Imperial College London

Imperial College London

Department of Mathematics

Distributional Prediction of Foreign Exchange Rates with Mixture Density Network

Author: Si Cheng Fong (CID: 01347033)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2020-2021

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

First and foremost, I would like to express my gratitude to Albert Chan and Louise Caprani from MUFG Securities for providing valuable comments, and guidance for my projects. Special thanks to Rupa Dhillon from Bloomberg for answering my questions in data exploration within the Bloomberg Terminal.

I would also like to thank my supervisor Dr. Alex Tse for providing insightful sharing and guidance whenever needed.

Finally, I would like to express my heartfelt thanks to my family for their unconditional support and continuous encouragement.

Abstract

This thesis aims to predict the probability distribution of foreign exchange rates log returns. The probability density function of the foreign exchange rates log returns are modelled as a mixture of normal distribution and Mixture Density Network is adopted to estimate the parameters in the mixture density. The Mixture Density Network is trained with market data of various asset prices and we compare the marginal contributions of each assets to the neural network predictions by applying techniques in global interpretability of the neural network. The application we are going to mainly focus on is to develop foreign exchange trading strategies. The constructed strategies are then evaluated on market data and we found out the performances of the trading strategies are improved by considering the dynamics of foreign exchange rates in different trading sessions. In addition, we present simple regime classification method base on the mixture density parameters estimations by the Mixture Density Network, and illustrate the classifications on market data.

Contents

List of Figures

List of Tables

Section 1

Introduction

Foreign Exchange (FX) is trading currency of one country for another. FX is currently the largest traded asset class, and the FX market is the most liquid financial market in the world, with daily volume around 6.6 trillion US dollar in 2019 according to Bank for International Settlements 2019 [11]. It plays a crucial role in international trade and investment that allows financial investors to buy and sell securities in foreign currency, corporations import and export goods and services worldwide. Large number of market participants, monetary policies, trades, and news lead to a volatile and complicated FX market. Therefore, it is important to consider the FX rate prediction problem in multi discipline across multiple asset classes to examine how prices from other asset classes impact the FX rate.

The main aim of this paper is to predict the probability density function of the FX rate mid log return using neural network with various asset classes as input features. We propose normal mixture density to model the distribution of the FX rate log return, and the parameters of the mixture density distribution are estimated by Mixture Density Network, which is a particular class of neural network first proposed by Christopher M. Bishop [6] (1994). There are numerous existing literature that investigate various neural network architectures to perform point estimation in financial time series price predictions and classification problem in predicting the direction of price movement using technical analysis indicators, lagged features and macroeconomic factors as input features. To the best of our knowledge, the combination of using multiple asset classes to train the neural network to predict the probability density function of financial time series has not been studied before. The density can give rich information about the direction of price movement, and various statistical properties such as skewness and kurtosis, which allows practitioners to have applications in risk management, algorithm trading, and derivatives hedging.

Research on FX and equities predictions using machine learning and deep learning has become very popular given the recent advancement in computational power and the proven general result of universal approximation property in neural network by Leshno et al. (1993) [20, Theorem 1 and Proposition 1, page 863], which guarantees the existence of neural networks of some architectures that can approximate any measurable functions. Assaf (2019) [2] and Nielsen (2018) [24] explore wild range of machine learning methods and neural networks to test the robustness of FX mid-price forecast using tick by tick data in limit order book (LOB) of several exchanges with feature creations via introducing various indicators using bid and ask prices of different levels. Long Short-Term Memory is a type of recurrent neural networks (RNNs), which is designed to solve the problem of vanishing and exploding gradient in RNNs. The use of Long Short-Term Memory (LSTM) has become popular in financial time series prediction given its successful achievements. Tu (2020) [35] adopts four architectures of neural network involving RNNs and LSTM using different price levels in LOB to predict the mid-price and direction of movement in equities. Rather than using LOB data, technical indicators are often used as features. Achchab and Lanbouri (2020) [19] use technical indicators such as exponential moving average (EMA), moving average convergence divergence (MACD), Bollinger Up and Bollinger Down to predict the S&P500 in one, five and ten minutes ahead. Sheth and Teeple (2020) [31] establish linear relationship between the FX and equity around the London 4pm Fix. They apply principle component analysis (PCA) to subset factors, and generates buy and sell signals based on the prediction of FX returns around the London 4pm Fix. The vast majority of existing research on FX rates and stock prices predictions with deep learning use information of individual asset and news as their input features for training. Hu

et al.(2021) [17] conduct a thorough survey on existing literatures focus on FX and stock prices predictions using deep learning, we can conclude from the survey that there are significant number of studies in using technical indicators, macroeconomic statistics, news headlines and sentiments as variables to train their neural networks. There are very few literature discuss about the reasoning on their choices of features, and also multiple asset classes have not yet been considered as variables to train the neural networks to the best of our knowledge. We therefore stress the equal importance of analysing the rationale behind the choices of the selected variables for training and quantitative justifications in this thesis.

Non-normality properties in log returns have been empirically observed in various asset prices for a long time. Alexander and Narayanan (2001) [1] show JPY/USD and EUR/USD empirically exhibit fatter tails with excess kurtosis known as the leptokurtosis. They attempt to model the non-normality of log returns by assuming the conditional distribution of log return is normal while the unconditional log return is not normal, which the normal mixture distribution satisfies such properties, and the normal mixture density is applied in option pricing and calibration. Similar derivatives pricing application was done by Brigo (2002) [8]. He use the Fokker-Planck equation derived analytical form of stochastic differential equation (SDE) which has a solution with risk-neutral marginal density satisfying the mixture density and to model the volatility smile problem.

Christopher M. Bishop (1994) [6] connects mixture density and neural network. Bishop originally wanted to solve the inverse problem motivated by robotic inverse kinetics. He described the inverse problems as having multi-valued mapping from the input to output space. Neural networks in classical regression problem can only approximate the average of multiple outcomes, but the average is not necessarily able to represent a solution. In the financial market, rarely we can predict a single-valued future price accurately. Once we have predicted some future prices, natural questions including what are the skewness and kurtosis of the predicted returns or how likely an extreme price drops will take place. Thus, the underlying probability distribution can provide desirable quantitative references for practitioners. Taylor (2000) [34] has studied probabilistic prediction on financial returns. He applies the nonparametric approach quantile regression by estimating the quantiles with neural network . Unlike quantile regression, mixture density not only provides interpretable parametric form that can recover the quantiles, it is also a natural choice under the Hidden Markov Model, which can access rich information of the returns under different market *regimes*. Therefore, we decide to estimate the FX rates log return probability density function using Mixture Density Network.

The outline of the thesis is as follows: In Section 2, theoretical framework of neural networks, mixture density will be reviewed. We will review the paper by Bishop (1994) [6] on the construction of the Mixture Density Network, and also the theoretical background of global interpretability of neural network will be introduced. In Section 3, details on data collection, data description and the rationale behind the choices of various asset classes will be discussed. Section 4 will present the performances and empirical results of the Mixture Density Network. The marginal contributions of the assets we have used in the neural network will also be compared. Section 5 will mainly introduce the application of our Mixture Density Network in algorithm trading and a brief demonstration on regime classification based on the estimations by our Mixture Density Network. In Section 6, conclusion and future research will be discussed.

Section 2

Theoretical Framework

In this section, we will review the basics of deep learning and the most common class Feedforward neural network (FNN). The vast majority of the fundamentals of deep learning and neural networks are taken from the great lecture notes written by Mikko Pakkanen [23]. We then gradually introduce the recurrent neural network (RNN), and Long Short-Term Memory neural network (LSTM). Mixture Density Network (MDN) will then be introduced and the minimisation of squared error in standard neural network is equivalent to the conditional mean of the probability density estimated by the Mixture Density Network which was proved by Bishop (1994) [6], will be demonstrated. Similar results hold for neural network classification and further details can refer to the original paper by Bishop (1994) [6]. The main objective of deep learning is to find a (non-linear) function $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$ given input features $\boldsymbol{x} := (x_1, \ldots, x_I) \in \mathbb{R}^I$ in some optimal way. It is a very general and broad description about deep learning tasks, and the key is to define optimality and how we can reach the optimality.

2.1 Feedforward Neural Networks

The lecture note by Pakkanen [23, Definition2.1, page 17] gives a very clear outline on the mathematical definition and notations of feedforward neural network (FNN), and we will first present the highlights from [23] to the readers, and explain the key components in FNN.

Definition 2.1.1 (Feedforward Neural Network)**.** Let $I, O, r \in \mathbb{N}$. A function $\boldsymbol{f}$ is a feedfoward neural network (FNN) with $r-1$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i-th hidden layer for any $i = 1, \ldots, r-1$, and activation functions $\boldsymbol{\sigma}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}, i = 1, \ldots, r$, where $d_r := O$, if

$$\boldsymbol{f} = \boldsymbol{\sigma}_r \circ \boldsymbol{L}_r \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{L}_1 \tag{2.1.1}$$

where $\boldsymbol{L}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, for any $i = 1, \ldots, r$ is an affine function

$$\boldsymbol{L}_i(\boldsymbol{x}) := W^i \boldsymbol{x} + b^i, \quad \boldsymbol{x} \in \mathbb{R}^{d_{i-1}} \tag{2.1.2}$$

parameterised by weight matrix $W^i = [W^i_{j,k}] \in \mathbb{R}^{d_i \times d_{i-1}}$ and bias vector $\boldsymbol{b}^i = (b^i_1, \ldots, b^i_{d_i}) \in \mathbb{R}^{d_i}$, with $d_0 := I$.
We denote the class of such functions $\boldsymbol{f}$ by

$$\mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_r) \tag{2.1.3}$$

If $\boldsymbol{\sigma}_i(\boldsymbol{x}) = (g(x_1), \ldots, g(x_{d_i})), \boldsymbol{x} = (x_1, \ldots, x_{d_i}) \in \mathbb{R}^{d_i}$ for some $g : \mathbb{R} \to \mathbb{R}$, write g in place of $\boldsymbol{\sigma}_i$ and omit $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_r$ for brevity.

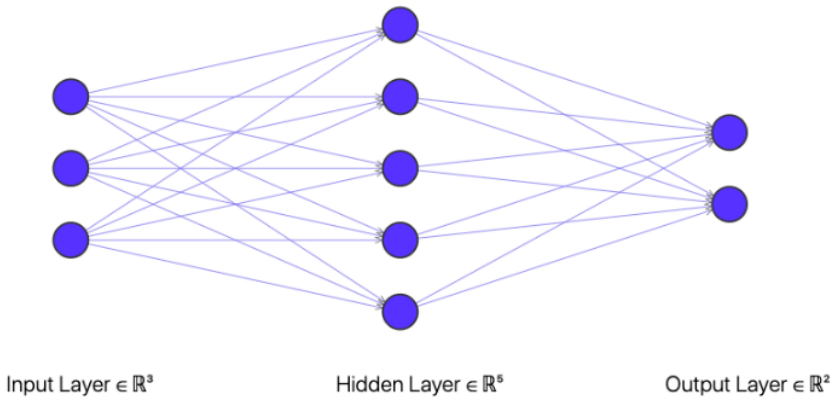

Figure 2.1: Feedforward Neural Network Example $\mathcal{N}_2(3, 5, 2)$

Key components of a typical feedforward neural network can be summarised as follows:

1. The function $\boldsymbol{f}$ is estimated by compositions of linear affine functions $\boldsymbol{L}$ and activation functions $\boldsymbol{\sigma}$
2. Linear affine functions $\boldsymbol{L}$ consist of the weight matrices W and the bias vectors $\boldsymbol{b}$
3. The activation functions $\boldsymbol{\sigma}$ are usually non-linear in order to capture non-linearities of the desired function $\boldsymbol{f}$. For example, the Rectified linear unit (ReLU) function is the most widely used, which is simply the function $g(x) = \max\{x, 0\}$. Other popular activation functions are Sigmoid, Hyperbolic tangent (tanh) etc.
4. The hyperparameters of the network are the integers $r, d_1, \ldots, d_{r-1}$. The parameters of the neural network are the weight matrices $W^1, \ldots, W^r$ and the bias vectors $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^r$. Together with the activation functions $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_r$ are called the architecture

Another key component that has not yet been introduced is the loss function that we are going to introduce in the next part.

2.1.1 Loss Function

Recall that the objective of general deep learning task is to find some non-linear function in some optimal way, and the loss function is crucial in characterising optimality.

Definition 2.1.2 (Characterisation of Optimality)**.** Suppose $\boldsymbol{f} \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O)$. Given input $\boldsymbol{x} \in \mathbb{R}^I$ and reference value $\boldsymbol{y} \in \mathbb{R}^O$, which are the realisations the joint vector $(\boldsymbol{X}, \boldsymbol{Y})$. Define loss function as $l : \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}$. The optimality of neural network is characterised by minimising the following quantity

$$\mathbb{E}[l(\boldsymbol{f}(\boldsymbol{X}), \boldsymbol{Y})]$$

By definition 2.1.2, the optimality is characterised by minimising the expected value of the loss function. However, the joint distribution is usually not observed, hence samples of realised loss $l(\boldsymbol{f}(\boldsymbol{x^i}), \boldsymbol{y^i})$ are observed and their empirical mean is computed as an estimator. Furthermore, instead of taking the mean of the whole sample space, subsets of sample called minibatches are usually taken. This is because taking the whole sample space is computationally expensive during the processes of optimisation for large sample size.

2.1.2 Choices of Loss Function

There are many variations of loss functions. We will state three choices of loss functions commonly used in regression, classification and fitting probability density function respectively.

Squared Error Loss

The square error loss is the most widely used loss function in regression. Suppose the predicted output in regression is given by $\hat{\boldsymbol{y}} \in \mathbb{R}^O$ and reference output $\boldsymbol{y} \in \mathbb{R}^O$, the square error is given by the Euclidean norm

$$\text{Squared Error Loss} = ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2$$

We often observe samples of realised $\hat{\boldsymbol{y}}^i$ for $i = 1, \ldots, N$, then the mean squared error loss is usually used for performance evaluations.

$$\text{Mean Squared Error (MSE)} = \frac{1}{N}\sum_{i=1}^{N} ||\hat{\boldsymbol{y}}^i - \boldsymbol{y}||_2^2$$

Categorical Cross Entropy

On the other hand if a multi-class classification problem of is considered, and the total number of class to be classified is $\mathcal{C}$ and suppose the the reference label $\boldsymbol{y} = (y_1, \ldots, y_{\mathcal{C}})$ is one-hot encoding, which means if $\boldsymbol{y}$ is in the i-th class $\mathcal{C}_i$, then $y_i = 1$ and $y_j = 0$ for any $j \neq i$. Denote the predicted class label $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_{\mathcal{C}}) \in [0,1]^{\mathcal{C}}$, the categorical cross-entropy is given by

$$\text{Categorical Cross Entropy} = -\sum_{i=1}^{\mathcal{C}} \mathbf{1}_{\{y_i=1\}} \log \hat{y}_i$$

Negative Log-Likelihood

Given we have some samples of data points $\{\boldsymbol{x}^i, \boldsymbol{y}^i\}_{i=1,\ldots,N}$, and denote their joint probability density function as $p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ for some parameters $\boldsymbol{\theta}$. The likelihood of the data set is given by

$$\mathcal{L} = \prod_{i=1}^{N} p(\boldsymbol{x}^i, \boldsymbol{y}^i; \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{y}^i | \boldsymbol{x}^i; \boldsymbol{\theta}) p(\boldsymbol{x}^i; \boldsymbol{\theta}) \tag{2.1.4}$$

Maximize the likelihood enables to determine the appropriate values of $\boldsymbol{\theta}$, and equivalently minimisation of the negative log-likelihood is often used in practice.

$$\text{Negative Log-Likelihood} = -\log \mathcal{L} \tag{2.1.5}$$

2.1.3 Training of Feedforward Neural Networks

The training of a feedforward neural network consists of three steps:

1. Weight Initialization
2. Forward Pass
3. Backward Pass

The last two steps are often combined known as backpropagation, which is described as a two-pass procedure in [14]. The three actions show how information flows within the neural network in order to approximate non-linear functions optimally.

Weight Initialization

The weight matrices $W^1, \ldots, W^r$ and the bias vectors $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^r$ require initialization. In practice, the biases are set to zero initially, and the weights are initialized by some efficient schemes such as He initialization [15] or Xavier initialization [36] to overcome the vanishing or exploding gradients problem.

Backpropagation

Backpropagation consists of forward and backward pass. In forward pass, predicted values will be determined based on the current weight matrices and biases. The realised loss is computed with the choice of loss function, then backward pass is performed to adjust the parameters to minimise the loss function. The two-pass procedure requires iterative optimization techniques namely gradient descent.

Gradient descent algorithm searches for the minimal solution of the loss function with iterative gradient updates. Denote the loss function $l(\boldsymbol{\theta})$ for $\boldsymbol{\theta} \in \mathbb{R}^n$ are the parameters of the neural network. Then the iterative update of the parameters at t-th step is given by the following recursive relationship

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \gamma \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}_{t-1})$$

where γ is the learning rate which has to be chosen appropriately during training. Too small or large values of γ will have slow convergence, converge to local minimum or divergence. Optimal γ is usually determined through trials, learning rate of 0.01 is usually an appropriate choice to begin with in practice.

A variation of the gradient descent algorithm, the Adam gradient descent algorithm is often used nowadays. It was first proposed by Kingma and Ba (2014) [18] and they described the algorithm as adaptive moment estimation. The Adam gradient descent outperforms other algorithms under sparse gradients and non-stationary setting. A brief description on the iterative updates of the algorithm [18, section 2, page 2] is given by:

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}_t), \quad \hat{m}_t = \frac{m_t}{1-\beta_1} \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}_t))^2, \quad \hat{v}_t = \frac{v_t}{1-\beta_2} \\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \frac{\gamma}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t
\end{aligned}$$

$\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ are usually selected, and the learning rate γ is chosen appropriately.

Therefore, in order to obtain an update of the parameters in neural network, the derivatives of the loss function with respect to the weights and biases, $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}_t)$, are required to compute. To compute the derivatives of the loss function $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}_t)$, chain rule in partial differentiation is applied. We first define some notations, then present the result of the backpropagation equation. Detail justifications can be found in the book Element of Statistical Learning [14, section 11.3, page 392-397] and the lecture notes written by Pakkanen [23, Proposition 3.13, page 39-40].
For $i = 1, \ldots, r$, and $j = 1, \ldots, d_i$

$$\begin{aligned}
\boldsymbol{x}^i &= \boldsymbol{\sigma}_i(W^i \boldsymbol{x}^{i-1} + \boldsymbol{b}^i) \\
\boldsymbol{z}^i &:= W^i \boldsymbol{x}^{i-1} + \boldsymbol{b}^i \\
\delta^i_j &:= \frac{\partial l(\boldsymbol{\theta})}{\partial z^i_j}
\end{aligned}$$

$\boldsymbol{x}^i$ is in fact the vectors computed in the i-th hidden layers of the neural network. $\boldsymbol{z}^i$ is simply the vectors by applying the i-th linear affine function on $\boldsymbol{x}^{i-1}$, and δ^i_j is the partial derivatives of the loss function l with respect to the components of $\boldsymbol{z}^i_j$. The main aim of backpropagaiton is to determine the expressions $\frac{\partial l}{\partial W^i_{j,k}}$ and $\frac{\partial l}{\partial b^i_j}$. By chain rule the desired expressions are given by:

$$\delta^i_j = \sigma_i'(z^i_j) \sum_{r=1}^{d_{i+1}} \delta^{i+1}_r W^{i+1}_{r,j} \tag{2.1.6}$$

$$\frac{\partial l}{\partial W^i_j} = \sum_{r=1}^{d_i} \frac{\partial l}{\partial z^i_r} \frac{\partial z^i_r}{\partial W^i_{j,k}} = \delta^i_j x^{i-1}_k \tag{2.1.7}$$

$$\frac{\partial l}{\partial b^i_j} = \sum_{r=1}^{d_i} \frac{\partial l}{\partial z^i_r} \frac{\partial z^i_r}{\partial b^i_j} = \delta^i_j \tag{2.1.8}$$

It is not hard to observe the components of $\boldsymbol{\delta}^i$ are determined recursively by $\boldsymbol{\delta}^{i+1}$ and W^{i+1}. The backward pass action is originated in this step, and thus called backpropogation.

To summarise, the feedfoward neural networks are the building blocks of all other classes of neural networks. The techniques stated above with some variations will be used throughout in the section of recurrent neural network and Mixture Density Network.

2.2 Recurrent Neural Network and LSTM

Recurrent neural network (RNN) is a class of neural network appropriate for processing sequential data. In contrast, feedfoward neural networks often ignore the ordering and sequential properties of the data, and the data are usually shuffled before forward pass into the neural network. This is undesirable in the analysis financial time series, since time series of price and volatility often exhibit autocorrelation structures and clustering phenomenon, which was first observed by Mandelbrot [22]. The general idea behind recurrent neural network is that sequential information are stored and passed on within the hidden states in the network, so the network has memory on the past information which are used to predict future values.

2.2.1 Forward Pass of Recurrent Neural Network

There are various architectures of recurrent neural network, and they differ in how layers are connected. The Goodfellow's book [13, Chapter 10, page 367 -415] has introduced several important examples of recurrent neural networks and the methodologies to unfold recursive computations into computational graphs. A general and universal recurrent neural network forward pass update equations [13, Chapter 10, page 372] can be represented by the following equations and Figure 2.2 is a simple illustration :

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{2.2.1}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}) \tag{2.2.2}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{2.2.3}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)}) \tag{2.2.4}$$

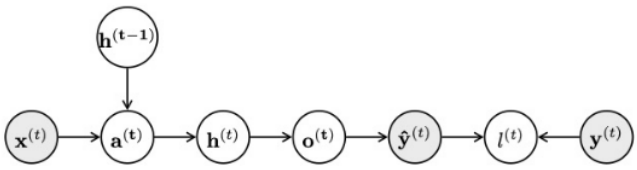

Figure 2.2: Recurrent Neural Network forward pass

In a forward pass at time sequence t, there are three recurrent connections weight matrices: input-to-hidden $\boldsymbol{U}$, hidden-to-hidden $\boldsymbol{W}$ and hidden-to-output $\boldsymbol{V}$. $\boldsymbol{b}$ and $\boldsymbol{c}$ are the bias terms and the hyperbolic tangent and the softmax activation functions are assumed to determine the current $\boldsymbol{h}^{(t)}$ and the predicted values $\boldsymbol{y}^{(t)}$. Analogy to feedfoward neural network, the predicted value $\hat{\boldsymbol{y}}^{(t)}$ can be written as composition of linear affine functions and activation functions, i.e.

$$\begin{aligned}
\boldsymbol{L}_1(\boldsymbol{z}_1, \boldsymbol{z}_2) &:= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{z}_1 + \boldsymbol{U}\boldsymbol{z}_2 \\
\boldsymbol{L}_2(\boldsymbol{z}_3) &:= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{z}_3 \\
\hat{\boldsymbol{y}}^{(t)} &:= \boldsymbol{\sigma}_2 \circ \boldsymbol{L}_2(\boldsymbol{h}^{(t)}) \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{L}_1(\boldsymbol{x}^t, \boldsymbol{h}^{(t-1)})
\end{aligned}$$

where the activation functions $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are the hyperbolic tangent and softmax function respectively.

2.2.2 Backpropagation Through Time

To train the parameters in recurrent neural networks, gradient based algorithm is applied and the gradients are computed by techniques similar to the backpropagation we have described in subsection 2.1.3, which is called Backpropagation through time (BPTT) algorithm. The objective of BPTT algorithm is to compute the gradients of loss function l with respect to the parameters $\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{c}$

In recurrent neural networks, the loss function is evaluated at each time sequence $l^{(t)}$, and the total loss function is the sum over of all time, i.e. $l := \sum_t l^{(t)}$. Therefore, the gradients have to sum over all time t, and they are given by the following. For details would advise readers refer to the Goodfellow's book [13, Equations 10.22-10.28, page 380]

$$\nabla_{\boldsymbol{c}} l = \sum_t (\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}})^\top \nabla_{\boldsymbol{o}^{(t)}} l$$

$$\nabla_{\boldsymbol{b}} l = \sum_t (\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}})^\top \nabla_{\boldsymbol{h}^{(t)}} l$$

$$\nabla_{\boldsymbol{V}} l = \sum_t \sum_i (\frac{\partial l}{\partial o_i^{(t)}}) \nabla_{\boldsymbol{V}^{(t)}} o_i^{(t)}$$

$$\nabla_{\boldsymbol{W}} l = \sum_t \sum_i (\frac{\partial l}{\partial h_i^{(t)}}) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)}$$

$$\nabla_{\boldsymbol{U}} l = \sum_t \sum_i (\frac{\partial l}{\partial h_i^{(t)}}) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)}$$

2.2.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) neural network is a particular type of recurrent neural network. LSTM was first introduced by Hochreiter and Schmidhuber (1997) [16] to address the problem of vanishing and exploding gradient in recurrent neural networks. Recurrent neural networks have problems in long-term dependency. They are great in learning past information of input sequence. However, gradients will tend to blow up or vanish during training of recurrent neural networks with BPTT algorithm due to long sequence of multiplications. The paper [16] provides detail analysis on the vanishing and exploding gradient problem. We will briefly highlight the main focus of the problem.

Recall the backpropagation equation 2.1.6, Hochreiter and Schmidhuber (1997) [16] describes the expression as j's backpropagated error signal.

$$\delta_j^i = \sigma_i'(z_j^i) \sum_{r=1}^{d_{i+1}} \delta_r^{i+1} W_{r,j}^{i+1}$$

In the training of recurrent neural network with BPTT, the following product relevant to the equation 2.1.6 determines the total error backward pass

$$\prod_m \sigma_{l_m}'(z^{l_m}(t-m)) W_{l_m, l_{m-1}}$$

If the above product greater than 1, then the error back flow increases exponentially and blows up, the weights oscillate and training is unstable. On the other hand, if the product is less than 1, the error back flow decreases exponentially and vanish, thus nothing can be learned. Since in the cases of blows up or vanishes mainly depend on the behaviours of multiplication of weight matrix, another simple intuition is by Singular Value Decomposition (SVD), the SVD of the weight matrix is given by

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$$
$$\tilde{\boldsymbol{W}}^n := (\boldsymbol{W}^\top \boldsymbol{W})^n = \boldsymbol{V}\tilde{\boldsymbol{\Sigma}}^n \boldsymbol{V}^\top$$

where $\boldsymbol{\Sigma}$ is the diagonal matrix of non-zero singular values (eigenvalues) of the matrix $\boldsymbol{W}$. If we raise $\tilde{\boldsymbol{W}}$ to the power of n, the singular values determine whether it blows up or vanish in the case the singular values are greater than 1 or smaller than 1 correspondingly.

Structure of LSTM

The core structure in LSTM is called gates. The idea of gates is carefully deciding the portion of information that can flow through in forward pass, which eliminates the effects of long-term dependency in ordinary recurrent neural networks. There are three main gates in LSTM:

1. Forget Gate
   When information leaves from the previous time sequence and enters the current state, the forget gate controls the portion of past information to be dropped out.
   $$\boldsymbol{f}_t = \text{sigmoid}(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f)$$
2. Input Gate
   The feature $\boldsymbol{x}^{(t)}$ at current time enters the LSTM unit, the input gate controls the portion of current information to be updated.
   $$\boldsymbol{i}_t = \text{sigmoid}(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i)$$
3. Output Gate
   The output gate decides the portion of information at current time to let through as input for the next LSTM unit.
   $$\boldsymbol{o}_t = \text{sigmoid}(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o)$$

Sigmoid activation function is applied componentwise amongst the gates in LSTM to describe how much information is allowed to let through, it outputs value between 0 and 1 which makes it an appropriate candidate for information control, i.e.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \quad x \in \mathbb{R}$$

After the information pass through the gates, there are central processors like structure called the

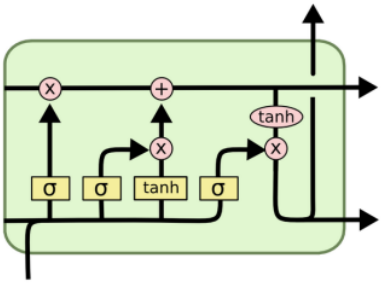

Figure 2.3: Single LSTM Unit [27]

cell state to *pack* information from the gates. The cell state consists of the following two steps:

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_{\tilde{c}} \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\tilde{c}} \boldsymbol{x}^{(t)} + \boldsymbol{b}_{\tilde{c}})$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \star \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \star \tilde{\boldsymbol{c}}_t$$

where $\star$ is the componentwise multiplication operation. $\boldsymbol{c}_{t-1}$ contains information of the cell state from the previous time sequence, and $\tilde{\boldsymbol{c}}_t$ contains current cell state information from the current time sequence, while $\boldsymbol{c}_t$ is the *packed* information from the forget and input gates by multiplying the vectors $\boldsymbol{c}_{t-1}$, $\tilde{\boldsymbol{c}}_t$ componentwise with $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ respectively and sum them up to update the current cell state $\boldsymbol{c}_t$. Lastly, the current cell state together with the output gate will determine the estimated $\hat{\boldsymbol{y}}_t$ with some activation function $\boldsymbol{\sigma}$:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{\sigma}(\boldsymbol{o}_t \star \tanh(\boldsymbol{c}_t))$$

Analogy to recurrent neural networks, BPTT algorithm is employed to train the LSTM with parameters within the three gates and the cell states. The expressions of the derivatives in BPTT of LSTM are variations to those in RNN we have described and they will not be stated for the sake of brevity.

2.3 Mixture Density

The insufficiency of using normal distribution to model high frequency returns has been well established empirically and motivates alternative choices of distribution such as the t-distribution in modelling extreme values. Norets (2010) [25, Proposition 2.1 and Corollary 3.1, page 5-11] proofs that mixture density can approximate any parametric and non-parametric density to an arbitrary degree of precision with sufficient number of components, which justifies the capability of mixture density to model non-normality characteristics. Moreover, the component densities in mixture density in financial returns modelling are relevant to the regimes in the market. The flexibility of mixture density is therefore make it a desirable candidate in financial modelling. We will first highlight some basic properties of mixture density, then Mixture Density Network will be introduced. Lastly, relationship between mixture density and market regimes will be justified.

2.3.1 Statistical Properties

Definition 2.3.1. (Finite Mixture Density) Finite mixture density is a probability density function $p(x)$ admits a finite weighted sum of component probability density function $p_1(x), \ldots, p_m(x)$ and weights $\lambda_1, \ldots, \lambda_m$, is given by

$$p(x) = \sum_{i=1}^{m} \lambda_i p_i(x)$$

where $m \in \mathbb{N}_{>0}$ is finite, $\sum_{i=1}^{m} \lambda_i = 1$, and $\lambda_i \geq 0$ for all $i = 1, \ldots, m$

We will consider the parametric family of densities $\mathcal{D} = \{p(\cdot; \boldsymbol{\theta}) | \boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d\}$. In particular, we will consider normal mixture where the parametric family of densities belongs to normal distribution, i.e. the probability density function for the i-th component follows a univariate normal distribution and the density function is given by

$$\phi_i(x; \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{(x-\mu_i)^2}{2\sigma_i^2})$$

where $\mu_i \in \mathbb{R}$ and $\sigma_i > 0$, and $p(x; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{i=1}^{m} \lambda_i \phi_i(x; \mu_i, \sigma_i)$. The flexibility of mixture normal density comes from its multi-modality properties and allowing skewness and kurtosis different from univariate noraml distribution.

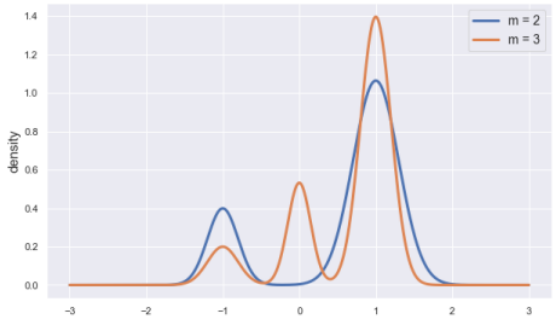

Figure 2.4: Multi-modality of Normal Mixture

Proposition 2.3.2. *(Properties of Finite Normal Mixture) Suppose the random variable X follows a finite mixture of normal distribution with m components has density $p(x; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{i=1}^{m} \lambda_i \phi_i(x; \mu_i, \sigma_i)$, also let $X_1, \ldots, X_m$ be the component random variables, where X_i has density function $\phi_i(x; \mu_i, \sigma_i)$, then we have the following:*

1. *Denote Φ be the cumulative distribution function of standard normal distribution (cdf). Then the cdf of finite mixture normal distribution $F_X(x)$ is given by*

$$F_X(x) = \sum_{i=1}^{m} \lambda_i \Phi(\frac{x - \mu_i}{\sigma_i}) \tag{2.3.1}$$

2. *For any measurable function f, the expectation of $f(X)$ is given by*

$$\mathbb{E}[f(X)] = \sum_{i=1}^{m} \lambda_i \mathbb{E}[f(X_i)] \tag{2.3.2}$$

In particular, denote $\mu := \mathbb{E}[X]$, then the k-th moment about zero and the k-th central moment is given by

$$\mathbb{E}[X^k] = \sum_{i=1}^{m} \lambda_i \mathbb{E}[X_i^k] \tag{2.3.3}$$

$$\mathbb{E}[(X - \mu)^k] = \sum_{i=1}^{m} \sum_{\substack{0 \le j \le k \\ j \text{ is even}}} \lambda_i \sigma_i^j (\mu_i - \mu)^{k-j} \frac{k!}{2^{\frac{j}{2}} (k-j)! (\frac{j}{2})!} \tag{2.3.4}$$

Proof. See Appendix A.1 □

By Proposition 2.3.2, one can compute the skewness and excess kurtosis of the mixture density by the following:

$$\beta_Y := \frac{\mathbb{E}[(Y - \mathbb{E}[Y])^3]}{(\mathbb{E}[(Y - \mathbb{E}[Y])^2])^{\frac{3}{2}}}, \quad \tilde{\kappa}_Y := \frac{\mathbb{E}[(Y - \mathbb{E}[Y])^4]}{(\mathbb{E}[(Y - \mathbb{E}[Y])^2])^2} - 3$$

Alexander and Narayanan (2001) [1, Table 4, Page 9] have shown empirically that it is able to construct all four possible cases of positive and negative skewness and excess kurtosis by choosing appropriate values of λ_i, μ_i and σ_i for $i = 1, 2$, in 2-component normal mixture density. This shows its ability to departure from normal distribution, which have zero skewness and excess kurtosis. Lastly the negative log-likelihood of finite normal mixture given observations $x_1, \ldots, x_N$ is given by

$$-\log \mathcal{L} = -\sum_{j=1}^{N} \log\{\sum_{i=1}^{m} \lambda_i \phi_i(x_j; \mu_i, \sigma_i)\} \tag{2.3.5}$$

2.3.2 Mixture Density Network

The architecture of Mixture Density Network only vary in the output layers of other conventional neural networks. Rather than having the output values $\boldsymbol{y}$ as some numerical values or labels, the Mixture Density Networks estimate the parameters $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma}$ of finite mixture density in the output layers. The input and hidden layers can connect through all other existing neural networks architectures, and only the output layers are wrapped in specific way to determine the parameters in finite mixture density. Denote $z_i^{\lambda}, z_i^{\mu}, z_i^{\sigma}$ be the networks outputs before transformation for the parameters $\lambda_i, \mu_i, \sigma_i$, then the following transformation proposed by Bishop (1994) [6] is given by

$$\lambda_i(\boldsymbol{x}) = \frac{\exp(z_i^{\lambda})}{\sum_{j=1}^{m} \exp(z_j^{\lambda})}, \qquad \mu_i(\boldsymbol{x}) = z_i^{\mu}, \qquad \sigma_i(\boldsymbol{x}) = \exp(z_i^{\sigma})$$

The weights $\lambda_i(\boldsymbol{x})$ is related to the softmax function, the location parameters $\mu_i(\boldsymbol{x})$ is an identity map, and the scale parameters $\sigma_i(\boldsymbol{x})$ is represented by exponential terms. As the neural network is making predictions on the probability distribution, negative log-likelihood stated in equation 2.3.5

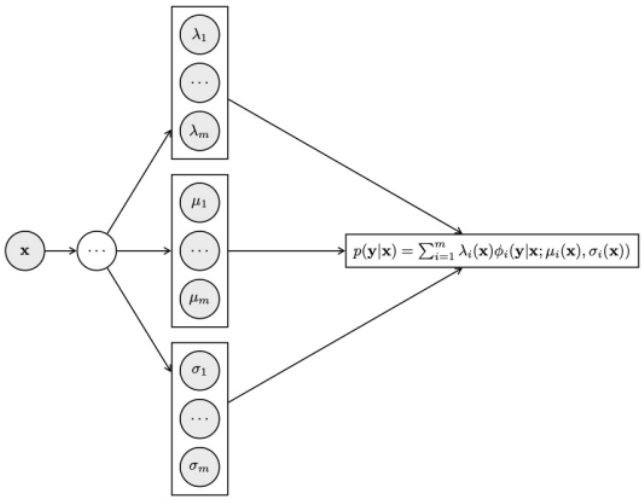

Figure 2.5: Mixture Density Networks

is chosen to be the loss function. Gradients of loss function with respect to $z_i^{\lambda}, z_i^{\sigma}, z_i^{\mu}$ can be determined analytically (see [6, Page 9-10]), and conventional backpropagation techniques are processed in the same way.

Lastly, it is not hard to show the standard approach to regression in neural network by minimizing sum-of-squares error is given by the mean of the probability density $p(\boldsymbol{y}|\boldsymbol{x})$, assuming the sample size of training data is sufficiently large. For $\boldsymbol{y} \in \mathbb{R}^O$ and training samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and consider the sum-of-squares error loss

$$E^S(\boldsymbol{\theta}) = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=1}^{N} \sum_{k=1}^{O} [f_k(\boldsymbol{x}^i, \boldsymbol{\theta}) - y_k^i]^2 = \frac{1}{2} \sum_{k=1}^{O} \int \int [f_k(\boldsymbol{x}, \boldsymbol{\theta}) - y_k]^2 p(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x}$$

By differentiating E^S with respect to f_k and set it to 0

$$\begin{aligned} \frac{\partial E^S}{\partial f_k} &= \int \int [f_k(\boldsymbol{x}, \boldsymbol{\theta^*}) - y_k] p(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x} = 0 \\ \implies f_k(\boldsymbol{x}, \boldsymbol{\theta}^*) &= \int \int y_k p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x} = \int y_k p(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y} \end{aligned}$$

where $\boldsymbol{\theta}^*$ is the optimal parameters that minimize the sum of square errors. For a classification problem with categorical cross entropy as the loss function can be proved in similar manner and will leave the details for readers refer to the original paper by Bishop (1994) [6]. Moreover, Bishop points out the E^S can be expressed as follows:

$$< y_k|\boldsymbol{x} > := \int y_k p(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y}, \quad < y_k^2|\boldsymbol{x} > := \int y_k^2 p(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y}$$

$$E^S(\boldsymbol{\theta}) = \frac{1}{2} \sum_{k=1}^{O} \int [f_k(\boldsymbol{x}, \boldsymbol{\theta}) - < y_k|\boldsymbol{x} >]^2 p(\boldsymbol{x}) d\boldsymbol{x} + \frac{1}{2} \sum_{k=1}^{O} \int [< y_k^2|\boldsymbol{x} > - < y_k|\boldsymbol{x} >^2] p(\boldsymbol{x}) d\boldsymbol{x}$$

Clearly, the minimum of E^S is achieved by the conditional mean of the density, and the error at its minimum can be interpreted as the average variance around the estimated conditional mean by the MDN. Therefore, the greater the variance of conditional distribution estimated by the MDN, the larger the error. We have further established similar arguments in Appendix A.2 involve SDEs, which show the deviation between the predicted and realised volatility is the cause of prediction errors in Mixture Density Network.

2.3.3 Hidden Markov Model and Mixture Density

When finite mixture density is applied in modelling, often encounter the very first natural question of the appropriate number of components m should be chosen. Brigo et al. (2013) [7, Section 4.4, Page 19] highlights from previous empirical studies the number of component densities are required to model the implied volatility surface in option pricing problems are typically 2 to 3, and typically one weight takes up most of the value. In [1] uses 2-component finite normal mixture to model the FX rates. Intuitively, by inspection of the statistical properties of mixture density we have described in Proposition 2.3.2, it hints that the appropriate number of components chosen is related to the number of regimes present in our data. In fact, the underlying relationship between the mixture density and different regimes can be justified by Hidden Markov Model (HMM). The following are inspired by the closely related work that is done by Berhane (2018) [4], we will briefly introduce the HMM, and how it relates to mixture density in our context. Firstly, we introduce some basic definitions and useful properties of Markov Chain.

Markov Chain

Definition 2.3.3. (Markov Chain) Let $\{Q_t\}_{t\geq 0}$ be a discrete time stochastic process take values in some state-space $E \subseteq \mathbb{Z}$. The stochastic process is a Markov Chain if it satisfies the Markov condition:

$$\mathbb{P}[Q_{t+1}|Q_0, \ldots, Q_t] = \mathbb{P}[Q_{t+1}|Q_t]$$

Definition 2.3.4. (Time-Homogeneous Markov Chain) The Markov Chain $\{Q_t\}_{t\geq 0}$ is time homogenous if

$$\mathbb{P}[Q_{t+1}|Q_t] = \mathbb{P}[Q_1|Q_0] \qquad \forall t \geq 0$$

Definition 2.3.5. (Transition Matrix) Denote the discrete state-space $E \subseteq \mathbb{Z}$, and let $|E| = D < \infty$. For $k \in \mathbb{N}$, the k-step transition matrix $P(k) = (p_{ij}(k))$ is the $D \times D$ matrix of k-step transition probabilities

$$p_{ij}(k) = \mathbb{P}[Q_{t+k} = j|Q_t = i] \qquad \forall i, j \in E, \quad t \geq 0$$

Theorem 2.3.6. *(Chapman–Kolmogorov) Let $m, n \geq 0$, then*

$$P(m+n) = P(m)P(n)$$

In particular, for any $k \geq 0$

$$P(k) = P^k$$

Definition 2.3.7. (Stationary Distribution) A vector $\boldsymbol{\pi}$ is a stationary distribution of discrete time Markov Chain $\{Q_t\}_{t\geq 0}$ with state-space E, and transition matrix P if:

1. $\sum_{i\in E} \pi_i = 1$
2. $\boldsymbol{\pi} P = \boldsymbol{\pi}$

In finite state-space, the stationary distribution $\boldsymbol{\pi}$ always exists and unique, which we omit the proofs, and the following theorem point out the convergence of transition probabilities and stationary distribution.

Theorem 2.3.8. *(Convergence of Stationary Distribution) For finite state-space $|E| < \infty$, then for any $i \in E$*

$$\lim_{t\to\infty} p_{ij}(t) = \pi_j$$

Furthermore, the stationary distribution is relevant to the quantity called the mean recurrence time. The mean recurrence time of state $i \in E$ is the average number of time steps ahead from current time that first revisit state i. The formal definition is given as follows:

Definition 2.3.9. (Mean Recurrence Time) Define T_i be the first hitting time of state $i \in E$ of the Markov Chain $\{Q_t\}_{t\geq 0}$:

$$T_i = \inf\{k \geq 1 : Q_k = i\}$$

The mean recurrence time M_i of state i is defined as

$$M_i = \mathbb{E}[T_i|Q_0 = i]$$

Theorem 2.3.10. *(Stationary Distribution and Mean Recurrence Time) For finite state-space E and stationary distribution $\boldsymbol{\pi}$, we have*

$$\lim_{t\to\infty} p_{ij}(t) = \pi_j = \frac{1}{M_j} \quad \forall i, j \in E$$

Hidden Markov Model

Before going into the formal definition of Hidden Markov Model, the intuition behind HMM is fairly simple. For example, assume there are only two states present in the market, the bearish and bullish market. However, the states of the market are *hidden* and we can only observe sequences of information such as prices, news, economic statistics etc. Therefore, the goal of HMM is to model the observed sequences and the hidden states jointly with by supposing the hidden states are Markov Chain.

Definition 2.3.11. (Hidden Markov Model) A Hidden Markov Model is a collection of random variables $\{(Q_t, O_t)\}_{t\geq 0}^T$, where $\{Q_t\}_{t\geq 0}^T$ is an unobserved discrete-value stochastic process and $\{O_t\}_{t\geq 0}^T$ is an observable discrete time stochastic process, satisfying the following conditions:

1. $\mathbb{P}[Q_{t+1}|Q_0, \ldots, Q_t] = \mathbb{P}[Q_{t+1}|Q_t]$
2. $\mathbb{P}[O_t|Q_0, \ldots, Q_T, O_0, \ldots, O_T] = \mathbb{P}[O_t|Q_t]$

for all t

The first condition in HMM states that the hidden $\{Q_t\}_{t\geq 0}$ is a Markov Chain, and the second condition induces conditional independence properties that the probability of the observed output O_t at time t only depends on the hidden state Q_t, while independent of the other observed outputs and states at other time.

Now we can think of the observed outputs are the log return of the prices $\{Y_t\}_{t\geq 0}$, while the unobserved states take values in some discrete state-space $E \subseteq \mathbb{Z}$ and in relevant with Mixture Density Network, given we have input features vector at time $\boldsymbol{x}_t$, for any time ahead $s \geq 1$, the Mixture Density Network predicts $\mathbb{P}[Y_{t+s}|\boldsymbol{x}_t]$ which can be expanded by the following:

$$\begin{aligned}
\mathbb{P}[Y_{t+s}|\boldsymbol{x}_t] &= \sum_{Q_{t+s}} \sum_{Q_t} \mathbb{P}[Y_{t+s}, Q_{t+s}, Q_t|\boldsymbol{x}_t] \\
&= \sum_{Q_{t+s}} \sum_{Q_t} \mathbb{P}[Y_{t+s}|Q_{t+s}, Q_t, \boldsymbol{x}_t]\mathbb{P}[Q_{t+s}, Q_t|\boldsymbol{x}_t] \\
&= \sum_{Q_{t+s}} \mathbb{P}[Q_{t+s}|\boldsymbol{x}_t]\mathbb{P}[Y_{t+s}|Q_{t+s}, \boldsymbol{x}_t]
\end{aligned}$$

The last equality makes use of the definition of HMM. Moreover, if we set the discrete state-space $E = \{1, \ldots, m\}$, and define the following:

$$\lambda_i(\boldsymbol{x}_t) := \mathbb{P}[Q_{t+s} = i|\boldsymbol{x}_t] \tag{2.3.6}$$

$$\phi(y|\boldsymbol{x}_t; \mu_i(\boldsymbol{x}_t), \sigma_i(\boldsymbol{x}_t)) := \mathbb{P}[Y_{t+s}|Q_{t+s} = i, \boldsymbol{x}_t] \tag{2.3.7}$$

Then we can write the expression as follows:

$$\mathbb{P}[Y_{t+s}|\boldsymbol{x}_t] = \sum_{i=1}^m \lambda_i(\boldsymbol{x}_t)\phi(y|\boldsymbol{x}_t; \mu_i(\boldsymbol{x}_t), \sigma_i(\boldsymbol{x}_t))$$

Thus, the number of component densities to be chosen to model the log return is equivalent to choose the number of hidden states. In the case of finite normal mixture, the i-th component mean μ_i and variance σ_i characterised the i-th regime in the market. Furthermore, the weights λ_i are in fact the probability of vising the i-th regime at time $t + s$ given they are estimated by $\boldsymbol{x}_t$.

Moreover, assume the predicted time frame s is sufficiently large, but not too large to violate the time-homogeneous property, then by the convergence property in Theorem 2.3.8, equation 2.3.6 can be expressed approximately as

$$\lambda_i(\boldsymbol{x}_t) = \sum_{Q_t=1}^m \mathbb{P}[Q_{t+s} = i|Q_t, \boldsymbol{x}_t]\mathbb{P}[Q_t|\boldsymbol{x}_t] \approx \sum_{Q_t=1}^m \pi_i(\boldsymbol{x}_t)\mathbb{P}[Q_t|\boldsymbol{x}_t] = \pi_i(\boldsymbol{x}_t) \tag{2.3.8}$$

where $\pi_i(\boldsymbol{x}_t)$ is the marginal stationary distribution of the hidden state i. We express the parameters in function of $\boldsymbol{x}$ to emphasis the parameters are predicted by our MDN which takes $\boldsymbol{x}$ as our input features. Also by Theorem 2.3.10, we can infer information about the mean recurrence time of the i-th regime once the $\lambda_i(\boldsymbol{x}_t)$ is realised, the greater the $\lambda_i(\boldsymbol{x}_t)$, the smaller the mean recurrence time, in other words, the more likely to revisit the i-th regime in closer time.

Lastly, back to the initial question about choosing appropriate number of components m is now clear that it is an equivalent to the classification of the number of regimes present in our market, which is one of the most popular research topics in quantitative finance. Recent study by Bilokon et al. (2021) [5] applies techniques in path signatures to classify the number of regimes in US equities time series data, and the study of regime classification is beyond our main scope. For readers interested in various regime classification theoretical background and recent algorithm development can refer to [5] and Nystrup et al. (2020) [26]. In our FX prediction problems, we fix the number of components $m = 2$ as suggested by most of the previous studies.

2.4 Interpretability of Neural Network

Interpretability, is a term to describe the level of how human understand how something or a system works. Neural network is often regarded as a "black box" model, where there are no straightforward interpretation between the parameters and the target variable, and also relationship between the input and output are unclear due to its non-linear nature. Therefore, interpretability of neural network has been gaining attention in finance, due to the broad applications of neural network, understanding the relationship between input and output is important to explain which features have significant impacts on our predicted variables, so that those subset of important features should be the main focus for future analysis.

Interpretability can be categorized into two types, local and global interpretability. Local interpretability focuses on understanding and explaining how an individual model make certain predictions. Global interpretability helps to understand an overall view on how the models make predictions and aims to recognize the average contributions of the features to the predictions. LIME is one of the most popular local interpretability methods introduced by Ribeiro et al. (2016) [29], and Shapley values from cooperative game theory [30] is used for global interpretability. We will focus on Shapley value to interpret the contribution of the different assets we choose to train our Mixture Density Network due to the non-linear nature and complicated dependencies of the assets amongst one another.

Let f be the model we desire to estimate, and let $\hat{f}$ be our neural network model. Also, denote our input features as $\boldsymbol{x} \in \mathbb{R}^p$. Interpretable representation $\boldsymbol{x}' \in \{0,1\}^p$ is used instead. In our main theme that the original input $\boldsymbol{x} \in \mathbb{R}^p$ corresponding to the prices of the p assets that we have chosen to train our neural network, however, they are just p numerical values but do not have much meaningful interpretation. One can give meaningful interpretation to the original input, for instances, in Brigo (2020) [9] defines a binary transformation to determine if x_i equals the average of the i-th features:

$$x_i' := \mathbb{1}_{\{x_i = m_i\}} = \begin{cases} 1, & x_i = m_i \\ 0, & x_i \neq m_i \end{cases}$$

where m_i is the average of the i-th features. Other examples such as determination of the presence of a particular word, where 1 indicates the word is present and 0 otherwise .
Then define the inverse map $h : \boldsymbol{x}' \to \boldsymbol{x}$ to recover the original input from the interpretable representation. In order to best interpret the non-linear model $\hat{f}$ estimated by the neural network, it is ideally to choose an explanation model g with linear structure. Furthermore, the explanation model g should also be able to interpret variations of the interpretable representation $\boldsymbol{x}'$, i.e. words in a sentence present and absent randomly This can be achieved by perturbing $\boldsymbol{x}'$ such that the perturbed observation $\boldsymbol{z}' \in \{0,1\}^p$ is similar to the interpretable representation, i.e. $\boldsymbol{z}' \approx \boldsymbol{x}'$. Therefore, the explanation model g should have following:

- Explanation model g can ideally approximate the model $\hat{f}$, where g is called local:

$$g(\boldsymbol{z}') \approx (\hat{f} \circ h)(\boldsymbol{z}')$$

- Explanation model g has additive feature attribution, and can be expressed as linear structure of z'_i for $i = 1, \ldots, p$:

$$g(\boldsymbol{z}') = \phi_0 + \sum_{i=1}^{p} \phi_i z'_i$$

In global interpretability, ϕ_i is determined by the Shapley values which are given by the following:

$$\phi_i(\hat{f}, \boldsymbol{x}) = \sum_{\boldsymbol{z}' \subseteq \boldsymbol{x}' \setminus \{i\}} \frac{|\boldsymbol{z}'|!(p - |\boldsymbol{z}'| - 1)!}{p!} (\hat{f}(h(\boldsymbol{z}' \cup \{i\})) - \hat{f}(h(\boldsymbol{z}'))) \tag{2.4.1}$$

where $\boldsymbol{x}' \setminus \{i\}$ is to set the $x_i = 0$ in $\boldsymbol{x}'$, in other words being "absence", while $\boldsymbol{z}' \cup \{i\}$ is to consider the i-th features being "presence". In equation 2.4.1, ϕ_i is not equivalent to the difference in our neural network prediction by removing the i-th features from training, it is the marginal contribution of the i-th feature across all possible "competitions" with other features, called the coalitions in cooperative game theory. In game theoretical point of view, the game is the prediction task, the features are the players in the game, and the difference between realised prediction and average prediction is the gain. Shapley values explain the fairly distribution of the total gains amongst all the players.

In the original cooperative game theory, a solution ψ in game G is a continuous map from the game G to some payout vectors, i.e. $\psi : G \to \mathbb{R}^N$, where for a good solution ψ satisfies at least one the four axioms: efficiency, dummy, symmetry and additivity. We will explain briefly about the four axioms by taking Shapley value as our example. Denote the gain function as v.

1. Efficiency: the sum of all features contributions is equal to the total gain, where the total gain in deep neural networks is the difference between the predicted $\hat{f}(\boldsymbol{x})$ at $\boldsymbol{x}$ and the mean $\mathbb{E}[\hat{f}(X)]$:

$$\sum_i \phi_i(\hat{f}, \boldsymbol{x}) = \hat{f}(\boldsymbol{x}) - \mathbb{E}[\hat{f}(X)]$$

2. Dummy: if the j-th feature never contributes, called dummy player, then $\phi_j = 0$. Moreover, for $S \subseteq \{x_1, \ldots, x_p\}$, the prediction will not affect by the joining of x_j.

$$v(S \cup \{x_j\}) = v(S) \qquad \forall S \subseteq \{x_1, \ldots, x_p\}$$

3. Symmetry: if any two features x_j and x_k, contribute equally to the total gain, then they have equal Shapley value.

$$v(S \cup \{x_j\}) = v(S \cup \{x_k\}) \implies \phi_j = \phi_k$$

4. Additivity: for two prediction tasks $\hat{f}_1, \hat{f}_2$, then the Shapley values $\phi(\hat{f}_1 + \hat{f}_2, \boldsymbol{x}) = \phi(\hat{f}_1, \boldsymbol{x}) + \phi(\hat{f}_2, \boldsymbol{x})$

Lastly, Shapley (1953) [30] proved that the Shapley values given in equation 2.4.1 is the unique solution that satisfies all of the four axioms.

Theorem 2.4.1. *(Shapley Theorem) There exists a unique solution ϕ that satisfies the efficiency, dummy, symmetry and additivity axioms, and the unique solution is given by equation 2.4.1.*

Lundberg and Lee (2017)[21] proposed SHAP (SHapley Additive exPlanations) to neural netwrok global interpretability, and apart from the four axioms, they state three additional properties that SHAP should satisfies: local accuracy, missingness and consistency. Lundberg and Lee (2017) [21, Theorem 1, Page 4] combined the Shapley Theorem that the Shapley values is the only possible solution satisfies all the four axioms and the three additional properties. For detail mathematical formulation of the three properties and further technical details can refer to Lundberg and Lee (2017)[21].

Section 3

Data Description

Our main aim is to use current prices of various asset classes and features to predict the probability distribution of the mid log return of our targeted currency pairs of 1-hour, 3-hour and 8-hour ahead. The three most liquid FX pairs will be used as our targeted currency pairs: GBP/USD, EUR/USD, USD/JPY, and the asset classes include bond, equity, derivative, currency SWAP and commodity will be used as input features to predict the target currency. The dataset is in 1-minute frequency, ranging from December 2020 to June 2021, resulting in a very large dataset, with over 200000 observations. We split our dataset from December 2020 to May 2021 as our training dataset, while the rest is our test set. All of our data is obtained through Bloomberg Terminal, where the access was kindly authorized Bloomberg. The variables of our dataset are summarised in Table 3.1, detail descriptions about the tickers can be found in Table B.1.

FX	**Government Bond**	**Equity**	**FX Option**	**SWAP**	**Commodity**
GBP/USD	GT10 GOV	SPX	GBPUSDV1M	USSW10	CL1 Comdty
EUR/USD	GTGBP10Y Govt	VIX	EURUSDV1M	BPSW10	XAU
USD/JPY	GTJPY10Y Govt	NKY	USDJPYV1M		

Table 3.1: Dataset Variables in Bloomberg Tickers

In addition, we compute the bid-ask spread which is given by spread := ask price − bid price as the additional features to train our neural networks. Therefore, suppose at current time t, we desire to predict the probability distribution of the GBP/USD log return of some future time s ahead, then our input features consist of the prices and bid-ask spread of EUR/USD, USD/JPY and all other assets, as well as the bid-ask spread of GBP/USD at current time t.

The use of government bond for prediction has straightforward intuition because it has direct linkage with the FX rates, since yield or interest rate of one government bond affects both the supply and demand of the currency. Interest rate of government bonds is affected by the central bank of one country. For example, the U.S. Fed "set" the interest rate by buying or selling government bonds in the open market from major financial institutions, so the Fed directly gives or takes liquidity of the U.S. dollar. Furthermore, the interest rate affects cost of funding and trades with different countries etc. and thus impacts the demand for the currency. Hence, one expect the government bonds and SWAPs have significant predictive powers on the FX rates. Equity may not have direct explanatory power on the FX rates. However, equity are often interconnected with bonds in portfolio constructions for portfolio diversification. FX options are often used by investors to hedge currency risks or sometimes for speculation on currency moves, thus the options might be indicative on investor expectations on the FX rates. Lastly, two commodities are selected, crude oil and gold. Before 1971 under the Bretton Woods system, countries worldwide adopted a fixed FX rate pegged with the U.S. dollars, and the dollars are pegged with Gold. After the termination of Bretton Woods, the U.S. dollar continue to dominate the currency market due to the fact that it is the only currency the crude oil prices are quoted. Therefore, the supply and demand of the oil will directly impact the U.S. dollar. Although gold has already lost its domination role in the currency rates, investors often invest gold to hedge against inflation, while monetary policies by most of the central banks from developed countries are inflation targeting, and thus gold nowadays are connected with currency indirectly via inflation.

3.1 Data Cleaning

Data cleaning is the process of handling missing data, detecting and removing potential outliers that may have negative impacts on the performance of model predictions and lead to draw false conclusion.

Our raw dataset has the issue of missing data, which are caused by two reasons: different trading hours among different assets and random data missing. The former situation is caused by different time zone or the nature of the asset itself. For example, the FX market opens 24 hours a day, while equity markets depend on the opening hours of the exchanges and the time zones where the exchanges located. The later situation might be caused by random error. When we examine the raw dataset closely, we found out for an individual asset, the total number of observations of bid, ask and mid prices are not the same. The total number of observations should be equal if the bid, ask and mid prices are well recorded in 1-minute frequency interval in ideal case, however, there are missing observations in some minutes and the chance of occurrence does not exhibit any particular pattern, so we consider that as random error.

We handle the missing data by replacing them with the value in previous time for both scenarios described above. By handling missing data in this way implicitly assumes the Efficient Market Hypothesis, which the prices adjustment are instantaneous that reflect all available information, and the current price is the best estimator of future prices.

To improve training in neural networks, it is common to normalize the data. There are several standard methods for data normalization, we employ the Min-Max transformation to scale data within 0 and 1:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

3.2 Exploratory Data Analysis

In this section, we will examine empirical statistical of our dataset to have deeper understanding on the structure of our dataset.

3.2.1 Correlation

We first compute the correlation coefficient amongst all the pairs of asset to explore how different assets are correlated to each other in the period of our dataset.

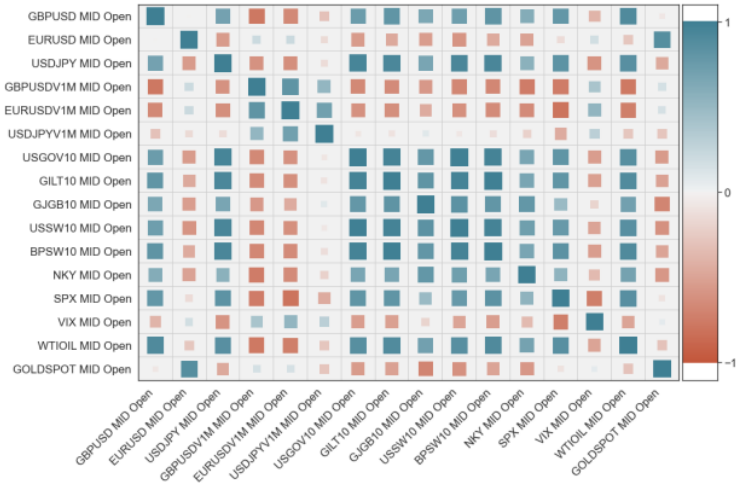

Figure 3.1: Heatmap of Correlation Matrix

The FXs are highly correlated with bonds and SWAPs. For example, the correlation coefficient between GBP/USD and US 10-Year Bond Yield is 0.79, and the correlation coefficient between

USD/JPY and US 10-Year SWAP is 0.95. Moreover, the GBP/USD, USD/JPY have high positive correlation with the WTI Crude Oil, while the EUR/USD are positive correlated with the Gold. Detail numerical descriptions can be found in Appendix B.1

3.2.2 Empirical Density and Realized Variance

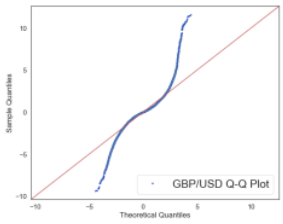

Figure 3.2: QQ-Plot: 1-hour GBP/USD log return

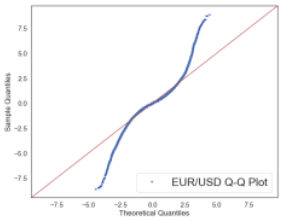

Figure 3.3: QQ-Plot: 1-hour EUR/USD log return

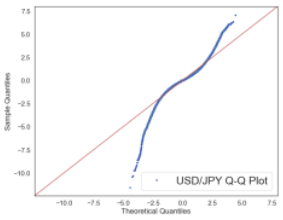

Figure 3.4: QQ-Plot: 1-hour USD/JPY log return

Non-normality of financial returns are often observed in the market. We examine the empirical distributions of our FX rates data. Figure 3.2, 3.3 and 3.4 are the QQ-Plot of the 1-hour mid log return of the three currency pairs. Points should fall on the red line if the log returns are normal distributed. The flipped S-shape observed in all of the three currency pairs indicate non-normality with heavier tails than normal distribution, and the distribution of the returns are all leptokurtic with kurtosis greater than 3. We report the empirical mean (μ) , standard deviation (σ) , skewness (β) and kurtosis (κ) to further justify the 1-hour log return distributions are non-normal.

	μ	σ	β	κ
GBP/USD	1.5×10^{-5}	1.02×10^{-3}	0.1812	8.1546
EUR/USD	3.2×10^{-8}	7.27×10^{-4}	-0.0963	5.1026
USD/JPY	1.6×10^{-5}	6.62×10^{-4}	-0.3231	4.9059

Table 3.2: 1-hour log return empirical statistics

Table 3.2 confirms the 1-hour empirical log return is leptokurtic and GBP/USD is positively skewed while EUR/USD and USD/JPY are negatively skewed, the kurtosis of the three FX pairs are all greater than 3 over the period of observations of our dataset. This confirms the empirical distributions of the FX pairs are different from normal distribution and motivate the use of finite normal mixture distribution to model the log return of the FX pairs.

Realized Volatility

The realised volatility of frequency n ($RV^{(n)}$) is a statistics often used to measure the instantaneous volatility of the underlying price dynamics. In fact, Nielsen and Shepherd (2001) [3] show the realised volatility converge in probability to the integrated instantaneous volatility of general stochastic volatility model, i.e. $\int \sigma(t)^2 dt$. Prediction of volatility is challenging and volatility is often the main cause of deviations of models predictions and realisation (see Appendix A.2). Therefore, it is important to understand its dynamics which allows us to outline the potential obstacles of the accuracy of our model predictions.

Denote the realised variance of frequency n at time t is $RV_t^{(n)}$, and we look back n discrete time points $t_1 < \cdots < t_n = t$, and the statistics is given by taking the sum of squared log return

$$RV_{t_n}^{(n)} = \sum_{k=1}^{n-1} (\log(\frac{S_{t_{k+1}}}{S_{t_k}}))^2$$

We can compute the 1-hour realized variance by taking $n = 60$ and taking the square of the 1-minute log return of the three currency pairs.

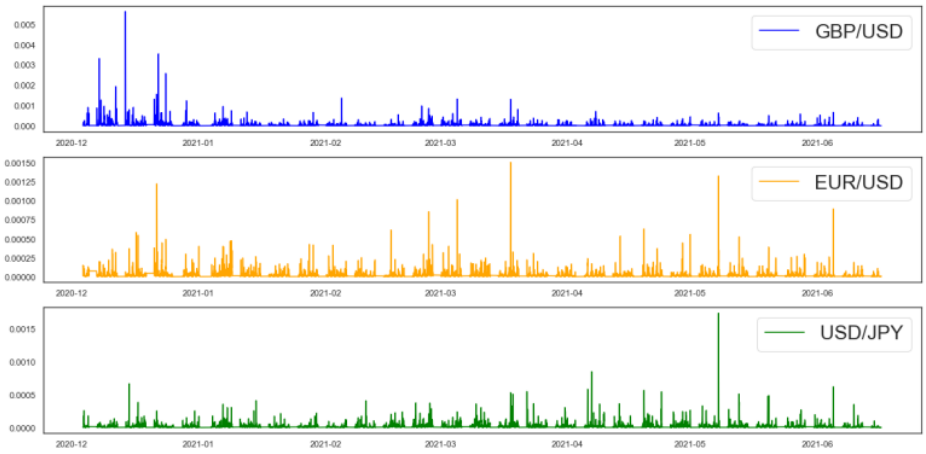

Figure 3.5: 1-hour Realized Variance $RV^{(60)}$

We first compute the realised variance $RV^{(60)}$ for every minute, then the computed values are grouped by each 24 hours in a day and their sample mean is determined. In Figure 3.6 is the plot of the $RV^{(60)}$ sample mean in every hours in a day.

The FX market opens 24 hours a day, but trading activities vary in different hours due to different international time zones. Hours of major trades take place in one region is called the trading session. Investors mainly concern about three important regions of trading sessions: Tokyo, London, and New York, where major financial institutes are located. In practice, investors set the Tokyo Session (12am - 9am UTC), the London Session (8am - 4pm UTC) and the New York Session (1pm - 10pm UTC). The dynamics of the mean realized variance in different sessions varies. The three FX pairs start to get more volatile when the London Session comes in, exhibit a U-shape pattern, and peak at a specific time point calls the "London 4pm Fix" (The Fix). The Fix is considered an important currency benchmark, and execution around the Fix, from 3:59:30 to 4:00:30 London local time, is used by global banks, asset managers and other institutional investors as the official benchmark rate that best values their portfolio holdings, hedging, measuring tracking error etc. [31]. Thus, the realized variance over the 24-hour period peeks for the three FX pairs at the Fix due to the large volume of transactions around the Fix. Moreover, the overlapping hours between the London and New York Session is considered to be the most volatile hours, due to frequent transactions and announcements of important economic data are often made in those hours. Thus, surprised changes in prices often occur and one may expect greater prediction errors within the London Session and New York Session.

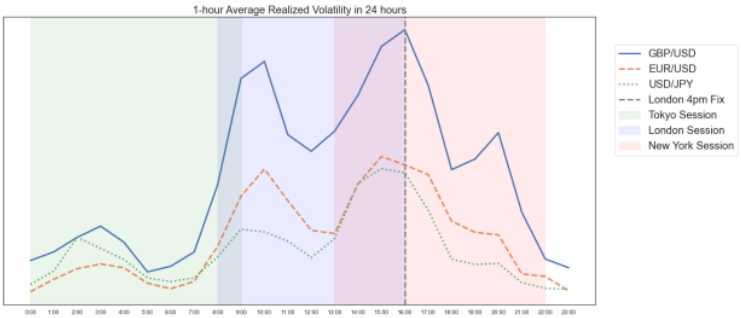

Figure 3.6: Mean $RV^{(60)}$ in 24 Hours

3.2.3 Bid-Ask Spread

We compute the bid-ask spread of each asset as additional features to train our neural network. Investors concern about the bid-ask spread is because it is the main source of transaction cost in trading FX, and it is an effective measure for the liquidity of the underlying asset. Existing literature and empirical studies have shown the bid-ask spread contains various components including inventory cost, adverse selection, volume of trading, risks expectation of market makers etc. Studies by Stoll (1978) [32], (1989) [33], Glosten and Milgrom (1985) [12] etc. investigated in the information of the bid-ask spread contains. Thus, we expect the bid-ask spread is an effective feature for predicting future prices.
Lastly, we report the mean (μ), standard deviation (σ), median and maximum of the bid-ask spread of the three FX pairs over the period of our dataset in Table 3.3

	μ	σ	median	max
GBP/USD	0.000279	0.000726	0.0001	0.014
EUR/USD	0.000117	0.000271	0.0001	0.0112
USD/JPY	0.013944	0.032725	0.01	0.58

Table 3.3: Bid-Ask Spread Statistics

FX trading in practice usually measures rates in pips. 1 pip refers to the the smallest price move that an exchange rate can make, which is the smallest decimal place that the FX rate can be changed and quoted. For GBP/USD and EUR/USD, a pip refers to 0.0001, while a pip in USD/JPY is 0.01. From Table 3.3 the average bid-ask spread for both EUR/USD and USD/JPY is around 1 pip and GBP/USD is arouind 3 pips. The maximum spread are 140 pips, 112 pips and 58 pips for GBP/USD, EUR/USD and USD/JPY respectively. Nielsen (2018) [24] points out large spread might have the problem of illiquid market that one may observe the move in mid price is dominated only by either the bid or ask price and such data should be avoided during training. By looking at the three FX pairs at their maximum spread, only the USD/JPY exhibits a one side domination on the ask price at very short instance and return to normal level very quickly as illustrates in Figure 3.9, while the other two FX pairs showed symmetric patterns on their bid and ask prices at the instance of maximum spread. It is uncommon to detect such phenomenon unlike frequent detections in [24], it is because the FX rates data from Bloomberg are based on collections of hundreds of currency data providers, while the data in [24] are collected from only three different providers. Hence FX rates from our data source have higher level of generality that we do not neglect such data in our training in order to build a robust neural network.

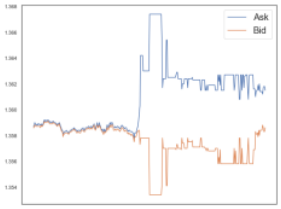

Figure 3.7: GBP/USD Bid Ask at Maximum Spread

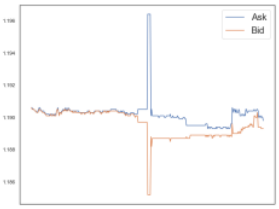

Figure 3.8: EUR/USD Bid Ask at Maximum Spread

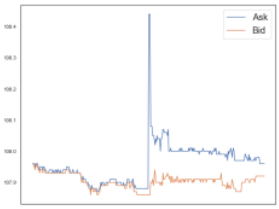

Figure 3.9: USD/JPY Bid Ask at Maximum Spread

Section 4

Mixture Density Network

4.1 Architecture

The proposed MDN architecture for training our dataset is a hybrid of dense layers and LSTMs. We fix $m = 2$, the number of components in finite normal mixture density, to model the mid log return of our targeted FX pairs as discussed in section 2.3.3

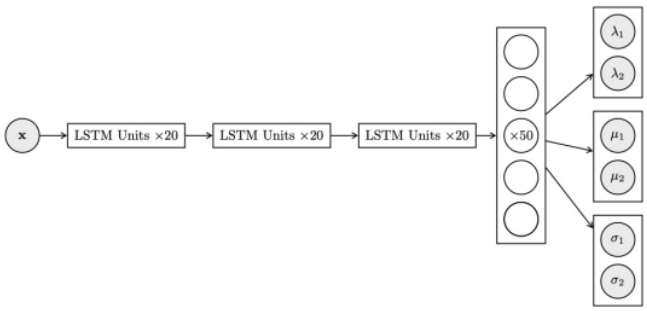

Figure 4.1: Mixture Density Network Architecture

In Figure 4.1 is an illustration of the architecture of our MDN. The input $\boldsymbol{x}$ passes to three stacked layers of LSTM, with 20 hidden units in each LSTM. A dense layer with 50 hidden units and ReLU activation function is added after the three stacked layers of LSTM, and the output parameters of the normal mixture with 2 components are then computed as in section 2.3.2. 0.2 dropout layers are added between all the layers to avoid the problem of overfitting. A summary of the architecture is shown in Table 4.1. The negative log likelihood stated in equation 2.3.5 is used as our loss function, the Adam optimizer with learning rate 0.001 is applied and with batch size of 256.

Units	Hidden Layer	Dropout
20	LSTM	0.2
20	LSTM	0.2
20	LSTM	0.2
50	Dense Layer ReLU Activation	0.2
6	Mixture Density Parameters Layer	0

Table 4.1: MDN Architecture Summary

The constructions of using stacked RNN are discussed by Pascanu et al.(2014) [28], in which they discuss the architecture of stacking layers of recurrent units potentially allows the hidden state at each level to operate at different timescale. They described stacked layers of RNN as deep

RNN, which they found out the deep RNNs outperform conventional RNNs in tasks of polyphonic music predictions and language modelling. Also, Chen et al.(2017) [10] improves their house price prediction problem with neural network architectures by stacking layers of LSTM. These form the rationale of designing our neural network architecture and it is worth to explore if such architecture is robust in predicting the probability density function with Mixture Density Network.

4.2 Empirical Results

4.2.1 Losses

In the following, we train our MDN to predict 1-hour, 3-hour and 8-hour mid log return of the three FX pairs. The empirical training and validation losses are presented in the Table 4.2.

	Training			Validation		
Hour(s)	1	3	8	1	3	8
GBP/USD	-1.7432	-1.4311	-1.3317	-0.6970	-0.5412	-0.0636
EUR/USD	-1.5821	-1.1143	-1.0497	-0.9048	-0.8783	-0.1012
USD/JPY	-1.2786	-1.1087	-0.7739	-0.6402	-0.6213	-0.2540

Table 4.2: MDN Empirical NLL Losses

The training losses are less than the validation losses for the three FX pairs, which indicates overfitting is not significant. The losses increase as the time ahead of prediction increases, which is sensible in general as prediction uncertainty increases.

We can also compute the conditional mean of the predicted probability density function by Proposition 2.3.2 which equals to the predicted values of conventional neural networks. We can pick out the mean square errors evaluated in a specified hour for every trading days and determine the sample average. Figure 4.2 shows the mean square errors with normalization (i.e. $x' = \frac{x-\mu}{\sigma}$, where μ, σ are the empirical mean and standard deviation) for the purpose of better visualisation and comparison.

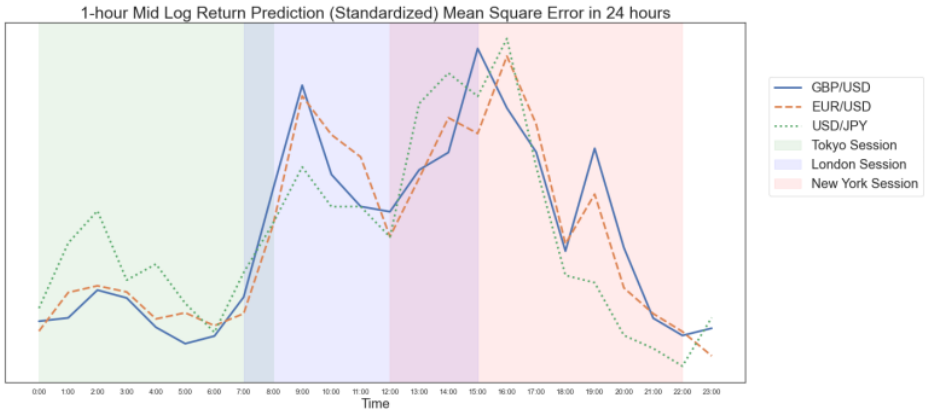

Figure 4.2: Mean Square Error in 24 Hours

The mean square errors remain low during the Tokyo Session, while the mean square errors of the three FX pairs show U-shape pattern over the London Session, peaks at around the start of the London Session and around the "Fix", follow by a decline after the end of London Session. The similar patterns in Figures 3.6 and 4.2 shows that the prediction performances in hours with high volatility, such as the overlapping hours London and New York Sessions, is relatively worse than other hours. We report the numerical values of the mean square error for the whole dataset in Table B.2.

4.2.2 Parameters

1 Hour	λ_1	λ_2	μ_1	μ_2	σ_1	σ_2
GBP/USD	0.7600	0.2400	6.27×10^{-5}	-1.67×10^{-5}	7.62×10^{-4}	141×10^{-4}
EUR/USD	0.7879	0.2121	1.65×10^{-4}	-1.00×10^{-4}	5.34×10^{-4}	11.0×10^{-4}
USD/JPY	0.7530	0.2470	1.03×10^{-3}	-1.22×10^{-3}	7.53×10^{-4}	13.9×10^{-4}

Table 4.3: Empirical Mean of Predicted 1-Hour Log Return Mixture Density Parameters

Table 4.3 shows the empirical mean of the 1-hour mid log return with 2-component normal mixture parameters predicted by our Mixture Density Network. One can observe some patterns on the parameters: the weight is mainly dictated by λ_1; μ_1 is positive and μ_2 is negative; σ_1 is smaller than σ_2. In section 2.3.3, we have discussed the component densities are correspond to different hidden states or regimes characterised by the mean and variance (μ, σ) in the case of using normal distribution as our component density. Nystrup et al.(2020) [26] proposed the market is in two states: the first state has positive mean and low volatility, and the second has negative mean and high volatility, which our predicted mixture density parameters of 1-hour log return are consistence with the properties of regimes suggested by previous studies. Moreover, information about the probability of being in the i-th regime is captured by λ_i as shown in equation 2.3.6. Therefore, there is a higher probability of being in Regime1 with positive mean and lower volatility than the Regime2 with negative mean and higher volatility by our empirical results in 1-hour FX rates log return.

Similar characteristics with Table 4.3 can be observed in the empirical results of 3-hour and 8-hour predictions are reported in Appendix B.3 (see Table B.3 and B.4). The decreasing accuracy for a much larger prediction time horizon weakens the model confidence of our Mixture Density Network. The characterisation of λ_i as the stationary probability of being in the i-th state require the assumption of time-homogeneous property of the underlying hidden Markov state, which may be violated in a large prediction time horizon. In general, we can draw conclusion on our overall results: the predicted 2-component mixture density is able to distinguish two regimes, a regime of higher return with lower volatility (Regime1) and a regime of lower return with higher volatility (Regime2).

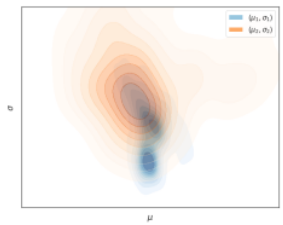

Figure 4.3: Empirical (μ, σ) 2D-Contour Plot: GBP/USD (1-hour)

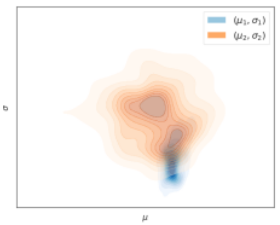

Figure 4.4: Empirical (μ, σ) 2D-Contour Plot: EUR/USD (1-hour)

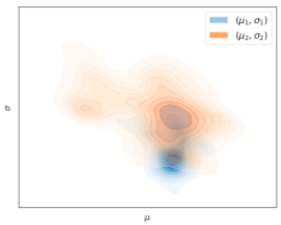

Figure 4.5: Empirical (μ, σ) 2D-Contour Plot: USD/JPY (1-hour)

We plot all the predicted pairs (μ_1, σ_1) and (μ_2, σ_2) for the three currency pairs estimated by our Mixture Density Network in Figures 4.3, 4.4 and 4.5, with the values of component mean on the x-axis and the values of component standard deviation on the y-axis. Moreover, we label the predicted pairs with respect to their components. The blue cloud represents all the predicted pairs (μ_1, σ_1), and we label it as Regime1 and the orange cloud represents all the predicted pairs (μ_2, σ_2), and we label it as Regime2. It is not hard to see the blue clouds generally have relatively smaller standard deviation and higher mean than the orange clouds, which confirms the capability of components of mixture density to capture information of different regimes as described in [26]. Moreover, as one might have noticed there are clusters formed in between the blue and orange clouds, which have similar mean returns as Regime1 with higher standard deviation similar to Regime2. Nevertheless, our empirical results justify the fact that either 2-component or 3-component mixture density is

sufficient in modelling FX rates log return, and the Mixture Density Network is able to capture characteristics of different regimes present in the market.

4.2.3 Global Interpretability

Rank	**GBP/USD**	**EUR/USD**	**USD/JPY**
1	EUR/USD	U.K. 10-Year SWAP	U.S. 10-Year Bond
2	U.S. 10-Year Bond	GBP/USD	U.K. 10-Year Bond
3	GBPUSDV1M	USDJPYV1M	USDJPYV1M
4	USDJPYV1M	S&P 500	EUR/USD
5	EURUSDV1M	GBP/USD Spread	Nikkei 225
6	Nikkei 225	U.K. 10-Year SWAP Spread	Japan 10-Year Bond
7	U.S. 10-Year Bond Spread	Japan 10-Year Bond	USD/JPY Spread
8	EUR/USD Spread	U.S. 10-Year SWAP	U.K 10-Year SWAP
9	VIX	U.K. 10-Year Bond Spread	GBPUSDV1M
10	EURUSDV1M Spread	U.S. 10-Year Bond Spread	WTI Crude Oil

Table 4.4: Rank by Shapley Values

In section 3 we have qualitative discussed about the relationship between FX and the selected assets, as well as their potential predictive powers. Interpretability of deep neural networks has introduced in our theoretical framework, now we report the top 10 of our features which have significant contribution on the prediction of our MDN in terms of Shapley values in Table 4.4.

Overall, the government bonds have the most significant contributions to the prediction of the FX rates and are confirmed by the Shapley values. The predictions in GBP/USD are mainly contribute from the EUR/USD and the US 10-Year government bond. Also, the UK 10-Year SWAP and GBP/USD have the most significant contributions to the EUR/USD predictions. The British Pound and the Euro are closely related has been known by investors, the Shapley values confirm their significant predictive powers from one another, despite of the EUR/USD and GBP/USD is nearly uncorrelated in the period of our dataset (see Table B.1.2). Major contributions in GBP/USD predictions also include the 1-month at the money implied volatility of the currency options. Possible explanation is that, institutional investors are the major market participants in the UK, and currency options are one of the major financial tools to hedge against currency risks in their portfolios. Therefore, the three FX options are listed with top ranks in contributing the predictions of GBP/USD in terms of Shapley values. The US 10-Year government bond has dominant contribution to the predictions of USD/JPY. The relationship between the US government bond and the USD/JPY is a well known fact amongst investors due to the popular usage of carry trades strategies by investors in trading the pair USD/JPY under the low interest rate policies of the Bank of Japan. Moreover, it is worth to highlight the fact that the bid-ask spread of certain assets has significant predictive powers. For instance, the EUR/USD bid-ask spread has significant contributions to the GBP/USD prediction. Therefore, the Shapley values justify that the bid-ask spread of the assets should be included in the prediction problem.

Section 5

Applications

With the ability to predict the probability distribution of the log return of the FX pairs, there are a broad range of potential applications by using the density estimated by our Mixture Density Network. Applications in derivative pricing, algorithm trading and risk management are possible. Derivative pricing with mixture density has been studied by Brigo (2002) [8]. He showed that given the marginal density is some mixture density, there exist some drift and diffusion coefficients in a stochastic differential equation (SDE) such that it admits a strong solution, and the diffusion coefficient can be obtained analytically by applying the Fokker-Planck equation. Therefore, applications in derivative pricing, hedging and volatility modelling (see [8] and [7]) are possible once the mixture density is estimated by the Mixture Density Network.

In this thesis, we will mainly focus on application in algorithm trading. We will construct simple trading strategies by generating trading signals based on the finite normal mixture densities predicted by the MDN we have described. The FX pair GBP/USD and its 1-hour mid log return prediction will be used for backtesting and demonstration throughout. In general, the strategy can be applied to other FX pairs. In addition, we will briefly demonstration how to classify regimes with simple classification rule based on the parameters estimated by our Mixture Density Network.

5.1 Algorithm Trading Strategy

Let Y_t be the random variable of the 1-hour mid log return at time t and is estimated mixture density predicted by our MDN by time t, where the density function is given by $p_t(y; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma})$. The probability of the FX rate will rise and decline is given by

$$\tilde{p}_t := \mathbb{P}[Y_t > 0] = \int_0^{+\infty} p_t(y; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = 1 - \mathbb{P}[Y_t \leq 0] \tag{5.1.1}$$

Denote $Z_t \in \{-1, 0, 1\}$ as our buy (1), sell (-1) and hold (0) trading signals at time t which are defined as follows:

$$Z_t := \begin{cases} 1, & \text{if } \tilde{p}_t \geq p_{\text{upper}} \\ -1, & \text{if } \tilde{p}_t \leq p_{\text{lower}} \\ 0 & \text{otherwise} \end{cases} \tag{5.1.2}$$

where $p_{\text{upper}}, p_{\text{lower}} \in [0, 1]$ are some upper and lower probability threshold respectively.

The probability given in 5.1.1 can be easily determined by Proposition 2.3.2 once the $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma}$ are estimated by the MDN. The probability thresholds correspond to the decision rule that if probability of rising FX rate is greater (less) than some p_{upper} (p_{lower}) to generate buy (sell) signals, and hold is the decision of neither buy nor sell. The choices of the thresholds can be chosen manually or through certain schemes of optimization in backtest according to the risk and reward preferences of different investors. In the following backtest, we explore three different sets of values of the threshold

1. Strategy 1: $p_{\text{lower}} = 0.5$, $p_{\text{upper}} = 0.5$
2. Strategy 2: $p_{\text{lower}} = 0$, $p_{\text{upper}} = 0.5$

3. Strategy 3: $p_{\text{lower}} = 0.4$, $p_{\text{upper}} = 0.6$

Strategy 1 corresponds to the long and short position of the FX when it is predicted to have over 50% probability of going up and down. Strategy 2 is a variation of Strategy 1, but short selling is not allowed, which is a long only strategy. Strategy 3 is in between with long and short position are generated when the predicted probability is exceeding 60% of going up and down.

5.1.1 Backtest

Evaluation Metrics

We split the data for backtesting into training and validation datasetas described in Section 3. The Profit and Loss (PnL) at time t of our strategy is expressed in terms of the trading signal Z_t and also the transaction cost as following

$$\text{PnL}_t := (S_t - S_{t-60})Z_{t-60} - \text{tc} \tag{5.1.3}$$

where S_t is the mid FX rate at time t measure in one minute interval. The PnL we have defined implicitly implies buying or selling unit amount of the currency whenever trading signals are generated. Also, the cumulative PnL is given by summing the PnL over all time

$$\text{Cumulative PnL} := \sum_t \text{PnL}_t$$

Institutional investors can often gain access to 1 pip constant spread for trading liquid FX pairs, hence the transaction cost tc in equation 5.1.3 is constant over time, and we set tc = 0.0001 for GBP/USD.

Sharpe ratio (SR) and maximum drawdown (MDD) are often used as metrics for evaluation of trading strategies, the definitions are given as follows

Definition 5.1.1. (Sharpe Ratio) Let r_f be the risk-free rate, R_s and σ_s be the mean and standard deviation of the strategy return, then the Sharpe ratio (SR) is given by

$$\text{SR} := \frac{R_s - r_f}{\sigma_s}$$

Definition 5.1.2. (Maximum Drawdown) Let T be the time horizon our strategy to consider. Let the cumulative PnL at time $t \in [0, T]$ be R_t^C, and define $M_t := \max_{0 \le u \le t} R_u^C$ be the running maximum of the cumulative PnL up to time t, then the maximum drawdown (MDD) over the time horizon T is given by

$$\text{MDD} := -\max_{t \in [0,T]} (M_t - R_t^C)$$

Moreover we can define the absolute of maximum drawdown so that the larger its value, the more undesirable of the strategy.

$$\text{AMDD} := |\text{MDD}|$$

Apart from using the Sharpe ratio and maximum drawdown, one may interest in the decisions of the strategy in consecutive time steps. Notice we can treat the trading signal Z_t takes discrete values from finite state $\mathcal{S} = \{-1, 0, 1\}$. For any state $i, j \in \mathcal{S}$, one may consider the one-step transition matrix $P = (P)_{ij} \in \mathbb{R}^{3\times 3}$ such that

$$P_{ij} := \mathbb{P}[Z_{t+1} = j | Z_t = i]$$

For the time horizon of our strategy T, one can estimate P_{ij} for any states i, j by

$$\hat{P}_{ij} := \frac{1}{T-1} \sum_{t=0}^{T-1} \mathbb{1}_{\{Z_{t+1}=j | Z_t = i\}}$$

and by weak law of large number $\hat{P}_{ij}$ converge in probability to P_{ij} given sufficiently large T.

We backtest our trading strategy in two different ways. Firstly, we implement the strategy generating trading signals continuously over 24 hours whenever the FX market is open. Secondly, certain hours are selected and we only implement our trading strategies within the specific range of hours we have chosen.

Trading Without Hours Specification

We backtest our strategy by trading continuously every one minute according to our trading signals over 24 hours a day. In Figure 5.1, we backtest our strategy on our training dataset (December 2020 - May 2021) with several particular values of p_{upper} and p_{lower} and their cumulative PnLs are illustrated.

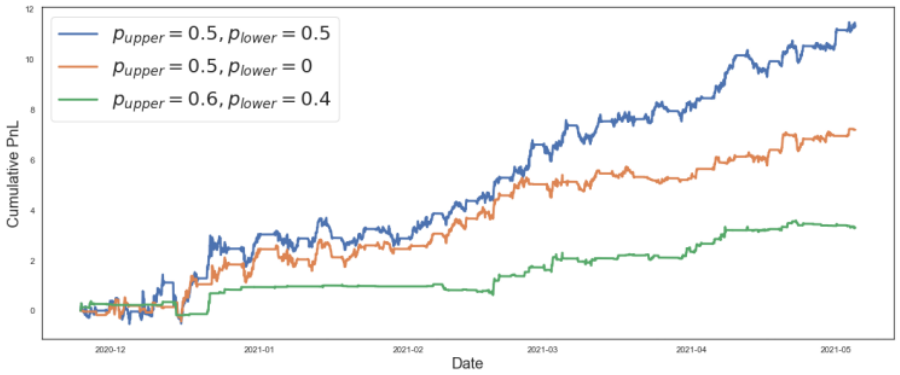

Figure 5.1: Trading Without Hours Specification: Cumulative PnL

We backtest the three strategies and their cumulative PnLs are illustrated in Figure 5.1. We can observe the cumulative PnL of Strategy 1 is greater than Strategy 2, this is because the former is able to profit from declining by short selling. The cumulative PnL of Strategy 3 grows relatively slow, it is because the probability thresholds are comparatively conservative, long or short signals will only be triggered if the predicted probability of increase or decrease is significant. Hence, one may inspect the flat line indicates a long period of rarely buy or sell.

We take the 1-Month U.S. Treasury rate as a proxy of the risk-free rate, and report our strategies performances for both training and validation data, with daily Sharpe ratio and absolute value of maximum drawdown in Table 5.1

	Training		Validation	
Strategy	Daily SR	AMDD	Daily SR	AMDD
1	0.2064	1.7863	0.1053	0.8984
2	0.1848	1.0920	0.1808	0.7406
3	0.1477	0.5605	-0.0425	0.7475

Table 5.1: Trading Without Hours Specification: Performance Metrics

In general, a good strategy should have large Sharpe ratio and small AMDD. By Table 5.1, Strategy 2 has better performances amongst the three strategies, whilst Strategy 3 has the worst performances. Our trading strategies can be considered as a type of momentum trading strategy, which buys the FX when they are rising and sell them when they are declining. If the trading signals at current time t is $i \in \{-1, 0, 1\}$, then the signal generates in the next minute is very likely, with probability $P_{ii} \approx 1$, to be i as well (see Appendix B.5)

Trading With Hours Specification

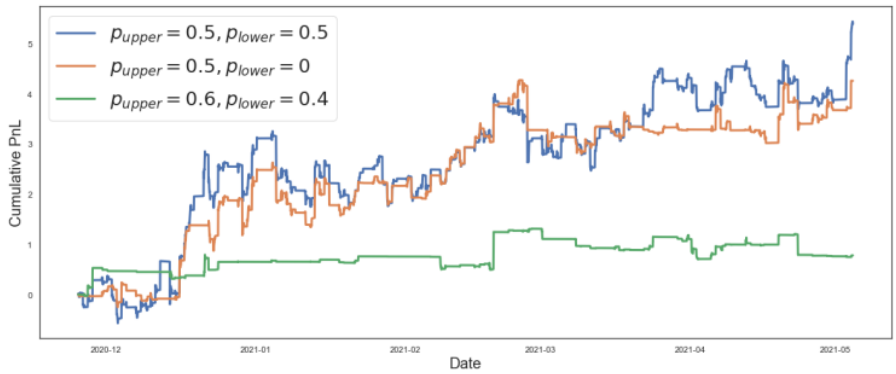

Figure 5.2: Trading With Hours Specification: Cumulative PnL

In order to improve our trading strategies, we propose to trade the FX in specific hours rather than continuously trading over 24 hours, then it comes to the question on choosing the range of specific hours. Recall that the performances of our neural network varies in different hours due to the changing in volatility as illustrated in Figures 3.6 and 4.2. In the spirit of that, we can try to avoid hours with high prediction errors, and only trade within hours that the neural network performs better. By inspection on Figure 4.2, we decide to devise our trading strategies from 21:00-7:00 (UTC), which avoid the London Session and the overlapping of the London and New York Session. Therefore, our strategies only start to trade from 21:00 (UTC), and closes all outstanding positions by 7:00 (UTC) in the next trading day. Figure 5.2 shows the cumulative PnL just trading within 21:00-7:00 (UTC) each day. Notice that the cumulative PnL of Strategy 2 is more closer to Strategy 1 than its cumulative PnL during continuous trading as shown in Figure 5.1. This implies there are more short positions during the trading hours we avoid. Similar to the analysis perform in continuous trading, we report the daily Sharpe ratio, AMDD to evaluate the performances.

	Training		Validation	
Strategy	Daily SR	AMDD	Daily SR	AMDD
1	0.1727	1.5434	0.3030	0.6786
2	0.1778	1.4636	0.3165	0.5864
3	0.0429	0.6115	0.1433	0.8672

Table 5.2: Trading With Hours Specification: Performance Metrics

From Table 5.2, we can notice the performances in the validation dataset significantly improve for both the Sharpe ratio and AMDD. These show the robustness of our Mixture Density Network in the validation as well as the stability of our trading strategies in the specific hours. Lastly, the transition matrices are reported similarly in Figures B.6 and B.7.

Relationship with Technical Indicator

As we discussed that our strategy is a type of momentum strategy, and some FX traders often look at technical indicators that are relevant to the momentum of the FX rates. One of the most popular momentum technical indicators that FX traders often use as trading signal is the moving average convergence divergence (MACD). Since our strategy uses the log return as the variable to generate trading signals, we will explore their mathematical relationship and how they are relevant to each other. To begin with MACD, we first define the exponential moving average (EMA).

Definition 5.1.3. (Exponential Moving Average) Suppose we have discrete time steps $\cdots < t_{-1} < t_0 < t_1 < \dots$. Denote S_{t_k} be the price at time t_k. The λ-period exponential moving average (EMA) at time t is defined by the following recursive formula:

$$\begin{aligned} &\text{EMA}_\lambda(t_0) = S_{t_0} \\ &\text{EMA}_\lambda(t_k) = \alpha S_{t_k} + (1-\alpha)\text{EMA}_\lambda(t_{k-1}), \qquad \text{for } k \geq 1 \end{aligned}$$

where α is the weighted multiplier, which in practice is given by $\alpha := \frac{2}{1+\lambda}$.

Definition 5.1.4. (Moving Average Convergence Divergence) The moving average convergence divergence is defined as the difference between 12-period and 26-period exponential moving average. At any discrete time t, the MACD is given by

$$\text{MACD}(t) = \text{EMA}_{12}(t) - \text{EMA}_{26}(t)$$

By the definition of exponential moving average (EMA), we can apply recursively to express the EMA as an infinite weighted sum of the prices

$$\text{EMA}_\lambda(t_k) = \sum_{i=0}^{\infty} \alpha(1-\alpha)^i S_{t_{k-i}} \tag{5.1.4}$$

The original idea of the MACD technical indicator is to measure whether there is a shift in momentum, and FX traders often look at whether the MACD is greater (smaller) than some threshold to indicate a buy (sell) signal. Traditional technical analysis set the 9-period EMA (EMA_9) as the threshold, and it is known as the signal-line. We propose the following proposition to characterise how our trading strategy using the probability of log return greater or less than some thresholds relates to trading strategies using the MACD.

Proposition 5.1.5. *(Characterisation of Log Return and MACD)*
Denote $\lambda_1 = 12$, $\lambda_2 = 26$, and $\alpha_1 = \frac{2}{1+\lambda_1}$, $\alpha_2 = \frac{2}{1+\lambda_2}$. Let G_k be the cdf of the m-component finite normal mixture density of the random variable $\log(S_{t_k}/S_{t_{k-1}})$, realized by the MDN. Then for any $\overline{x}$, we have the following:

$$\mathbb{P}_{k-1}[MACD(t_k) > \overline{x}] = 1 - G_k(\eta_{k-1} + \log(\overline{x} + C_{k-1})) \tag{5.1.5}$$

where $\mathbb{P}_{k-1}$ is the probability given all information up to time t_{k-1} are known, and η_{k-1}, C_{k-1} are given by

$$\begin{aligned} \eta_{k-1} &:= -\log(S_{t_{k-1}}(\alpha_1 - \alpha_2)) \\ C_{k-1} &:= (1-\alpha_2)EMA_{\lambda_2}(t_{k-1}) - (1-\alpha_1)EMA_{\lambda_1}(t_{k-1}) \end{aligned}$$

In particular, consider the case $\overline{x} = x^*$, where $x^* := e^{-\eta_{k-1}} - C_{k-1}$, then

$$\mathbb{P}_{k-1}[\log(\frac{S_{t_k}}{S_{t_{k-1}}}) > 0] = \mathbb{P}_{k-1}[MACD(t_k) > x^*] \tag{5.1.6}$$

Furthermore, x^* can be expressed as follows:

$$x^* = \sum_{i=1}^{\infty} \psi_i S_{t_{k-i}} \tag{5.1.7}$$

$$\psi_i := \begin{cases} (\alpha_1 - \alpha_2)(2 - \alpha_1 - \alpha_2), & i = 1 \\ \alpha_1(1-\alpha_1)^i - \alpha_2(1-\alpha_2)^i, & i \geq 2 \end{cases} \tag{5.1.8}$$

Proof. See Appendix A.3 □

With the above characterisations, the probability of the log return greater or less than 0 is in fact equivalent to the probability of the MACD greater or less than some x^*. Moreover, one can use equation 5.1.5 to determine the probability of the MACD crossing above or below the signal line by setting $\overline{x} = \text{EMA}_9$, which can be used to examine the probability of the shift in momentum from the perspective of technical analysis.

Further Discussion

In the previous we have discussed the applications of mixture density in algorithm trading. The trading strategies described can be further optimized that is worth to highlight but beyond our main theme in this thesis, and will leave the implementations as future research.

The trading strategy can be optimized in two ways, the probability thresholds and the weight allocation on the FX. The optimal probability thresholds p^*_{lower} and p^*_{upper} can be determined through optimization with respect to certain objective function. The most popular objective function is the Sharpe ratio, however, as one can see in Table 5.1, strategies can have promising Sharpe ratio overall, but sharp losses might be problematic. The Sharpe ratio only measures the first two moments of the PnL, it ignores the non-normality characteristics and does not consider statistics such as the drawdown, skewness, kurtosis etc. One can define alternative performance measure by considering both the Sharpe ratio and maximum drawdown similar to the mean variance optimization as follows:

$$\delta_\lambda = \text{Daily SR} - \lambda|\text{MDD}|$$

where λ is the risk aversion level preference of investors, and p^*_{lower} and p^*_{upper} can be determined by maximising the δ_λ.
Another consideration is the optimal weight allocation to trade whenever trading signals are triggered. In order to compute the optimal weight, one can apply utility optimization. A utility $U : \mathbb{R} \to [-\infty, \infty)$ is a non-decreasing function, which quantifies one satisfactory or happiness level of consumption. Often the utility function is chosen as a concave function to model the risk-aversion behaviour of investors. To determine the optimal allocation on the FX to be traded, we can choose the constant absolute risk aversion (CARA) as our utility, where fix $\gamma > 0$, the CARA_γ takes the form

$$U(x) = \text{CARA}_\gamma(x) = -\exp(-\gamma x)$$

Consider a one-period optimization setting, with random variables Y_0 be the current log return, and Y_1 be the log return of the next time frame. As we predict the random variable Y_1 admits a m-component finite normal mixture density $p(y; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{i=1}^m \lambda_i \phi(y; \mu_i, \sigma_i)$. Let $\theta \in \mathbb{R}$ be the weight that are desired to optimised. One can maximise the expected utility $\mathbb{E}[U(\theta Y_1)]$ subject to the initial condition θY_0. By considering the Lagrangian function with the Lagrange multiplier α, we get (see Appendix A.4)

$$\mathcal{L}(\theta, \alpha) = -\sum_{i=1}^m \lambda_i \exp(-\gamma\theta\mu_i + \frac{\gamma^2\theta^2\sigma_i^2}{2}) - \alpha\theta Y_0 \tag{5.1.9}$$

Hence, it remains to solve $\frac{\partial \mathcal{L}}{\partial \theta} = 0$ and $\frac{\partial \mathcal{L}}{\partial \alpha} = 0$ numerically to obtain the optimal allocation θ^*.

5.2 Regime Classification

In this short session, we present another application using the mixture density of the 1-hour log return estimated by our MDN. We propose a simple classification rule by using the estimated weight parameters λ to classify the regime which the prices are belonged to. Intuitively, when there is an increased in λ_i, then the i-th component density increases its impact on the overall distribution. Moreover, recall that the weights $\lambda_1(\boldsymbol{x}_t), \lambda_2(\boldsymbol{x}_t)$ are the approximated stationary probability (condition on $\boldsymbol{x}_t$) of being in the i-th regime at time t and the reciprocal of the mean recurrence time of the i-th regime. Thus an increase in $\lambda_i(\boldsymbol{x}_t)$ is equivalent to a rise in the probability of being in the i-th regime and a decrease in the mean time of first revisit the i-th regime. Denote the empirical mean of the weight parameters as $\overline{\lambda}_1, \overline{\lambda}_2$, to be the representative of the Regime1 and Regime2 respectively, if $\lambda_1(\boldsymbol{x}_t) \geq \overline{\lambda}_1$, then Regime1 is labeled at time t, otherwise Regime2 is labeled. Figures 5.3 labeled the FX rates time series into Regime1 (blue) and Regime2 (orange).

The proposed classification scheme is able to label regimes which are interpretable through sub-period analysis. For instances, Regime2 is labelled throughout the beginning of the GBP/USD realised time series. It corresponds to the volatile period from December 2020 to January 2021 which we have seen its high realised volatility in Figure 3.5. Furthermore, one can label unseen data with appropriate regime once the parameters of the mixture density are estimated by the Mixture Density Network. This provides tractable and interpretable metrics to measure how likely the current FX rate belongs to different regimes.

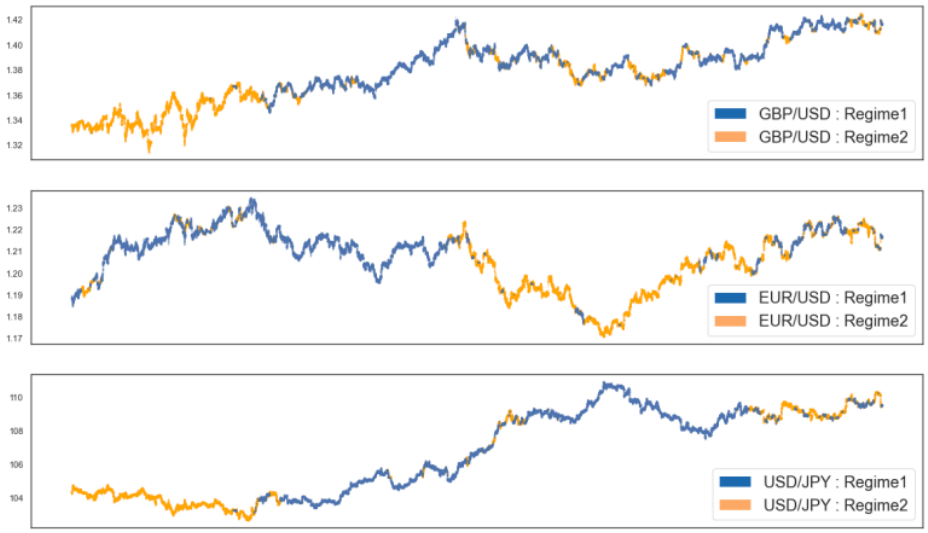

Figure 5.3: Regime Classification

Section 6

Conclusion and Future Research

In this paper, our main objective is to predict the probability density function of FX rates log return with Mixture Density Network by using prices and bid-ask spread of multiple assets as our input features. Throughout, we have shown the ability of mixture density to model non-normality of the returns and demonstrated interpretation of the component densities correspond to different market regimes with Hidden Markov Model. The properties of different market regimes are able to be captured by our Mixture Density Network and showed consistency with previous studies. Shapley values from game theory is adopted as global interpretability of our neural network to compare the marginal contribution of different assets in the prediction. Furthermore, we have extended beyond the prediction task and developed trading strategies by inferring the probability of having positive and negative returns, and also highlighted the significance of considering trading sessions in strategy implementations. We demonstrated the rich information we can gain from the density estimated by the Mixture Density Network, and its robustness to apply in a broad range of financial applications. These show the area of distributional prediction of financial time series is worth to be further established in future research and applications.

Firstly, future work can enhance our current framework in several aspects. Larger training dataset with a wider historical window can be used to train the neural networks to adapt the financial market more well rounded. This enables to predict the FX rates for a longer time horizon, and capable to have applications in derivatives trading, hedging etc. As discussed before, the number of component density selected can be optimized by applying existing regime classification algorithms to determine the number of regimes exist in the FX market. We can also choose alternative distribution family as the component density to capture heavier tail than the normal components we have chosen, such as the t-distribution, Pareto distribution etc. Moreover, our Mixture Density Network only predicts univariate probability density function, which can be extended to multivariate version to consider the joint distribution of the assets.

Future research in regimes classification by using Mixture Density Network is worth to be explored. Possible extensions can be done on multivariate regime classification by inferring the joint distribution of the assets and development of regime-switching trading strategies which are able to adapt optimal trading strategies under different regimes. We will leave these extensions for future justifications.

Lastly, one should notice global interpretability of neural network using Shapley values only explain the marginal contributions of each features in the task of predictions, but ignoring causal relationship between the chosen features and the target variables. The financial market is complicated in a way that there is no one-way causality, but instead assets are constantly influencing and interacting with one another. In this paper, input features are chosen manually through qualitative analysis based on subjective domain knowledge. An ideal predictive *agent* should be able to identify assets from a diversify set of assets with the highest level of causality to the targeted assets we desire to predict. This hints the possibility of applying reinforcement learning techniques in choosing the optimal *actions* that maximise future rewards. Thus, input features can be chosen with more sophisticated techniques in future works.

Nevertheless, substantial research in predictive model should be expected in the future to have deeper understanding about the complex structure and the behaviour in the financial market.

Appendix A

Supplementary Proofs

A.1 Statistical Properties of Mixture Density

Proof. The proofs of proposition 2.3.2 are given as follows:

Equation 2.3.1 and 2.3.2 is clear by linearity i.e.

$$F_X(x) = \int_{-\infty}^{x} \sum_{i=1}^{m} \lambda_i \phi_i(x; \mu_i, \sigma_i) dx = \sum_{i=1}^{m} \lambda_i \int_{-\infty}^{x} \phi_i(x; \mu_i, \sigma_i) dx$$
$$\mathbb{E}[f(X)] = \int_{-\infty}^{+\infty} f(x) \sum_{i=1}^{m} \lambda_i \phi_i(x; \mu_i, \sigma_i) dx = \sum_{i=1}^{m} \lambda_i \int_{-\infty}^{+\infty} f(x) \phi_i(x; \mu_i, \sigma_i) dx$$

By taking $f(y) = y^k$ to obtain equation 2.3.3, then it is clear that $\mu = \sum_{i=1}^{m} \lambda_i \mu_i$ and equation 2.3.4 can be determined by the following

$$\begin{aligned}
\mathbb{E}[(X-\mu)^k] &= \sum_{i=1}^{m} \lambda_i \mathbb{E}[(X_i - \mu_i + \mu_i - \mu)^k] \\
&= \sum_{i=1}^{m} \lambda_i \mathbb{E}[\sum_{j=0}^{k} \binom{k}{j} (X_i - \mu_i)^j (\mu_i - \mu)^{k-j}] \\
&= \sum_{i=1}^{m} \sum_{j=0}^{k} \lambda_i \sigma_i^j \binom{k}{j} (\mu_i - \mu)^{k-j} \mathbb{E}[(\frac{X_i - \mu_i}{\sigma_i})^j] \\
&= \sum_{i=1}^{m} \sum_{\substack{0 \le j \le k \\ j \text{ is even}}} \lambda_i \sigma_i^j \binom{k}{j} (\mu_i - \mu)^{k-j} \frac{j!}{2^{\frac{j}{2}} (\frac{j}{2})!} \\
&= \sum_{i=1}^{m} \sum_{\substack{0 \le j \le k \\ j \text{ is even}}} \lambda_i \sigma_i^j (\mu_i - \mu)^{k-j} \frac{k!}{2^{\frac{j}{2}} (k-j)! (\frac{j}{2})!}
\end{aligned}$$

in the second last equality we have used the fact that for $Z \sim \mathcal{N}(0,1)$, then $\mathbb{E}[Z^j] = \frac{j!}{2^{\frac{j}{2}}(\frac{j}{2})!}$ if j is even and 0 whenever j is odd. □

A.2 Stochastic Differential Equations and Mixture Density

The flexibility of normal mixture density can also establish relationship with stochastic differential equations (SDEs). Let Y_t admits a finite normal mixture density $p_{Y_t}(y) = \sum_{i=1}^{m} \lambda_i \phi(y; \mu_i, \sigma_i)$. Brigo (2002) [8, Proposition 1.2 and Theorem 2.1, Page 4-7] have shown the existence for some SDE with drift and diffusion coefficient f and σ^f admits a unique strong solution and its solution has as the marginal density the final normal mixture density $p_{Y_t}(y)$, and the SDE is given by

$$dY_t = f_t(Y_t)dt + \sigma_t^f(Y_t)dW_t$$

where $(W_t)_{t\geq 0}$ is a standard Brownian motion. In the original proof, the Fokker-Planck equation is applied and one is able to derive analytical form of the diffusion coefficient σ^f. We do not go through technical details, but strongly recommend interested readers refer to [8] for further details. If we let Y_t be the true log return, and $\hat{Y}_t$ be the predicted log return which admits a finite normal mixture density $p_{\hat{Y}_t}(y)$, then one can write

$$dY_t = f_t(Y_t)dt + \sigma_t^f(Y_t)dW_t \tag{A.2.1}$$

$$d\hat{Y}_t = \hat{f}_t(\hat{Y}_t)dt + \hat{\sigma}_t^{\hat{f}}(\hat{Y}_t)d\hat{W}_t \tag{A.2.2}$$

$$\hat{Y}_0 = Y_0, \quad d[W, \hat{W}]_t = \rho dt \tag{A.2.3}$$

where $(W_t)_{t\geq 0}$ and $(\hat{W}_t)_{t\geq 0}$ are standard Brownian motions in probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and $\rho \in [-1, 1]$. One can establish relationship between mean square error and the diffusion coefficients of the SDEs by the following

Proposition A.2.1 (Mean Square Error and Diffusion Coefficient)**.** *Assume A.2.1, A.2.2 and A.2.3, then the expected value of the square error process* $\{(Y_t - \hat{Y}_t)^2\}_{t\geq 0}$ *is given by*

$$\mathbb{E}[(Y_t - \hat{Y}_t)^2] = \int_0^t \mathbb{E}[(\sigma_s^f(Y_s) - \hat{\sigma}_s^{\hat{f}}(\hat{Y}_s))^2]ds + 2(1-\rho)\int_0^t \mathbb{E}[\sigma_s^f(Y_s)\hat{\sigma}_s^{\hat{f}}(\hat{Y}_s)]ds \tag{A.2.4}$$

In particular, consider the special case $\rho = 1$*, then the expected square error process given in equation A.2.4 achieves a minimum and is given by*

$$\mathbb{E}[(Y_t - \hat{Y}_t)^2] = \int_0^t \mathbb{E}[(\sigma_s^f(Y_s) - \hat{\sigma}_s^{\hat{f}}(\hat{Y}_s))^2]ds \tag{A.2.5}$$

Proof. It is straightforward by applying Itô's formula to $(Y_t - \hat{Y}_t)^2$

$$\begin{aligned} d(Y_t^2) &= 2Y_t dY_t + \sigma_t^f(Y_t)^2 dt \\ d(\hat{Y}_t^2) &= 2\hat{Y}_t d\hat{Y}_t + \hat{\sigma}_t^{\hat{f}}(\hat{Y}_t)^2 dt \\ d(2Y_t\hat{Y}_t) &= 2\hat{Y}_t dY_t + 2Y_t d\hat{Y}_t + 2\rho\sigma_t^f(Y_t)^2\hat{\sigma}_t^{\hat{f}}(\hat{Y}_t)^2 dt \end{aligned}$$

then by linearity we can get the integral form as follows

$$(Y_t - \hat{Y}_t)^2 = (Y_0 - \hat{Y}_0)^2 + \int_0^t (\sigma_s^f(Y_s) - \hat{\sigma}_s^{\hat{f}}(\hat{Y}_s))^2 ds + 2(1-\rho)\int_0^t \sigma_s^f(Y_s)\hat{\sigma}_s^{\hat{f}}(\hat{Y}_s)ds$$

Finally taking expectation on both sides and applying Fubini theorem to interchange the expectation and integral to obtain equation A.2.4. Moreover, equation A.2.5 follows immediately by the fact that $\rho \leq 1$. □

In our context relevant to Mixture Density Network, proposition A.2.1 tells us that for any robust normal mixture density predictions on the log return using the MDN has to capture the trend of the true process $\rho \approx 1$, also MDN not only are able to minimize the mean square error on the conditional expectation of point estimation, and also able to best approximate the diffusion coefficient of the true underlying stochastic volatility model, and this indicates the source of realised prediction errors comes from the deviation between the predicted and the realised volatility processes.

A.3 Characterisation of Log Return and MACD

In order to obtain equation 5.1.5, one have to first derive an expression of log return in terms of the MACD. By expanding out the EMA terms in the MACD, one will obtain the following:

$$\begin{aligned}
\text{MACD}(t_k) &= \text{EMA}_{\lambda_1}(t_k) - \text{EMA}_{\lambda_2}(t_k) \\
&= [\alpha_1 S_{t_k} + (1-\alpha_1)\text{EMA}_{\lambda_1}(t_{k-1})] - [\alpha_2 S_{t_k} + (1-\alpha_2)\text{EMA}_{\lambda_2}(t_{k-1})] \\
S_{t_k} &= \frac{1}{\alpha_1 - \alpha_2}[\text{MACD}(t_k) + (1-\alpha_2)\text{EMA}_{\lambda_2}(t_{k-1}) - (1-\alpha_1)\text{EMA}_{\lambda_1}(t_{k-1})] \\
\log(\frac{S_{t_k}}{S_{t_{k-1}}}) &= -\log(S_{t_{k-1}}(\alpha_1 - \alpha_2)) + \log[\text{MACD}(t_k) + (1-\alpha_2)\text{EMA}_{\lambda_2}(t_{k-1}) - (1-\alpha_1)\text{EMA}_{\lambda_1}(t_{k-1})] \\
&= \eta_{k-1} + \log[\text{MACD}(t_k) + C_{k-1}]
\end{aligned}$$

Therefore, it is clear that equation 5.1.5 is straightforward from the above expression

$$\begin{aligned}
\mathbb{P}_{k-1}[\text{MACD}(t_k) \leq \overline{x}] &= \mathbb{P}_{k-1}[\log(\frac{S_{t_k}}{S_{t_{k-1}}}) \leq \eta_{k-1} + \log(\overline{x} + C_{k-1})] \\
&= G_k(\eta_{k-1} + \log(\overline{x} + C_{k-1}))
\end{aligned}$$

Next, we are going to derive the expression $x^* = e^{-\eta_{k-1}} - C_{k-1}$ stated in equation 5.1.6, by writing the EMAs as infinite summation

$$\begin{aligned}
e^{-\eta_{k-1}} - C_{k-1} &= S_{t_{k-1}}(\alpha_1 - \alpha_2) + (1-\alpha_1)\alpha_1 \sum_{i=1}^{\infty} S_{t_{k-i}}(1-\alpha_1)^{i-1} - (1-\alpha_2)\alpha_2 \sum_{i=1}^{\infty} S_{t_{k-i}}(1-\alpha_2)^{i-1} \\
&= S_{t_{k-1}}[2(\alpha_1 - \alpha_2) - (\alpha_1^2 - \alpha_2^2)] + \sum_{i=2}^{\infty} S_{t_{k-i}}(\alpha_1(1-\alpha_1)^i - \alpha_2(1-\alpha_2)^i) \\
&= \sum_{i=1}^{\infty} \psi_i S_{t_{k-i}}
\end{aligned}$$

where the coefficients ψ_i is given by equation 5.1.8

A.4 Weight Optimization Lagrangian function

To obtain the expression in equation 5.1.9, it remains to to express the expectation of the CARA_γ with the log return is given by the m-component finite normal mixture as follows:

$$\begin{aligned}
\mathbb{E}[U(\theta Y_1)] &= \mathbb{E}[-\exp(-\gamma\theta Y_1)] \\
&= -\int_{-\infty}^{+\infty} \exp(-\gamma\theta y) p(y; \boldsymbol{\mu}, \boldsymbol{\sigma}) dy \\
&= -\int_{\infty}^{+\infty} \exp(-\gamma\theta y) \sum_{i=1}^{m} \lambda_i \phi(y; \mu_i, \sigma_i) dy \\
&= -\sum_{i=1}^{m} \lambda_i \int_{-\infty}^{+\infty} \exp(-\gamma\theta y) \phi(y; \mu_i, \sigma_i) dy \\
&= -\sum_{i=1}^{m} \lambda_i \exp(-\gamma\theta\mu_i + \frac{\gamma^2\theta^2\sigma_i^2}{2})
\end{aligned}$$

The last equality is by the moment generating function (MGF) of normal distribution with mean μ and variance σ^2, which has the known form $\text{MGF}(t) = \exp(\mu t + \sigma^2 t^2/2)$

Appendix B

Supplementary Tables and Diagrams

B.1 Data Description

This section provides the descriptions of the Bloomberg tickers and also numerical values of correlation coefficients across assets in Section 3

B.1.1 Bloomberg Tickers Descriptions

Ticker	Description
GBP/USD	Exchange rate of the British Pound against the US dollar
EUR/USD	Exchange rate of the Euro against the US dollar
USD/JPY	Exchange rate of the Yen against the US dollar
GT10 Govt	US 10-Year Treasury Yield
GTGBP10Y Govt	UK 10-Year Gilt Yield
GTJPY10Y Govt	Japan 10-Year Government Bond Yield
SPX	S&P 500 Index
VIX	Chicago Board Options Exchange Volatility Index
NKY	Japan Nikkei 225
GBPUSDV1M	GBP/USD 1-month at the money (ATM) option implied volatility
EURUSDV1M	EUR/USD 1-month at the money (ATM) option implied volatility
USDJPYV1M	USD/JPY 1-month at the money (ATM) option implied volatility
USSW10	US 10-Year interest rate SWAP
BPSW10	UK 10-Year interest rate SWAP
CL1 Comdty	Generic Crude Oil Futures
XAU	Gold Spot price in US dollar

Table B.1: Bloomberg Tickers Descriptions

B.1.2 Correlation Matrix

	GBP/USD	EUR/USD	USD/JPY	GBPUSDV1M	EURUSDV1M	USDJPYV1M	USGG10YR	GIGB10	GJGB10	USSW10	BPSW10	NKY	SPX	VIX	CL1 Comdty	XAU
GBP/USD																
EUR/USD	0.023															
USD/JPY	0.7	-0.54														
GBPUSDV1M	-0.76	0.21	-0.59													
EURUSDV1M	-0.6	0.29	-0.59	0.8												
USDJPYV1M	-0.19	-0.11	-0.041	0.46	0.66											
USGG10YR	0.79	-0.46	0.96	-0.68	-0.61	-0.035										
GTGB10	-0.85	0.39	-0.95	0.69	0.65	0.08	-0.98									
GIGB10	0.75	-0.45	0.75	-0.64	-0.54	0.059	0.83	-0.85								
USSW10	0.79	-0.49	0.95	-0.7	-0.63	-0.034	0.99	-0.97	0.87							
BPSW10	0.83	-0.42	0.95	-0.68	-0.64	-0.065	0.98	-1	0.86	0.97						
NKY	0.68	-0.39	0.6	-0.76	-0.64	-0.16	0.72	-0.7	0.83	0.77	0.7					
SPX	0.81	-0.12	0.83	-0.74	-0.74	-0.34	0.85	-0.87	0.61	0.83	0.85	0.62				
VIX	-0.29	0.23	-0.55	0.34	0.47	0.22	-0.49	0.5	-0.22	-0.46	-0.49	-0.31	-0.65			
CL1 Comdty	0.91	-0.21	0.85	-0.76	-0.65	-0.14	0.91	-0.94	0.84	0.92	0.93	0.76	0.86	-0.39		
XAU	-0.14	0.86	-0.54	0.23	0.28	-0.21	-0.55	0.51	-0.66	-0.6	-0.53	-0.55	-0.18	0.16	-0.39	

Correlation Matrix

B.2 Mean Square Error

Given the mixture density is estimated by the Mixture Density Network, it is able to compute the conditional mean of the mixture density and evaluate the performances between the predicted mean and realised FX rates log return by using mean square error.

Hour(s)	1	3	8
GBP/USD	2.18×10^{-6}	6.00×10^{-6}	15.0×10^{-6}
EUR/USD	1.78×10^{-6}	2.82×10^{-6}	8.00×10^{-6}
USD/JPY	1.90×10^{-2}	2.85×10^{-2}	5.36×10^{-2}

Table B.2: Empirical Mean Square Error

B.3 Mixture Density Network Estimated Parameters

We compute the empirical mean of the parameters of the mixture density estimated by the Mixture Density Network for 3-hour and 8-hour mid log return for the three FX pairs. The results are reported in the following two tables:

3 Hours	λ_1	λ_2	μ_1	μ_2	σ_1	σ_2
GBP/USD	0.7492	0.2508	2.16×10^{-5}	-2.56×10^{-5}	1.29×10^{-3}	2.10×10^{-3}
EUR/USD	0.7333	0.2667	1.63×10^{-4}	-1.94×10^{-4}	0.991×10^{-3}	1.84×10^{-3}
USD/JPY	0.9930	0.0070	9.14×10^{-4}	-3.85×10^{-3}	1.25×10^{-3}	10.5×10^{-3}

Table B.3: Empirical Mean of Predicted 3-Hour Log Return Mixture Density Parameters

8 Hours	λ_1	λ_2	μ_1	μ_2	σ_1	σ_2
GBP/USD	0.7823	0.2177	2.94×10^{-4}	-6.50×10^{-4}	2.09×10^{-3}	2.49×10^{-3}
EUR/USD	0.9989	0.0011	-1.10×10^{-4}	-82.3×10^{-4}	1.62×10^{-3}	12.0×10^{-3}
USD/JPY	0.5144	0.4856	1.22×10^{-3}	0.931×10^{-3}	1.32×10^{-3}	2.17×10^{-3}

Table B.4: Empirical Mean of Predicted 8-Hour Log Return Mixture Density Parameters

B.4 Shapley Values Diagrams

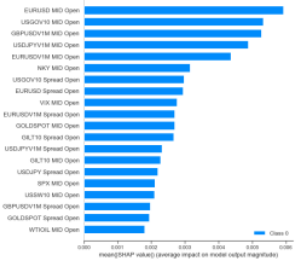

Figure B.1: Shapley Values: GBP/USD

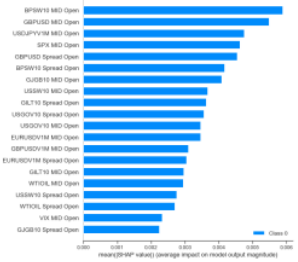

Figure B.2: Shapley Values: EUR/USD

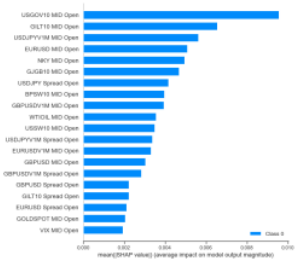

Figure B.3: Shapley Values: USD/JPY

B.5 Trading Signals Transition Matrices

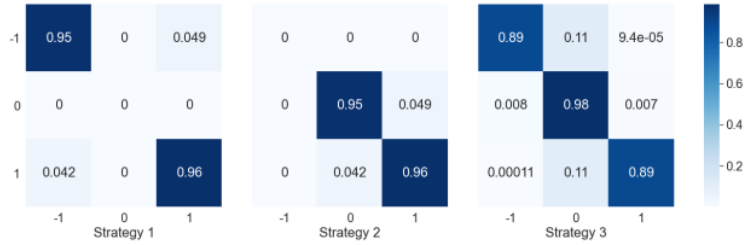

Figure B.4: Trading Without Hours Specification: Transition Matrices of Training Data

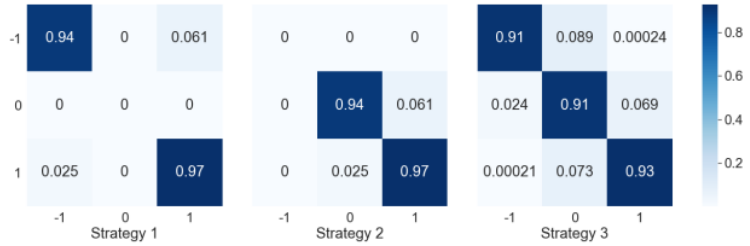

Figure B.5: Trading Without Hours Specification: Transition Matrices of Validation Data

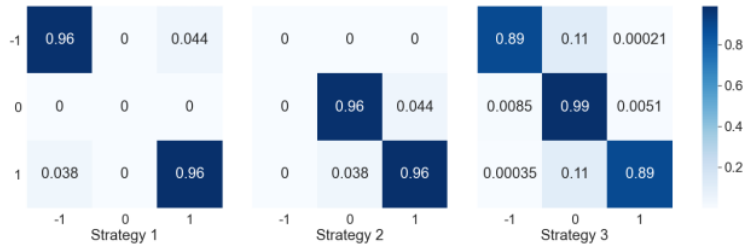

Figure B.6: Trading With Hours Specification: Transition Matrices of Training Data

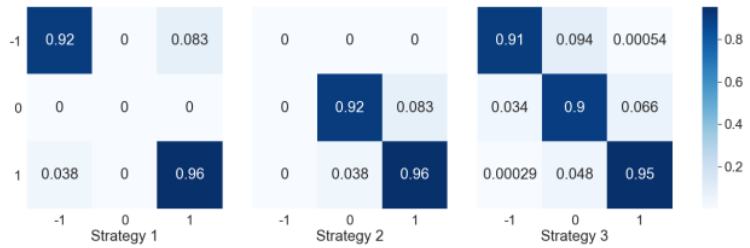

Figure B.7: Trading With Hours Specification: Transition Matrices of Validation Data

Bibliography

[1] C. Alexander and S. Narayanan. Option pricing with normal mixture returns: Modelling excess kurtosis and uncertainty in volatility. ICMA Centre Discussion Papers in Finance icma-dp2001-10, Henley Business School, University of Reading, 2001.

[2] R. Assaf. HIGH FREQUENCY FOREIGN EXCHANGE RATES FORECASTING USING MACHINE LEARNING: A micro-structure approach. Master's thesis, Imperial College London, 2019.

[3] Ole E. Barndorff-Nielsen and Neil Shephard. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(2):253–280, 2002.

[4] J. Berhane. Zero-inflated hidden markov models and optimal trading strategies in high-frequency foreign exchange trading. Master's thesis.

[5] Paul Bilokon, Antoine Jacquier, and Conor McIndoe. Market regime classification with signatures, 2021.

[6] Christopher M. Bishop. Mixture density networks. 1994.

[7] D. Brigo, F. Rapisarda, and Abir Sridi. The arbitrage-free multivariate mixture dynamics model: Consistent single-assets and index volatility smiles. *ERN: Asset Pricing Models (Topic)*, 2014.

[8] Damiano Brigo. The general mixture-diffusion sde and its relationship with an uncertain-volatility option model with volatility-asset decorrelation, 2008.

[9] Damiano Brigo, Xiaoshan Huang, Andrea Pallavicini, and Haitz Saez de Ocariz Borde. Interpretability in deep learning for finance: a case study for the heston model, 2021.

[10] Xiaochen Chen, Lai Wei, and Jiaxin Xu. House price prediction using lstm, 2017.

[11] Bank for International Settlements Monetary and Economic Department. Triennial Central Bank Survey Foreign exchange turnover in April 2019, 2019.

[12] Lawrence R. Glosten and Paul R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14(1):71–100, 1985.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[14] Tibshirani R. Hastie, T. and J. Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition*. Springer, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[17] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9, Feb 2021.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[19] Zineb Lanbouri and Said Achchab. Stock market prediction on high frequency data using long-short term memory. *Procedia Computer Science*, 175:603–608, 2020. The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology.

[20] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[21] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[22] Benoit Mandelbrot. The variation of some other speculative prices. *The Journal of Business*, 40(4):393–413, 1967.

[23] P. Mikko. Deep learning lecture notes. 2020.

[24] L. G. Nielsen. MACHINE LEARNING FOR FOREIGN EXCHANGE RATE FORECASTING. Master's thesis, Imperial College London, 2018.

[25] A. Norets. APPROXIMATION OF CONDITIONAL DENSITIES BY SMOOTH MIXTURES OF REGRESSIONS. *The Annuals of Statistics*, 38(3):1733–1766, 2010.

[26] Peter Nystrup, Petter N. Kolm, and Erik Lindström. Greedy online classification of persistent market states using realized intraday volatility features. *The Journal of Financial Data Science*, 2(3):25–39, 2020.

[27] Christopher Olah. Understanding LSTM Networks, 2015.

[28] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks, 2014.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[30] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, 2016.

[31] A. Sheth and K. Teeple. Connecting equity and foreign exchange markets through the WM "Fix": a trading strategy. *Journal of Investment Strategies*, 8(4):33–53, 2020.

[32] Hans R. Stoll. The supply of dealer services in securities markets. *The Journal of Finance*, 33(4):1133–1151, 1978.

[33] Hans R. Stoll. Inferring the components of the bid-ask spread: Theory and empirical tests. *The Journal of Finance*, 44(1):115–134, 1989.

[34] J.W. Taylor. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. 19:299–311, 2000.

[35] Y. Tu. Predicting High-Frequency Stock Market by Neural Networks. Master's thesis, Imperial College London, 2020.

[36] G. Xavier and B. Yoshua. Understanding the difficulty of training deep feedforward neural networks. *Aistats*, 9:249–256, 2010.

FONG_SICHENG_01347033

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49