorithm achieves the desired goal, the reduction in the error until it reaches a minimum. In Figure 5.9 we show how the error changes with iteration for different values of r for the three different algorithms. We will use the same terminology throughout: NMF results are from the algorithm which minimised $||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1||^2_{\text{Fro}}$, XNMF (exogenous inputs NMF) is for the minimisation of $||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}$ and EX (exogenous inputs alone) is for the minimisation of $||\mathbf{V} - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}$. The blue lines are for different values of r for NMF and the red lines for different values of r_1 for XNMF.

The EX algorithm produces a poor approximation as it contains no information from the actual stocks themselves. The results of particular note are those of the XNMF algorithm which works as we expect it to. We see a fall in the objective function with each iteration until it reaches a minimum where the error plateaus. The XNMF algorithm is much slower than the NMF algorithm at reaching a minimum which should not be a surprise as we have three matrices to optimise rather than two. In addition, the third matrix may make the objective function more non-convex than with just two matrices to optimise.

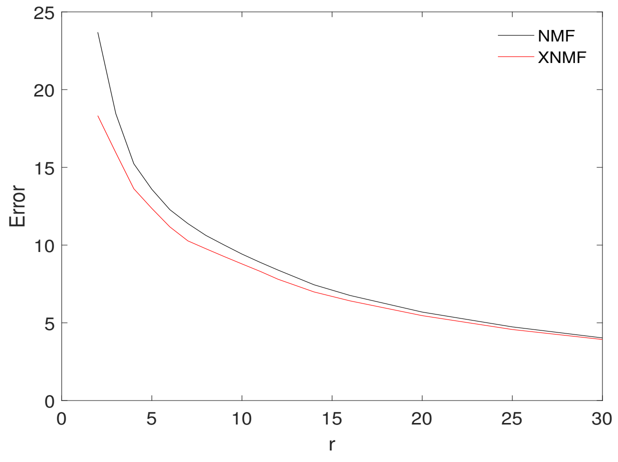
We now consider the effect of adding macro-variable data into our representation. In principle, we would expect the additional information to reduce the error somewhat. In Figure 4.1(b) we show the errors from performing normal NMF (black line) and the XNMF (red line) for different sizes of the subspace, r. At low values of r the model does not have enough subspace dimensions (columns of $\mathbf{W}_1$) to effectively fit the data and so the errors are high. The additional macro-variables here make a significant difference to the quality of the fit. As r increases the benefit of the additional information decreases as the increased capacity of the $\mathbf{W}_1\mathbf{H}_1$ part of the model means that a good fit to the data is possible without any additional information. As r increases, it is likely that the model is overfitting the data, so any use of NMF requires a sensible choice of r to be made.

A particular appeal of NMF is noise suppression, by reducing the noise we might expect to be able to extract more real features from the data. A key result demonstrated with gene expression data is that the reduction in noise achieved by matrix factorization leads to stable clustering and biologically relevant inference about genes, as shown in Brunet et al. (2004) and Devarajan (2008). In financial data we are often interested in how stocks and shares move together through time; a balanced portfolio would not contain lots of shares which are likely to fall in the same period. If we can effectively cluster the shares we can then build a more resilient portfolio.

We can easily cluster stock data into groups using a range of techniques, K-means clustering being a popular method. We are then interested in how the clustering works into



(a)



(b)

FIGURE 4.1: (a) The extended multiplicative update algorithm reduces the error monotonically with iteration until a plateau is reached. The multiple blue (NMF algorithm) and red (XNMF algorithm) lines are for different sizes of the subspace, r. Generally, the XNMF algorithm requires more iterations to reach a minimum than the NMF algorithm, but reaches a lower minimum. Here, we demonstrate that the algorithm reduces the error with iteration by the inclusion of exogenous variables. Then (b) shows the final errors for different sizes of the subspace, r, for NMF and XNMF. At all values of r that were implemented, XNMF produces smaller errors than NMF. As r increases the difference between the error produced by the algorithms reduces as the capacity of the NMF model increases. Hence XNMF clearly produces a lower error than standard NMF.

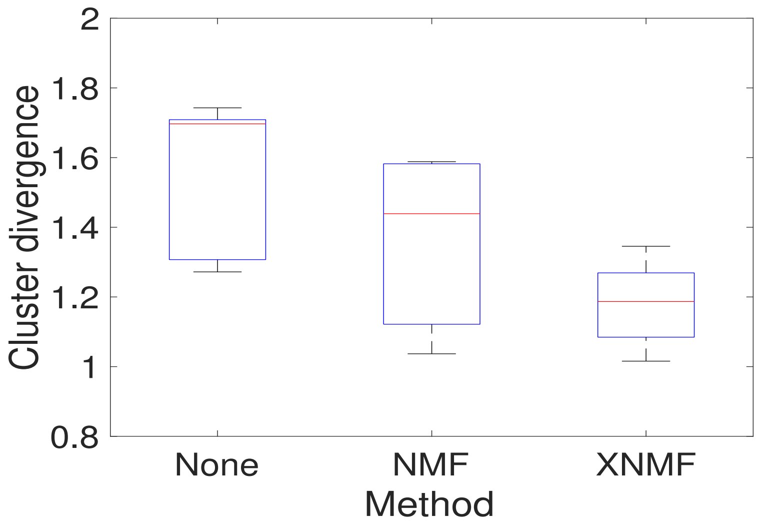
the future. If the clustering has been successful then we might expect the clusters to stay together through time. NMF itself is not a clustering technique, although variants with a high level of sparsity imposed on the columns of $\mathbf{H}$ can turn NMF into a clustering technique. NMF is a dimensionality reduction technique that creates a new sub-space in which we can apply clustering.

We have performed; K-means clustering on three versions of the data: a) no dimensionality reduction; b) dimensionality reduction using NMF; c) dimensionality reduction using XNMF. A measure of the similarity of a cluster is the average distance to the cluster centre using the non-dimensionality reduced data. What we are interested in is the change in the average distance to the cluster centre as this gives us a measure of how similar the cluster is at different time points. In general, we would expect an increase in distance as clusters will tend to diverge with time. If we see a smaller increase using the dimensionality reduced versions, it shows that the NMF techniques are allowing us to produce clusters which generalise better than a standard method.

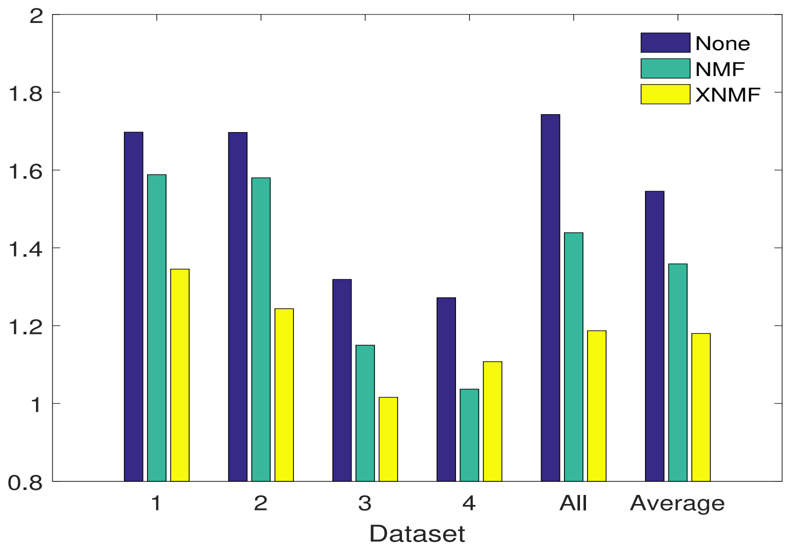
We illustrate in Figure 4.2 the results of this forward prediction of clustering. We split the data in half into a “training" set, the first half of the data in time, and a “testing" set, the second half of the data. The training data was then clustered into seven cluster centres using, respectively: raw data, $\mathbf{H}_1$ from NMF and $\mathbf{H}_1$ from XNMF. The y-axis shows the ratio of the average distances from each data-point to its cluster centre between the testing data and the training data. A smaller value means the cluster stayed closer together. We see a clear trend; the raw data performs the worst with the highest change whilst the XNMF gives the best performance, and NMF gives a result in between the other two. The results of this forward prediction of clustering is illustrated in Figure 4.2

It is reasonable to assume that stocks that are part of the same market do not behave independently of each other, but are rather driven by some underlying forces, normally significantly fewer than the number of stocks themselves. In this way, groups of stocks exhibit correlated behaviour, and these groups may or may not coincide with the sectors. We analyse it with a group of FTSE100 stocks constituents with the ten sectors that they belong to. Figure 4.3 shows the groups inferred using XNMF did not coincide with the known sector structure. It could mean that some stocks are more sensitive to the underlying factors from XNMF. We analyse the underlying factors that result from the matrix $\mathbf{W}_1$ of the XNMF model. The interval of time was including an important event in the UK, Brexit in June of 2016. In Figure 4.4 it is observable in a sharp transition which is not seen in any of the individual stocks. This would imply that the asset classes modelled by these factors are more sensitive to this political reality.

Thus, it can support our initial hypothesis considering that XNMF got better performance than NMF due to the inclusion of exogenous information. On the other hand, we analyse if it can help with interpretability in the next chapter.



(a) Models



(b) Dataset

FIGURE 4.2: A representation of how much clusters diverge with time. K-means clustering was applied to non-dimensionality reduced data (dark blue bars), dimensional reduction using NMF (light blue bars) and dimensional reduction using XNMF (yellow bars) for four times periods and for a combination of the four periods. The clusters produced from data with no dimensional reduction diverge the most, with the application of NMF the divergence is reduced and with XNMF we see the smallest divergence, the clusters tend to hold together better through time.

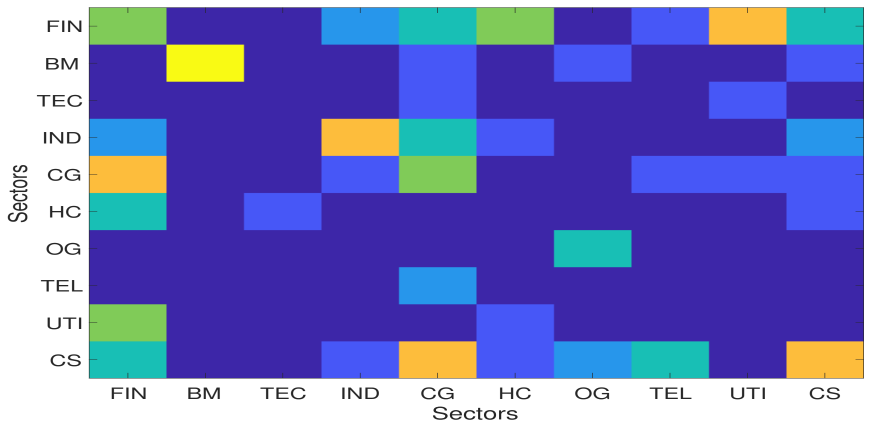


FIGURE 4.3: The confusion matrix shows that the 10 (rank dimension) groups that we got from XNMF do not coincide with the sector industries that the 94 FTSE100 stocks belong to. It means that some stocks are more sensible to the representative NMF factor than to their corresponding sectors (Where FIN-Finance, BM-Basic Materials, TEC-Technology, IND-Industrial, CG-Consumer Goods, HC-Health Care, OG-Oil&Gas, TEL-Telecomunication, UTI-Utilities, CS-Consumer Service)

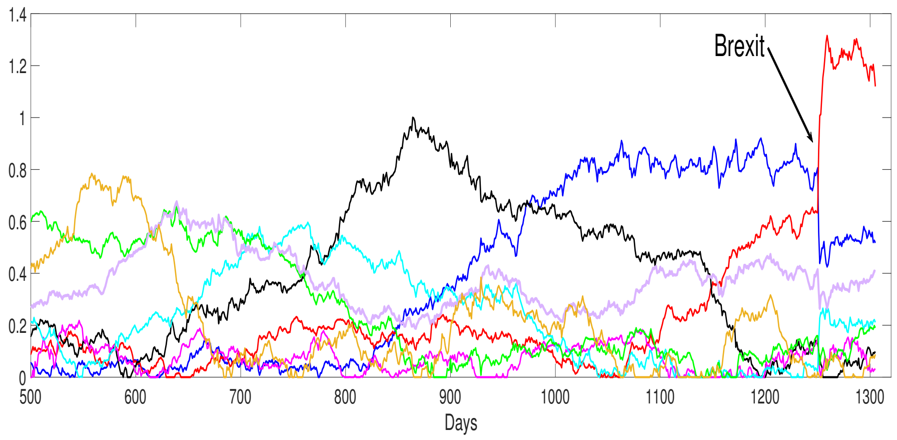


FIGURE 4.4: The weights of the matrix $\mathbf{W}_1$. This shows an example of an interpretable output from XNMF modelling. Close to the Brexit event, it has a sharp transition, which is not seen in any of the individual stocks. This would imply that the asset classes modelled by these underlying factors are more sensitive to this political reality.

4.5 Summary

In this chapter we introduced a novel matrix factorization model suitable for multivariate financial time series that includes known exogenous macroeconomic variables. We report an empirical study that XNMF produces a lower reconstruction error than standard NMF for different sizes of the subspace by the inclusion of exogenous variable. However, this effect is more visible when the subspace is lower. We assume it is reasonable because in higher dimension the models consider much more information is included. We showed that the groups of stocks by underlying factors from XNMF did not coincide with the known sector structure, which may be help for portfolio diversification strategy. We use real FTSE100 stock data to show that stock clusters formed with the addition of exogenous data stay tighter bound through time. Hence, these empirical results may help to prove that the inclusion of exogenous information enhance performance.

Chapter 5

On Quantifying the Role of Exogenous Macro-Economic Information in Financial Time Series Analysis

We compare two approaches to quantifying the roles of exogenous financial information which are taken to be proxies of the underlying macro-economic environment in which trading on assets takes place and thus prices set. We start from a state-spaces dynamical systems formulation and model the innovation signal by means of a sparse regression taken over a set of exogenous variables. The second, Non Negative Matrix Factorization with exogenous inputs (XNMF) works by dimensionality reduction, in which the factorization is enhanced by the incorporation of the macro-economic signals. We extend this model by including additional sparsity constraints using the Hoyer's Algorithm with adaptation to additional information. We consider time series analysis on cryptocurrencies such as Bitcoin and data from three different market indices with their constituent stocks to identify the common macro-economic variables that are most influential in price movements.

5.1 Introduction

Machine learning techniques are widely used for modelling multivariate time series (Tkacz, 2001; Cook and Hall, 2017). For example, Mahler (2009) introduced a method that uses the Kalman filter and the LagLasso to predict upward and downward monthly variations of the S&P 500 index by using a group of 7 macro-economic and financial explanatory variables (explained in Section 5.2.2). Prediction with LagLasso can be done by the selection of the most important variables and lags. This is an interactive method that requires the management of a set of covariates. It helps to shrink and select variables for linear regression by minimising the sum of squared errors, subject to a bound on the sum of the coefficients absolute values. Thus Kalman LagLasso assumes that a small subset of macro-economic and financial factors can efficiently represent the exogenous influence on the financial indices previously mentioned, where the influence of each of these factors can change over time and can be lagged.

We start our chapter by using Kalman LagLasso on time series of cryptocurrencies and stock indices. The hypothesis would be that the differing nature of these two market instruments would mean that the LagLasso method should find very different macroeconomic variables as explanatory variables. The difference arises from the fact that the stock index is driven by its constituent assets whose values are determined largely by their performance such as profitability, market capture and dividend payments. Our hypothesis is, cryptocurrencies do not have any such underlying fundamentals that influence them. Their values would be dominated largely by speculative behaviour of investors and traders.

Then we apply Kalman LagLasso on three different markets (USA, UK and Japan) to make predictions about the variations of the S&P 500, FTSE100 and Nikkei indices over the next month, with an important pool of 15 exogenous financial variables. And finally, we extend our XNMF work explained in Section 4.2 by adding sparseness in order to analyse the sectors that are influenced by those variables. Also, we compare our XNMF model with the Kalman LagLasso model to find an overlap between the exogenous variables that influenced the three countries market information that we chose (Japan, USA and UK).

This chapter is structured as follows: in Section 5.2 we present our model including the data that we used; in Section 5.4 we display our results; and in Section 5.5 we conclude and discuss future research directions.

5.2 Model Implementation and Data

We start by using the autoregressive model (AR), which is expressed in Equation 5.1 and to then fit using the Kalman filtering that is described in Section 5.2.1.

$$\hat{x}_t = \sum_{j=1}^{p} \theta_{jt} y_{t-j}, \tag{5.1}$$

we can write it as vector notation: $\hat{x}_t = \boldsymbol{\theta}^T \boldsymbol{y}_t$. here the past values of the time series are in $\boldsymbol{y}_t$ and the regression coefficients in $\boldsymbol{\theta}$.

5.2.1 Kalman Filtering

Financial stock indices are sensitive to the variations of macro-economic and financial predictors around their own trend rather than to the variations themselves. For that reason, filtering the predictors is needed, which is possible by computing the Kalman algorithm (Kalman, 1960). Let $\hat{x}_t$ be the observations and let $\boldsymbol{\theta}$ be a hidden random vector (unobserved variables). Thus we can use the equation 5.2 and 5.3:

$$\hat{x}_t = \boldsymbol{y}_t^T \boldsymbol{\theta}_t + \mathrm{v}_t, \tag{5.2}$$

where $\boldsymbol{y}_t$ is known and v_t is Gaussian white noise, $\mathrm{v}_t \sim \mathcal{N}(0, \boldsymbol{R})$ and $\boldsymbol{R}$ is the observation covariance matrix. $\boldsymbol{\theta}_t$ is given by:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{w}_t \tag{5.3}$$

where w_t is Gaussian white noise, $\mathrm{w}_t \sim \mathcal{N}(0, \boldsymbol{Q})$ and $\boldsymbol{Q}$ is the evolution covariance matrix.

The Kalman filter is a process that recursively estimates the coefficients of the model represented by a vector $\boldsymbol{\theta}$ based on the values of the previous days coefficients, an uncertainty matrix adjusted every day, and some tuning parameters that model the error.

The Kalman filter equations based on Mahler (2009) are given by:

$$\begin{aligned}
\boldsymbol{\theta}_{t|t-1} &= \boldsymbol{\theta}_{t-1|t-1} \\
\mathbf{P}_{t|t-1} &= \mathbf{P}_{t-1|t-1} + \mathbf{Q} \\
r_t &= \hat{x}_t - \mathbf{y}_t^T \boldsymbol{\theta}_{t|t-1} \\
\mathbf{k}_t &= \mathbf{P}_{t|t-1}\mathbf{y}_t(\mathbf{y}_t^T \mathbf{P}_{t|t-1}\mathbf{y}_t + \boldsymbol{R})_t^{-1} \\
\boldsymbol{\theta}_{t|t} &= \boldsymbol{\theta}_{t|t-1} + \mathbf{k}_t r_t \\
\mathbf{P}_{t|t} &= \big(\mathbf{I} - \mathbf{k}_t \mathbf{y}_t^T\big)\mathbf{P}_{t|t-1}
\end{aligned} \tag{5.4}$$

where the signal modelled is r_t, from the vector of past values $\boldsymbol{y}_t$. $\boldsymbol{\theta}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$ are predictions of the parameters and error covariances in them respectively. $\boldsymbol{R}$ and $\boldsymbol{Q}$ tune the entire Kalman filter.

The Kalman algorithm makes a prediction of the signal and calculates its residual error r_t by predicting $\boldsymbol{\theta}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$ from $\boldsymbol{\theta}_{t-1|t-1}$ and $\boldsymbol{P}_{t-1|t-1}$. Then we use this residual with the term *Kalman gain* ($\mathbf{k}_t$) in the posterior updates.

5.2.2 LagLasso

LagLasso is a linear model that explains the residual r, from applying Kalman filtering, by taking into account exogenous variables as is described in Equation 5.5.

$$r_t = \sum_{k=1}^{K} \sum_{j=1}^{J} w_{jk} u_j(t-k), \tag{5.5}$$

where we need to estimate the unknown parameters w_{jk} by the variables $u_j(t)$; $j = 1, ..., J$ and the number of lags k depends on the application. Here we chose the same value number of lags for all the exogenous variables for convenience.

The application of the L_1 penalty to regression is usually applied to select a subset of variables by searching all possible combinations (Tibshirani, 1996; Takeda et al., 2013; Weston et al., 2003). The resulting minimisation problem is:

$$\min_{w} \|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1, \tag{5.6}$$

where the residual signal is $\boldsymbol{r}$, the design matrix is $\boldsymbol{X}$ consisting of the exogenous time series $u_j(t)$; $j = 1, ..., J$, the parameter that controls the level of sparsity is λ and the unknown parameters to be estimated are $\boldsymbol{w}$.

We describe the computations of LagLasso in pseudo-code format in Algorithm 1.

Algorithm 1 LagLasso Algorithm

```
1: **Input:** $\boldsymbol{M} \in \mathbb{R}^{m \times n} \leftarrow$ Independent financial variables
2: Initialize $\boldsymbol{z} := \{\}$
3: Initialize $\mathbf{r}$ with Residual values from applying Kalman filtering, with dimension $m$
4: Initialize $k \in \mathbb{R}_{>0} \leftarrow$ Number of lags desired
5: Choose the desired $L_1$ penalty value $\lambda \in \mathbb{R}_{>0}$
6: Set $\boldsymbol{X} \in \mathbb{R}^{p \times q}$ with values from $\boldsymbol{M}$ considering $k$ lags $\{p = m - k$ and $q = n \times k\}$
7: Apply Lasso to get $\boldsymbol{w}$ with value $\lambda_i$ constraint
   $\min\{\|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1\}$
8: Set $\boldsymbol{z}$ with the non-zero values from the weight vector $\boldsymbol{w}$
9: **return** $\boldsymbol{z}$
   Above, $m$ is the number of observations and $n$ is the number of macro-economic variables.
```

5.2.3 Sparse XNMF

NMF tends to be sparse (Lee and Seung, 1999) and it has been suggested to control its sparsity by more direct means. However, several studies have introduced various extension in NMF that can incorporate sparseness constraint. Those extension with sparsity are scheme in which a large population is represented with only few units. Sparse coding concept is used to effectively represent typical data vector (Field, 1994). Some of the spareness measures proposed are (Peharz and Pernkopf, 2012; Karvonen, 2008) that quantify the amount of energy that a vector packs in order to get only a few components. However, the most common sparsity measurement is by the L_1-norm. Hoyer (2002) described how to apply sparseness for non-negative matrix factorization to minimize the objective function:

$$\|\mathbf{V} - \mathbf{WH}\|^2 + \lambda \sum_{ij} |\mathrm{H}_{ij}|, \tag{5.7}$$

where $\mathbf{V}$ is the original matrix to be factorized in two matrices $\mathbf{W}$ and $\mathbf{H}$ with a L_1-norm penalty value of λ.

Further, Hoyer (2004) introduced a variation of sparseness measurement for NMF that take account the relationship between the L_1-norm (Hoerl and Kennard, 1970) and the L_2-norm (Tibshirani, 1996) in any sparseness degree desired. It finds the nearest, euclidean distance, non-negative vector $\mathbf{s} = \text{sparseness}(\mathbf{x})$

$$\text{sparseness}(\mathbf{x}) = \frac{\sqrt{n} - \sum \mid \mathbf{x}_i \mid}{(\sqrt{n} - 1)\sqrt{\sum \mathbf{x}_i^2}}, \tag{5.8}$$

where $\mathbf{x}$ can be any vector of n dimensionality.

The algorithm of Hoyer (2004) is a projected gradient descent that allows to apply sparsity to either the basis vectors $\mathbf{W}$ or the coefficients $\mathbf{H}$ (from Equation 5.7). A step in this algorithm follows the direction of the negative gradient, then it is projected to the constraint space to assure the taken step is smaller than the objective function $f = \|\mathbf{V} - \mathbf{WH}\|^2$, which is shortened at each step.

We extended Hoyer's algorithm to implement the sparseness constraint in our XNMF model (Algorithm 2). The unique difference is that we modified the objective function to $f = \|\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2\|^2$.

With the aim of enforcing sparseness to the matrices is required to apply Hoyer's new projection operator. This operator is defined in Algorithm 3. The operator projects the given vector onto the hypersphere $\sum s_i = L_1$, where the nearest point is projected on the intersection of the sum and the L_2-norm constraint. The point where all the components have same values is the center of the sphere, and from the center it moves radially outward. It converges (arrive to destination), when the result is fully non-negative. Otherwise, the components take a value of zero and a new point follows same path with additional constraints.

Algorithm 2 Sparse XNMF Algorithm

```
1: **Input:** $\mathbf{V} \in \mathbb{R}^{m\times n} \leftarrow m$ is the observations number and $n$ the constituents numbers
   {Matrix with the FTSE 100 constituents daily prices}
2: Initialize random positive matrices $\mathbf{W}_1 \in \mathbb{R}^{m\times r_1}$, $\mathbf{H}_1 \in \mathbb{R}^{r_1\times n}$ and $\mathbf{H}_2 \in \mathbb{R}^{r_2\times n}$
3: **Input:** $\mathbf{W}_2 \in \mathbb{R}^{m\times r_2} \leftarrow$ Exogenous macro-economic information
4: Set the desired sparseness $S_{W_1}$, $S_{H_1}$ and $S_{H_2}$ desired for $\mathbf{W}_1$,$\mathbf{H}_1$ and $\mathbf{H}_2$ respectively
5: **if** sparseness constraints on $\mathbf{W}_1$ **then**
6:   project each column of $\mathbf{W}_1$ to be non-negative, have unchanged $L_2$-norm, and set
     $L_1$-norm to get desired sparseness $S_{W_1}$ by Algorithm 3
7: **end if**
8: **if** sparseness constraints on $\mathbf{H}_1$ **then**
9:   project each row of $\mathbf{H}_1$ to be non-negative, have unit $L_2$-norm, and set $L_1$-norm
     to get desired sparseness $S_{H_1}$ by Algorithm 3
10: **end if**
11: **if** sparseness constraints on $\mathbf{H}_2$ **then**
12:   project each row of $\mathbf{H}_2$ to be non-negative, have unit $L_2$-norm, and set $L_1$-norm
      to get desired sparseness $S_{H_2}$ by Algorithm 3
13: **end if**
14: **while** Number of iterations **do**
15:   **if** sparseness constraints on $\mathbf{W}_1$ **then**
16:     Set $\mathbf{W}_1 := \mathbf{W}_1 - \mu_{W_1}(\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2 - \mathbf{V})\mathbf{H}_1^T$
17:     project each column of $\mathbf{W}_1$ to be non-negative, have unchanged $L_2$-norm, and
        set $L_1$-norm to get desired sparseness $S_{W_1}$ by Algorithm 3
18:   **else**
19:     take standard multiplicative step:
        $\mathbf{W}_1 \leftarrow \mathbf{W}_1 \otimes \left[\mathbf{V}\mathbf{H}_1^T\right] \oslash \left[\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T\right]$
20:   **end if**
21:   **if** sparseness constraints on $\mathbf{H}_1$ **then**
22:     Set $\mathbf{H}_1 := \mathbf{H}_1 - \mu_{H_1}\mathbf{W}_1^T(\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2 - \mathbf{V})$
23:     project each row of $\mathbf{H}_1$ to be non-negative, have unit $L_2$-norm, and set $L_1$-norm
        to get desired sparseness $S_{H_1}$ by Algorithm 3
24:   **else**
25:     take standard multiplicative step:
        $\mathbf{H}_1 \leftarrow \mathbf{H}_1 \otimes \left[\mathbf{W}_1^T\mathbf{V}\right] \oslash \left[\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2\right]$
26:   **end if**
27:   **if** sparseness constraints on $\mathbf{H}_2$ **then**
28:     Set $\mathbf{H}_2 := \mathbf{H}_2 - \mu_{H_2}\mathbf{W}_2^T(\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2 - \mathbf{V})$
29:     project each row of $\mathbf{H}_2$ to be non-negative, have unit $L_2$-norm, and set $L_1$-norm
        to get desired sparseness $S_{H_2}$ by Algorithm 3
30:   **else**
31:     take standard multiplicative step:
        $\mathbf{H}_2 \leftarrow \mathbf{H}_2 \otimes \left[\mathbf{W}_2^T\mathbf{V}\right] \oslash \left[\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2 + \mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1\right]$
32:   **end if**
33: **end while**
34: **return** non-negative matrices $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$
    Above, $\otimes$ and $\oslash$ are element-wise multiplication and division respectively. $\mu_{W_1}$, $\mu_{H_1}$
    and $\mu_{H_2}$ are the step-sizes of $\mathbf{W}_1$,$\mathbf{H}_1$ and $\mathbf{H}_2$ respectively. The subpace dimension
    is $r_1$ and the number of exogenous variables is $r_2$.
```

Algorithm 3 Projection operator from Hoyer (2004)

```
1: Initialize $\boldsymbol{z} := \{\}$
2: Input: An arbitrary vector $\mathbf{x}$
3: Input: $L_1$-norm and $L_2$-norm constraints
4: Start by projecting the point to the sum constraint hyperplane
   $s_i := x_i + (L_1 - \sum x_i)/\dim(\mathbf{x}), \forall_i$
5: while Iterate do
6:   Set midpoint $m_i := L_1/(\dim(\mathbf{x}) - \mathrm{size}(\boldsymbol{z}))$
7:   if $i \in \boldsymbol{z}$ then
8:     Set midpoint $m_i := 0$
9:   end if
10:  Set $\boldsymbol{w} := \boldsymbol{s} - \boldsymbol{m}$
11:  Set $a := \sum \boldsymbol{w}^2$
12:  Set $b := 2 \times \boldsymbol{w}^T \times \boldsymbol{s}$
13:  Set with $L_2$-norm constraint
     $c := \sum \boldsymbol{s}^2 - L_2$
14:  Set $\alpha := (-b + \sqrt{(b^2 - 4 \times a \times c})/(2 \times a)$
15:  Set $\boldsymbol{s} := \boldsymbol{m} + \alpha(\boldsymbol{s} - \boldsymbol{m})$
16:  if All components of $\boldsymbol{s}$ are non-negative then
17:    return $\boldsymbol{s}$
18:  else
19:    Set $\boldsymbol{z} := \boldsymbol{z} \cup \{i; s_i < 0\}$
20:    Set $s_i := 0, \forall i \in \boldsymbol{z}$
21:    Set $r := (\sum s_i - L_1)/(\dim(\mathbf{x}) - \mathrm{size}(\boldsymbol{z}))$
22:    Set $s_i := s_i - r, \forall i \notin \boldsymbol{z}$
23:  end if
24: end while
25: return A closest non-negative vector $\boldsymbol{s}$ to $\mathbf{x}$
    Above, $x_i \in \mathbf{x}$, $s_i \in \boldsymbol{s}$ and $m_i \in \boldsymbol{m}$.
```

5.3 Data

5.3.1 Japan, USA and UK Market Data

We aim to compare the effect of macro-economic variables in three countries (Japan, USA and UK). For that reason, we chose the macro-variables that are available for those three countries with the same time frequency (monthly) and period of time (from July 1996 to July 2018). We wanted to consider China as well that has the second biggest economy in the world, but the available Chinese macro-economic variables differ from the initial three countries. We found fifteen macro-economic variables that are available for each of three countries. We discarded several macro-variables that contains information for USA and UK but not of Japan, as the aim is to compare all three together. We did not include macro-economic variables that are with lower time frequency to avoid linearly interpolation to compensate the differences in frequency as we did in the previous chapter, for example GDP. Our dataset, collected from DataStream Thomson Reuters, is formed of macro-economic variables with some of the most popular commodities (Gold,

TABLE 5.1: Macro-economic variables available for each of the three countries (Japan, USA and UK) with monthly values

Macro-Economic Variables			
Consumer Confidence	index	WTI Oil	USD/barrel
10Y Bonds	%	20Y Bonds	%
Gold	USD/oz	Unemployment	%
Exports CURN	10^6 GBP	Imports CURN	10^6 GBP
Foreign Currency Reserves	10^6 ¥,\$,£	Government Budget	10^6 ¥,\$,£
Copper	USD/MT	Gas	USD/m^3
Target Rate	%	Production	USD
Exchange Rate	index		

Copper, Gas and Oil) which have been analysed for their effect in the stock market by for example Kilian and Park (2009). They are listed in Table 5.1. Besides, we collected several cryptocurrency variables to compare with macro-economic variables from USA, using the Kalman Laglasso. This data is composed of daily values from August 2011 to March 2019 and similarly we only considered those that have same time frequency to avoid linearly interpolation.

5.3.2 Cryptocurrency Data

We have collected Blockchain data and financial information from August 2011 to February 2019. It consists of 34 exogenous variables information that contains macroeconomic variables of USA market, others countries stock market index, currencies exchange rate and Blockchain information related to Bitcoin. Table 6.1 shows this data that was acquired from Thomson Reuters Datastream Platform at the University of Southampton with daily and monthly values. In addition to Blockchain information and USA market, we consider the commodities: oil, gas and gold price because some works analyze the effect of those on stocks and currencies (Sujit and Kumar, 2011; Pukthuanthong and Roll, 2011).

Table 5.2: Macro-economic and Cryptocurrency variables

Macro-economic Data	Units	Macro-economic Data	Units
USA Amount Market	10^6 USD	Government Budget	10^6 USD
Equity Risk Premium	%	Oil WTI	USD/Barrel
Production	USD	Gas	USD/m^3
Market Issues	10^2	Global Investors	10^6 USD
Trade Balance	USD	10Y Bonds	%
EUR/USD	Ratio	GBP/USD	Ratio
Personal Incomes	10^6 USD	Yen/USD	Ratio
Yuan/USD	Ratio	Nikkei 225	Yen
DAX 30	EUR	Policy Uncertainity	Index
Infl-LKD 10Y BId	USD	Gold	USD/T Ounce
FTSE100	GBP		
Crypto Data	**Units**	**Crypto Data**	**Units**
CPTRA: Cost/Trans.	USD	ETRAV: Estimated Trans. Vol.	10^6
TOUTV: BTC Total Output Vol.	10^3	MIREV - Miners Revenue	10^2 USD
NADDU: Num. Unique Addresses	10^3	NTRBL: Num. Trans/Block	10^3
NTREP: Trans. Exc. Popular Addr.	10^3	TRFEE: BTC Total Trans. Fees	USD
TRVOU: Exch. Trade Vol.	10^6	NTRAT-Total Num Trans.	10^6
Virtual Crypto Technologies	USD	MKTCP-BTC Market Cap.	10^6
HRATE: BTC Hash Rate	10^6/sec		

5.4 Results

5.4.1 Kalman LagLasso on three different countries

We started including all the macro-economic variables and let the algorithm determine which one pops out as relevant, without considering the government budget variable in our XNMF due to this variable having negative value. This model demonstrates that the combination of filtering method with a selection method can achieve encouraging results on multi-variate financial data taking into account exogenous information. Figure 5.1 shows the time series being modelled and the corresponding residuals when an autoregressive model of order three is applied.

Figure 5.2 shows how the number of macro-economic variables getting non-zero values at increasing levels of the regularization parameter. There is a monotonic decrease in the number of parameters, as expected. Reviewing from this graph we chose 10 variables for the S&P 500 data and 14 for the FTSE100 data where in both of them there is a 'knee' in the graph, and 7 variables for the Nikkei data where there is a flat region.

This selection is indeed a matter of convenience and in a practical situation of applying such a technique some higher level consideration needs to be brought in. Here, it suffices to say that the prominent explanatory variables is what we seek.

Figure 5.3 shows the LagLasso influence of the different macro-economic variables on the Nikkei, S&P500, FTSE100 index time series, separated by the three lags used. Unemployment and Gold prices are seen as two macro-economic variables that have influence in any index for any lag used. Otherwise, Production, WTI oil and 20Y bonds rate are

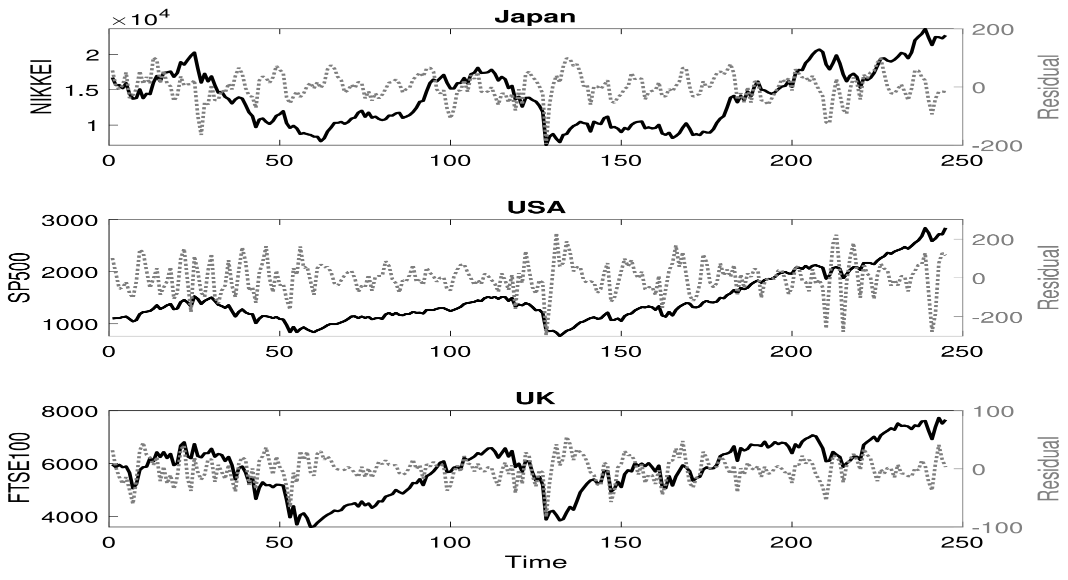


FIGURE 5.1: The time series targets (Nikkei, S& P 500 and FTSE100 values) and the corresponding residuals after Kalman filtering. For convenience the x-axis starts after converging to better illustration.

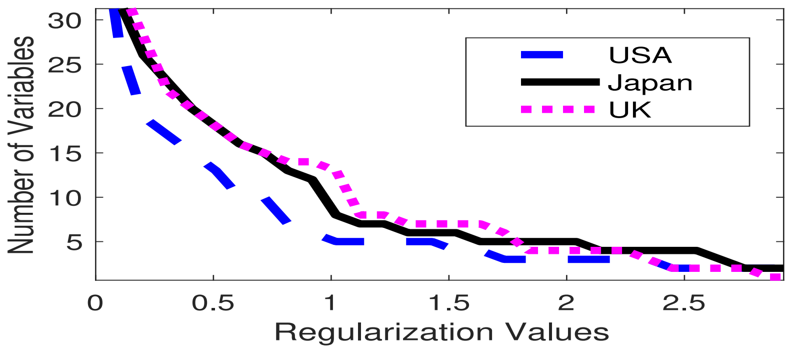


FIGURE 5.2: The number of macro-economic variables (and lags) getting non-zero values at increasing levels of the regularization parameter. Here there is a monotonic decrease where a plateau or knee in the graph help us to select a convenient subset.

the most influential macro-economic variables in all 3 indices. An important effect that is shown in Figure 5.3 is that of oil price has a low influence in Japanese data compared with USA and UK. It has been consistent with the claim of Zhu et al. (2014) that provides evidence of a weak dependence of Asian-Pacific stock market returns with the crude oil prices. Furthermore, Lee et al. (2012) analysed the influence of oil on the G7 countries (where Japan, USA and UK are part of this group), they found that oil price changes were led by stock price changes in Germany, the UK and the US. Another support of this observation is Degiannakis et al. (2014) that found an influence of oil on the European market. This influence on Oil prices linked to index time series has been widely analysed and is supported by several works Sujit and Kumar (2011); Degiannakis et al. (2013);

Chang et al. (2013); Broadstock and Filis (2014). And finally, the negative influence of oil price is also found in Filis et al. (2011) who show that oil prices exercise a negative impact in all stocks markets that were used with lagged correlation, disregarding the origin of the oil price shock.

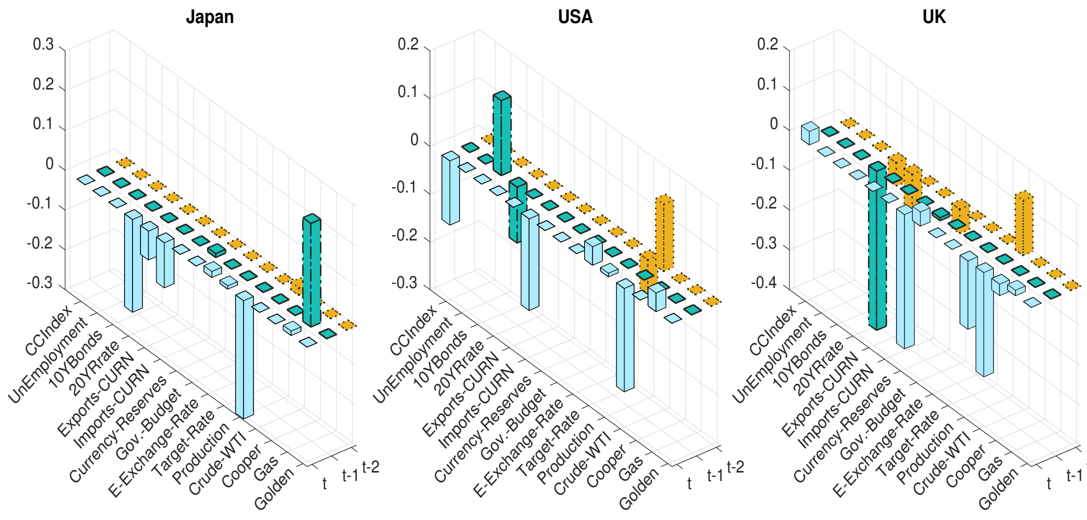


FIGURE 5.3: The LagLasso weights on macro-variables from three different countries. Here we show the influence of the different macro-economic variables on the Nikkei, S& P500, FTSE100 index time series, separated by the three lags used. Unemployment and Gold prices are seen as two macro-economic variables that have not influenced in any index for any lag used. Otherwise, Production, WTI oil and 20Y bonds rate are the most influential macro-economic variables in all 3 indices.

Figure 5.4 shows the exogenous information variables that influence each country market by using the Kalman LagLasso model to monthly variations of the S&P 500, FTSE100 and Nikkei indices. It illustrates that some variables influence either one, two or all the three countries markets. Here we consider even those macro-economic variables that have low influence.

5.4.2 Comparing with Cryptocurrency

Figure 5.5 shows the time series being modelled and the corresponding residuals when an autoregressive model of order three is applied. We note a clear reduction in the variance of the residual which also appears to be zero mean, as expected. Figure 5.6 shows the number of macroeconomic variables getting non-zero values when we increase the levels of regularization, λ, for the daily time-scale (similar effect was found for the monthly values). There is a monotonic decrease in the number of parameters, again as expected. Inspecting this graph we chose 23 variables for the Bitcoin data and 28 variables for the S&P 500 data (where there is a flat region in the graph). This selection is indeed a matter of convenience and in a practical situation of applying such a technique

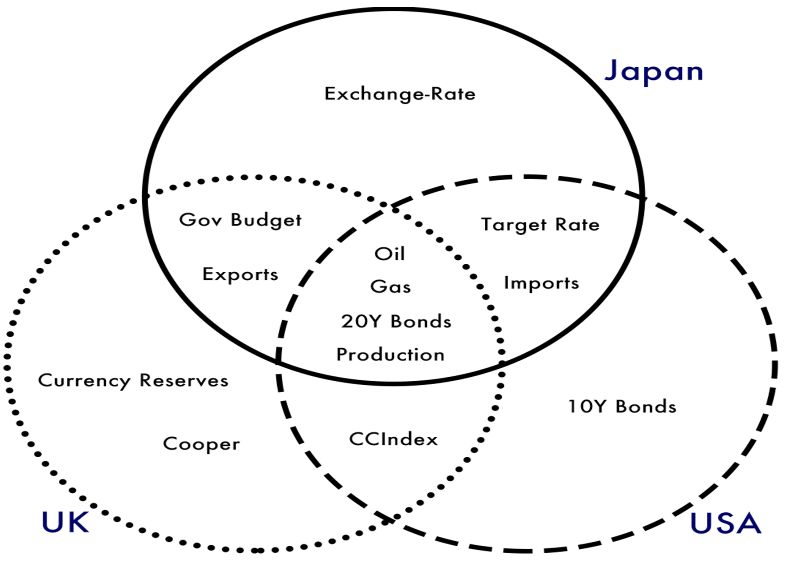
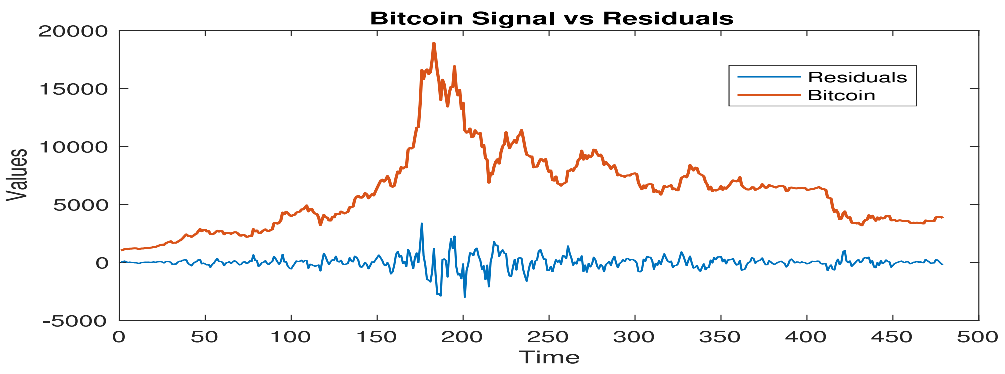


FIGURE 5.4: The selected macro-economic variables that were found using Kalman LagLasso in each country market. Sources of energy, manufacturing and long range interest rates appear to dominate all three.



some higher level consideration needs to be brought in. Here, it suffices to say that the prominent explanatory variables are what we seek.

Figure 5.7 shows the influence of the variables as given by the corresponding weights of the regression solution, separated by the three lags used on daily values, similar to Figure 5.8 where monthly values were analysed. Finally we also found clear difference in the type of variables that influence Ethereum (another popular cryptocurrency) and Dow Jones, which support our analysis of cryptocurrencies do not have any such underlying fundamentals that influence them, unlike the stock market indices.

Unlike stock indices, cryptocurrencies do not have any underlying assets of economic performance to modulate their values. Therefore we can show that cryptocurrency values respond primarily to trader sentiments and objectives. Related to this we identify the work of Phillips and Gorse (2018) who found that there is a relationships between cryptocurrency price changes and topic discussion on social media. They show several examples where topics precede positive and negative price movements on Ethereum and Bitcoin.

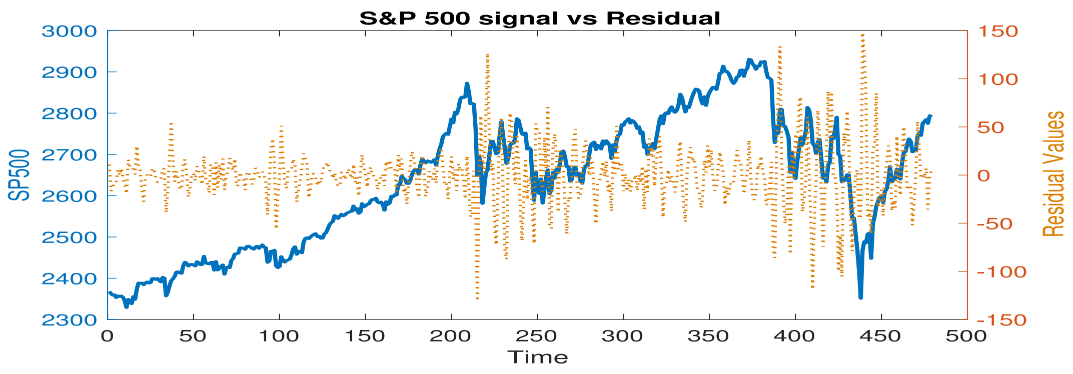


FIGURE 5.5: The target time series (Bitcoin values (a) and S&P 500 index (b)) and the corresponding residuals after Kalman filtering. Initial X axis values are from the time that it converged for better illustration.

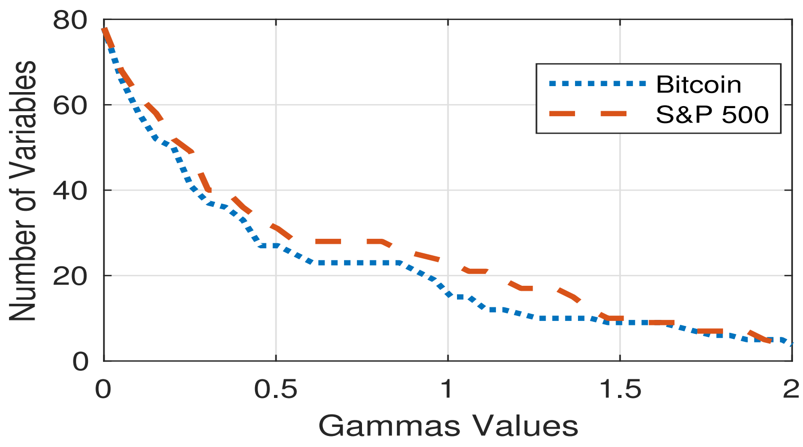


FIGURE 5.6: The number of variables (and lags) selected as the amount of regularisation is increased. In general, there will be a monotonic decrease in the number of non zero coefficients, and we look for a plateau or knee in the graph to select a covenient subset.

5.4.3 Convergence of XNMF Estimation

We show empirically that our XNMF algorithm achieves the desired goal for reducing the error until it reaches a minimum which is illustrated in Figure 5.9. It illustrates the errors of the NMF and XNMF; in every iteration the error is reduced, and the subspace dimension r is also changed. XNMF reaches the minimum error plateaus, despite the its convergence is slower than the NMF algorithm. Additionally, we show the error for different sizes of subspace r to compare NMF (dash line) and the XNMF (solid line). In principle, we would expect the additional information to reduce the error somewhat.

5.4.4 XNMF with sparseness constraint

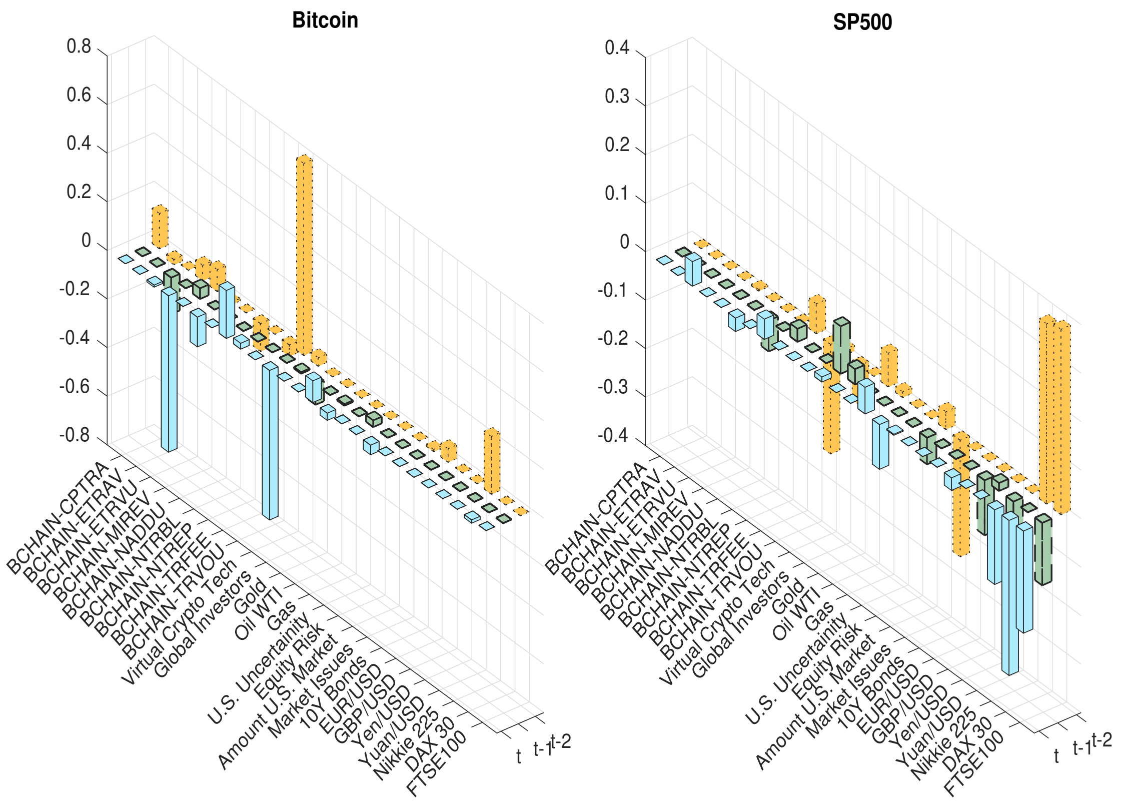


FIGURE 5.7: The LagLasso weights on Bitcoin and S&P 500. Here we show the influence of the different macroeconomic variables on the cryptocurrency and stock index time series with daily values, separated by the three lags used. The financial factors that affected Bitcoin, are not same that influenced S&P 500.

We applied sparseness as we explained in Section 5.2.3 to our XNMF model to get an effective representational scheme of the selected macro-economic variables, with lower number of units. The sparseness applied was 0.7 on $\mathbf{H}_1$, (levels 0 to 1), where high levels of sparseness mean few elements are approximately active and at low levels otherwise. Thus most coefficients are zero while only few take significant values. We applied sparseness constraint to identify the stocks that were influential from each macro-economic variables. Then, we group the constituents by the industrial sector that they belong, to analyse which industrial sectors were the most impacted from those exogenous information variables. These results are shown in Table 5.3. Besides, we show these values in a bar graph in Figure 5.10, where the y-axis values represent the number of stocks that were influenced by each one of the exogenous financial variables grouped by each industry which they belong to. Thus, we can see which industries were most influenced.

5.4.5 Kalman LagLasso and XNMF

After getting the most relevant macro-economic variables from Kalman LagLasso, we compare with those obtained by our XNMF model and we found that three of them are

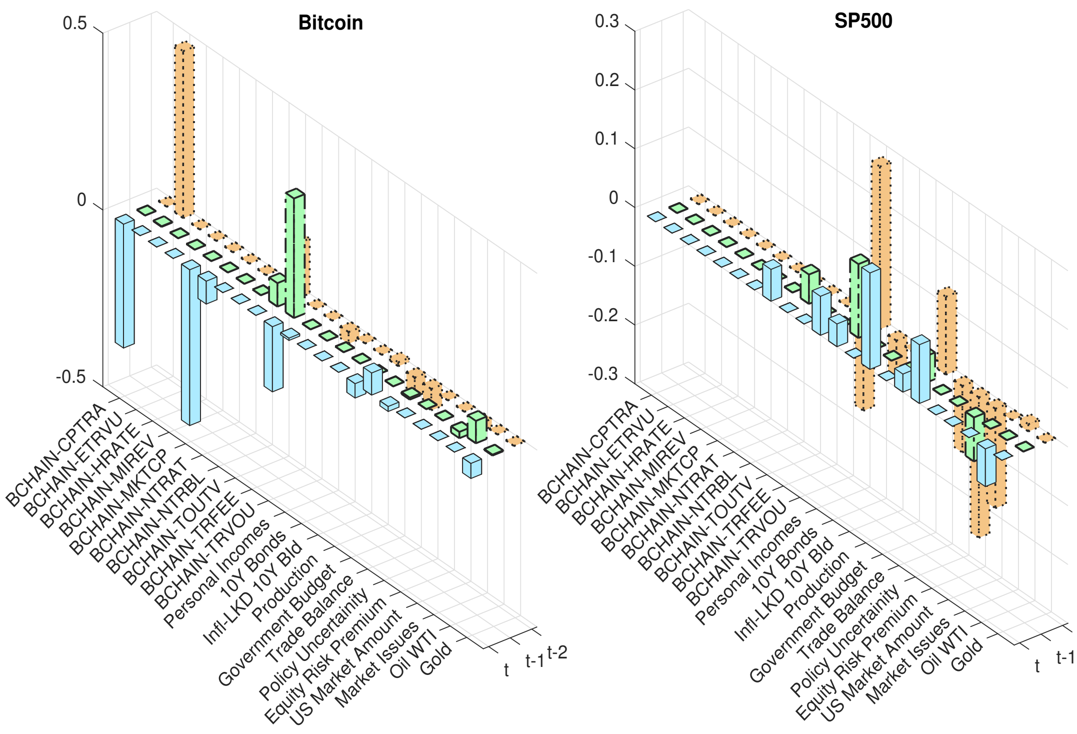


FIGURE 5.8: The LagLasso weights on Bitcoin and S&P 500. Here we show the influence of monthly macroeconomic variables values. There is clear difference similar to Figure 5.7 where the type of variables that influence Bitcoin and S&P 500 are not same.

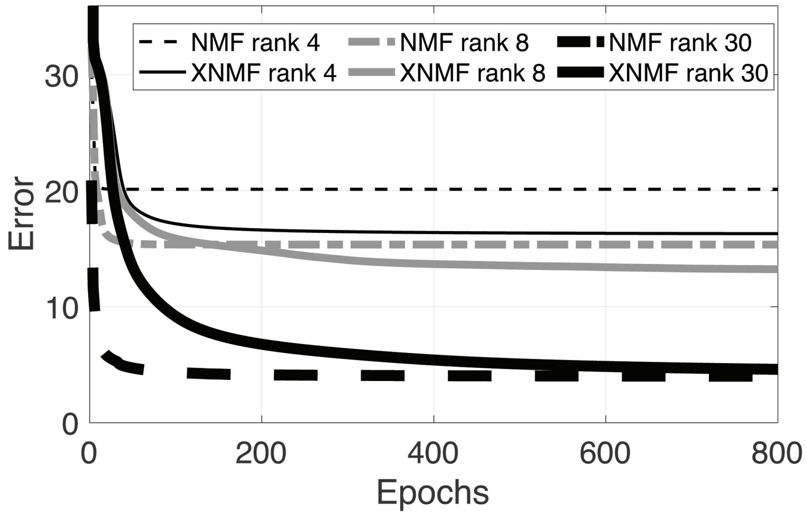


FIGURE 5.9: Performance of XNMF compared to NMF at various ranks. The difference between the two models is more pronounced at low ranks. Generally the XNMF require more iterations to converge.

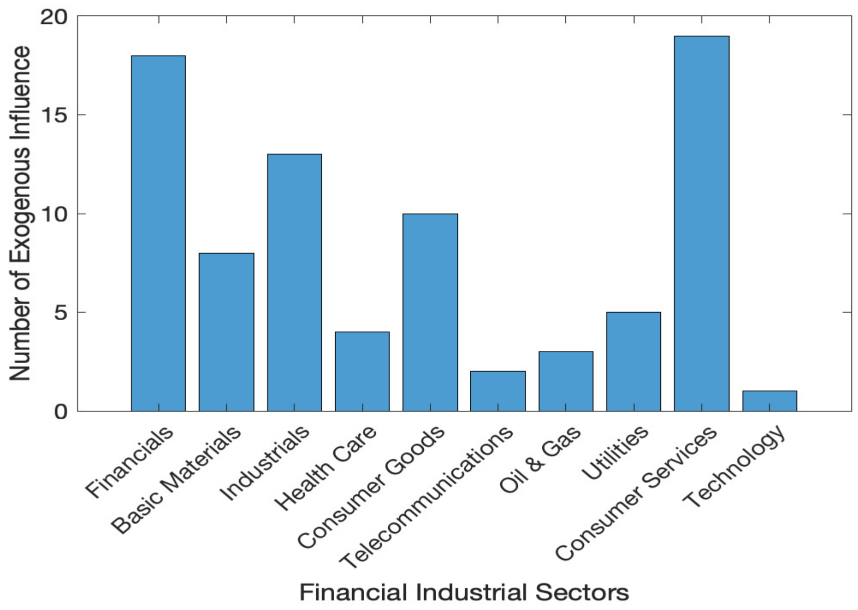


Figure 5.10: The UK industries sectors that were influenced by the exogenous variables. The y-axis values represent the number of FTSE100 constituents that correspond to each industrial sector specified in x-axis. The stocks that belong to Financial and Consumer Services sectors are the most influenced.

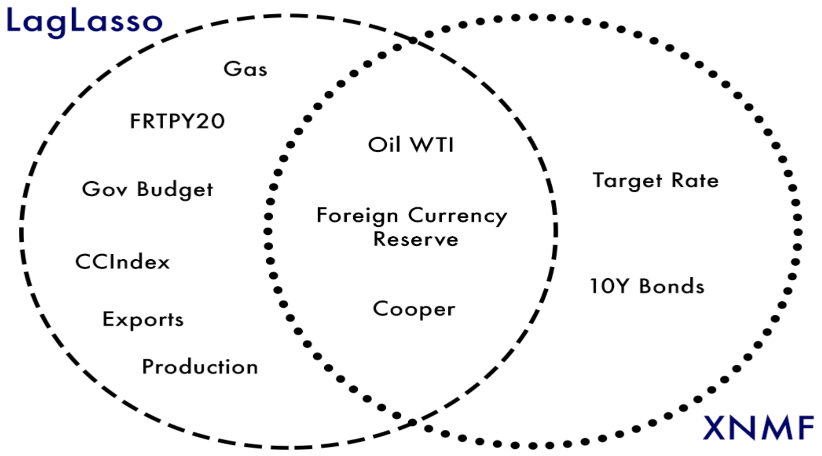
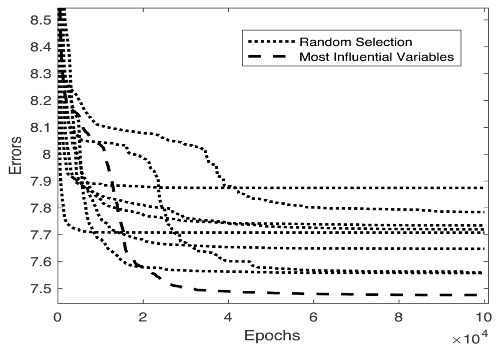


Figure 5.11: The most relevant macro-economic variables that were found by using Kalman LagLasso in Figure 5.4 and our XNMF model in the UK market. Here we can see that three exogenous variables were common in both models and those macro-economic variables were applied to make the comparison that is shown in Figure 5.12

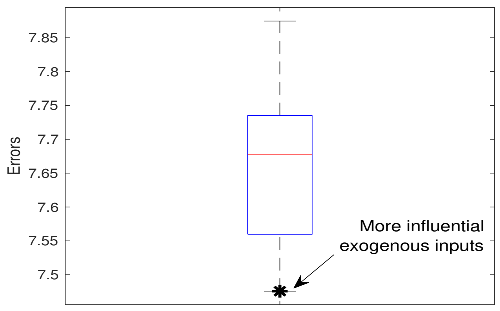
reselected in both models and they are shown in Figure 5.11. The left side group are the financial variables that have been selected by Kalman LagLasso and the right side are those selected by the XNMF model.

5.4.6 Comparing with random selection

Getting these results, we started to analyse the errors found by adding the five most relevant exogenous variables from the XNMF model, comparing with 8 random selections of five exogenous variables. In Figure 5.12 (a) we show that including those five more influential exogenous variables, our XNMF model gets the lowest error compared with a random selection of variables. Besides, we can see that the errors converge for all sets of exogenous variables chosen and the set of inputs with the most relevant financial variables converges with lower errors than the others with random selection. A better representation of these observations is made by Figure 5.12 (b) where it shows all the errors of these comparisons are demonstrating that the selection of the most relevant variables has the lowest the error.



(a) Models



(b) Dataset

FIGURE 5.12: How influential are exogenous macro-economic variables selected in a data-driven way? In (a) we show the variation in approximation error by selecting five random macro-economic variables comparing with the most influential variables. In (b) shows that XNMF with the five more influential exogenous variables (the cross) finds the lowest error compared with the others random set of variables as inputs.

TABLE 5.3: XNMF with sparseness constraint. The number of the FTSE100 constituents that were influenced by the exogenous variables with a 0.7 of sparsity on $\mathbf{H}_1$. We show them with respect to the UK industrial sectors that they belong to. Squares with dark green colour are those with higher influence.

Exogenous Variables	**8000 Financials**	**1000 Basic Materials**	**2000 Industrials**	**4000 Health Care**	**3000 Consumer Goods**	**6000 Telecommunications**	**0001 Oil & Gas**	**7000 Utilities**	**5000 Consumer Services**	**9000 Technology**	**Total**
CCI	1	0	0	0	0	0	0	0	0	0	1
Unemployment	0	0	0	0	0	0	0	0	0	0	0
10 Y Bonds	0	0	1	2	1	0	1	4	3	0	12
20 Y Bonds	0	0	1	0	0	0	0	0	3	0	4
Exports CURN	0	0	1	0	0	0	0	0	0	0	1
Imports CURN	1	0	1	0	0	0	0	1	0	0	3
Foreign Currency Reserves	2	2	4	1	2	0	0	0	2	0	13
Exchange Rate Index	0	0	0	0	0	0	0	0	0	0	0
Target Rate	6	0	0	0	1	0	0	0	2	0	9
Production	0	0	1	0	0	0	0	0	0	0	1
Oil WTI	2	3	3	0	1	0	1	0	3	0	13
Copper	5	2	0	1	2	1	0	0	3	1	15
Gas	1	0	1	0	0	0	1	0	1	0	4
Gold	0	1	0	0	3	1	0	0	2	0	7

5.5 Summary

In this chapter, we report an empirical study comparing two methods for quantifying or identifying the influence of macro-economic variables in financial time series: the dynamical systems model using a Kalman filter followed by a sparsity inducing linear regression on the innovation signal (Kalman LagLasso), and a matrix factorization method that factors in exogenous information as additional terms in its reconstruction. By applying the models to data from stock indices and the prices of their constituent markets in three different geographical areas (UK, USA and Japan), we identify the common variables that have a dominant influence at a macro-economic level. Some of the relationships we identify via these models are confirmed in independent reports of previous authors. By extending our XNMF model with sparseness constraints, we show the influence of exogenous information in the industrial sectors. It helps to prove our second hypothesis that exogenous information help to improve interpretability. We undertake a similar exploration using Kalman LagLasso on cryptocurrency values where one would expect fundamental differences due to the lack of an underlying production activity worthy of any value or merit. We illustrate a fundamental difference between the traded values of cryptocurrencies (such as Bitcoin) and other financial assets (such as stock indices), in that explanatory exogenous variables of relevance are not the same between them in the two time-scales analysed (daily and monthly values). We postulate that this is because cryptocurrency values respond primarily to trader sentiments and objectives. This is shown by the technique of sparsity inducing regularized linear regression modelling residual signals of an autoregressive process applied to Bitcoin and S&P 500 data.

Chapter 6

Deep Learning for Financial Data

We explore the use of two deep learning models for problems with financial data. Deep learning may be able to detect and exploit interactions in the data that are not easy to see in an existing financial economic theory. Multi-variate time series that arise in financial data, for example, are likely to be driven by underlying lower dimensional latent variables. Extracting such latent spaces can be useful in representing the data efficiently and as a means of explaining aspects of the system from which they are generated. Here, we first study an extension to the Variational Autoencoder model specifically cast in a probabilistic setting to deal with positive valued data to extract a non-negative matrix factorization model (NMF) in a probabilistic setting (PAE-NMF). To model financial data, where information about some underlying macroeconomic system may be observed, we extend the PAE-NMF model to include exogenous variables (PAE-XNMF). We present the learning algorithm for this model and illustrate its operation on financial data of constituents of the FTSE100 index and a set of relevant macroeconomic variables. We show an example of the latent space detecting a sharp transition around the Brexit event that is not readily apparent on any of the individual time series. Secondly, we implement the Long Short Term Memory (LSTM) model that is a class of RNN for one step ahead time series prediction on derivative option from its past values. And we aim to analyse the LSTM gating signals in its response to changes in the underlying interest rate. And finally we applied the dual attention mechanism to the LSTM to see if it can explains the residual r from 5.5 by exogenous variables.

6.1 Introduction

Predicting stock prices is difficult work. Many studies are trying to find an alternative to improve the selection of input variables to achieve good performance when forecasting with multivariate time series. Machine learning techniques for modelling multivariate time series have been widely applied (Tkacz, 2001; Cook and Hall, 2017; Mahler, 2009), including Variational Autoencoders (VAEs) (Kingma and Welling, 2013). VAEs have become a popular generative model for dimensionality reduction. VAEs work with diverse types of data, sequential, non-sequential (Hsu et al., 2017), continuous, discrete (Dupont, 2018), labelled and unlabelled data (Kawachi et al., 2018; Xu et al., 2018), making them highly powerful generative tools. Promising results have been shown in different research domains, such as handwritten digits (Kingma and Welling, 2013; Salimans et al., 2015), CIFAR images (Gregor et al., 2015), physical models of scenes (Kulkarni et al., 2015), segmentation (Sohn et al., 2015) and predicting the future events that might happen from static images (Walker et al., 2016).

We adopt the VAE framework and combine it with our XNMF method by transforming the input data with exogenous inputs. This model we call probabilistic non-negative matrix factorization with exogenous information (PAE-XNMF). As XNMF is an extension of the NMF technique, PAE-XNMF is an extension of PAE-NMF (Squires et al., 2019) which performs NMF using an Autoencoder framework. Unlike XNMF, PAE-XNMF is a probabilistic technique that helps to generate new financial data taking into account exogenous information. Probabilistic methods are widely applied for analysis of financial data, for instance using a Bayesian approach (Kim and Nelson, 1999; Jacquier et al., 2002; Smets and Wouters, 2007; Kim and Nelson, 1999). Using generative models with deep learning to learn a representation has gained attention recently. To the best of our knowledge, this approaches mainly use generative adversarial networks (GANs) (Goodfellow et al., 2014) with financial data (Jang and Lee, 2019; Zhou et al., 2018; Takahashi et al., 2019).

On the other hand, an emerging interest in the financial area is the regime-switching model, which are reflected changes in the financial market (Assaf, 2017; Chatziantoniou et al., 2017; Zhang and Chan, 2016; Hu and Wang, 2017; Bazzi et al., 2017). A change in the market can be a trend in the general market because underlying movements of an individual stock. The Market regime can be modelled by a finite-state Markov chain which modulates the rate of return and volatility. In a regime-switching (RS) model, the stock parameters are considered on the regimes which are limited number of states. These regime points up states on the underlying asset, behaviours of traders, the mood of the shareholders and any additional economic aspect (Hamilton, 1989). An example of the modelling regime switching is the work of Goutte et al. (2017) who introduced a homogeneous continuous time Markov chain on a finite space, which represents the regime state of volatility.

A powerful tool for capturing/modelling statistically significant properties underlying time varying data is the Recurrent Neural Network (RNN), which are neural networks with feedback in them. The first work to compute back-propagation for such networks is Pineda (1987). RNNs have formed a range of applications including Speech recognition (Robinson and Fallside, 1991), low bit rate coding (Wu et al., 1994), EKF training of RNN (Puskorius and Feldkamp, 1994), stock data and expected returns (Saad et al.,

1998) and to predict stock return with a hybrid model between ARMA model, an exponential smoothing model and RNNs (Rather et al., 2015). A particular version of RNN introduced by Hochreiter and Schmidhuber (1997) is the Long Short Term Memory (LSTM) model. It has gating signals (input gate, output gate and forget gate) which are introduced into the network through a memory cell that allows the neuron to choose when to forget and when to remember things. Each memory cell is associated with every gate that feeds into itself across time steps. This neural network is popular for sequential data processing such as translations, sound, time series, written natural language and handwriting recognition.

Finally, we apply attention mechanism that was created in order to make neural networks more interpretable (Vaswani et al., 2017). Attention mechanism allows RNNs to focus on relevant parts of series for prediction. This has been applied to sales prediction. Chen et al. (2018) applied dual attention to ensure sales prediction by compensating the unknown states of influential factors in the future sales volume values and align the upcoming trend with the most relevant one from the past. Here we extend the application of the the DA-RNN (Qin et al., 2017). A dual attention mechanism with two stages that are integrated within a LSTM. The first one extracts relevant input features and the second stage uses a temporal attention mechanism to get the most relevant hidden states from all the time steps. In their work related to financial data, they used the constituents of the NASDAQ index at $t - 1$ to predict the NASDAQ index value at time t. Their inputs NASDAQ constituents are highly correlated to the values of the index price that they try to predict and it can not be a good example of the effect of the attention mechanism. For that reason, we extended their work with a different set of inputs to make predictions on the residual that was obtained by Kalman filtering (Section 5.5), in order to improve interpretability in the model.

This chapter is organised as follows. In Section 6.2 is the explanation of the methodologies. In Section 6.2.3, the list of data used in our work. In Section 6.2.4 are the experiments we have run and the results we achieved. Section 6.3.2 has the implementation of the attention mechanism, Section 6.3.2.1 has the illustration of applying attention mechanism with LSTM. Finally, the conclusion are stated in Section 6.4.

6.2 Implementation of PAE-XNMF

6.2.1 XNMF and AE-XNMF

The aim of XNMF (Squires et al., 2017a) is to produce the approximation $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2$ where $\mathbf{V} \in \mathbb{R}^{m\times n}$ is our input matrix with m dimensions and n data-points, $\mathbf{W}_2 \in \mathbb{R}^{m\times r_2}$ is a matrix with constant elements produced from some external data-source, and $\mathbf{W}_1 \in \mathbb{R}^{m\times r_1}$, $\mathbf{H}_1 \in \mathbb{R}^{r_1\times n}$ and $\mathbf{H}_2 \in \mathbb{R}^{r_2\times n}$ are all factorised matrices to be found by the algorithm. The external dimension r_2 depends on the choice of exogenous data while r_1, the chosen subspace size for the data, should be picked either with domain knowledge or using some form of automated method (Squires et al., 2017b). The external data is expected to act as a driver to force an improved representation of the data because we are adding prior knowledge into the NMF formulation.

The original XNMF work in Section 4 used an extended form of the multiplicative updates of Lee and Seung (Lee and Seung, 1999) to find the factorised matrices but this method does not allow for the use of Variational Autoencoder methods. However, it has been shown that XNMF can be performed using an Autoencoder (Squires et al., 2019) by imposing additional constraints onto the Autoencoder framework. We show the structure of an Autoencoder designed to perform XNMF in Figure 6.1.

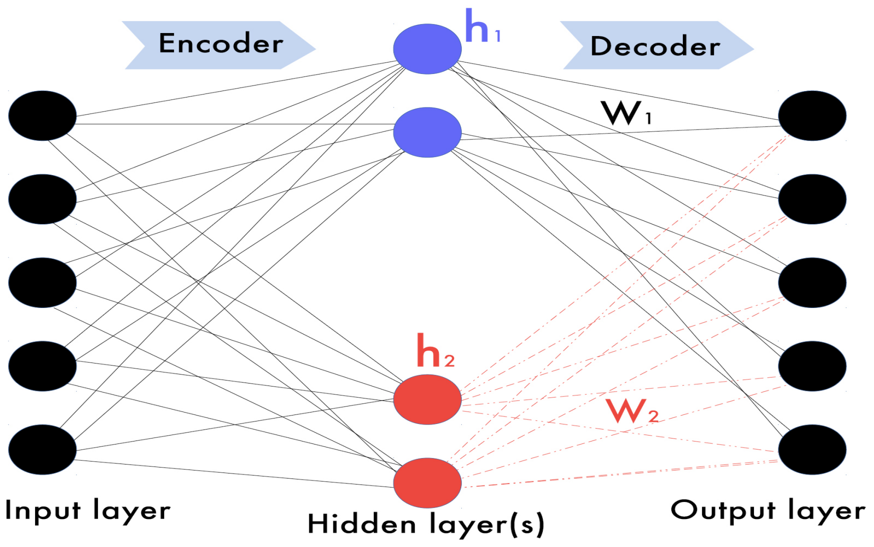


FIGURE 6.1: A standard Variational Autoencoder for performing XNMF (adapted from Squires et al. (2019)) with one hidden layer that is a low dimensional compression of inputs variables into latent factors, which can be expressed as weighted linear combinations of input variables. The red nodes represent the exogenous information.

A major downside of XNMF (and NMF) is that they are deterministic which means we get no uncertainties on the lower dimensional representation, are not able to sample new data-points from the distributions and do not have a principled method to decide on the level of regularisation. However there are some studies that have worked with a Bayesian treatment of non-negative matrix factorization (NMF) (Févotte and Cemgil, 2009; Schmidt et al., 2009; Mohammadiha et al., 2013, 2012). But to the best of our knowledge, they have not been applied to financial data.

In this section we address these drawbacks by producing a probabilistic form of XNMF using a modified Variational Autoencoder called PAE-XNMF.

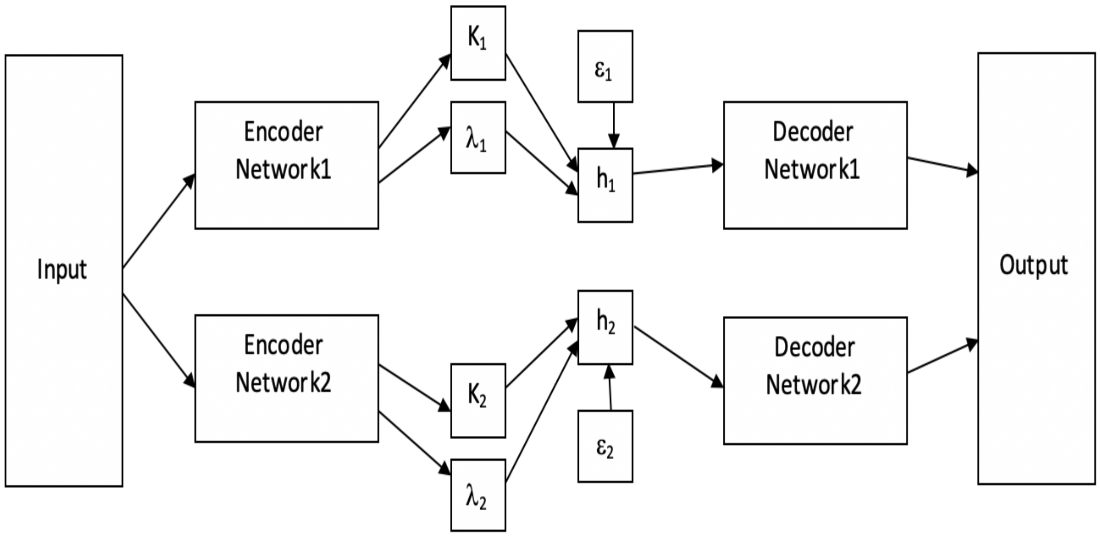


FIGURE 6.2: PAE-XNMF with stochasticity provided by the input vectors ϵ_1 and ϵ_2.

6.2.2 PAE-XNMF

The structure of the PAE-XNMF is displayed in Figure 6.2. Here, we need to learn the Encoder Networks 1 and 2 and the Decoder Network 1. The Decoder Network 2 is the external data and remains constant throughout the training process. We do not specify the exact nature of the encoding and decoding networks because variations are possible, for example including multiple layers before the construction.

A VAE uses the encoding network to find the parameters of a distribution, usually a Gaussian. However, for XNMF we need a distribution which does not contain negative values so we utilise the Weibull distribution, as it is used in Squires et al. (2019). This distribution is parametrised by k and λ with a probability density function given by:

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}\exp(-(x/\lambda)^k) & \text{if x} \geq 0 \\ 0 & \text{if x} < 0 \end{cases} \tag{6.1}$$

We choose the Weibull distribution for the same reason as the PAE-NMF authors, namely that it is a non-negative distribution and has a simple inverse cumulative function, which is given by $C^{-1}(\epsilon) = \lambda(-\ln(\epsilon))^{1/k}$, which is important for sampling from the distributions, as will be discussed later.

The structure of the network shown in Figure 6.2 is very general we will now discuss the specific version of this network we have used. Once the network has been trained the flow through the network starts with an input vector $\mathbf{v}_i$. This input is then fed through the encoding networks which produces four vectors $\mathbf{k}_1$, $\boldsymbol{\lambda}_1$, $\mathbf{k}_2$ and $\boldsymbol{\lambda}_2$; which contain the parameters of the distributions. From $\mathbf{k}_1$ and $\mathbf{l}_1$ we want to draw samples that form

$\mathbf{h}_1$ and equivalently for $\mathbf{h}_2$ but we cannot do so because if we sample directly from the distributions we cannot differentiate through the network. To get around this problem we utilise the reparameterisation trick (Kingma and Welling, 2013) which involves injecting stochasticity into the network from at input node, denoted by $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ in Figure 6.2. As we do not need to do backpropagation through these input nodes we have a fully differentiable network. The two vectors $\mathbf{h}_1$ and $\mathbf{h}_2$ are now passed through their respective decoder networks which produce the output, $\hat{\mathbf{v}}$.

When we perform backpropagation we do not update the weights of the second decoder network as this is the external data we are inserting into the network. For the first decoder network we set any negative values back to zero after the backwards pass as this first decoder network is the equivalent to $\mathbf{W}_1$ in XNMF. The network designed in this chapter has two matrices for each encoder network. For Encoder Network 1 we take in an input vector, $\mathbf{v}$, and perform: $\mathbf{k}_1 = \sigma(\mathbf{W}_1^{(k)}\mathbf{v})$ and $\boldsymbol{\lambda}_1 = \sigma(\mathbf{W}_1^{(\lambda)}\mathbf{v})$. Equivalently for Encoder Network 2: $\mathbf{k}_2 = \sigma(\mathbf{W}_2^{(k)}\mathbf{v})$ and $\boldsymbol{\lambda}_2 = \sigma(\mathbf{W}_2^{(\lambda)}\mathbf{v})$. The activation function, σ, for all of these is the rectified linear unit (ReLU) which acts element-wise. The next step is to sample from the distributions to get $\mathbf{h}_1$ and $\mathbf{h}_2$ which we do by performing $\mathbf{h}_1 = C^{-1}(\boldsymbol{\epsilon}_1) = \boldsymbol{\lambda}_1(-\ln(\boldsymbol{\epsilon}_1))^{1/k_1}$ and $\mathbf{h}_2 = C^{-1}(\boldsymbol{\epsilon}_2) = \boldsymbol{\lambda}_2(-\ln(\boldsymbol{\epsilon}_2))^{1/k_2}$. In the decoding layer we have two matrices for Decoder Network 1 and 2 respectively which perform: $\mathbf{k}_1 = \sigma(\mathbf{W}_1^{(k)}\mathbf{v})$. The objective function we are minimising is given by:

$$\begin{aligned}\text{ojb} &= \mathbb{E}_{q_\phi(\mathbf{h}_1,\mathbf{h}_2|\mathbf{v})}(-\log(p_\theta(\mathbf{v}|\mathbf{h}_1,\mathbf{h}_2)) + D_{KL}\big(q_\phi(\mathbf{h}_1|\mathbf{v})\|p(\mathbf{h}_1)\big) + D_{KL}\big(q_\phi(\mathbf{h}_2|\mathbf{v})\|p(\mathbf{h}_2)\big) \\ &\approx \frac{1}{2\sigma^2}\|\mathbf{v}-\hat{\mathbf{v}}\|^2 + D_{KL}\big(q_\phi(\mathbf{h}_1|\mathbf{v})\|p(\mathbf{h}_1)\big) + D_{KL}\big(q_\phi(\mathbf{h}_2|\mathbf{v})\|p(\mathbf{h}_2)\big) \qquad (6.2)\end{aligned}$$

Our implementation of PAE-XNMF using the pytorch library (Paszke et al., 2017) is described in pseudocode in Algorithm 4.

6.2.3 Data

We have collected financial information from December 2014 to April 2019. It consists of the FTSE100 constituents and 16 arbitrary exogenous information that contains macro-economic variables of UK market, others countries stock market index and currencies exchange rate. Table 6.1 shows the variables with daily values that were acquired from Thomson Reuters Datastream Platform. The initialisation of the parameters of the network is likely to be important. As we effectively have two networks which are merged at the end we do not want one of the two networks to come to dominate the solution just because of the parameter initialisation. To counter this effect we initialise the network such that the matrices are similar in size as each other and add up to a similar result to the average of $\mathbf{V}$.

Algorithm 4 PAE-XNMF Algorithm

```
 1: Input: A matrix $\mathbf{V} \in \mathbb{R}^{m \times n} \leftarrow m$ observations and $n$ constituents numbers
 2: Initialize $\mathbf{W}_1 \in \mathbb{R}^{m \times r_1}$, to random positive values
 3: Input: A matrix $\mathbf{W}_2 \in \mathbb{R}^{m \times r_2} \leftarrow$ Exogenous macro-economic information
 4: while epochs do
 5:     Encoding Network 1 $\mathbf{k}_1 = \sigma(\mathbf{W}_1^{(k)}\mathbf{v})$, $\boldsymbol{\lambda}_1 = \sigma(\mathbf{W}_1^{(\lambda)}\mathbf{v})$
 6:     Encoding Network 2 $\mathbf{k}_2 = \sigma(\mathbf{W}_2^{(k)}\mathbf{v})$ and $\boldsymbol{\lambda}_2 = \sigma(\mathbf{W}_2^{(\lambda)}\mathbf{v})$
 7:     Sampling from the distributions using the inverse cumulative function.
        $\mathbf{h}_1 = \boldsymbol{\lambda}_1(-\ln(\boldsymbol{\epsilon}_1))^{1/k_1}$ and $\mathbf{h}_2 = \boldsymbol{\lambda}_2(-\ln(\boldsymbol{\epsilon}_2))^{1/k_2}$
 8:     reconstruct data $\hat{\mathbf{V}} = \mathbf{W}_1(\mathbf{h}_1) + \mathbf{W}_2(\mathbf{h}_2)$
 9:     Get $\mathbf{e}_r$, the reconstruction error by MSE
10:     Get $\mathbf{KLD}_1$ and $\mathbf{KLD}_2$, KL-divergences which perform regularisation
11:     Back-propagate loss= $\mathbf{e}_r + \mathbf{KLD}_1 + \mathbf{KLD}_2$
12:     Update: $\mathbf{k}_1, \boldsymbol{\lambda}_1, \mathbf{k}_2, \boldsymbol{\lambda}_2, \mathbf{W}_1$
13: end while
14: return $\mathbf{k}_1, \boldsymbol{\lambda}_1, \mathbf{k}_2, \boldsymbol{\lambda}_2, \mathbf{W}_1, \mathbf{h}_1, \mathbf{h}_2, \hat{\mathbf{V}}$
    Above, $r_1$ and $r_2$ are the dimension reduction and number of exogenous respectively.
```

Table 6.1: The exogenous macro-economic variables

Variables	Units	Variables	Units
ASR Equity Risk Premium	%	TR Gvt BMK BID YLD 20Y	%
TR GVT BMK BID YLD 10Y	%	Eco Policy Uncertainty	index
10 Year DS Govt.	Index	1M FX Volatility	GBP
GBP/USD	ratio	Eur/GBP	ratio
YUAN/GBP	ratio	Yen/GBP	ratio
S&P 500	USD	Hang Seng	HKD
Shanghai SE	Yuan	Gold	USD/T oz
OIL	USD/barrel	NIKKEI 225	Yen

6.2.4 Results PAE-XNMF

We first analyse if the data generated from our PAE-XNMF model is close to the real data. Figure 6.3 shows the reconstruction of four FTSE100 constituents that were selected randomly from different UK industrial sectors. It shows that PAE-XNMF method can generate new data fairly close to the real values with the respective uncertainty (standard deviation). Figure 6.4 shows the original matrix $\mathbf{V}$ that is composed by all the FTSE100 constituents and the reconstruction $\hat{\mathbf{V}}$ by using our PAE-XNMF model. For that reason we can state that exogenous information enhance the model for generation of new financial data in order to prove our first hypothesis.

Table 6.2 shows the values of the matrix $\mathbf{H}_2$ of our model. These values are the influence of each exogenous variables to each one of the FTSE100 constituents (the largest UK conmpanies). Every of those stock belongs to a specific industrial sector that have been grouped for getting the total of the number of exogenous influence. It helps to understand the effect of the exogenous variable for the new generation. Thus, it can contribute in the second hypothesis of improving interpretability.

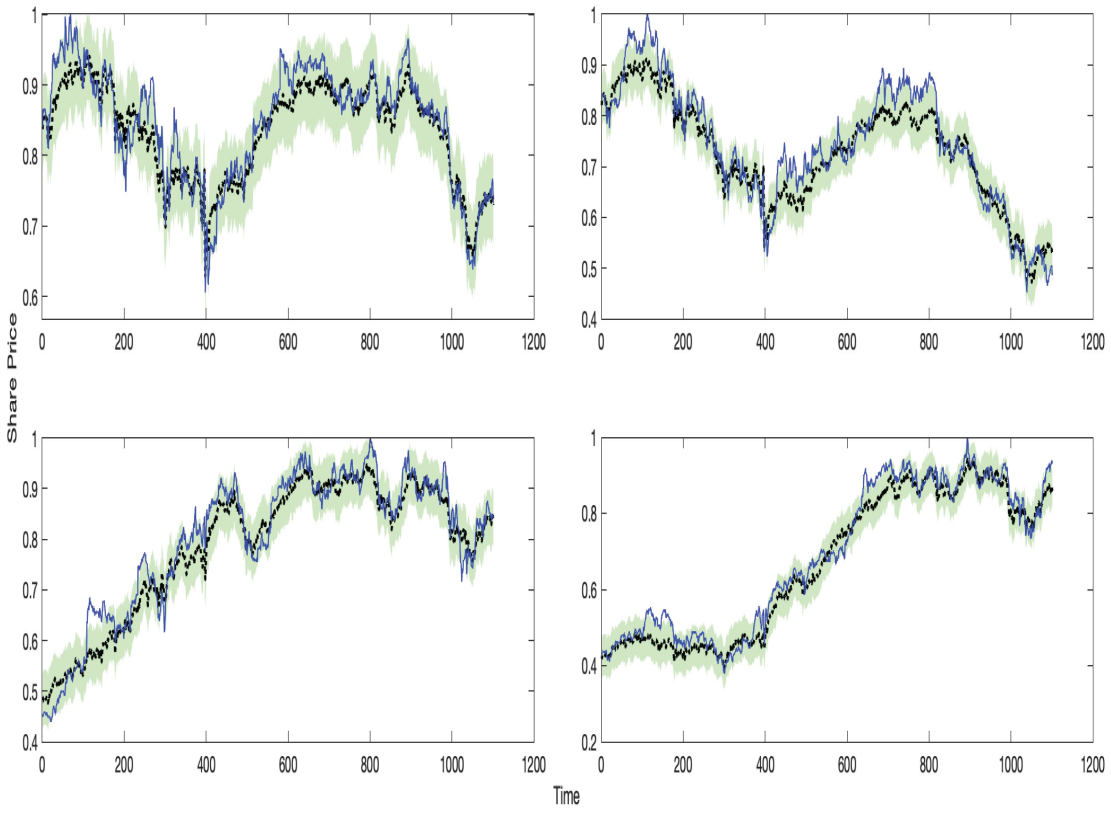


Figure 6.3: Reconstruction of four FTSE100 constituents. Blue lines correspond to the real values of those constituents, the black dashed line is the reconstructed values with the uncertainty (standard deviation) as the shaded outline with colour green. We can see that our PAE-XNMF method can generate new data fairly close to the real values.

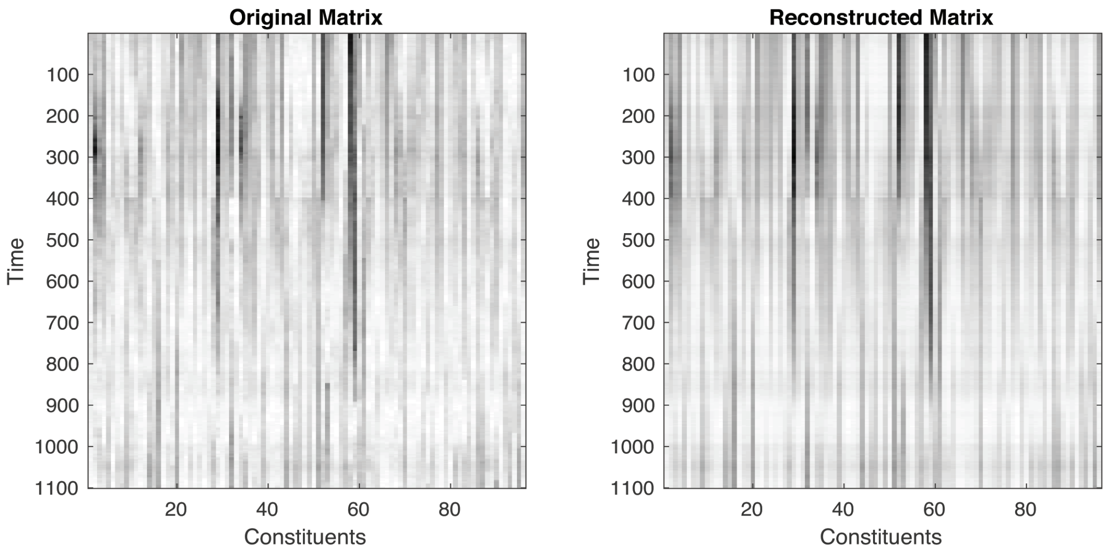


Figure 6.4: Reconstruction of the matrix that is composed by all the FTSE100 constituents using our PAE-XNMF model by $\hat{\mathbf{V}} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2$

TABLE 6.2: The matrix $\mathbf{H}_2$ of our model shows the influence of each exogenous variables to each one of the FTSE100 constituents, then we grouped them by the UK industrial sector that each stock belong to, for getting the total of the number of exogenous influence.

	0001 Oil & Gas	**2000 Industrials**	**3000 Consumer Goods**	**4000 Health Care**	**5000 C. Services**	**6000 Telecom.**	**7000 Utilities**	**8000 Financials**	**9000 Techn.**	**1000 Basic Materials**
ASR EQUITY RISK PREMIUM	0	3	1	1	0	0	0	3	0	7
TR UK GVT BMK BID YLD 20Y	0	0	0	0	0	0	0	0	0	0
TR UK GVT BMK BID YLD 10Y	0	0	0	0	0	0	0	0	0	0
UK ECON POLICY UNCERTAINTY	2	8	6	0	0	0	2	4	2	3
10 YEAR DS GOVT. INDEX	0	0	0	0	0	0	0	0	0	0
POUND STERLING 1M FX VOLATILITY	1	3	1	3	0	0	0	2	0	0
GBP/USD	0	1	0	0	0	0	0	0	0	0
EURO/USD	0	0	4	0	0	2	3	12	0	0
YUAN/GBP	0	0	0	0	0	0	0	0	0	0
YEN/GBP	0	0	0	0	0	0	0	0	0	0
S&P 500	0	0	0	0	0	0	0	0	0	0
HANG SENG - PRICE INDEX	0	0	0	0	0	0	0	0	0	0
SHANGHAI SE A SHARE - PRICE INDEX	0	0	0	0	0	0	0	0	0	0
GOLD	0	0	0	0	0	0	0	0	0	0
CRUDE OIL	0	0	0	0	0	0	0	0	0	0
NIKKEI 225 STOCK AVERAGE	0	0	0	0	0	0	0	0	0	0

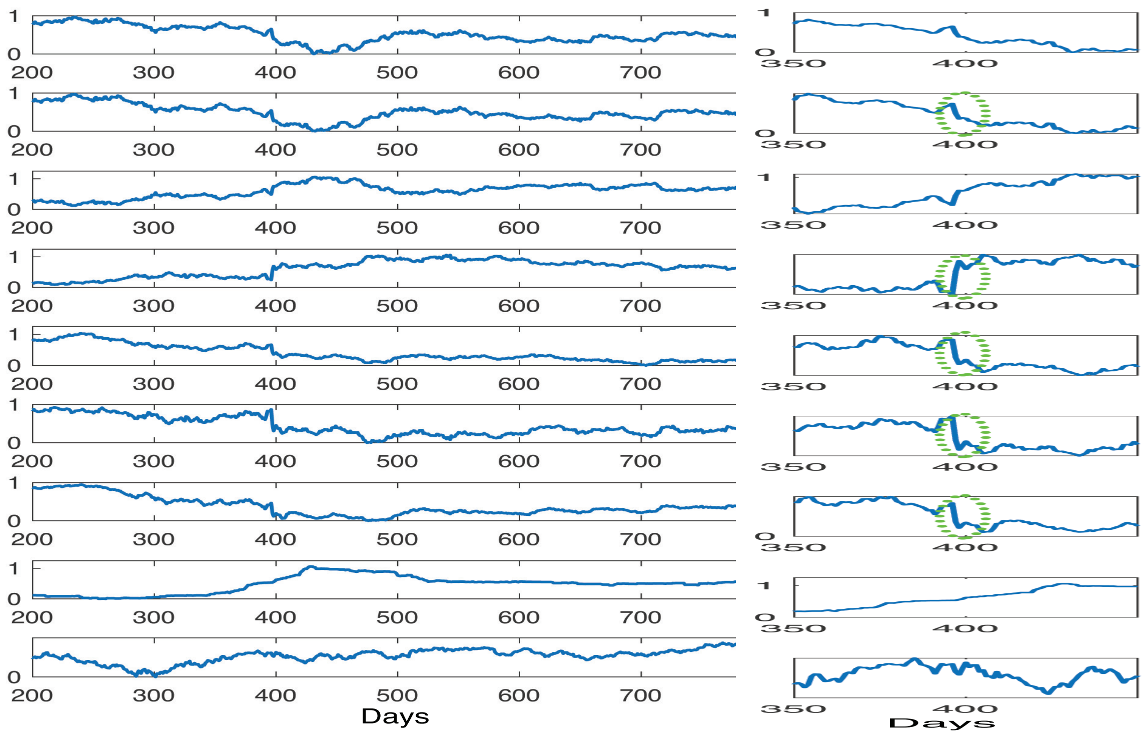


FIGURE 6.5: Nine dimensional latent space representation of the 96 FTSE100 Data (matrix W1 in the model). The zoomed version around time samples 350 to 450 shows a sharp discontinuity in June 2016 when the Brexit referendum took place. Such sharp change, seen in the latent representation, is not seen elsewhere in the original data.

Figure 6.5 shows the lower dimension representation of the FTSE100 constituents extracted from Matrix $\mathbf{W}_1$. It can be interpreted as the underlying components that forces the FTSE100 constituents according to Liu, Tang (2009). For convenience we chose the rank value of $r = 9$ where $r < n$ and $n = 96$ is the number of constituents. Here we can see different effects of the Brexit referendum results announcement that happened on day 398 (23/06/2016), where it influences some of the underlying components. In the zoomed part shows a sharp discontinuity when this event happen. We can extract which are those constituents that are forced for each underlying component by analysing the matrix $\mathbf{H}_1$. It can be used in the analysis of the Brexit impact, which is another topic of research that has gained attention in the last three years. Some studies are finding a way to avoid a negative impact and also analyse the effect of this uncertainity in the stock market (Dhingra et al., 2016b; Ebell and Warren, 2016; Dhingra et al., 2016a; McGrattan and Waddle, 2017).

Finally we compare our PAE-XNMF model with the probabilistic NMF to find if adding exogenous information can contribute in the reconstruction of the financial data that we want to generate after training our model. Figure 6.6 shows our model getting lower reconstruction error than the Probabilistic NMF in all the ranks values that we tested. For clarity of illustration, we have included the learning curves taken only two values of the ranks.

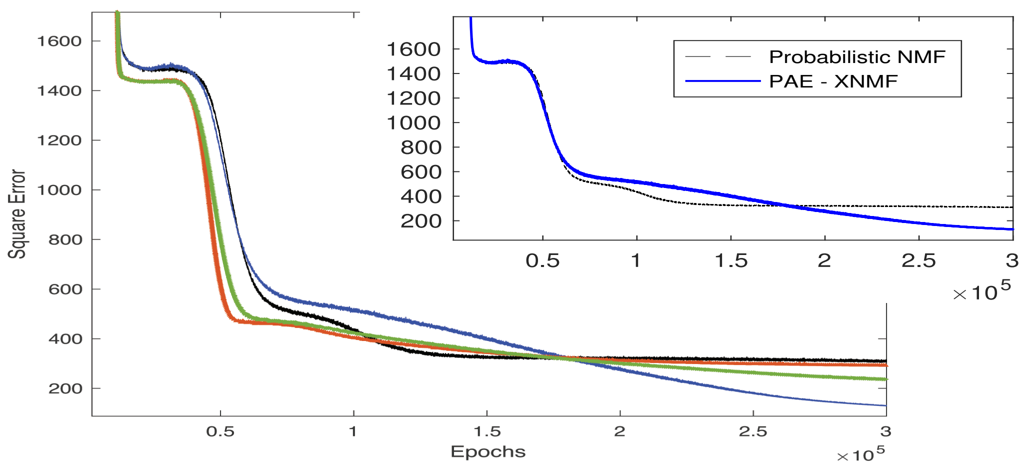


FIGURE 6.6: Comparison between PAE-XNMF and Probabilistic NMF. We compared the models at two different ranks of nine (in colours blue and black, also shown expanded in inset) and twelve (in colours green and orange). The use of exogenous information gives a marginally lower reconstruction error in both cases.

6.3 Implementation of Recurrent Neural Network LSTM

We implement the Long Short Term Memory (LSTM) model that is a class of RNN for one step ahead time series prediction on derivative option from its past values. And we aim to analyse the LSTM gating signals in its response to changes in the underlying interest rate.

We use Python framework for fast computation of mathematical expressions (Theano Development Team, 2016), because it gives more freedom to compute our models. Programing this architecture was done using the library Keras (Chollet, 2015b) with Google TensorFlow (Abadi et al., 2016) which has many advantages such as it supports recurrent networks and it runs seamlessly on CPU and GPU. The training of our model is by gradient descent using Adam (Kingma and Ba, 2014), with a learning rate of 0.001.

6.3.1 Gating Signals Analysis

We focus on understanding the dynamic system inside of the gating signal of the LSTM. We started asking if LSTM can detect any underlying patterns in the financial data. For this, we decide to work with regime switching in the European and American stock market. We are considering historical data from the European financial crisis in 2007-2009, because the interest rate values were changing frequently, and it is our target for the regime switching analysis. Our data is the FTSE-100 index value from that crisis period time.

We initially analyse if a recognized model in regime detection called Markov Regime Switching model (MRSM) can detect regime or variation on the FTSE100. This model was implemented using matlab toolbox Ms_Regress from Perlin (2015). In Figure 6.7,

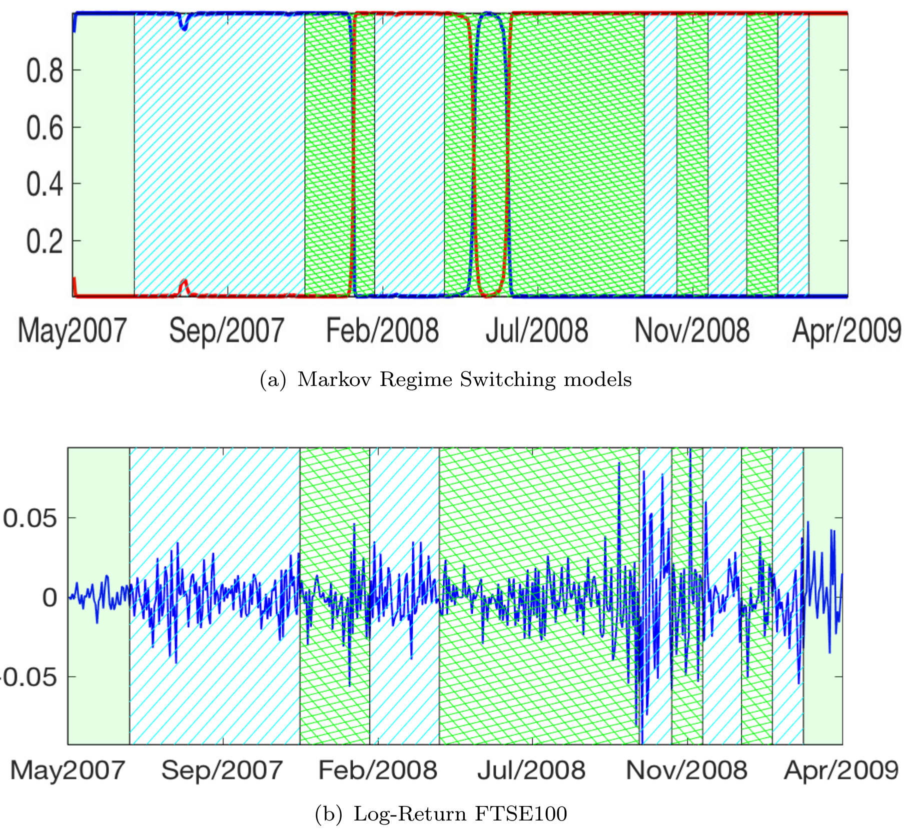


FIGURE 6.7: The Markov Regime Switching model result. The colours in the background represent changes in the interest rate of the UK. In Figure 6.7(a) are the two states from the Markov model and in Figure 6.7(b) are the Log-return of the FTSE100 to compare if any pattern on it is detected as well. It shows that Markov Switching model can barely detect any variation or regimes on the FTSE100 values and can hardly detect the changes on interest rate.

in which the different colours in the background represent changes in the interest rate of the UK, it shows that MRSM fails to detect changes in the interest rate of the UK. Also, it hardly detect variations in the the Log-return of the FTSE100.

LSTM gating signals also is not capable to detect the regime switching, but it in some parts detected the changes in the interest rate. Figure 6.8 shows the the gating signals which we analyse for this propose. Furthermore, we see that the gating signals get significant changes on the log-return of the FTSE-100 index value.

Thus, understanding the LSTM signal may help in our second research question of getting interpretability.

6.3.2 Dual-Stage Attention

In this section we implement the dual stage attention mechanism introduced from Qin et al. (2017). Figure 6.9 shows the structure of this, which is composed of two stages of

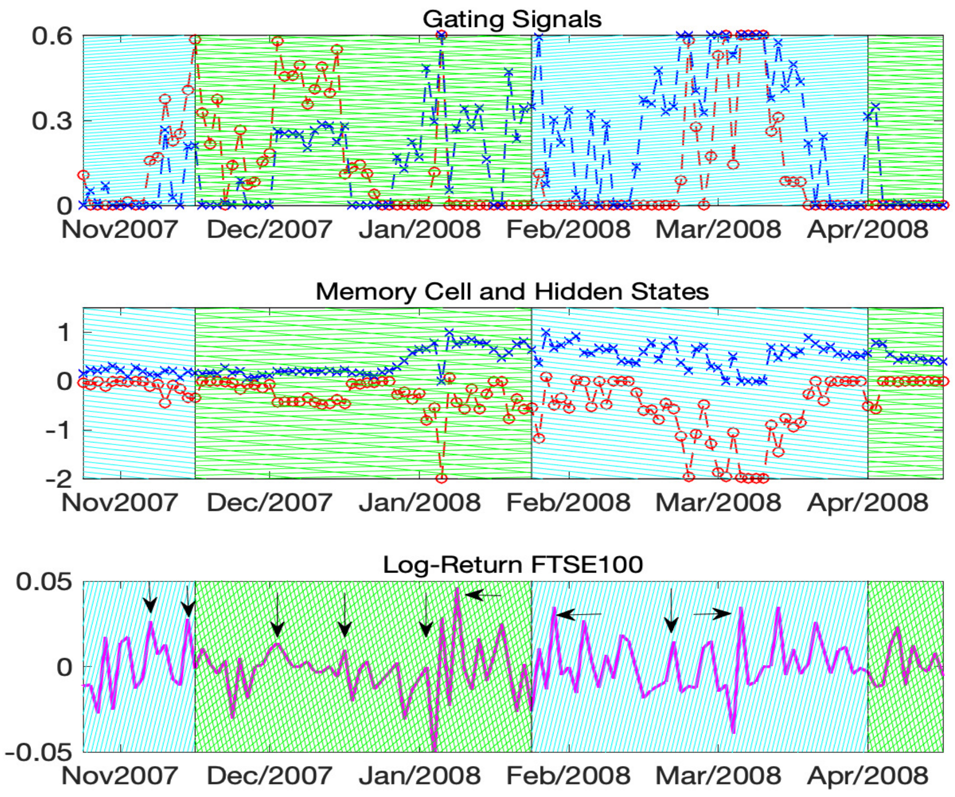
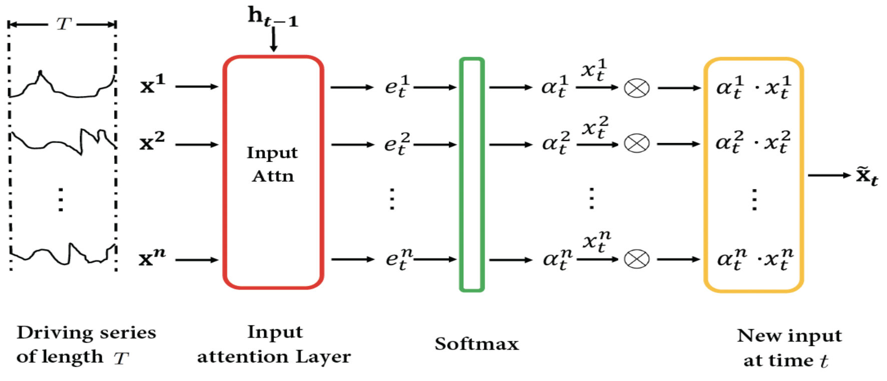


FIGURE 6.8: The gating signals of the LSTM estimating the FTSE-100 index from the financial crisis in 2007-2009. The gating signals are slightly detecting the change on interest rate in UK which are represented in different colours in the background. Also, the forget gate signal (colour blue) is storing in memory when the log return values are keeping steady, meanwhile it starts to spike when big changes in log-return, which can indicates an increase on volatility at that moment that lead to a bear market regime. The input gates signal (colour red) is accepting or rejecting the new states values.

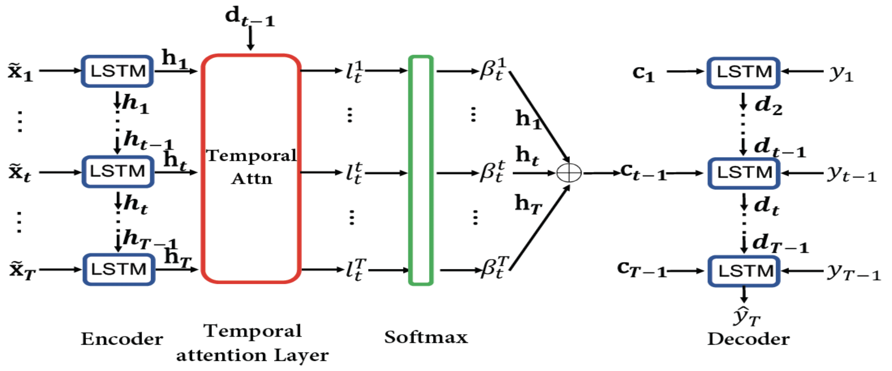
the dual attention recurrent neural network. First stage generates the input attention weights $\boldsymbol{\alpha}$ for multiple time series that are the inputs $\boldsymbol{x}$ and returns a new computed $\hat{\boldsymbol{x}}$ that was multiplied with the activations weight $\boldsymbol{\alpha}$. Second stage, the temporal attention, that represents the weight of the input information of the encoder hidden states across all the time steps. Finally the output $\boldsymbol{c}$ is the input to the decoder LSTM and y is the output from the decoder. (This figure is taken from Qin et al. (2017)).

The first part is an encoder, formed of a RNN, that learns to map from the input sequence $\boldsymbol{x}$ to a feature representation

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) \tag{6.3}$$



(a) Input Attention



(b) Temporal Attention

FIGURE 6.9: The two stages of the dual attention recurrent neural network. (a) generates the input attention weights $\boldsymbol{\alpha}$ for multiple time series that are the inputs $\boldsymbol{x}$ and returns a new computed $\hat{\boldsymbol{x}}$ that was multiplied with the activation weights $\boldsymbol{\alpha}$. (b) is the temporal attention that represents the weight of the input information of the encoder hidden states across all the time steps. Finally the output $\boldsymbol{c}$ is the input to the decoder LSTM and y is the output from the decoder. (This figure is taken from Qin et al. (2017))

where f is the LSTM. Then the input attention is built by referring the previous value of the hidden state $\boldsymbol{h}_{t-1}$ and the cell state $\boldsymbol{s}$.

$$e_t^k = \mathbf{v}_e^T \tanh(\mathbf{W}_e[\boldsymbol{h}_{t-1}; \boldsymbol{s}_{t-1}] + \mathbf{U}_e \boldsymbol{x}^k) \tag{6.4}$$

where $\mathbf{v}, \mathbf{W}$ and $\mathbf{U}$ are the parameters that are going to be learnt. And to get the values between 0 to 1, we apply a softmax function to e_t^k. Thus we address to output $\hat{\boldsymbol{x}} = (\alpha_t^1 x_t^1, ..., \alpha_t^n x_t^n)^T$, where $\alpha_t^k = \frac{\exp(e_t^k)}{\sum_{i=1}^n \exp(e_t^i)}$ that will update the hidden state in Equation 6.3.

Similarly, the decoder with temporal attention is based on LSTM, which decodes the new input information.

$$l_t^i = \mathbf{v}_d^T \tanh(\mathbf{W}_d[\boldsymbol{d}_{t-1}; \boldsymbol{s}_{t-1}] + \mathbf{U}_d \boldsymbol{h}_i) \tag{6.5}$$

Here also $\mathbf{v}, \mathbf{W}$ and $\mathbf{U}$ are the parameters that are going to be learnt. The β represents the relevance of the hidden encoder state $\boldsymbol{h}_i$. As it is mapped to the temporal component form the inputs, we can compute the context vector $\boldsymbol{c} = \sum_{j=1}^{T} \beta_t^i \boldsymbol{h}_i$, where $\beta_t^i = \frac{\exp(l_t^i)}{\sum_{j=1}^{T} \exp(l_t^j)}$. Thus we can combine the weighted context vectors with the targets as Equation 6.6

$$\hat{y}_{t-1} = \hat{\mathbf{w}}^T [y_{t-1}; \boldsymbol{c}_{t-1}]. \tag{6.6}$$

Then $\hat{y}_{t-1}$ is applied to update the decoder states:

$$\boldsymbol{d}_t = f_2(\boldsymbol{d}_{t-1}, \hat{y}_{t-1}) \tag{6.7}$$

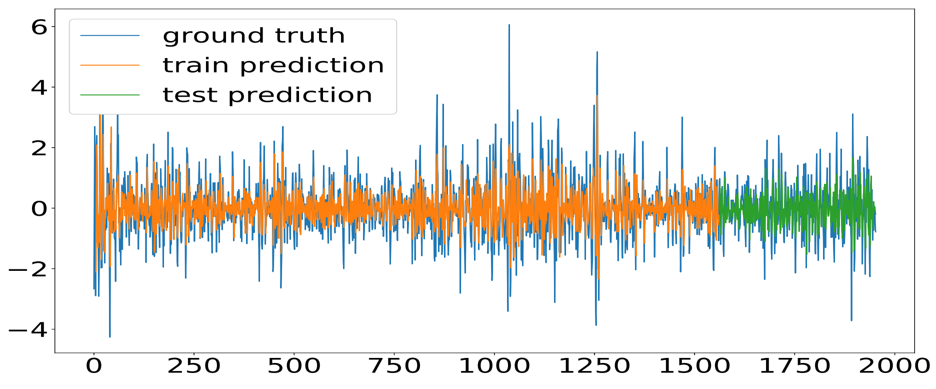
Here f_2 is the LSTM unit to then update $\boldsymbol{d}_t$ as:

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_f [\boldsymbol{d}_{t-1;\hat{y}_{t-1}}]) \tag{6.8}$$

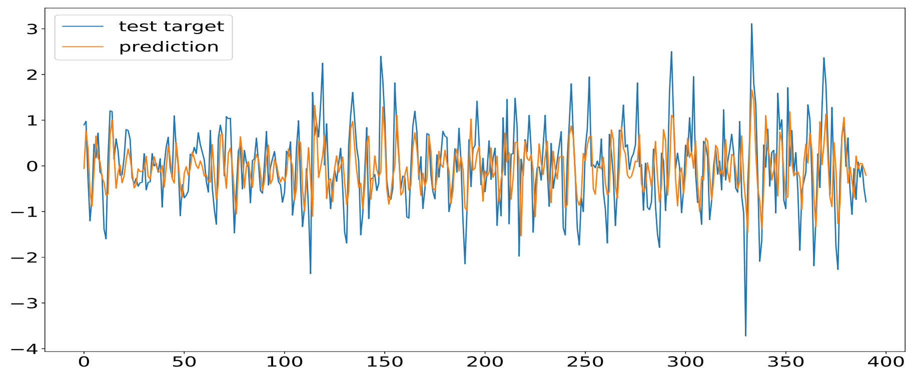
6.3.2.1 Experiment of Dual Attention on UK market data

We choose the dataset from Section 6.2.3 but adding 3 more financial indices to the input sequence. We aim to test the dual attention mechanism on predicting the residual r of the FTSE100 from Section 5.5. Our input is composed of 19 financial time series from the UK market with 5 time steps. We show in Figure 6.10 the prediction of the residual by the Dual Attention mechanism. As it is explained in the work of Qin et al. (2017), Dual Attention outperforms many baselines for this work. For this work we compared with the ARIMA model and Dual Attention got better mean square error (0.62 vs 0.76).

Thus, from this result we can consider the importance of working with multivariate time series and how machine learning is outperforming by the inclusion of macro-economic variables. It also contributes to answer our first research question.



(a) Training Set



(b) Testing set

FIGURE 6.10: Prediction of the residual signal. Here we show that dual attention mechanism is generalizing the prediction closely to the ground truth values

6.4 Summary

In this chapter, we introduced a novel method of applying probabilistic NMF on financial data using a Variational Autoencoder with exogenous information. This model is an extension from PAE-NMF in which the Gaussian prior of standard VAE is replaced with the Weibull distribution to enforce non-negative values. The advantage over a VAE is that we should see improved interpretability due to the parts based nature of the representation formed and the inclusion of exogenous financial information. We illustrated that our model can generate financial data by considering macroeconomic variables and the influence of them in the UK industrial sectors. Thus, it can help to prove the contribution of exogenous information to enhance the model, which is the aim of our research. Also we show an example of nine latent representation in which our model detected a sharp discontinuity around the Brexit event, which can help to extract useful information in financial analysis. Further, we explore the use of Long Short Term

Memory recurrent neural networks for modelling the temporal dynamics of financial time series. A special feature of LSTM is that architecture has special circuitry to enable forgetting the influence of distant past. This is particularly important in financial time series which are often characterized by non-stationary behaviour. The LSTM often responds to changes in the underlying interest rate which are important determinants of the financial market.

Chapter 7

Conclusion and future work

7.1 Conclusion

This dissertation considered several problems in empirical finance to show how additional variables may help in improving predictions, and some insights could be drawn from them. We addressed these problems by pricing derivatives, learning a subspace using matrix factorization and exogenous variables, time series prediction followed by explaining the residual using exogenous data, and designing a Variational Autoencoder, again with side information. We also include a study of a dynamic neural network and attempt to explain how its gating signal behaves.

In Chapter 3, we reported how additional information might help improve a non-parametric mapping between option prices and underlying price of an asset (on which the option contract is based). This work published by Hutchinson et al. (1994) showed a good approximation could be derived using a radial basis function model whose inputs were the asset price and the time to maturity only. We showed in this work that additional data such as asset volatility, traded volume and interest rate can be included to improve the prediction accuracy. While Hutchinson et al.'s work was on a single underlying asset (S&P 500 index options) and simulated data, we showed here that their results hold for a wide range of data (other indices and equity options). An interesting outcome of this study is the relationship between the volume of an asset traded and its volatility. There is a debate in empirical finance literature if these are correlated. In the study here, we show that while the correlation between these is not observable directly, their influence in pricing derivatives is significant.

In Chapter 4, we developed a matrix factorization method. We extend the method of non-negative matrix factorization (NMF) to include known eXogenous information to specify the XNMF model. We derive a multiplicative update algorithm for this model and show that the resulting factors are tightly clustered as a result of using external information. This tight clustering is maintained over time. We also showed that XNMF achieved lower reconstruction error than NMF as a result of the inclusion of exogenous information. The factors formed deviated from the sector structure.

In Chapter 5, we studied the effect of external information on explaining the residual in time series analysis. We model time series data using a simple autoregressive model, estimated using a Kalman filter, and apply Lasso regression using external macroeconomic variables to explain the residual signal. This identifies a small number of variables, some of which are intuitive, that are relevant in explaining the residual. By comparing the effect of such modelling on stock index time series and Bitcoin, we demonstrate the fundamental difference between them, one based on underlying economic activity and the other simply of trading behaviour. Thus, we showed that the inclusion of exogenous macro-economic information helped to gain more interpretability for financial time series analysis. Then we compare those selected variables with the XNMF and we found that some of them overlap.

In Chapter 6, we introduced a probabilistic XNMF (PAE-XNMF). We extended a Variational Autoencoder (VAE) with inductive exogenous variables, replacing the Gaussian prior with the Weibull distribution to enforce positive values. Many studies are modifying the structure of the VAEs to get disentanglement. We address it by inducting the exogenous financial information. We showed that PAE-XNMF can detect sharp changes from the latent representation generated and a lower reconstruction error than PAE-NMF. After this, we explored the use of recurrent neural networks, specifically LSTM, for modelling the temporal dynamics of financial time series. We show how it often captures dynamics of the time series by analysing the states of the gating signals. The LSTM often responds to changes in the underlying interest rate which can be an indicator for financial use. Finally, to increase the interpretability of the LSTM on multivariate time series, we applied the dual attention mechanism.

This our work showed that exogenous information improved performance and in some cases helped to extract useful information when they were added into a model fitting procedure. Thus the interpretability of machine learning models in their role on financial time series analysis can have a better approach. We answered the research question by the analysis of different financial problems with a range of different machine learning models. We expect that this work carried out can be followed for new researchers to increase the understanding of the role of machine learning in finance.

7.2 Future Work

A potential future path could be to explore the use of more machine learning methods with others financial instruments such as bonds. This idea is jointly carrying out with a colleague that is also interested in the use of machine learning for financial data. We are aiming to compare different models for these matters.

A particular interesting future work could be the analysis of PAE-XNMF (Chapter 6) in its role for anomaly detection in financial data. Anomalies are patterns with a clear deviation from the normal data. They can be detected by finding the probability distribution from the data. Then, comparing that probability distribution with a specified threshold when new observation arrives. If the probability value is lower than the threshold, we could consider as an anomaly. Normal observations tend to have a large probability, higher than the threshold. Thus, PAE-XNMF may be used to get a new anomaly threshold on financial data. PAE-XNMF can generate new data points, and whenever new observations veer too far away from the bounds from the generated patterns, we can consider that something unusual is happening, which could be an anomaly. PAE-XNMF can be useful specially that several works with VAE in anomaly detection have problems in separating noise and real anomalies (Suh et al., 2016; Wang et al., 2018). Thus, we believe PAE-XNMF could overcome this issue thanks to the inclusion of exogenous information.

Additionally, we recommend an extensive and very challenging work that address the inclusion of exogenous macro-economic information in Reinforcement Learning (RL). In RL the learning is based on trajectory samples from the Markov decision process, state-space and transition probabilities that are not directly expressed. Reinforcement learning is increasingly used in finance, and some of the areas applied include portfolio optimization, optimal trade execution and market-making. In portfolio optimization, the aim of reinforcement learning is to create an optimal portfolio when the model receives specific factors that should be maximized (e.g. expected return) or minimized (e.g. financial risk). In optimal trade execution the aim of RL is to create strategies for trading (buy or sell) a financial instrument (e.g. share of stock) that maximize (e.g. revenues) or minimize (e.g. capital used) in a fixed time period. In market-making, RL aims to create a price setting strategy that maximizes the profit from buying or selling stock; and minimizes the inventory risk. For example, Grassl (2010) has applied reinforcement learning techniques for pricing options to write the derivatives pricing as a Markov decision process. The pricing model can learn directly from data, and it does not need to assume that a specific price process is followed by the underlying's price. Thus, we may extract a fair price, estimated from random samples generated.

In order to achieve better results, we recommend variations to the base of the standard reinforcement learning approach. For example, by adapting regime changes via a new researched method that keep updating the trading strategy continually through new observations that include exogenous macro-economic information. We think that linking the selection of relevant factors using our XNMF method may improve the performance of RL.

Besides, we propose a variation of reinforcement learning with a recurrent convolutional neural network agent that evaluate an asset's past prices, similar to Zhang et al. (2017), for predicting time series. In that work, they formulated a RCNN agent trained with RL for tracking videos that learn to predict bounding box locations. There is a potential in the use of recurrent convolutional reinforcement learning that accounts for current and previous inputs in order to get temporal correlations of the past share prices. Dynamic decision making compared with conventional learning task, is more challenging due to the lack of supervised information from human experts. Thus, it requires the agent to explore an unknown environment all by itself and to make correct decisions in an online manner simultaneously. Our agent for sequential pricing would take a sequence of the underlying asset value of some macro-economic variables as input, and the price of an option as the target input. Thus, at each point in time, the agent extracts representative features from macro-economic variables, underlying assets and integrates information over time.

In the work carried out until now we have modelled continuous variables. However, a much finer time scale trading happen at discrete points in time. Multiple bids and asks are available between ticks at which some equilibrium is reached, a price agreed and a transaction occurs at that agreed price. When the market is volatile and liquid, the rate at which trading happens will be high and at low volatility periods inter-trade intervals may be long. The underlying market volatility, macroeconomic variables, and sentiments may influence different assets differently, yet they may be related in behaviour. An appropriate modelling paradigm suitable to capture the above scenario is state space models with point process observations. Such a model consists of a slowly changing dynamical system which is continuous in time, driving point process which are discrete events in time. Smith and Brown (2003) developed a maximum likelihood approach to estimate parameters. It avoids the somewhat artificial change to inter-event times and handles the discrete events directly. This model assumes a first-order autoregressive process driven by an exogenous stimulus as state dynamic and an approximate Bernoulli process with a parameterised intensity function as its observation model. Therefore, we recommend developing state spaces point process models for financial data. We postulate that slowly varying underlying information, be it market sentiments or other macroeconomic variables will drive several parallel points process during times of high volatility trading will be frequent. We expect that an univariate state space model may be inadequate to capture underlying state dynamics.

As we showed that exogenous macro-economic variables improved the performance of machine learning methods and the extraction of useful information. We consider that our work can help to clear many doubts of the influence of machine learning in financial data and its enhancement by the inclusion of macro-variables.

Appendix A

Libraries and Tools

A.1 Programming Languages For NMF Implementations

Many publications focused on improving, adapting, extending and re-designing algorithms for computing NMF. Most of them are written in Matlab.

A.1.1 Matlab

MathWorks developed Matrix Laboratory known as Matlab, which is a proprietary programming language with multi-paradigm numerical computing environment. It eases the manipulation of matrices, plot the data and incorporate programs of others programming languages. Matlab has a simple implementation of divergence-reducing NMF, which is the built-in function `nnmf`. It uses the multiplicative update rules (MU) Lee and Seung (1999) for a beta-divergence cost (including Kullback Leibler divergence Kullback and Leibler (1951) and Froebenius distance) and alternating least squares (ALS) Paatero and Tapper (1994). This function receives $n \times m$ matrix $\mathbf{V}$ and factorizes into 2 matrices $\mathbf{W}$ and $\mathbf{H}$. This algorithm converges to r lower rank solution. $[\mathbf{W}, \mathbf{H}] = \texttt{nnmf}(\mathbf{V}, k)$ is the in-built function provides in Matlab. In Cemgil (2009) is shared a variational Bayes implementation on Matlab, It works for Kullback-Leibler divergence. Furthermore in Cichocki et al. (2006) is introduced a toolboxes to implement NMF with with image data and signal processing. It provides Multiplicative update, projected gradient, conjugate gradient, exponentiated gradient, and quasi-Newton implementations. Additionally, Hoyer (2004) contributes with a Matlab package which can perform a projected gradient algorithm with sparseness. There are other packages that provides Matlab codes of fast Newton-type NMF methods Kim et al. (2007), that gives two projected gradient methods for NMF Lin (2007), and finally code of Bayesian NMF and sparse NMF with least squares in Schmidt and Laurberg (2008).

A.1.2 R

A free alternative similar to Matlab is R programming language. It provides environment for statistical computing and graphics, which has increased in popularity in recent year. Implementation of a framework for NMF algorithms in R package is found in Gaujoux and Seoighe (2010). Which contains standard algorithms and it ease the implementation of new methods into a common framework. The last updated version is from March Gaujoux and Seoighe (2018). R provides the function `nmf` to run NMF algorithms, which requires 4 parameters: `nmf(x,rank,method,seed,...)`. Where x correspond to the target matrix, *rank* is the factorization rank, *method* the algorithm to be used for the factorization, and *seed* is the method for computing the starting point.

A.1.3 Python

NMF can be implemented on an interpreted high-level programming language, which is Python. It provides an automatic memory management with multiple programming paradigms. It has many comprehensive standard libraries. The code in Python to compute NMF can be found in the library provides in Janecek et al. (2012), and also in Lin (2007). On the other hand, in Battenberg and Wessel (2009) was analysed the performance of parallel NMF in Python. Also in this programming language can used OpenMP for shared-memory multi-core systems and CUDA for many-core graphics processors. A highly used library in python is Scikit-Learn Pedregosa et al. (2011), also can compute NMF on Python. This library use *sklearn.decomposition.NMF* to find the two non-negative matrices ($\mathbf{W}$,$\mathbf{H}$) which approximates the non-negative matrix $\mathbf{V}$.

A.1.4 C/C++

C is an imperative computer programming language where NMF can be implemented. This language supports structured programming, recursion and lexical variable scope. For implementation of NMF, C provides a high performance library called *libNMF*, which is computationally efficient Janecek et al. (2012). It consists of external routines from Basic Linear Algebra Subprograms (BLAS), Linear Algebra package (LAPack) and ARPack. It can perform central vector and matrix operations by efficient building block provides. Furthermore, Another C++ library that provides NMF algorithms and can exploit the perfomance gains is introduced in Sra and Dhillon (2006). The algorithms from this library comprise the basic MU algorithm, ALS and MY in a hybrid form, variants of ALS algorithm, and 2 algorithms that consists on Bregman divergence. Also in Greene et al. (2008) is the implementation of more NMF algorithms that can be applied for hierarchical clustering. Lastly, in Wang et al. (2006) is shared a code in C++ to compute least squares non-negative matrix factorization (LS-NMF).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. ISSN 0018-9286.

P Geoffrey Allen and Bernard J Morzuch. Twenty-five years of progress, problems, and conflicting evidence in econometric forecasting. What about the next 25 years? *International Journal of Forecasting*, 22(3):475–492, 2006.

Pongpitch Amatyakul. The relationship between trading volume and jump processes in financial markets. *Duke Journal of Economics*, XXII, 2010.

Torben G Andersen. Return volatility and trading volume: An information flow interpretation of stochastic volatility. *The Journal of Finance*, 51(1):169–204, 1996.

M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

Ata Assaf. The stochastic volatility model, Regime Switching and Value-at-Risk (VaR) in international equity markets. *Journal of Mathematical Finance*, 7(02):492, 2017.

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

AS Babu and SK Reddy. Exchange rate forecasting using ARIMA, neural network and fuzzy neuron. *Journal of Stock and Forex Trading*, 4(3):–, 2015. ISSN 2168-9458.

Flavio Barboza, Herbert Kimura, and Edward Altman. Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, 83:405–417, 2017.

Eric Battenberg and David Wessel. Accelerating non-negative matrix factorization for audio source separation on multi-core and many-core architectures. In *International Society for Music Information Retrieval (ISMIR)*, pages 501–506, 2009.

Marco Bazzi, Francisco Blasques, Siem Jan Koopman, and Andre Lucas. Time-varying transition probabilities for Markov regime switching models. *Journal of Time Series Analysis*, 38(3):458–478, 2017.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*, pages 1–4. Springer, 2009.

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

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: Forecasting and Control*. Holden-Day, New San Francisco, CA, 1970.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Paolo Brandimarte. *Numerical Methods in Finance and Economics: a MATLAB-based introduction*. John Wiley and Sons, 2013.

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

David C Broadstock and George Filis. Oil price shocks and stock market returns: New evidence from the United States and China. *Journal of International Financial Markets, Institutions and Money*, 33:417–433, 2014.

Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101(12):4164–4169, 2004.

Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized non-negative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

Fan Cai, Nhien-An Le-Khac, and Tahar Kechadi. Clustering approaches for financial data analysis: A survey. *arXiv preprint arXiv:1609.08520*, 2016.

John Y Campbell and Ludger Hentschel. No news is good news: An asymmetric model of changing volatility in stock returns. *Journal of Financial Economics*, 31(3):281–318, 1992.

Mark Cecchini, Haldun Aytug, Gary J Koehler, and Praveen Pathak. Detecting management fraud in public companies. *Management Science*, 56(7):1146–1160, 2010.

Sibel Celik. New evidence on the relation between trading volume and volatility. *Business and Economic Research*, 3(1):176–186, 2013.

Ali Taylan Cemgil. Bayesian inference for non-negative matrix factorization. *Computational Intelligence and Neuroscience*, 2009:17, 2009.

Wesley S Chan. Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260, 2003.

Chia-Lin Chang, Michael McAleer, and Roengchai Tansuchat. Conditional correlations and volatility spillovers between crude oil and stock index returns. *The North American Journal of Economics and Finance*, 25:116–138, 2013.

Ioannis Chatziantoniou, George Filis, and Christos Floros. Asset prices regime-switching and the role of inflation targeting monetary policy. *Global Finance Journal*, 32:97–112, 2017.

Fei Chen and Charles Sutcliffe. Pricing and hedging short sterling options using neural networks. *Intelligent Systems in Accounting, Finance and Management*, 19(2):128–149, 2012.

Hua Chen, Jinlin Ma, Jiaying Liu, and Jingnan Wang. Non-negative matrix factorization via projected gradient method for credit risk analysis. In *6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)*, volume 2, pages 119–122. IEEE, 2013.

Ning Chen and Armando Vieira. Bankruptcy prediction based on Independent Component Analysis. In *1st International Conference on Agents and Artificial Intelligence (ICAART)*, pages 150–155, 2009.

Shanxiong Chen, Weiguo Zhang, and Maoling Peng. Anomaly detection based on non-negative matrix factorization in stock market. *Rev. Téc. Ing. Univ. Zulia*, 40(1):9–18, 2016.

Tong Chen, Hongzhi Yin, Hongxu Chen, Lin Wu, Hao Wang, Xiaofang Zhou, and Xue Li. Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 49–58. IEEE, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

ByoungSeon Choi. *ARMA model identification.* Springer Science and Business Media, 2012.

Ki-Hong Choi, Zhu-Hua Jiang, Sang Hoon Kang, and Seong-Min Yoon. Relationship between trading volume and asymmetric volatility in the Korean stock market. *Modern Economy*, 3(05):584, 2012.

François Chollet. Keras. `https://github.com/fchollet/keras`, 2015a.

François Chollet. keras. `https://github.com/fchollet/keras`, 2015b.

Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. Csiszars divergences for non-negative matrix factorization: Family of new algorithms. In *International Conference on Independent Component Analysis and Signal Separation*, pages 32–39. Springer, 2006.

Peter K Clark. A subordinated stochastic process model with finite variance for speculative prices. *Econometrica: Journal of the Econometric Society*, pages 135–155, 1973.

Mauro Coli, Ricardo Di Nisio, and Luigi Ippoliti. Exploratory analysis of financial time series using Independent Component Analysis. In *27th International Conference on Information Technology Interfaces*, pages 169–174. IEEE, 2005.

Thomas Cook and Aaron Smalter Hall. Macroeconomic indicator forecasting with deep neural networks. *Federal Reserve Bank of Kansas City*, (17-11), 2017.

John C Cox, Stephen A Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

Ali F Darrat, Maosen Zhong, and Louis TW Cheng. Intraday volume and volatility relations with and without public news. *Journal of Banking and Finance*, 31(9):2711–2729, 2007.

Caleb De Bernardis, Fernando Vicente-Guijalba, Tomas Martinez-Marin, and Juan Lopez-Sanchez. Particle filter approach for real-time estimation of crop phenological states using time series of NDVI images. *Remote Sensing*, 8(7):610, 2016.

Ruairí de Fréin, Konstantinos Drakakis, and Scott Rickard. Portfolio diversification using subspace factorizations. In *42nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1075–1080. IEEE, 2008a.

Ruairí de Fréin, Konstantinos Drakakis, Scott Rickard, and Andrzej Cichocki. Analysis of financial data using non-negative matrix factorization. In *International Mathematical Forum*, volume 3, pages 1853–1870. Journals of Hikari Ltd, 2008b.

Stavros Degiannakis, George Filis, and Christos Floros. Oil and stock returns: Evidence from European industrial sector indices in a time-varying environment. *Journal of International Financial Markets, Institutions and Money*, 26:175–191, 2013.

Stavros Degiannakis, George Filis, and Renatas Kizys. The effects of oil price shocks on stock market volatility: Evidence from European data. *The Energy Journal*, pages 35–56, 2014.

Karthik Devarajan. Nonnegative matrix factorization: an analytical and interpretive tool in computational biology. *PLoS Computational Biology*, 4(7), 2008.

Swati Dhingra, Gianmarco Ottaviano, Thomas Sampson, and John Van Reenen. The impact of Brexit on foreign investment in the UK. *London School of Economics and Political Science, Centre For Economic Performance CEP*, 2, 2016a.

Swati Dhingra, Gianmarco IP Ottaviano, Thomas Sampson, and John Van Reenen. The consequences of Brexit for UK trade and living standards. *London School of Economics and Political Science, Centre For Economic Performance CEP*, 2016b.

Emilien Dupont. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*, pages 710–720, 2018.

Monique Ebell and James Warren. The long-term economic impact of leaving the EU. *National Institute Economic Review*, 236(1):121–138, 2016.

Julian Faraway and Chris Chatfield. Time series forecasting with neural networks: A comparative study using the air line data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(2):231–250, 1998.

Cédric Févotte and A Taylan Cemgil. Nonnegative matrix factorizations as probabilistic inference in composite models. In *17th European Signal Processing Conference*, pages 1913–1917. IEEE, 2009.

David J Field. What is the goal of sensory coding? *Neural Computation*, 6(4):559–601, 1994.

George Filis, Stavros Degiannakis, and Christos Floros. Dynamic correlation between stock market and oil prices: The case of oil-importing and oil-exporting countries. *International Review of Financial Analysis*, 20(3):152–164, 2011.

Hong-Tao Gao, Tong-Hua Li, Kai Chen, Wei-Guang Li, and Xian Bi. Overlapping spectra resolution using non-negative matrix factorization. *Talanta*, 66(1):65–73, 2005.

Renaud Gaujoux and Cathal Seoighe. A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367, 2010. ISSN 1471-2105.

Renaud Gaujoux and Cathal Seoighe. *The package NMF: Manual pages*, 2018. R package version 0.21.0.

Gyozo Gidofalvi and Charles Elkan. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*, 2001.

Clyde Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Rule inference for financial prediction using recurrent neural networks. In *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering (CIFEr)*, pages 253–259. IEEE, 1997.

Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12(257):257–291, 2014.

Ahmad Golbabai, Davood Ahmadian, and Mariyan Milev. Radial basis functions with application to finance: American put option under jump diffusion. *Mathematical and Computer Modelling*, 55(3-4):1354–1362, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Stéphane Goutte, Amine Ismail, and Huyên Pham. Regime-switching stochastic volatility model: Estimation and calibration to vix options. *Applied Mathematical Finance*, pages 1–38, 2017.

Thomas Grassl. A reinforcement learning approach for pricing derivatives. *The Stanford Encyclopedia of Philosophy*, 2010.

Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

Derek Greene, Gerard Cagney, Nevan Krogan, and Pádraig Cunningham. Ensemble non-negative matrix factorization methods for clustering protein–protein interactions. *Bioinformatics*, 24(15):1722–1728, 2008.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *ArXiv Preprint ArXiv:1502.04623*, 2015.

Chonghui Guo, Hongfeng Jia, and Na Zhang. Time series clustering based on ICA for stock data analysis. In *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–4. IEEE, 2008.

James D Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.

Wolfgang K Härdle, Yuh-Jye Lee, Dorothea Schäfer, and Yi-Ren Yeh. The default risk of firms examined with smooth support vector machines. 2007.

Wolfgang Karl Härdle, Linda Hoffmann, and Rouslan Moro. Learning machines supporting bankruptcy prediction. In *Statistical Tools for Finance and Insurance*, pages 225–250. Springer, 2011.

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

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Sean B Holden and Mahesan Niranjan. On the practical applicability of VC dimension bounds. *Neural Computation*, 7(6):1265–1288, 1995.

Sean B Holden and Mahesan Niranjan. Average-case learning curves for radial basis function networks. *Neural Computation*, 9(2):441–460, 1997.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

Patrik O Hoyer. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. IEEE, 2002.

Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

Tsung-Jung Hsieh, Hsiao-Fen Hsiao, and Wei-Chang Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2510–2525, 2011.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised domain adaptation for robust speech recognition via Variational Autoencoder-based data augmentation. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23. IEEE, 2017.

Fengxia Hu and Rongming Wang. Optimal investment–consumption strategy with liability and regime switching model under value-at-risk constraint. *Applied Mathematics and Computation*, 313:103–118, 2017.

John C Hull. *Options, Futures, and Other Derivatives.* Prentice Hall, New York, 8th edition, 2011.

James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.

Eric Jacquier, Nicholas G Polson, and Peter E Rossi. Bayesian analysis of stochastic volatility models. *Journal of Business and Economic Statistics*, 20(1):69–87, 2002.

Prem C Jain and Gun-Ho Joh. The dependence between hourly prices and trading volume. *Journal of Financial and Quantitative Analysis*, 23(3):269–283, 1988.

Andreas Janecek, Stefan Schulze Grotthoff, and Wilfried N Gansterer. `libNMF`–a library for nonnegative matrix factorization. *Computing and Informatics*, 30(2):205–224, 2012.

Huisu Jang and Jaewook Lee. Generative Bayesian neural network model for risk-neutral pricing of American index options. *Quantitative Finance*, 19(4):587–603, 2019.

Visakan Kadirkamanathan and Mahesan Niranjan. A function estimation approach to sequential learning with neural networks. *Neural computation*, 5(6):954–975, 1993.

Visakan Kadirkamanathan, Mahesan Niranjan, and Frank Fallside. Sequential adaptation of radial basis function neural networks and its application to time-series prediction. In *Advances in Neural Information Processing Systems*, pages 721–727, 1991.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Bhargav Kanagal and Vikas Sindhwani. Rank selection in low-rank matrix approximations: A study of cross-validation for NMFs. In *Proceedings Conference on Advances in Neural Information Processing Systems*, volume 1, pages 10–15, 2010.

J.M. Karpoff. The relation between price changes and trading volume: A survey. *Journal of Financial and Quantitative Analysis*, 22:109–126, 1987.

Juha Karvonen. Non-negative matrix factorization with sparseness constraints for sea ice SAR feature extraction. In *7th European Conference on Synthetic Aperture Radar*, pages 1–4. VDE, 2008.

Yuta Kawachi, Yuma Koizumi, and Noboru Harada. Complementary set Variational Autoencoder for supervised anomaly detection. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2366–2370. IEEE, 2018.

Lutz Kilian and Cheolbeom Park. The impact of oil price shocks on the US stock market. *International Economic Review*, 50(4):1267–1287, 2009.

Chang-Jin Kim and Charles R Nelson. Has the US economy become more stable? a Bayesian approach based on a Markov-Switching model of the business cycle. *Review of Economics and Statistics*, 81(4):608–616, 1999.

Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. Fast Newton-type methods for the least squares nonnegative matrix approximation problem. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 343–354, 2007.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2013.

Ralf Korn and Elke Korn. *Option pricing and portfolio optimization: Modern methods of financial mathematics*, volume 31. American Mathematical Society, 2001.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Bi-Juan Lee, Chin Wei Yang, and Bwo-Nung Huang. Oil price movements and stock markets revisited: A case of sector stock price indexes in the G-7 countries. *Energy Economics*, 34(5):1284–1300, 2012.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

Yue Li and Khaldoun M Khashanah. The predictive power of volatility pattern recognition in stock market. In *Symposium Series on Computational Intelligence*, pages 742–748. IEEE, 2015.

Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

Robert Litterman and Jose Scheinkman. Common factors affecting bond returns. *Journal of Fixed Income*, 1(1):54–61, 1991.

Weixiang Liu, Nanning Zheng, and Qubo You. Nonnegative matrix factorization and its applications in pattern recognition. *Chinese Science Bulletin*, 51(1):7–18, 2006.

Liu, Tang. Non-negative matrix factorization for stock market pricing. In *2nd International Conference on Biomedical Engineering and Informatics. BMEI'09*, pages 1–5. IEEE, 2009.

Nicolas Mahler. Modeling the sp 500 index using the Kalman filter and the LagLasso. In *International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2009.

Harry Markowitz. Portfolio selection. The Journal of Finance, vol. 7, no. 1, 1952.

Ellen R McGrattan and Andrea Waddle. The impact of Brexit on foreign investment and production. Technical report, National Bureau of Economic Research, 2017.

Marcelo C Medeiros, Gabriel FR Vasconcelos, Álvaro Veiga, and Eduardo Zilberman. Forecasting inflation in a data-rich environment: the benefits of machine learning methods. *Journal of Business and Economic Statistics*, pages 1–22, 2019.

Felix Ming, Fai Wong, Zhenming Liu, and Mung Chiang. Stock market prediction from WSJ: Text mining via sparse matrix factorization. In *International Conference on Data Mining (ICDM)*, pages 430–439. IEEE, 2014.

Nasser Mohammadiha, W Bastiaan Kleijn, and Arne Leijon. Gamma hidden Markov model as a probabilistic nonnegative matrix factorization. In *21st European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2013.

Nasser Mohammadiha, Jalil Taghia, and Arne Leijon. Single channel speech enhancement using Bayesian NMF with recursive temporal updates of prior distributions. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4561–4564. IEEE, 2012.

Christophe Molina and Mahesan Niranjan. Pruning with replacement on limited resource allocating networks by F-Projections. *Neural Computation*, 8(4):855–868, 1996.

Luis Montesdeoca and Mahesan Niranjan. Extending the feature set of a data-driven artificial neural network model of pricing financial options. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6, Dec 2016.

Luis Montesdeoca and Mahesan Niranjan. On comparing the influences of exogenous information on Bitcoin prices and stock index values. In *1st International Conference on Mathematical Research for Blockchain Economy*. Springer, 2019.

Luis Montesdeoca, Steven Squires, and Mahesan Niranjan. Variational autoencoder for non-negative matrix factorization with exogenous inputs applied to financial data modelling. In *11th International Symposium on Image and Signal Processing and Analysis*. IEEE, 2019.

John Moody and Lizhong Wu. What is the “true price"? State space models for high frequency FX data. In *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering (CIFEr)*, pages 150–156, 1997.

Mahesan Niranjan. Sequential tracking in pricing financial options using model based and neural network approaches. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 960–966. MIT Press, 1997.

Mahesan Niranjan and Frank Fallside. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4(3):275–289, 1990.

Mohamed F Omran. Nonlinear dependence and conditional heteroscedasticity in stock returns: UK evidence. volume 4, pages 647–650. Taylor and Francis, 1997.

Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2): 111–126, 1994.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons. Text mining using non-negative matrix factorization. In *Proceedings of the SIAM International Conference on Data Mining*, pages 452–456. Society for Industrial and Applied Mathematics, 2004.

Andrea Pazienza, Sabrina Francesca Pellegrino, Stefano Ferilli, and Floriana Esposito. Clustering underlying stock trends via non-negative matrix factorization. In *Proceedings of the First Workshop on Mining Data for Financial Applications*, pages 5–16. Riva del Garda, 2016.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

Robert Peharz and Franz Pernkopf. Sparse nonnegative matrix factorization with $\ell0$-constraints. *Neurocomputing*, 80:38–46, 2012.

Marcelo Perlin. Ms_regress-the MATLAB package for Markov regime switching models. 2015.

Ross C Phillips and Denise Gorse. Mutual-excitation of cryptocurrency market returns and social media topics. In *Proceedings of the 4th International Conference on Frontiers of Educational Technologies*, pages 80–86. ACM, 2018.

Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229, 1987.

John Platt. A resource-allocating network for function interpolation. *Neural computation*, 3(2):213–225, 1991.

Kuntara Pukthuanthong and Richard Roll. Gold and the Dollar (and the Euro, Pound, and Yen). *Journal of Banking and Finance*, 35(8):2070–2083, 2011.

Gintaras V Puskorius and Lee A Feldkamp. Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2627–2633. AAAI Press, 2017.

Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241, 2015.

Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pages 1431–1439, 2014.

Bernardete Ribeiro and Ning Chen. Graph weighted subspace learning models in bankruptcy. In *The International Joint Conference on Neural Networks (IJCNN)*, pages 2055–2061. IEEE, 2011.

Bernardete Ribeiro, Catarina Silva, Armando Vieira, and João Neves. Extracting discriminative features using non-negative matrix factorization in financial distress data. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 537–547. Springer, 2009.

Tony Robinson and Frank Fallside. A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5(3):259–274, 1991.

Emad W Saad, Danil V Prokhorov, and Donald C Wunsch. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, 9(6):1456–1470, 1998.

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

Mikkel N Schmidt and Hans Laurberg. Nonnegative matrix factorization with Gaussian process priors. *Computational Intelligence and Neuroscience*, pages 3:1–3:10, 2008. ISSN 1687-5265.

Mikkel N Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 540–547. Springer, 2009.

Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 792–799. ACM, 2005.

Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180. IEEE, 2003.

Frank Smets and Rafael Wouters. Shocks and frictions in US business cycles: A Bayesian DSGE approach. *American Economic Review*, 97(3):586–606, 2007.

Anne C Smith and Emery N Brown. Estimating a state-space model from point process observations. *Neural Computation*, 15(5):965–991, 2003.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

Steven Squires, Adam Prügel Bennett, and Mahesan Niranjan. A Variational Autoencoder for probabilistic non-negative matrix factorisation. *arXiv preprint arXiv:1906.05912*, 2019.

Steven Squires, Luis Montesdeoca, Adam Prügel-Bennett, and Mahesan Niranjan. Non-negative matrix factorization with exogenous inputs for modeling financial data. In *International Conference on Neural Information Processing*, pages 873–881. Springer, 2017a.

Steven Squires, Adam Prügel-Bennett, and Mahesan Niranjan. Rank selection in non-negative matrix factorization using Minimum Description Length. *Neural Computation*, 29(8):2164–2176, 2017b.

Suvrit Sra and Inderjit S Dhillon. Generalized nonnegative matrix approximations with Bregman divergences. In *Advances in Neural Information Processing Systems*, pages 283–290, 2006.

Suwon Suh, Daniel H Chae, Hyon-Goo Kang, and Seungjin Choi. Echo-state conditional variational autoencoder for anomaly detection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1015–1022. IEEE, 2016.

KS Sujit and B Rajesh Kumar. Study on dynamic relationship among gold price, oil price, exchange rate and stock market returns. *International Journal of Applied Business and Economic Research*, 9(2):145–165, 2011.

Andrew Sun, Michael Lachanski, and Frank J Fabozzi. Trade the tweet: Social media text mining and sparse matrix factorization for stock market prediction. *International Review of Financial Analysis*, 48:272–281, 2016.

Hao Sun, Zhiqian Chen, and James Chen. Credit risk analysis using sparse non-negative matrix factorizations. In *2nd International Conference on Information Science and Control Engineering (ICISCE)*, pages 181–184. IEEE, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. pages 3104–3112, 2014.

Hellinton H Takada and Julio M Stern. Non-negative matrix factorization and term structure of interest rates. In *American Institute of Physics (AIP) Conference Proceedings*, volume 1641, pages 369–377, 2015.

Hellinton H Takada and Julio M Stern. Intraday trading volume and non-negative matrix factorization. In *American Institute of Physics (AIP) Conference Proceedings*, volume 1757, pages 60006:1–60006:8, 2016.

Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modelling financial time-series with generative adversarial networks. *Physica A Statistical Mechanics and Its Applications*, 527:121261, 2019.

Akiko Takeda, Mahesan Niranjan, Jun-ya Gotoh, and Yoshinobu Kawahara. Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. *Computational Management Science*, 10(1):21–49, 2013.

Mehrdad Tamiz, R Hasham, DF Jones, B Hesni, and EK Fargher. A two staged goal programming model for portfolio selection. In *Multi-Objective Programming and Goal Programming*, pages 286–299. Springer, 1996.

Halit Alper Tayalı and Seda Tolun. Dimension reduction in mean-variance portfolio optimization. *Expert Systems with Applications*, 92:161–169, 2018.

Paul C Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168, 2007.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Elina Tjioe, Michael Berry, Ramin Homayouni, and Kevin Heinrich. Using a literature-based NMF model for discovering gene functional relationships. *BMC Bioinformatics*, 9(7):P1, 2008.

Greg Tkacz. Neural network forecasting of Canadian GDP growth. *International Journal of Forecasting*, 17(1):57–69, 2001.

Haakon Andreas Trønnes. Pricing options with an artificial neural network: A reinforcement learning approach. Master's thesis, NTNU, 2018.

Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Yap Bee Wah and Irma Rohaiza Ibrahim. Using data mining predictive models to classify credit card applicants. In *6th International Conference on Advanced Information Management and Service (IMS)*, pages 394–398. IEEE, 2010.

Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using Variational Autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

Guoli Wang, Andrew V Kossenkov, and Michael F Ochs. LS-NMF: A modified non-negative matrix factorization algorithm utilizing uncertainty estimates. *BMC Bioinformatics*, 7(1):175, 2006.

Jie Wang. Stock trend extraction via matrix factorization. In *International Conference on Advanced Data Mining and Applications*, pages 516–526. Springer, 2012.

Tian Wang, Meina Qiao, Zhiwei Lin, Ce Li, Hichem Snoussi, Zhe Liu, and Chang Choi. Generative neural networks for anomaly detection in crowded scenes. *IEEE Transactions on Information Forensics and Security*, 14(5):1390–1399, 2018.

Andreas S Weigend, Bernardo A Huberman, and David E Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1(03): 193–209, 1990.

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

Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3(Mar):1439–1461, 2003.

Lizhong Wu, Mahesan Niranjan, and Frank Fallside. Fully vector-quantized neural network-based code-excited nonlinear predictive speech coding. *IEEE Transactions on Speech and Audio Processing*, 2(4):482–489, 1994.

Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via Variational Autoencoder for seasonal KPIs in web applications. In *Proceedings of the World Wide Web Conference*, pages 187–196, 2018.

Rui Xu and Donald C Wunsch. Survey of clustering algorithms. 2005.

Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273. ACM, 2003.

Ching-Chiang Yeh, Der-Jang Chi, and Yi-Rong Lin. Going-concern prediction using hybrid Random Rorests and rough set approach. *Information Sciences*, 254:98–110, 2014.

David Yermack. Good timing: CEO stock option awards and company news announcements. *The Journal of Finance*, 52(2):449–476, 1997.

Huanhuan Yu, Rongda Chen, and Guoping Zhang. A SVM stock selection model within PCA. *Procedia Computer Science*, 31:406 – 412, 2014.

Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.

Mengzhe Zhang and Leunglung Chan. Saddlepoint approximations to option price in a regime-switching model. *Annals of Finance*, 12(1):55–69, 2016.

Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. Stock market prediction on high-frequency data using generative adversarial nets. *Mathematical Problems in Engineering*, 11, 2018.

Hui-Ming Zhu, Rong Li, and Sufang Li. Modelling dynamic dependence between crude oil prices and Asia-Pacific stock market returns. *International Review of Economics and Finance*, 29:208–223, 2014.