國立政治大學經濟系研究所

碩士學位論文

深度學習於台灣指數期貨之應用 ：

經驗模態分解下之長短期記憶神經網路建模

Application of Deep Learning in Taiwan Index Futures : Long-term and Short-term Memory Neural Network Modeling Based on Empirical Mode Decomposition

國立政治大學 National Chengchi University

指導教授 ： 陳樹衡博士

研究生 ： 莊彥哲 撰

中華民國一〇八年七月

DOI:10.6814/NCCU201900955

摘要

本文中我們主要的目標是想要基於深度學習模型(Long Short Term Memory Network，縮寫 LSTM)，並結合經驗模態分解(Empirical Mode Decomposition，EMD 分解)將期貨的分鐘頻率日內資料分解為有意義的頻率信號，將人工智慧應用於預測金融時間序列走勢，且實際用於期貨市場的當沖交易。預測金融時間序列走勢的一直都不是個簡單的任務，主要是因為金融時間序列的非定態，具有序列相關。於是我們想結合專門將時間序列分解成多個獨立且有頻率意義信號的經驗模態分解(EMD 分解):以及具有長期記憶、寫入、清除、輸出，專門處理時間序列資料的長短期記憶神經網路模型(LSTM)，並運用模型輸出結果實際在歷史資料上交易回測，然後計算模型效能、策略績效，最後與傳統的機器學習(本文中以具有隱藏層以及多神經元的深度學習區分傳統上統計學的機器學習方法)演算法 K-近鄰演算法(K Nearest Neighbor. KNN)做比較。經過實驗我們成功找出 EMD 分解與 LSTM、KNN 的最佳預測區間長度，且經由實驗證明 EMD 分解確實能有效幫助中、短期的金融時間序列趨勢預測，以及深度學習模型 LSTM 的效能在相同資料處理方式下明顯優於傳統機器學習方法 KNN。

關鍵字:人工智慧、深度學習、神經網路、長短期記憶模型、遞迴神經網路、機器學習、K-近鄰演算法、經驗模態分解、日內資料、當沖交易、金融時間序列趨勢預測

DOI:10.6814/NCCU201900955

目 次

摘要 1

一、緒論 4

1.1 研究緣起 4

1.2 本文貢獻 6

1.3 本文架構 7

二、研究背景及文獻回顧 8

2.1 機器學習模型用於分類簡介 8

2.2 深度學習模型用於分類簡介 9

2.3 文獻回顧 10

三、研究方法 13

3.1 資料處理 13

3.1.1 台灣指數期貨資料簡介 13

3.1.2 資料集切割與標籤 14

3.2 機器學習演算法，K-近鄰演算法(K Nearest Neighbor，KNN) 20

3.3 深度學習演算法-長短期記憶模型((Long short-term memory. LSTM) 21

3.3.1 遞迴神經網路(Recurrent Neural Networ,RNN) 21

DOI:10.6814/NCCU201900955

3.3.2 長短期記憶模型(Long short-term memory. LSTM) 24

3.3.3 本文實驗採用的 LSTM 架構 27

四、實驗 36

4.1 實驗設計 36

4.2 衡量指標 37

4.2.1 判斷模型效能的指標 37

4.2.2 交易績效指標 38

4.3 實驗數據 39

4.3.1 不同模型與資料處理方式的效能交叉比對 40

4.3.2 不同模型與資料處理方式的交易效能交叉比對 44

五、結論與展望 50

5.1 結論 50

5.2 展望 51

參考文獻 52

國立政治大學 National Chengchi University

DOI:10.6814/NCCU201900955

一、緒論

1.1 研究緣起

2016 年，Google 人工智慧實驗室 DeepMind 所開發的人工智慧 AlphaGo 跌破各方專家眼鏡，以 4 比 1 懸殊的比數，打敗南韓最強職業圍棋九段棋士李世乭[1]，從此將人工智慧(Artificial Intelligence，縮寫為 AI)，這個名詞從電影中帶到現實。而所謂人工智慧背後的關鍵技術，廣義來說就是一種稱為深度學習[2]的神經網路模型，也因為 AI 的爆紅，各種深度學習相關的研究也在學界、業界蓬勃發展。近年來，人工智慧除了在學術界取得許多重大突破外，也被廣泛應用於解決現實世界中的各種難題。其中最經典且最廣為人知的應用就是機器視覺智能影像辨識以及自然語言處理語音辨識。而其中專門用於自然語言處理任務的是一種稱為遞歸神經網路(Recurrent Neural Networks，縮寫為 RNN)**[18]**的深度學習模型，因為 RNN 模型的特殊構造，會去考慮其隱藏層上一次輸出的倒傳導特性，使得其特別適合處理序列相關資料，如語言的前後相關、PM2.5 的循環性資料，其中當然也包含了金融商品如期貨、股票的時間序列資料。然而因為 RNN 只是單純考慮前一期的輸出，會有長期依賴(Long-Term Dependencies)[3]**[1]**的問題，因此現在主流

---

[1] AlphaGo 打敗李世乭: https://en.wikipedia.org/wiki/AlphaGo_versus_Lee_Sedol

[2] 現今的人工智慧大部分是指一種稱為深度學習的模型，而深度學模型廣義來說就是由多個權重隱藏層組成的神經網路模型，因為其多層的結構故稱為深度學習。

[3] 在 RNN 中每次訓練時，每次隱藏層的輸出會單純傳導到下一次訓練迴圈，且每次訓練共享權重，會造成當下訓練迴圈可能受到長時間之前的狀態影響，稱為長期依賴。

DOI:10.6814/NCCU201900955

使用的是將 RNN 的記憶體改善為具有長期記憶、忘卻、輸出功能的長短期記憶神經網路(Long Short Term Memory Network，縮寫 LSTM)**[8]**。本文致力於研究將 LSTM 加上資料處理方法用於預測台灣指數期貨的趨勢為上漲或者下跌，資料頻率採用分 k 日內資料(Intraday Data)[4]。過去的研究中較沒有詳細的針對預測區間的長度做實驗**[12][23]**，其中也有例外如[16]我們會在 2.3 節中詳細解釋。因此本文以 5 分鐘為一個單位，比較各個不同預測區間的結果，期望能找出最佳的預測區間長度，並嘗試將我們訓練好的 LSTM 用於台灣指數期貨的當沖交易(Day Trading)[5]。會如此設定實驗是因為台灣指數期貨為國人最常交易之期貨商品，其手續費固定且交易稅相對較低[6]的特性很適合拿來做當沖交易。而本文中 LSTM 神經網路所預測出未來某段時間內的趨勢的結果，也剛好可以運用在當沖交易的操作上。

然而由於金融時間序列資料普遍具有時間趨勢、非平穩的特性會使得資料尺度不一致，造成神經網路訓練上不易收斂的困難點，使得預測金融商品時間序列的走勢一直都是個很大的挑戰。因此我們嘗試在研究過程中處理資料時加入經驗模態分解(Empirical Mode Decomposition，縮寫 EMD 分解)[7]，來將原始時間序列分解成有頻率意義、獨立的時間的信號**[9]**。觀察經過訓練之後神經網路能不能從 EMD 分解後的信號中學習出能解釋未來上漲、下跌趨勢的規則。並經由實驗與傳

---

[4] K 線圖指的每單位時間的開盤、最高、最低、收盤價，而分 K 指的就是每分鐘的 K 線圖，是一種日內資料，台灣期貨市場一日有 300 分鐘交易時間。

[5] 當沖交易，指的是在每日收盤前將持有之部位平倉賺取利差，避開非交易時段變故的風險。

[6] 台灣證券市場股票交易稅為 0.3%，手續費 0.1425%，而台灣指數期貨交易稅為 0.002%，且手續費為固定，每家期貨商略有高低，交易稅成本股票為期貨的 150 倍。

[7] 經驗模態分解，由黃鍔（N. E. Huang）於 1998 年提出，是一種自適應信號頻率處理方法，特別適用於非線性、非平穩信號的分析處理。**[9]**

DOI:10.6814/NCCU201900955

統機器學習方法 K-近鄰演算法(K Nearest Neighbor，縮寫 KNN)比較，觀察我們所使用的 EMD 分解加上 LSTM 神經網路是否可以得到較好的結果，且可以實際在期貨或其他金融市場上進行當沖交易，同時具有獲利能力，能成為投資人、交易員做投資決策時的參考。

1.2 本文貢獻

在決定本文研究方向前，作者有先嘗試單純的使用 LSTM 加上輸入過去的原始時間序列來預測未來的趨勢，但是並沒有得到太好的結果，主要是因為台指期的原始資料具有明顯的時間趨勢，所以在資料處理的過程加上 EMD 分解做信號拆解，並與傳統上的 Z 值標準化做比較。而過去也有一些學者將 EMD 分解搭配 LSTM 做時間序列趨勢的預測(以下簡稱 EMD-LSTM)**[24]**，但缺乏在較高的頻率如分 K 資料下詳細的比較不同預測長度之間預測的效果差距。因此我們整個實驗都會詳細展示在不同預測長度的效果，並將我們訓練好的 EMD-LSTM 神經網路實際應用在當沖交易做交易績效比較，整體會在深度學習模型(LSTM)與機器學習模型(KNN)以及兩種資料處理方式 Z 值標準化與 EMD 分解之間做交叉比較，綜合以上本文的貢獻可以歸納為以下幾點:

I. 交叉比較深度學習演算法(LSTM)與傳統機器學習演算法(KNN)，分別使用 EMD 分解與傳統 Z 值標準化處理後的資料做訓練後的模型效能。而經由實驗過後發現 EMD 分解能有效提升 KNN、LSTM 模型對於未來趨勢預測的準確度。在相同的資料處理流程下，LSTM 的效能又會優於 KNN，因此在本實驗中我們可以得到 EMD-LSTM 是相對較有效可以用來預測期貨指數走勢的方法之結論。

II. 在交叉比較 LSTM 與 KNN 與兩種資料處理方式的同時，我們也詳細的觀察設

DOI:10.6814/NCCU201900955

定不同長度的預測區間對於模型預測效能的影響，最後發現大致上呈現越長的預測區間模型的效能越差的趨勢。

III. 實際依照模型預測出的結果在測試資料集內進行交易回測，並以交易的報酬率、夏普比率指標來衡量模型交易的績效，最後得到短預測區間的EMD-LSTM能穩定獲利，有最好的交易績效之結果。

1.3 本文架構

- 第 2 章主要描述本研究的背景，會大概介紹機器學習與深度學習在發展上的重要里程碑，以及為何 LSTM 模型適合用來做金融時間序列的趨勢預測，此外為了處理金融時間序列資料的序列相關、非平穩性我們也會在本章簡介如何將金融時間序列資料拆解成有意義的信號，同時我們也會簡單描述台灣的期貨交易市場，以利後續的交易回測實驗，本章的最後會回顧一些實際將機器學習、深度學習用於金融時間序列趨勢預測的文獻。

- 第 3 章主要描述本文之研究方法，會從資料處理方法包含資料來源、Z 值標準化、EMD 分解、移動窗格、訓練以及預測監督式資料集的裁切，到本研究使用的機器學習模型(KNN)，與深度學習模型(LSTM)架構，做詳細的定義。

- 第 4 章我們開始展示實驗的結果，會包含整詳細描述體實驗的設計，以及定義用來衡量模型效能、交易績效的指標，經由數據、圖表來證實本文使用的EMD-LSTM 對於台指期走勢以及當沖交易的輔助有較好的效能。

- 第 5 章我們會總結前面幾章的研究結果，並探討未來還可以延伸研究的方向。

DOI:10.6814/NCCU201900955

二、研究背景及文獻回顧

如前述，本文首要研究目標在於提出一個可以基於長短期記憶深度學習模型預測金融時間序列走勢用於當沖交易的深度學習模型，屬於分類問題，以下就過往所提出用於分類以及金融時間序列走勢預測方法的文獻做深入探討。2.1 節會把重點放在過去在分類問題有重大進展的經典機器學習方法 KNN 上。2.2 節則簡述神經網路深度學習模型發展的歷史，並探討近幾年最廣泛用於時間序列走勢預測的深度學習模型 LSTM 模型，並研究如何使用 EMD 分解做資料處理。最後在 2.3 節我們回顧過去已經使用機器學習以及深度學習方法預測金融時間序列走勢的文獻，作為本文研究的基礎。

2.1 機器學習模型用於分類簡介

機器學習用於分類或是時間序列走勢的預測已經有悠久的歷史，最早可以追朔到 1936 年由 Fisher 所發明的線性區別分析(Linear Discriminant Analysis.LDA)**[6]**，是一種將資料投影降維後找出一條線性函數做為切割依據，並使得切割後的組內變異最小的方法，還有同樣歷史悠久於 1958 年被發表的羅吉斯迴歸(Logistic Regression)**[3]**，因為可以直接預測出一個樣本屬於正樣本的機率，所以在當時被廣泛應用在廣告點擊率預測、罹患疾病機率診斷等場景，但這些線性的分類模型，在實務上還是有其限制，因為現實生活中的資料有很大部分都不是線性的，而且線性模型甚至無法解決異或問題，因此，經過改良也發展出許多非線性的分類模型，而早期最廣為人知的分類演算法則是 K-近鄰演算法(K Nearest Neighbor，縮寫 KNN)，KNN 算法最初由 Cover 和 Hart 于 1967 年提出**[2]**，其分類原理為計算樣本點最近的 K 個訓練樣本並取最頻繁的類別，也因為其簡單、無需估計參數、特別適合多分類

DOI:10.6814/NCCU201900955

問題的特性，KNN 算法被沿用至今，也有許多人拿來做金融時間序列走勢的預測，作為挖掘交易信號的工具，我們會在 2.3 小節中介紹過去使用 KNN 於金融時間序列趨勢預測的文獻，並於第 3 章中詳細解釋 KNN 的原理。

2.2 深度學習模型用於分類簡介

深度學習模型，也就是所謂的人工智能，是近年來學界、業界的重大發展項目，被運用在各行各業、各種應用場景，其中當然也包含了金融時間序列趨勢的預測，但深度學習模型的概念最早是於 1943 年由神經科學家 W.S.McCilloch 和 W.Pitts 提出[15]，仿照生物神經元的工作原理構造了神經網路的數學模型，包含輸入訊號、求和、非線性啟用三個過程，之後於 1958 年由電腦科學家 Rosenblatt 提出兩層神經元的神經網路稱為感知器(Perceptrons)[17]，使用梯度下降法從訓練樣本中自動學習更新權重完成二分類的任務，也是本文研究能成立的基礎，神經網路真正爆發性的成長是由於神經網路之父 Geoffrey Hinton 與 Ronald J. Williams 於 1986 發明適用多層感知器(MLP)的反向傳播(Backpropagation)演算法[18]，並於 2006~2012 年間不斷改良，改變網路結構、激活函數(Activation Function)，於語音識別、影像識別領域都取得突破性的進展，也發展出許多基於基礎神經網路的變化模型，然而提到人工智能、神經網路用於時間序列預測就不得不提到長短期記憶模型(Long short-term memory. LSTM)，它是一種時間遞迴神經網路，也稱反向傳播神經網路(Recurrent Neural Networks. RNN)，因為其倒傳導的特性，很適合處理前後相關的任務，一開始是運用在自然語言處理，因為語言通常是需要考慮前後語來判斷語意，像是 Google 的翻譯軟體就是用自家開源框架 Tensorflow 所開發的 RNN 網路，也因為它適合前後相關任務的特性，也有許多人拿來做時間序列分析，取得突不錯的結果，我們會在 2.3 小節中研究過去使用 LSTM 於金融時間序列趨勢預測的文獻。

DOI:10.6814/NCCU201900955

2.3 文獻回顧

LA Teixeira ，ALI De Oliveira(2010)[21]使用 KNN 分類模型預測隔日的股價走勢並將技術指標[8]分析工具如 RSI、SMA[9]指標當作輸入資料，輸出隔日股價的走勢並將之用於聖保羅證券交易所真實公司的股價，在考慮交易成本的狀況下，最後得到根據 KNN 模型預測結果進行交易的績效優於單純使用 SMA 黃金、死亡交叉[10]判斷買賣點的結果。

M.V.SUBHA(2012)[19]使用 K Nearest Neighbor 預測印度股市中兩檔熱門股票未來一天的走勢，跟前文較大不同的地方是作者單純使用前一天股價的開高低收原始資料作為輸入，並未額外使用過去歷史資料計算技術指標，同樣預測隔日股價的走勢，並與 Logistic Regression 做比較，在 118 天的測試期中得到 KNN 表現較優秀的結果成功預測出 83 天的上漲趨勢，在現今由於運算資源的成長，我們想嘗試輸入更多過去的資料，也就是讓模型接收過去更多期數的資訊後，期望能更精確的預測未來走勢。此外，這兩篇論文都是單純的預測隔天的走勢，並沒有比較不同預測區間長度之間的效果，我們也在本文中加強了這點。

國立政治大學 National Chengchi University

---

[8] 技術指標(Technical Indicator)，是指利用過去的歷史資料，經由各種數學公式運算，期望能計算出對於未來價格走勢有解釋力的指標。

[9] RSI(相對強弱指標)，為一種衡量過去一定區間內上漲力道的強弱，RSI 又稱為逆勢指標，也就是當 RSI 過大時可能代表市場過熱，價格可能要開始走低，SMA 則是計算過去固定期數的移動平均。

[10] 黃金交叉，指的是較短期數的 SMA(快線)向上穿過較長期數的 SMA(慢線)，可能代表股價趨勢即將止跌反升的信號，反之則稱為死亡交叉，則代表股價可能要開始下跌的信號。

DOI:10.6814/NCCU201900955

Ariel Navon, Yosi Keller(2017)**[16]**將普通的前傳導神經網路[11]用於預測股票、ETF 的走勢，使用分鐘頻率的高頻資料，交易標的物選擇紐約、納斯達克交易所中的多檔股票以及 ETF，並做了兩年的回測，神經網路輸入過去 60 分鐘的原始開盤價、最高價、最低價、收盤價資料，輸出未來走勢為上漲或是下跌的類別機率，並由該機率是否大於設定的閥值來判斷是否為交易訊號，並比較預測未來各種長度區間的效能，得到 28 期長度的預測期間有最佳效能的結果，並使用高頻交易夏普比率來評斷模型的交易績效，該篇論文未使用 LSTM 神經網路但同樣也使用分鐘頻率的資料，並比較不同預測區間長度的結果與我們的構想類似，因此我們會參考這篇論文所使用的輸入期數(60)，並使用修改過的夏普比率來衡量神經網路的交易績效。

EDWIN LI(2018)**[8]**將 LSTM 用於預測 S&P500 走勢並實際上用於交易，僅輸入前一天的開盤價、最高價、最低價、收盤價再加上 RSI、MACD[12]指標預測隔天指數的走勢，使用一層含有 32 個 LSTM Cell 的隱藏層，並接上 ReLU 激活函數，優化器的選擇則是採用 Adam**[4]**，能夠有效避免梯度消失的問題，以及具有計算效率較高的好處，訓練時也採用 Drop Out 技巧避免 Over Fitting，在共 838 天的交易中獲得約 30%的報酬率，我們會參考其神經網路架構，同樣也使用 Drop Out 技巧，但會增加輸入的期數，期望在獲得更多過去資訊的狀態下 LSTM 可以表現得更好。

---

[11] 前傳導神經網路，是一種最基礎的神經網路，其每次隱藏層的輸出都是傳到下一個隱藏層，並沒有反饋到下次訓練的機制，故稱為前傳導神經網路，而有反饋機制的神經網路則是前面有提到的遞迴神經網路(RNN)，會在第 3 章中詳細介紹。

[12] MACD 指標，是一種計算長期與短期移動平均線差離狀態改變的指標，可以衡量過去移動平均線收斂、發散的趨勢，

DOI:10.6814/NCCU201900955

Zhang, Boning. (2018) **[23]**同樣使用 LSTM 來預測金融時間序列的走勢，但不同的是作者預測的是時間序列的值而不是趨勢，但本文想要借鏡的是作者在資料處理的步驟加入的改良，為了處理金融時間序列都有的時間趨勢問題，他在資料處理時入經驗模態分解(Empirical Mode Decomposition ，簡稱 EMD)，將複雜的時間序列信號拆解成有限個獨立且有頻率意義的本徵模函數（Intrinsic Mode Function，簡稱 IMF），然後使用這些 IMFs 送進 LSTM 訓練，並做出預測，最後得到使用 IMFs 訓練的 LSTM 效能優於只有使用原始資料訓練的 LSTM 的結論，本文的一大重點也會比較使用 EMD 與 Z 值標準化方法資料處理，對於機器學習、深度學習訓練的影響。

Jonathan Doering & Michael Fairbank(2017)**[5]**將捲積神經網路(convolutional neural network，縮寫 CNN)[13]用於高頻交易上，預測標的物使用倫敦交易所中的 GB0031348658 (Barclays PLC) 資料區間為 2007 年 7 月至 2008 年 8 月中共 217 天的逐筆交易資料，共約 72000 筆，輸入資料使用區間長度 180 的移動窗格，預測下 20 筆交易的價格走勢，作者提到因為在高頻交易中資料的漲跌幅較小，為了能區分出合理的閥值所以標記價格走勢的標準採用原始價格的報酬率，作者經由計算採用 0.05 的漲跌幅作為閥值，可以剛好讓資料集保持在二分類平衡的狀況，較不會發生極端不平衡[14]難以訓練的狀況，本文中我們也會首先考量資料集的原始狀況為衡量模型預測效能的基準點，並使用移動窗格比較各種預測長度的結果。

---

[13] 捲積神經網路，是一種用於影像辨識的經典神經網路，由於其捲積核構造可以像掃描器一樣將圖片中的像素萃取成能代表圖片的關鍵特徵值，近幾年主導機器視覺領域的發展

[14] 在訓練深度學習模型時，如果某一類別數量過少，造成該類別就算全部分錯對於損失函數的影響很小，會造成該類別的準確度被犧牲掉，稱為不平衡資料。

DOI:10.6814/NCCU201900955

三、研究方法

3.1 資料處理

3.1.1 台灣指數期貨資料簡介

在本文中我們使用元大期貨 Multichart[15]匯出的台灣指數期貨每分鐘歷史資料，主要因為本文致力於研究長短期記憶模型(LSTM)用於金融時間序列趨勢預測的效能如何並實際應用於當沖交易。為了能充分展現出高頻高金融時間序列的資料特性，我們需要交易頻率夠高，且穩定持續高頻買賣搓合的商品。然而股票因為有相對較高證交稅的關係不太適合做頻繁進、出場的當沖交易。而期貨的交易稅、手續費相對較低，加上不管看多、看空都能進場能做多、放空使我們可以依照模型預測出來的趨勢自由的決定我們的策略。因此本文會以期貨為研究的商品，而在標的物的選擇上，我們選擇台灣人最常交易的台灣指數期貨(TFX)。

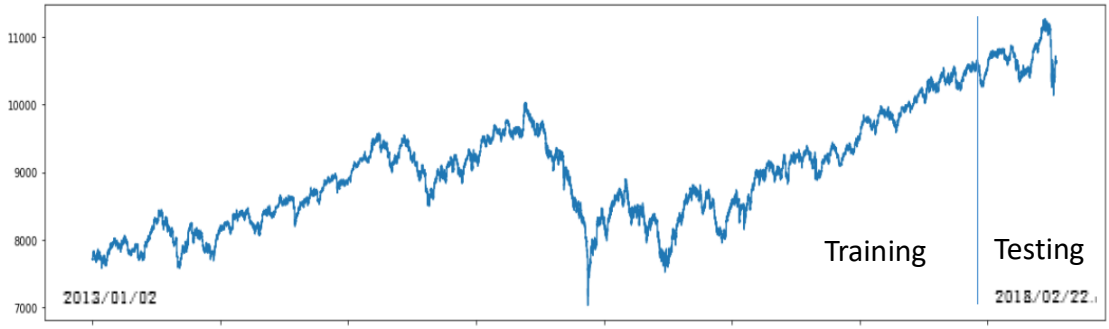
台指期貨是台灣期貨交易所推出的產品，屬於期貨的一種，依據台灣的股票加權指數去做漲跌變動，早上八點四十五分開盤至下午一點四十五分收盤，共 5 個小時也就是 300 分鐘，投資人可以依照自身判斷決定做多(買進一口多單)或放空(賣出一口空單)，其中一點的漲跌幅是 200 元，依照投資人做多或放空計算損益。台指期在 2017 年就達到每日平均 138270 口的交易量約占台灣期貨總

[15]資料來源 ： 元大期貨 Multichart http://www.multicharts.com.tw/characteristic.aspx

DOI:10.6814/NCCU201900955

交易量的 47%[16]，為我們的研究提供高頻、穩定的交易資料。資料頻率使用分資料、資料區間是 2013 年 01 月 02 號到 2018 年 02 月 22 號包含每分鐘的開盤價、最高價、最低價、收盤價、交易量。我們將每分鐘收盤價的結果初步視覺化如圖一所示。

圖一 ： 2013/01~2018/02 台指期分資料



3.1.2 資料集切割與標籤

為了產生能讓深度學習、機器學習演算法訓練的監督式學習(Supervised Learning)[17]資料集，首先我們將資料集切成大約 9:1 的訓練(2013/01/02~2017/06/30)、測試資料集(2017/07/01~2018/02/22)，然後在資料集內用移動窗格(Sliding Window)[18]的方式做出一對一關係的監督式資料集，如圖二所示參數 T 代表輸入的區間長度，t 代表我們要預測未來趨勢的區間長度，也就是我們輸入 T 期的資料區間當作訓練的輸入資料，而第 T 期到 T+t 期的趨勢則是輸出的

[16] 資料來源 ： 台灣期貨交易所 https://www.taifex.com.tw/cht/index

[17] 監督式學習，指在訓練過程中提供模型答案(標籤)，在本研究中就是真實的上漲、下跌。

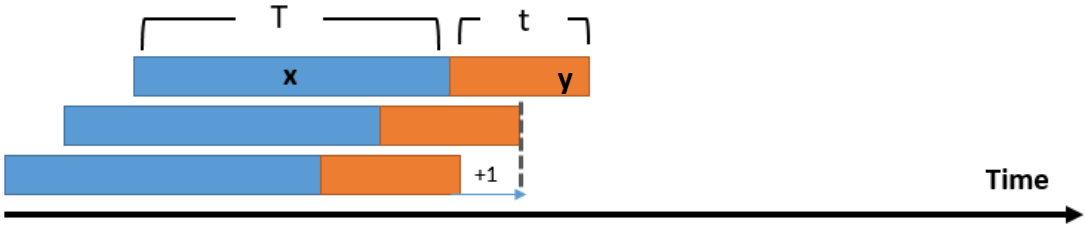
[18] 移動窗格，指將時間序列資料以固定長度以及固定橫移長度切割成多個區段

DOI:10.6814/NCCU201900955

訓練目標 y，稱為標籤(Label)[19]，如圖二所示反覆執行以上動作，但要注意的是我們切割資料集時不做跨日切割，也就是讓交易在當日了結，可以避免掉隔日台指期跳躍的風險，對於訓練也有正面的影響。

此外，由圖一中我們可以發現在測試區間(2017/07/01~2018/02/22)內，整體台指期呈現多頭走勢(牛市)，這也代表著如果我們單純的在測試區間內持有一口多單而不做任何交易(buy-and-hold)，我們基本上就能夠得到正報酬。因此我們會在 4.3 小節中比較依照 EMD-LSTM 預測進行買多賣空與單純持有一口多單兩種策略的交易績效，並且考慮交易手續費。如果實驗證明我們的 EMD-LSTM 交易策略能在牛市行情中達到優於單純持有一口多單的績效，則證明我們提出了一套有實際應用價值，可用於當沖交易的深度學習方法。

圖二 ： 移動窗格資料處理說明



[19] 標籤，指模型訓練的目標，在本研究中必須人工標記真實資料的上漲、下跌標籤。

DOI:10.6814/NCCU201900955

在本文中，輸入過去期數 T 的長度我們主要參考 Ariel Navon, Yosi Keller(2017)[16]的做法使用 60 這個長度區間，一方面因為在該篇論文中所採取訓練架構與本文相近，另一方面則是因為考量到在不跨日交易的狀況，輸入的區間長度意味著需要犧牲掉開盤之後的多少時間，而 T=60 已經代表開盤之後的一小時都還未交易，以及考慮到運算資源的限制，所以決定將 T 參數固定在 60，而預測未來趨勢的區間則嘗試 t=5 至 t=60 之間每 5 期訓練一個模型來找出最合適的區間，因此一天之內在沒有缺失值的狀況會有 300-T-t+1 筆訓練資料(一天有 300 分鐘的交易時間)，而因為使用開、高、低、收、交易量，所以會有 60*5=300 維的特徵值，而訓練標籤的處理方式，則是如(1)式中，簡單的將第當期 n 與第 n+t 期作比較，來判斷為漲勢或是下跌。

$$y_{(n)} = \begin{cases} 1, \ if\ x_{(n+t)} > x_{(n)} & 上漲 \\ 2, \ if\ x_{(n+t)} \leq x_{(n)} & 下跌 \end{cases} \quad (1)$$

3.1.3 資料歸一化

經過實驗，使用報酬率標準(ln(y_t-y_t-1))化會造成訓練上的困難使得模型總是只能預測出 1 類別的答案，因此本文探討兩種資料標準化的方式 ： 1.標準化(z-value) 2.經驗模態分解，會在後續的章節藉由實驗研究經驗模態分解是否能較有效的處理序列相關的問題，本實驗採用的標準化做法簡單使用一般常見的 z-value，為了撇除掉時間趨勢的問題僅在每一個輸入的序列中獨立做標準化:

DOI:10.6814/NCCU201900955

$$z = \frac{x-\bar{x}}{\sigma} \qquad (2)$$

$\bar{x}$ ：樣本平均數

σ：樣本標準差

在每次輸入模型的序列中，我們依照式(2)對每個樣本做 Z 值標準化，類似於我們在 3.3.3 中討論的 Batch Normalization。

經驗模態分解(Empirical Mode Decomposition，EMD)，以下簡稱 EMD 分解，是由 N. E. Huang(1998)**[9]**在美國太空總署 NASA 提出，是一種自適應的信號序列處理方法，特別適合處理非線性非平穩的信號處理，被廣泛使用在各種工程領域如海洋、大氣、天體觀測等，由於金融時間序列資料大部分都是非平穩，近年來也有文獻將 EMD 分解用在金融時間序列的拆解上。在物理學上如果一段頻率是有意義的，則該片段函數必須是對稱且片段均值為零，且有相同的過零點和極值點數目，符合這個條件的函數則被稱為本徵模函數(Intrinsic Mode Function，簡稱 IMF)，EMD 分解的目的就是把原始帶有序列相關非平穩的時間序列，分解成數個有獨立且有頻率意義的 IMFs，其做法如下:

1. 找出原始信號中全部的局部極大和極小值

2. 將這些局部極大、極小值擬合成上、下包絡曲線

3. 由上下包絡曲線找出他們的均值曲線，m(t)，再用原始時間序列 f(t)減掉 m(t)得到 H(t)就是 IMF

$$H(t) = f(t) - m(t) \qquad (3)$$

DOI:10.6814/NCCU201900955

4. 但通常 1~3 步驟剛開始找出的 IMF 不會符合上述的 IMF 條件，必須不斷重複 1~3 步驟，直到 SD 值小於閥值為止

$$\mathrm{SD} = \sum_{t=0}^{T} \frac{\left|H_{k-1}(t) - H_{k(t)}\right|^2}{H_{k-1}^2(t)} \qquad (4)$$

(4)式中的 T 代表樣本點的總數，k 則表示已經做了 1~3 步驟的次數，直觀的意義可以看做是做了 k 次與 k-1 次的分解之 IMF 之間的差異，如果之間的差異小於我們設定的閥值，則表示我們認為已經分解出的這些 IMF 就足以表示原始序列，不需要繼續分解下去。

5. 我們可以在上述過程中計算殘差 :

$$\mathrm{r(t) = f(t) - H(t)} \qquad (5)$$

當分解停止後，我們可以觀察殘差，殘差的曲線越接近 0 則表示，我們分解出的 IMFs 已經可以充分代表原始序列資料。

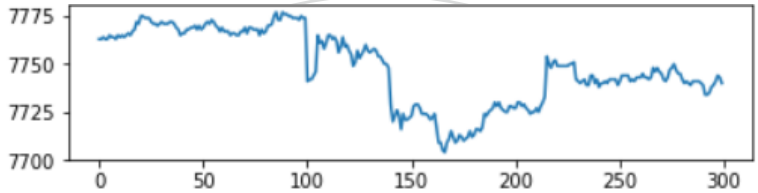
我們取資料區間中某一日 300 分鐘的收盤價來做 EMD 分解看看效果，為了方便展示先簡單先設定分解成 3 個 IMFs ，但實際上做實驗時我們會分解成 5 個 IMFs。我們有以下幾個考量 1.為了方便與傳統 Z 值標準化做比較，因為在使用 Z 值標準化時我們使用每分鐘的開盤、最高、最低、收盤價以及交易量共 5 維。但是在使用 EMD 分解時我們僅採用每分鐘的收盤價，因為我們預測的目標是收盤價的走勢，因此理論上收盤價的 IMF 會比較能代表收盤價的走勢訊號。為了統一神經網路的架構(輸入資料的維度會影響神經網路的隱藏層架構)，我們也採用 5 個 IMFs，讓使用 EMD 分解以及傳統 Z 值標準化的 LSTM 神經網路架構可以一模一樣，降低實驗的不確定因素。2.考量到運算的資源，有嘗試過不要限制 IMFs 的數量做 EMD 分解，所花費的時間幾乎讓實驗無法進行。綜合以上統一 LSTM 神經網路架構以

DOI:10.6814/NCCU201900955

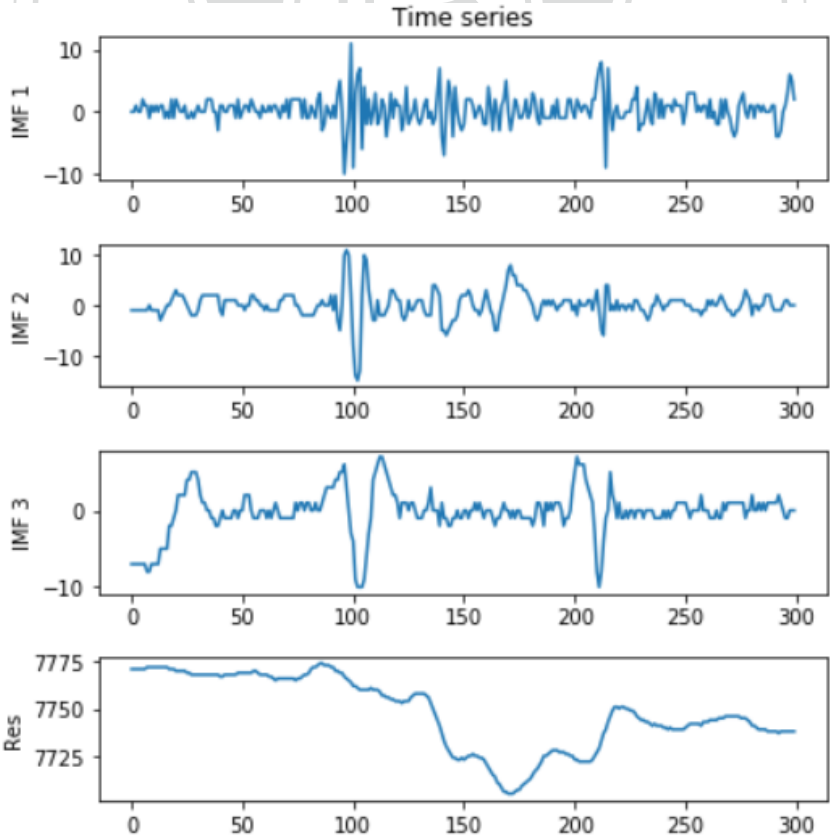
及運算時間的考量我們決定將 IMFs 數量固定在 5。而拆解的對象為每分鐘收盤價，原因是我們預測的目標是未來的趨勢，而趨勢為上漲或者下跌是使用每分鐘的收盤價來判斷，因此使用收盤價拆解出來的訊號理論上會含有較多能代表未來趨勢的信息。

圖三，EMD 分解效果

原始資料



EMD 分解後的 IMFs 與殘差



DOI:10.6814/NCCU201900955

如圖三所示，原始的時間序列資料被拆解成 3 個互相獨立、沒有時間趨勢在固定區間內上下震盪的訊號，我們會將這些 IMFs 送入模型中，來測試 EMD 分解是否對於預測趨勢上有明顯的幫助。

3.2 機器學習演算法，K-近鄰演算法(K Nearest Neighbor，KNN)

KNN 演算法它是由 Cover 和 Hart 於 1967[2]年提出，因為其簡單有效的特性被沿用至今，它的原理為找出距離未分類樣本點最近的 k 個已知樣本點，而 k 個樣本中最頻繁類別即為該未分類樣本點的類別，其算法流程如下:

我們首先定義資料集為 T ，T 中共有 N 個成對的資料 ：

$$T = (x_1, y_1), (x_1, y_1), \ldots\ldots, (x_N, y_N) \qquad (6)$$

N ： 總樣本數量

其中$y_i \in Y = \{c_1, c_2, \ldots . c_N\}$為實例的類別，i=1, 2, 3. . N

c：表示類別

依照下式先算出個樣本之間的距離，這裡使用較常見的歐式距離

$$L(x_i, x_j) = \left[\sum_{l=1}^{n}\left(\left|x_i^{(l)} - x_j^{(l)}\right|^2\right)\right]^{\frac{1}{2}} \qquad (7)$$

l：一個樣本點的第 l 個特徵

n：一個樣本點的特徵總數

DOI:10.6814/NCCU201900955

根據(7)式計算出的距離，在訓練集中找出與 x 最鄰近的 k 個點，涵蓋 k 個點的範圍記為$N_k(x)$，並在$N_k(x)$中多數表決，決定 x 的類別 y

$$y = \arg max_{cj \in N_k(x)} I(y_i = c_j), i = 1,2,3, \dots \dots, N; \qquad (8)$$

C_j： 表示第 j 個類別，為了區別樣本與種類使用 j 符號

(8)式中直觀意義為找尋在範圍$N_k(x)$的 k 個樣本中，出現次數最多的類別。

3.3 深度學習演算法-長短期記憶模型((Long short-term memory. LSTM)

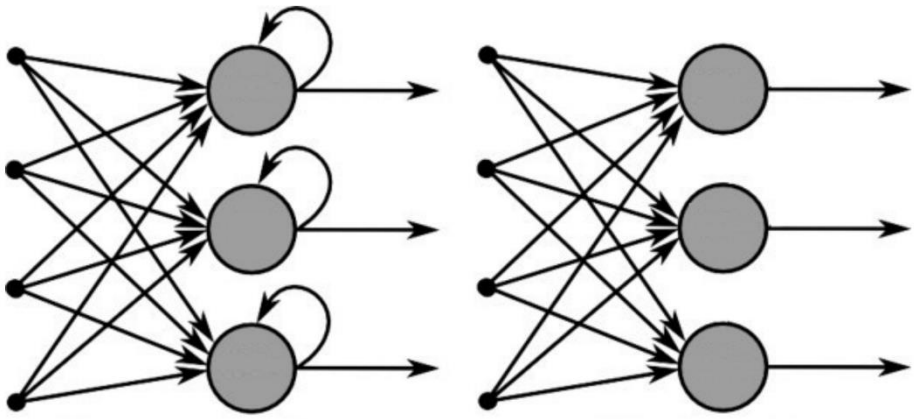
在本章節中將會介紹我們所使用的 RNN、LSTM 演算法，以及會詳細的描述本文中所使用的神經網路架構，以及選擇神經網路中各個模組、參數的原因，如激活函數、損失函數、優化器等等。

3.3.1 遞迴神經網路(Recurrent Neural Networ,RNN)

LSTM 是遞迴神經網路(Recurrent Neural Networ, RNN)[18]的變化型，因此要研究 LSTM 我們不得不先了解 RNN，然而 RNN 是由前饋神經網路發展而來，我們可以先比較遞迴和前饋神經網路的差別。

DOI:10.6814/NCCU201900955

圖三，前饋神經網路 VS 遞迴神經網路



遞迴神經網路　　前饋神經網路

Source：https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

遞迴神經網路和前饋神經網路最大的差別在於前饋網路每次在更新參數時，只會參考下一批輸入的資料，但遞迴神經網路除了參考下一批新的資料外，還會參考隱藏層(Hidden Layer)[20]在前一次代送時的輸出，也因為這樣的特性讓遞迴神經網路特別適合應付資料前後有相關的任務如自然語言處理、時間序列等等，其中也包含本研究中使用的金融時間序列資料，我們可以從數學式中理解遞迴神經網路的原理 ：

[20] 隱藏層(Hidden Layer)，神經網路中最小的單位是一個神經元，一個神經元就是一組用來乘上輸入資料的權重加上一個偏差值，而將很多個神經元組合在一起就是一個隱藏層。

DOI:10.6814/NCCU201900955

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \qquad (9)$$

$$y_t = \sigma_y(w_y h_t + b_y) \qquad (10)$$

x_t：*輸入的資料矩陣*

h_t：*隱藏層* $(Hidden\ Layers)$

y_t：*神經網路的輸出* $(Output\ Vector)$

W,U：權重矩陣

b：偏誤項

σ_h, σ_y：*自訂義激活函數* $(Activation\ Function)$

式(9)、(10) 以僅有一層隱藏層的遞迴神經網路為例，其中的W_h、U_h、b_h、w_y、b_y是我們神經網路訓練的目標，藉由訓練更新這幾個矩陣或是向量中的權重參數來讓損失函數(Loss Function)達到最小，進而使神經網路的輸出y_t越來越接近真實正確的資料標籤，我們會在後續的章節清楚定義我們的神經網路架構。

我們可以從式子中清楚的看到在 t 時刻的隱藏層，其輸入包含了 t 時刻新一批的 x 資料以及 t-1 時刻的隱藏層h_{t-1}，因為這樣的構造讓 RNN 在每次訓練的時候，除了考慮當前的輸入資料外，也會參考前一次代送時隱藏層的狀況，使得 RNN 模

DOI:10.6814/NCCU201900955

型很適合用來處理序列相關的資料，此外會在隱藏層中加上偏誤項[21]b_h，有了這個誤差項可以讓整個 RNN 具有整體橫移的能力，能幫助訓練。我們還會在隱藏層或是輸出層(Output Layer)上加上一個激活函數(Activation Function)，激活函數有很多選擇，但都有相同的作用就是將隱藏層或是輸出層的輸出壓縮到一個較小的區間，如此可以讓整個神經網路變成非線性的函數，可以解決異或問題[22]或是處理更複雜非線性的任務，在後續的章節我們也會詳細定義本文中使用的激活函數。

3.3.2 長短期記憶模型(Long short-term memory, LSTM)

然而現在比較主流的做法是將 RNN 的記憶體改良成 LSTM[8]，是因為原生的 RNN 有幾個比較大的缺點，第一 ： 梯度消失(The vanishing gradient problem for RNNs)[22][13]，因為 RNN 會單純的參考前一個隱藏層狀態，再加上使用激活函數將輸出值壓縮，使梯度下降法更新參數時，微分後的鏈鎖率會產生一連串小於 1 的值連乘導致梯度消失使神經網路無法訓練。第二 ： 過去較長時刻的輸入對當前時刻的影響很小，因為 RNN 僅參考前一次的輸出，沒有參考較長時刻的機制。因此後來的學者提出具有長期記憶、提取、遺忘能力的記憶體(Cell)的 LSTM。

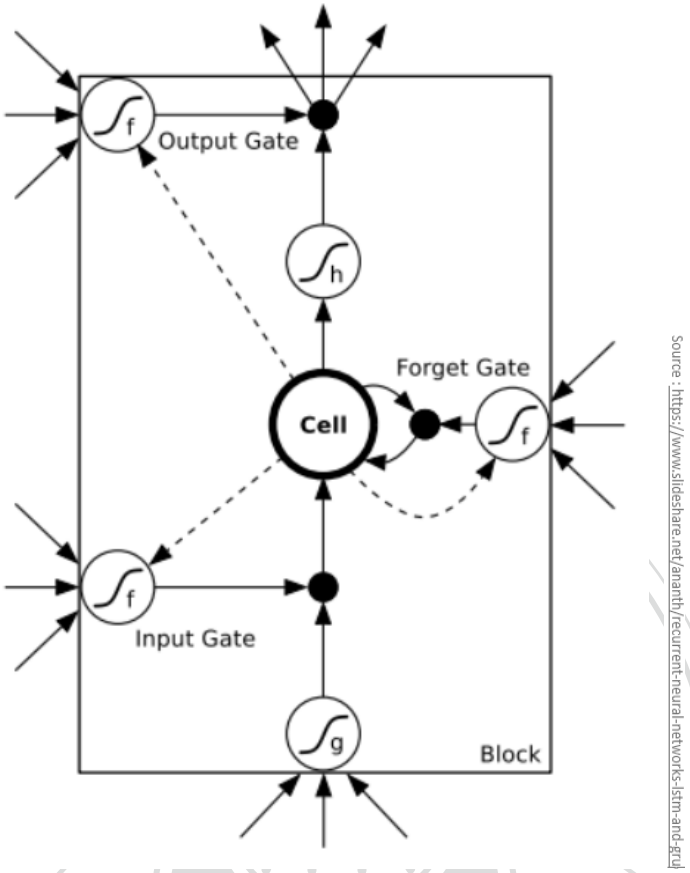
---

[21] 偏誤項(Bias)，其主要作用為讓神經網路具有整體橫移的能力，能在訓練神經網路時更快速的收斂

[22] 異或（XOR）問題可以看做是我們要將單位正方形的四個角分成兩類，四個角的座標（0,0），（0,1），（1,1），（1,0）為線性不可分，因此這樣的任務線性的分類模型無法達到。

DOI:10.6814/NCCU201900955

圖四，長短期記憶體運作示意圖



圖四是將 LSTM 以圖形化的方式呈現，如圖四所示，一個長短期記憶體包含 3 個閘門，分別是 ： 輸入閘門 (Input Gate)、遺忘閘門(Forget Gate)、輸出閘門(Output Gate)，為了能更理解實際上 LSTM 運作的原理，我們將數學式列出，並詳細探討 LSTM 是如何做到長期的記憶功能，如何做到可以自動的判斷何時要將新資料寫入記憶體，何時要將記憶體中的資料忘卻，LSTM 的數學式如下:

DOI:10.6814/NCCU201900955

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (11)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (12)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (13)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_t h_{t-1} + b_c) \quad (14)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (15)$$

f_t：*遺忘閘門*

i_t：*輸入閘門*

o_t：*輸出閘門*

c_t：*記憶體狀態*

σ_g: *Sigmoid Function*

σ_c、σ_h: *自訂義激活函數*

- 符號代表 Hadamard Product 即矩陣內相對應位置的元素做乘積

國立政治大學 National Chengchi University

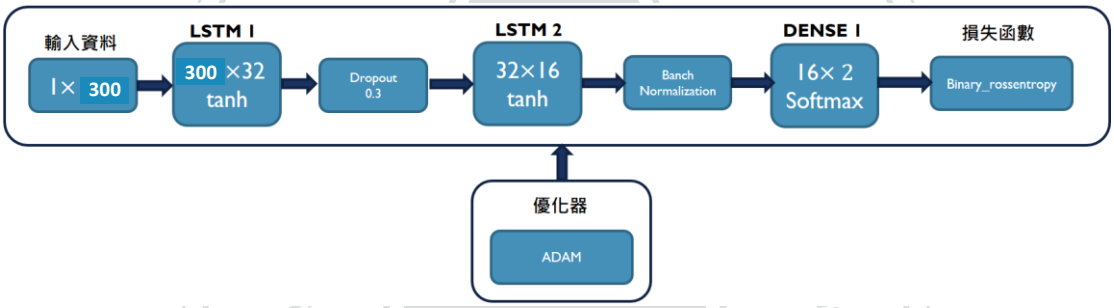
(11)、(12)、(13)式的遺忘閘門f_t、輸入閘門i_t、輸出閘門o_t，三個閘門的輸入都是x_t與h_{t-1}，只是其中各自有自己的 W 與 U 權重分別乘上當前的輸入資料x_t與前一時刻隱藏層的狀態h_{t-1}，且最後都需要經過一個激活函數σ_g將值壓縮在[0, 1]之間，(14)式則是當下時刻記憶體的狀態，由前一時刻的記憶體狀態c_{t-1}與當期的輸入計算得到，可以看到如果(14)式內的f_t也就是遺忘閘門的值為 0，則前一時刻的記憶體狀態c_{t-1}會被遺忘，然後只採納當前時刻的輸入來計算。

DOI:10.6814/NCCU201900955

同樣在(14)式中也可以看到由輸入閘門i_t來決定是否接受當期時刻的輸入，最後是(15)式代表的隱藏層輸出，這邊的隱藏層就是對應 3.2.1 章節中(9)式前一時刻的 RNN 隱藏層，從(15)式中也可以看到由輸出閘門o_t控制著是否輸出記憶體的狀態。經由訓練之後 LSTM 將可以學到甚麼時候要進行遺忘、輸出、輸入的處理，達到可以長期記憶，選擇性的從記憶體中提取記憶體狀態，必要的時候將記憶體清空的效果。

3.3.3 本文實驗採用的 LSTM 架構

圖五，本文實驗的 LSTM 架構、損失函數、優化器



圖五為本文中所使用的 LSTM 神經網路的完整架構，整體架構包含兩層 LSTM(其中一層是隱藏層)、一層 Dense 輸出層、一層隱藏層、一層批量標準化，以及計算 Dense 1 輸出層結果與真實資料差距的損失函數 Binary Crossentropy，以及用來更新參數使損失函數降到最低的優化器 Adam，以下將詳細解釋這些架構的用途，以及為何選擇這些元件。

DOI:10.6814/NCCU201900955

Long Short-Term Memory

這邊使用的 LSTM 就如同 3.3.2 章節中介紹的神經網路模型，本實驗採用的模型包含有 2 個 LSTM 層，整體的架構由 LSTM 1 當作資料輸入層、LSTM 2 當作隱藏層，而最後的 DENSE 1 則是輸出層，會用這樣一個較精簡的配置是因為本實驗的目標是預測未來趨勢的上漲或是下跌，屬於單純二分類任務，加上參考 EDWIN LI(2018)[12]一文中也是僅使用一層隱藏層的 LSTM 模型就有不錯的結果。因為要接收輸入的資料加上過去訓練小型神經網路模型的經驗，所以將 LSTM 1 權重矩陣的尺寸設為 300× 32，本實驗採用 Tensorflow 架構的 Keras 高端接口[23]，在 Keras 的套件[24]設計下 300× 32的 LSTM 層代表著有 32 個長短期記憶體，然而，因為模型最後的輸出會是一個 2 維的向量(代表兩個類別的機率)，所以我們會將神經網路的架構設計的越來越小，會得到一個比較穩定的結果且收斂較快，所以將 LSTM 2 的尺寸設為 32× **16**。

向前傳導全連接層

Dense 1 層就是普通神經網路的向前傳導全連接層(Fully Connected Neural Networks)，在本實驗架構下是當作輸出層來使用，用來來將 LSTM 2 的輸出，經過 Dense 1 之後能得到我們要的輸出尺寸，也就是一個 2 維的向量，因此他的尺寸會依照 LSTM 2 的輸出來調整，在本實驗的設計下就是 16× **2**。

---

[23] Tensorflow，是一個由 Google Brain 所開發的深度學習、神經網路框架並於 2015 年 11 月開源發布，為目前世界上最多人使用的開源 AI 框架，而 Keras 則是 Tensorflow 的高端接口，將 Tensorflow 功能整合為較簡單的使用介面。

[24] Keras 接口詳細使用細節可以參考 Keras 官網 : https://keras.io/

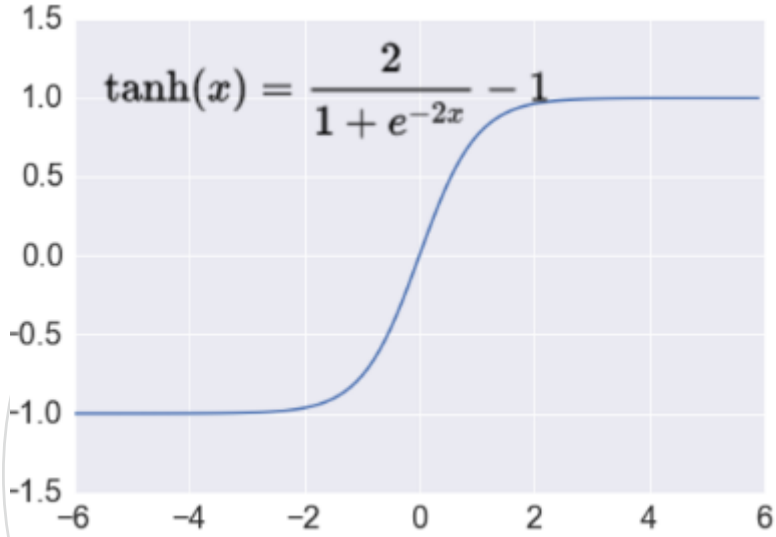
DOI:10.6814/NCCU201900955

激活函數與損失函數

LSTM 1、LSTM 2 搭配的激活函數，我們選擇 tanh

$$\mathbf{tanh(x)} = \frac{\mathbf{2}}{\mathbf{1}+e^{-2x}} - \mathbf{1} \qquad (16)$$

圖六，tanh 函數



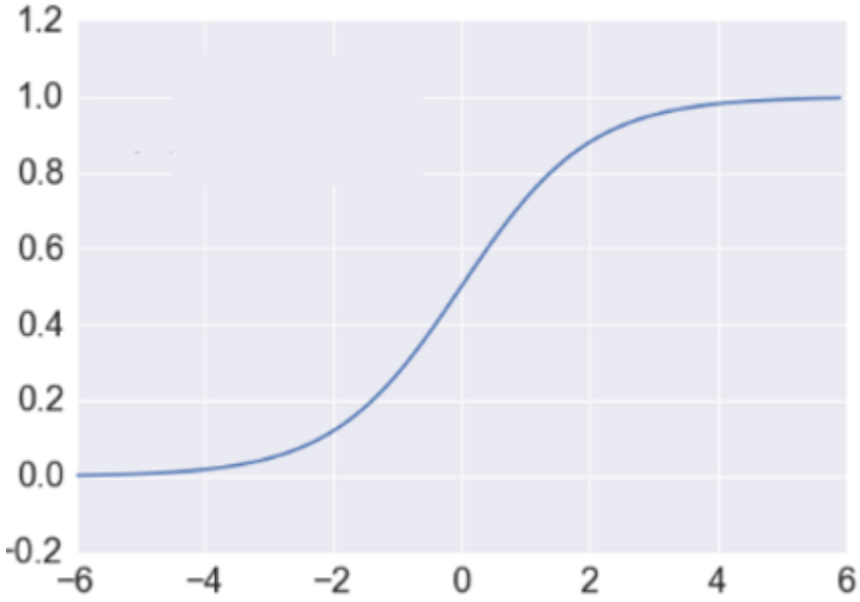
LSTM 搭配 tanh 有幾個好處[11]：1.如圖六所示 tanh 將值壓縮於[-1,1]之間可以讓 LSTM 狀態值增加以及減少，不會像某些激活函數如 ReLU、Sigmoid，只有大於 0 的輸出值。2.梯度消失是遞迴神經網路的主要潛在問題，為了讓梯度不要太快趨近於 0，我們需要一個函數，其二階導數可以在變為零之前維持較久廣範圍的值，tanh 具有此良好的特性。3.LSTM 的記憶體具有長期記憶的功能，因此如果記憶體中存有極端值對後續的訓練可能造成不良影響，而 tanh 函數強制將值壓縮在[-1,1]之間，相對於 ReLU 這類保持輸入值大於零部分原值的激活函數能有效規避掉極端值的問題。

DOI:10.6814/NCCU201900955

然而在 Dense 1 我們搭配 Softmax 形成一個能輸出 2 維機率向量的輸出層，Softmax 在深度學習中有非常廣泛的應用，如式(17)，它能將一個含任意實數的 K 維向量 x 壓縮到另一個 K 維實向量 Softmax(x)中，並使得每一個元素的範圍都介於(0,1)之間，且所有元素的和為 1。因此可以在分類問題中輸出每個類別各自的機率且保證各類別機率的總和為 1，如此良好的特性使得其常被應用在分類神經網路的輸出層。

$$\mathbf{Softmax}(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \quad \text{for } j = 1, 2, \ldots, K \qquad (17)$$

圖七，Softmax 函數



DOI:10.6814/NCCU201900955

在分類任務中使用 Softmax 有以下幾點好處:

1. Softmax 的輸出可以直接拿來計算損失函數，這邊的損失函數定義為交叉熵

$$\mathbf{Crossentropy}_j = -\ln\left(\frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}\right) \qquad (18)$$

(18)式中取 log 括號內的值就是訓練時輸出正確分類的 Softmax 值，因為 log 前面加上負號，使得整個數學式的含意就是輸出為正確類別的值越大則損失函數的值越小，可以透過訓練使這個交叉熵函損失函數達到最小，來極大化分類正確的機率。

2. 計算上非常簡易，當我們透過梯度下降法來使分類任務的損失函數達到最小時，我們需要對損失函數中每個權重矩陣偏微分來決定每次更新參數的幅度，以下我們對(18)式偏微分。

$$\frac{d\,L_j}{dx_j} = \frac{d-\ln\left(\frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}\right)}{dx_j} = \boldsymbol{P}_{x_j} - \mathbf{1} \qquad (19)$$

p_{x_j} *:即式(17)計算出的機率向量中對應正確類別之機率值*

由(19)式我們可以發現，我們只要將計算出來之機率向量中對應正確類別的機率減掉 1，就可以得到損失函數的偏導數了。因上述的優良特性，以及本文屬於分類任務，決定採用 Softmax 當最後輸出層的激活函數。

DOI:10.6814/NCCU201900955

Adam (Adaptive Moment Estimation)優化器

優化器是藉由梯度下降法(Gradient Descent)，來決定神經網路中權重更新的方向，因為梯度的方向是走向局部最大(Local Maximum)的方向，所以只要往梯度的反方向新參數，且調整適當的學習率參數(Learning Rate)理論上就可以到達損失函數的全局最小值(Global Minimum)。優化器從原始的單純往梯度反方向更新參數，經過一連串改良產生了許多種不同特性的優化器，而本文中 LSTM 模型訓練是使用 Adam 優化器，我們將解釋為何選擇使用 Adam，由 OpenAI 的 Diederik Kingma 於 2015 年提出[4]。

Adam 這個名稱是 Adaptive Moment Estimation 的縮寫，代表其根據損失函數對每個參數的一階梯度矩陣(式 20)以及二階梯度矩陣(式 21)來自適應的動態調整針對每個參數的學習率。其中(式 20)代表的意義為當下代送的動量項m_t等於前一期之動量m_{t-1}加上當下代送的梯度項g_t。(式 21)則表示v_t為梯度矩陣平方的移動平均。假設 t 時刻對損失函數內神經網路權重的一階導數(梯度)為$\boldsymbol{g_t}$，我們可以首先計算梯度與梯度平方的移動平均:

$$\boldsymbol{m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t} \qquad (20)$$

$$\boldsymbol{v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2} \qquad (21)$$

其中:

$$\boldsymbol{m_0 = 0}$$

$$\boldsymbol{v_0 = 0}$$

$$\boldsymbol{\beta_1 、\beta_2 \in [0,1)}$$

DOI:10.6814/NCCU201900955

然而因為(20)、(21)式移動均值的初始化為 0 向量，所以估計值會偏差向 0，特別是在初始時間步和衰減率非常小($\boldsymbol{\beta}$接近$\mathbf{1}$)的時候，所以我們透過(22)、(23)式操作來得到偏差修正(Bias-Corrected)的估計:

$$\widehat{\boldsymbol{m}}_{\boldsymbol{t}} = \frac{\boldsymbol{m}_{\boldsymbol{t}}}{\mathbf{1}-\boldsymbol{\beta}_{\mathbf{1}}^{\boldsymbol{t}}} \qquad (22)$$

$$\widehat{\boldsymbol{v}}_{\boldsymbol{t}} = \frac{\boldsymbol{v}_{\boldsymbol{t}}}{\mathbf{1}-\boldsymbol{\beta}_{\mathbf{2}}^{\boldsymbol{t}}} \qquad (23)$$

最後我們可以得到 Adam 梯度下降優化器的結果 :

$$\boldsymbol{\theta}_{\boldsymbol{t}+\mathbf{1}} = \boldsymbol{\theta}_{\boldsymbol{t}} - \boldsymbol{\tau} \cdot \frac{\widehat{\boldsymbol{m}}_{\boldsymbol{t}}}{\sqrt{\widehat{\boldsymbol{v}}_{\boldsymbol{t}}}+\boldsymbol{\epsilon}} \cdot g_t \qquad (24)$$

(24)式中$\boldsymbol{\tau}$是學習率，學習率越高更新參數的步伐就越快，但過大的學習率有可能導致收斂不了的情況，一般會建議取值 0.001 或以下，$\boldsymbol{\epsilon}$則是取值$\mathbf{10^{-8}}$，只是單純避免分母為 0 的情況產生。

然而從(24)式中我們理解到 Adam 的幾個良好特性: 1.Adam 會根據梯度以及梯度平方的移動平均來自適應的調整學習率，因為公式中參考了過去的梯度，帶入物理學中動能的概念，可以避免梯度消失或是卡在局部最小值。2.Adam 除了一階梯度外，還充分利用了梯度的二階移動平均，並使用(22)、(23)式中的$\boldsymbol{\beta_1}$、$\boldsymbol{\beta_2}$控制這些移動平均的衰減率，並有效修正估計的偏差，使得 Adam 在處理非穩態任務有很優秀的性能。

如果連續多次代送的梯度皆很小，使用 Adam 優化器仍然能保有一定的學習率來更新參數；相反的如果每次代送的梯度波動較大，則梯度平方的移動平均v_t會變大，而將學習率保持在一定的大小，不會產生梯度爆炸無法收斂的情形。在本文中基於 LSTM 的金融時間序列高頻資料的分類任務，Adam 優化器可以有很好的發

DOI:10.6814/NCCU201900955

揮。

批量標準化(Batch Normalization)

批量標準化由 Google(2015)[20]年提出的一種優化神經網路訓練的方法，Batch 就是在神經網路訓練中一批一批送進激活函數的資料，批量標準化指在每個 Batch 中做標準化，可以使得神經網路訓練更穩定，收斂的更快。

首先定義要進入激活函數的一批數據為一個 Batch 記為β：

$$\beta = \{x_1、x_2、\ldots\ldots x_m\} \quad (25)$$

我們可以在(25)式內做標準化:

$$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad (26)$$

$$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_\beta\right)^2 \quad (27)$$

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \quad (28)$$

經過 26)、(27)、(28)式運算，我們可以使得一組要進入激活函數前的數值較為壓縮，去掉極端值的影響，可以有效解決梯度消失以及梯度爆炸的問題。

DOI:10.6814/NCCU201900955

Dropout

Dropout 也是一個訓練神經網路的技巧，他第一次被提出是 Geoffrey E. Hinton(2012)神經網路之父發表的機器視覺領域開山之作《ImageNet Classification with Deep Convolutional》**[10]**，他的概念非常簡單，我們在訓練大型神經網路時，如果訓練數據不夠多或是數據缺乏多樣性無法反應出真實母體。會造成神經網路在訓練時針對訓練數據集客製化神經網路的權重，造成我們訓練好的神經網路在訓練資料集上得到很好的表現，但在神經網路沒看過的測試資料集上則因為權重過於針對訓練資料集客製化而效能大幅退步，產生所謂的過度擬合(Over Fitting)。因此我們在神經網路層與層連接之間加入一個 Dropout 層，也就是每次訓練代送時就隨機的讓固定比例的神經網路不被激活，也就是暫時不參與訓練。如此降低神經網路因為自身架構龐大去過度擬合訓練資料集的狀況產生**[20]**，在本文模型中考量到網路的尺寸與架構並不大，因此簡單選取 Dropout 比例為 0.3。

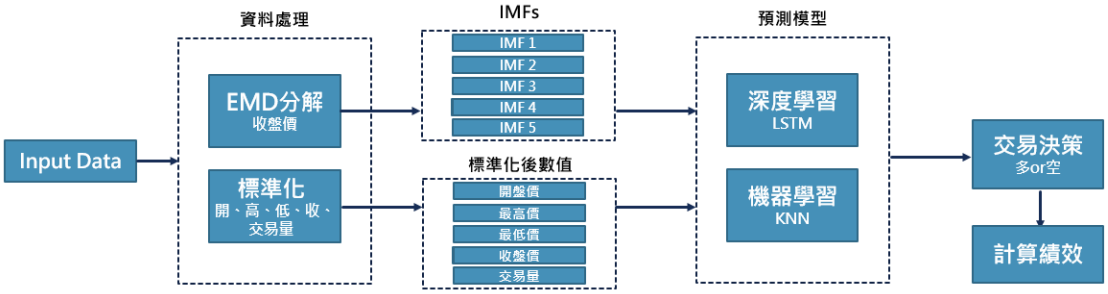
本實驗使用以上架構的 LSTM 神經網路訓練，每個神經網路訓練 50 次，在單張 GTX1060 顯示卡以及 i7-8750h 處理器的配備下，一個神經網路訓練 50 次大約需要 8 小時的時間，可供需要自行訓練的研究人參考。

DOI:10.6814/NCCU201900955

四、實驗

4.1 實驗設計

圖八，實驗流程圖



如圖八所示，本文實驗包含 2 種資料處理方式：1.EMD 分解、2.Z 值標準化，以及 2 種模型：1.LSTM、2.KNN，所以共會有 2×2=4，4 種模型，而我們分別計算這四個模型的準確度(Accuracy)、精密度(Precision)，然後將模型實際放到歷史資料上做回測，之後計算交易策略的夏普比例(Sharpe Ratio)衡量交易績效，並比較不同預測區間長度之間的效能，我們會在 4.2 小節中詳細描述我們使用的衡量指標。

DOI:10.6814/NCCU201900955

4.2 衡量指標

4.2.1 判斷模型效能的指標

首先先定義幾個計算指標會使用到的名詞：

1.True Positive(TP)，表示預測為真且真實為真的樣本數，定義為真就是上漲

2.True Negative(TN)，表示預測為假且真實為假的樣本數，定義為假就是下跌

3.False Positive(FP)，表示預測為真，但是真實為假的樣本數

4.False Negative(FN)，表示預測為假，但真實為真的樣本數

我們可以用 TP、TN、FP、FN 這 4 個數字來計算我們需要的幾個衡量模型效能的指標，如準確率(Accuracy)、精確度(Precision)、招回率(Recall)。

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \tag{29}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{30}$$

準確率可以讓我們初步的判斷模型一個較概括的效能，而我們可以再細部的去觀察精確度，來看模型對於上漲、下跌趨勢的命中率，可以視為交易上常觀察的空單、多單勝率。

DOI:10.6814/NCCU201900955

4.2.2 交易績效指標

首先我們列出原始的夏普比率(Sharpe Ratio)公式：

$$\text{SHARPE RATIO} = \mathrm{E}\left(R_x - R_f\right)/StdDev(x) \qquad (31)$$

原始的夏普比率的公式為期望年化報酬率去扣掉市場無風險報酬率，再除上整體交易報酬率的標準差，其直觀意義為投資人每承擔一單位風險可以帶來的超額報酬有多少，而衡量風險的標準就是報酬率的標準差，也代表著交易策略的波動程度，但是在本文中研究的任務是不跨日當沖交易策略，所以我們參考 Navon, Ariel. Yosi Keller(2017)**[16]**以及演算法交易學習網站 Quantinsti[25]的做法將原始的夏普比率做修改，用來判斷當沖或者較頻繁進出場之交易策略的績效。

修改過後的夏普比率:

$$\text{SHARPE RATIO} = \sqrt{N} \cdot \mathrm{E}(R_x)/StdDev(x) \qquad (32)$$

其中因為不跨日交易市場無風險利率可視為 0，而我們可以計算所有交易的平均報酬率以及標準差來估計$\mathrm{E}(R_x)$、$StdDev(x)$，N 則代表每日執行的當沖交易次數(例如 t=10，則 N=(300-60)/10 = 24)，並依照(32 式)計算，我們可以用這個新的夏普比率來衡量我們依照模型預測的結果做當沖交易的績效。

[25] Quantinsti 演算法交易學習網：https://blog.quantinsti.com/sharpe-ratio-applications-algorithmic-trading/

DOI:10.6814/NCCU201900955

4.3 實驗數據

模型測試的區間是(2017/07/01~2018/02/22)，其中不進行跨日交易，首先我們要先確定，在各預測區間長度之下，不同預測區間長度的監督式測式資料集內標籤(上漲、下跌)的狀況：

表一，監督式資料集內原始 1 類別(上漲)比例

	5	10	15	20	25	30	35	40	45	50	55	60
One_Class_Percentage	0.571933	0.55978	0.549838	0.546287	0.544203	0.54369	0.541006	0.539399	0.539148	0.536804	0.533195	0.531371

表一內的原始比例，可以協助我們衡量模型學習的狀況，由圖一我們可以發現在測試區間內台指期大概呈現上漲走勢也反應出在預測長度區間資料集中 1 類別(上漲)的數量都較多，因此我們以各預測長度區間中 1 類別(上漲)的比例為比較基準點，我們實驗的 4 個模型預測出的結果必須優於各自的比較基準點，我們才能判斷模型預測是有實質意義的。以預測區間 5 期的測試資料集為例，其 1 類別(上漲)的比例就佔了約 57%，由於在完全不使用模型預測單純全部猜測為上漲類別的狀況下，我們就可以得到至少 57%的準確率，因此我們模型預測的結果必須優於 57%才能算是比較好的結果。

DOI:10.6814/NCCU201900955

4.3.1 不同模型與資料處理方式的效能交叉比對

首先我們總共會有 4 個不同訓練方法的模型分別是 ：

1. Nor_KNN：使用 Z 值標準化資料處理與 KNN 演算法

2. Nor_LSTM：使用 Z 值標準化資料處理與 LSTM 演算法

3. EMD_KNN：使用 EMD 分解資料處理與 KNN 演算法

4. EMD_LSTM：使用 EMD 分解資料處理與 LSTM 演算法

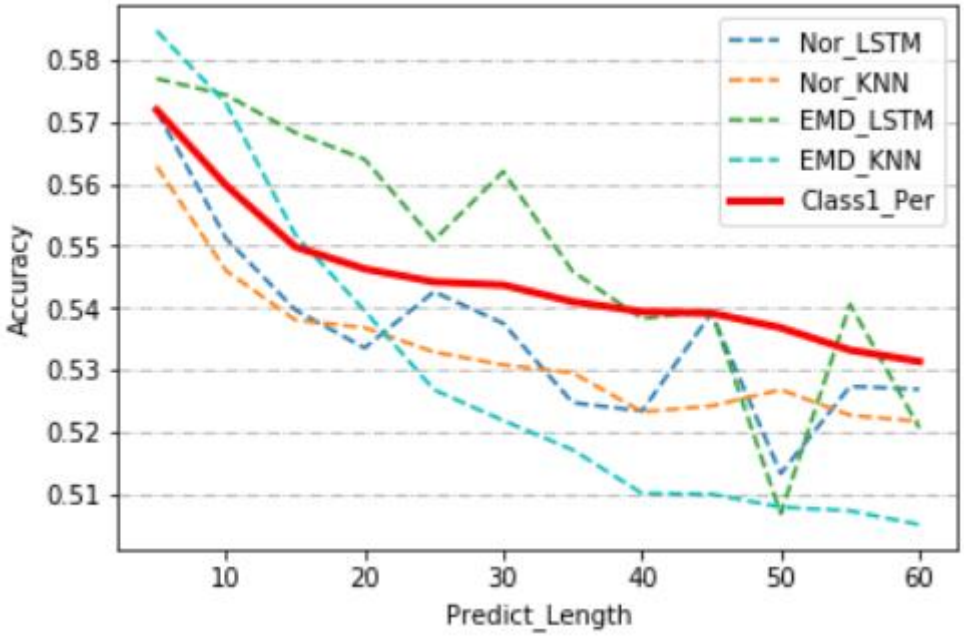
我們先簡單觀察 4 種模型的整體準確度:

表二，整體準確率

Predict Length	Class1_Per	Nor_LSTM	Nor_KNN	EMD_LSTM	EMD_KNN
5	0.571933	0.571758	0.562855	0.576970	0.584679
10	0.559780	0.551226	0.546012	0.574357	0.572998
15	0.549838	0.539676	0.538060	0.568253	0.551810
20	0.546287	0.533546	0.536822	0.563961	0.539548
25	0.544203	0.542617	0.532886	0.550924	0.526805
30	0.543690	0.537511	0.530829	0.561991	0.521838
35	0.541006	0.524719	0.529542	0.545876	0.517128
40	0.539399	0.523375	0.523248	0.538268	0.510109
45	0.539148	0.539166	0.524186	0.539558	0.509992
50	0.536804	0.513292	0.526819	0.506814	0.507888
55	0.533195	0.527354	0.522700	0.540591	0.507291
60	0.531371	0.526890	0.521682	0.520619	0.505066

DOI:10.6814/NCCU201900955

圖九，整體準確率與基準比較圖



表二，顯示 4 種模型在輸入過去 60 分鐘資料預測未來不同區間走勢的整體準確度，而 Class1_per 則是測試資料集中 1 類別(上漲)的占比，整體的準確度必須大於這這個基礎的 1 類別占比，才有實質上的意義。圖九，是我們將表一的結果視覺化呈現，紅色粗體線就是在不同的預測區間測試集中 1 類別的占比，也就是在粗體紅色線以上的部分就是表現優於單純全部預測上漲的模型組合，從圖中我們可以觀察到有使用 EMD 分解的 LSTM(綠)與 KNN(淺藍)，有較多的部分在紅線之上，其中綠色線的 EMD_LSTM 表現最優異，只有預測 40、50、60 的區間落在紅線之下，淺藍色的 EMD_KNN 則是在較短的預測區間內表現不錯，在短期 5 分鐘內表現甚至超越 EMD_LSTM，而使用 Z 值標準化的模型表現都不理想幾乎都在紅線之下，另外可以觀察到在 EMD 分解的模型在較短的預測區間內效能都會明顯優於使用 Z 值標準化處理的模型，KNN 模型的系統內使用 Z 值標準化甚至在 20 期以後大幅超過使用 EMD 分解的模型，可見 EMD 分解對於短期(5~20)的預測區間有較好的效果，也因此我們實驗的結果與 Ariel Navon, Yosi Keller(2017)**[16]**文中

DOI:10.6814/NCCU201900955

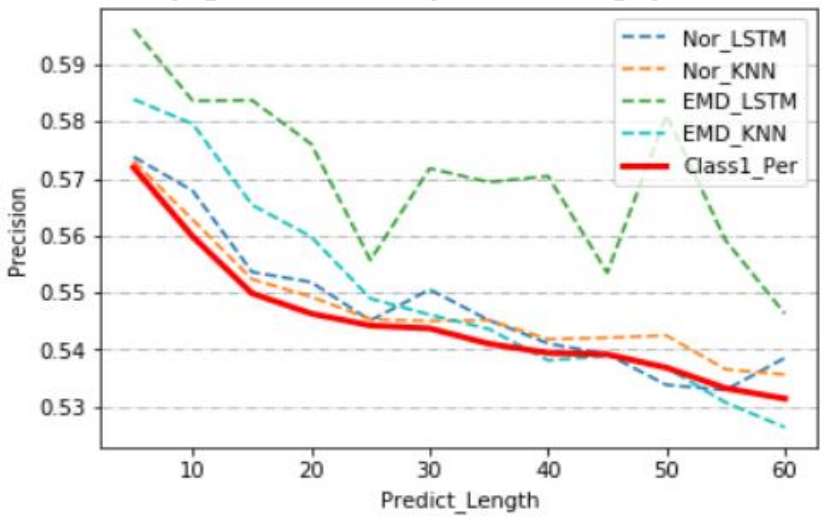
28 天為最佳預測區間長度的結果有所差異，但仔細觀察可以發現，使用 Z 值標準化([16]中所使用的方法)資料處理的 LSTM 約在 25 的預測區間長度有較接近 EMD 分解的 LSTM，與**[16]**中的最佳長度 28 十分接近。此外整體的趨勢基本上呈現預測區間越長整體準確率越低。

接著我們細部觀察兩個類別的精確度(Precision)

表三，上漲類別精確度

Predict Length	Class1_Per	Nor_LSTM	Nor_KNN	EMD_LSTM	EMD_KNN
5	0.571933	0.573815	0.573081	0.596225	0.583905
10	0.559780	0.567821	0.562739	0.583622	0.579568
15	0.549838	0.553591	0.552338	0.583742	0.565371
20	0.546287	0.551821	0.549253	0.576051	0.559787
25	0.544203	0.545074	0.545302	0.555683	0.548925
30	0.543690	0.550525	0.545011	0.571782	0.546099
35	0.541006	0.545140	0.545158	0.569371	0.543593
40	0.539399	0.541060	0.541807	0.570421	0.538084
45	0.539148	0.539166	0.542070	0.553468	0.538871
50	0.536804	0.533792	0.542462	0.581209	0.536816
55	0.533195	0.532884	0.536502	0.559140	0.530690
60	0.531371	0.538516	0.535595	0.546274	0.526337

圖十，精確度以及比較基準圖



DOI:10.6814/NCCU201900955

在將模型預測結果用於交易的狀況下，上漲類別的精確度可以看做是多單的勝率，也就是我每次下多單有多少的比例會賺錢，同樣的精確度必須要高於測試資料集中本身上漲樣本的比例才有價值，表三展示的是各個模型在不同預測區間長度的上漲類別精確度表現，藉由圖十的視覺化我們可以觀察到，有使用 EMD 分解的 LSTM 模型在對多單的處理上整體優於其餘模型，且 KNN 算法在使用 EMD 分解下的短預測區間有顯著提升，其餘模型與預測區間基本上就是沿著一類別比例基準線移動，只有些許震盪，個體差異並不大。

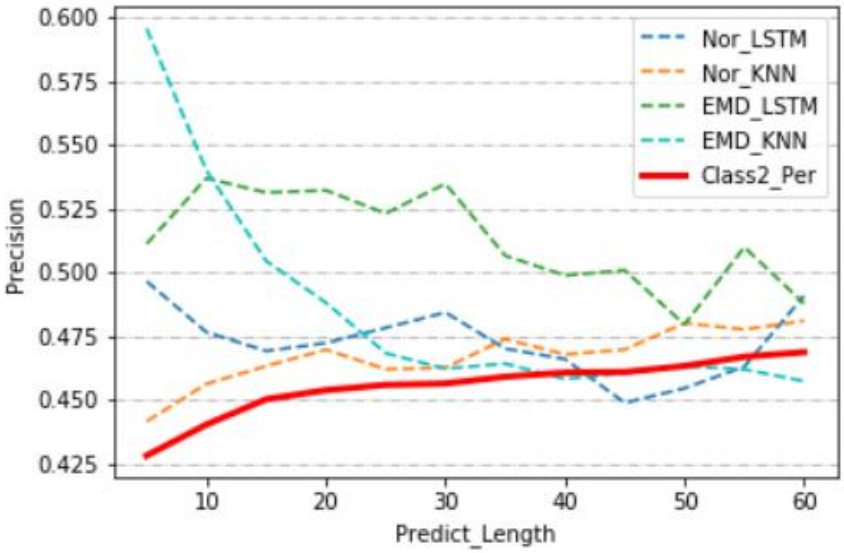
表四、下跌類別精確度

Predict Length	Class2_Per	Nor_LSTM	Nor_KNN	EMD_LSTM	EMD_KNN
5	0.428067	0.496425	0.441495	0.511100	0.595519
10	0.440220	0.476496	0.456184	0.536835	0.539916
15	0.450162	0.469100	0.463173	0.531205	0.504398
20	0.453713	0.472215	0.469638	0.532097	0.488023
25	0.455797	0.478191	0.461967	0.523014	0.468239
30	0.456310	0.484215	0.462571	0.534661	0.462177
35	0.458994	0.470009	0.473752	0.506495	0.464126
40	0.460601	0.465926	0.467736	0.498720	0.458109
45	0.460852	0.448752	0.469685	0.500744	0.460327
50	0.463196	0.454486	0.480043	0.479484	0.463169
55	0.466805	0.462947	0.477544	0.509809	0.461889
60	0.468629	0.490758	0.480930	0.487706	0.457255

政 治

DOI:10.6814/NCCU201900955

圖十一，精確度以及比較基準圖



表四展示的是各個模型在不同預測區間長度的類別下跌精確度表現，藉由圖十一的視覺化我們可以觀察到，可以觀察到下跌類別(紅色粗體線)，本身在測試資料集中的占比就比較少，反映出台灣近年來大盤呈上漲的趨勢，而有使用 EMD 分解的 LSTM 模型整體還是優於其餘模型，但值得注意的是 KNN 搭配 EMD 分解在短預測區間內的效果十分顯著，在 5、10 兩個預測區間內甚至超越 EMD_LSTM。

4.3.2 不同模型與資料處理方式的交易效能交叉比對

我們首先列出在期初投入 10000 元資金，經過約半年的不跨日交易後，帳戶內的餘額會是多少，交易規則：舉例來說如果預測區間 10 期的模型在 T 預測出上漲，則我們就相信模型的預測持有一口多單直到 T+10 期，再將 T – 50~T+10 這 60 期的資料送入模型得到下一個 10 期的預測結果，碰到跨日則在前一個完整的交易區間結束後出場不再交易，等到下一個交易日再從第 61 分鐘開始進場，並參考

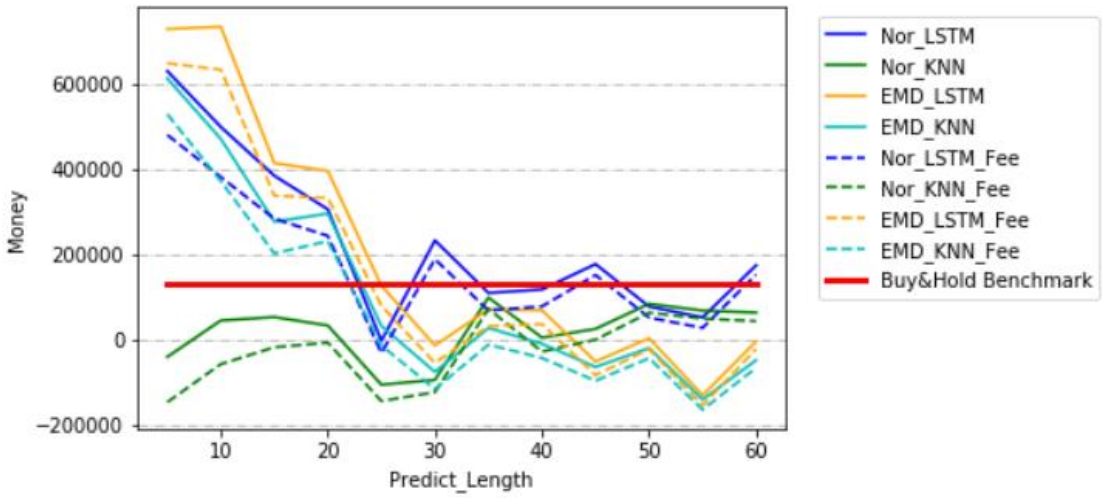
DOI:10.6814/NCCU201900955

台灣期貨商常用的手續費單次交易是 70 元[26]，交易區間為 2017/07/01~2018/02/22。

表五，各模型、區間交易後帳戶餘額

	5	10	15	20	25	30	35	40	45	50	55	60
Nor_KNN	-24600	-59200	109600	14600	-62200	-119400	73200	-5200	41000	-41200	19200	176200
Nor_LSTM	630600	499200	384400	305400	-3800	232200	108800	116800	176600	76800	51200	173800
EMD_KNN	613400	471200	276400	295400	31000	-77000	26400	-10800	-65400	-20800	-142000	-49400
EMD_LSTM	729400	734800	414000	395800	125800	-14600	70000	68000	-52600	2400	-132000	-5400

圖十二，各模型、區間交易後帳戶餘額與 Benchmark 比較



表五為有考慮交易手續費之交易後帳戶最後餘額(這邊的交易收入為了方便比較，並未考慮在交易過程中虧損過多被強制出場的狀況，換句話說我們假設投資人有足夠的保證金可以承受虧損)，為確定我們實驗中較短預測區間當沖交易不會因

[26] 依照各家期貨商的規定以及交易口數的多寡，交易一口台指期手續費在 50~100 台幣之間，我們取中間值 70 作為實驗中交易的手續費。

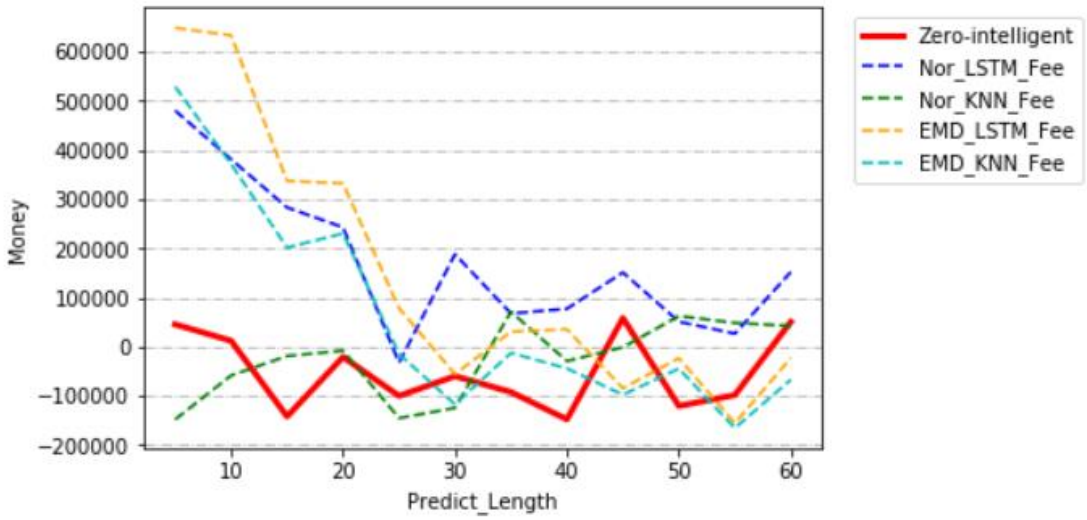
DOI:10.6814/NCCU201900955

為較頻繁的做多、放空造成獲利被手續費吃掉。因此我們在圖十二中把無考慮交易手續費的結果與有考慮交易手續的結果同時畫出，發現各模型以及各交易區間基本上小幅度往下平移並沒有太大的影響獲利情形。

然而在圖十二所示我們同時畫出(Buy&Hold)策略的交易餘額(粗紅線，129400 元)，主要考量到整體測試區間呈現上漲趨勢(牛市)，也就是我單純持有一口多單不做任何交易就可以獲得正報酬，因此任何的策略在考慮交易手續費的情況下，必須要有高於(Buy&Hold)策略的報酬才有採用的意義。

從圖十二中我們可以整體發現績效最好的模型是 EMD_LSTM，其次 Nor_LSTM，再來是 EMD_KNN，這三個模型在短、中期預測區間的獲利(5~25)都有在單純持有一口多單(Buy&Hold)的獲利以上。而 Nor_KNN 則是各個區間都在平均以下較不適合拿來實際交易，另外 Nor_LSTM 在 25 期之後的獲利反過來超過 EMD_LSTM，且 Nor_LSTM 分別在 30、40、60 的預測區間以有大於平均的表現，也反映出前面分析的 EMD 分解對於短、中期區間的預測幫助較大。

圖十三，各模型、預測區間與單純亂數猜測獲利比較圖



DOI:10.6814/NCCU201900955

上圖十三為各種模型和單純隨機亂數預測(Zero-intelligent)[27]的交易利潤比較，主要是考量到我們模型整體預測的準確度與 Benchmark(原始上漲類別比例)相當接近，可能會有根據模型預測結果做交易並未明顯優於隨機猜測交易之疑慮[7]，為了證明我們模型的交易績效因此我們會在實驗中加入與亂數猜測(Zero-intelligent)的結果比較。

從圖十二中我們可以發現在中短期預測區間我們的模型普遍可以表現得比單純 Buy&Hold 還好。但是在較長的預測區間下模型普遍表現不如單純 Buy&Hold。當我們在圖十三中和單純的隨機猜測做比較時，同樣可以發現在中短期預測區間中(5~25)，除了 Nor_KNN 以外基於模型的交易大致上都有明顯優於隨機亂數猜測交易的利潤。但是隨著預測區間的拉長(30~60)，我們可以發現基於模型的交易與隨機亂數猜測交易的利潤越來越接近，甚至在某些區間(45、60)隨機亂數交易的利潤還超過基於模型的交易，也反映出圖十二中較常的預測區間利潤低於單純的 Buy&Hold。因次我們可以斷定除了 Nor_KNN 之外的模型，其交易績效是有顯著高於隨機亂數的，但在長期尤其是使用 EMD 分解的模型，交易利潤則不理想。

接下來我們觀察各模型、區間的夏普比率，來衡量風險承擔以及獲利之間的關係。

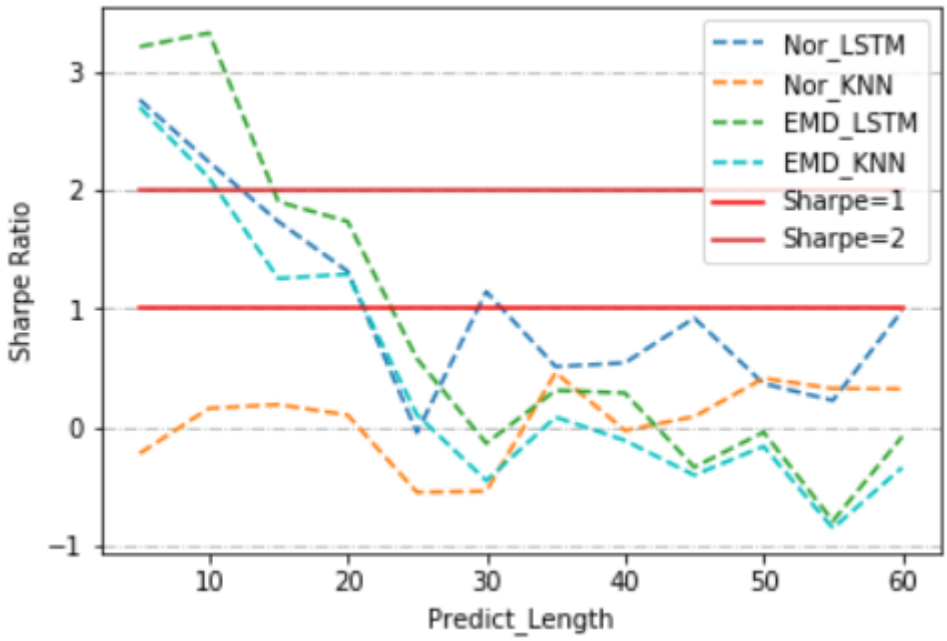
[27] 本文使用的隨機亂數預測結果稱為 Zero-intelligent，使用 Python Numpy 的 Random 套件產生，而圖十三中的利潤為做 10 次隨機亂數的平均結果。

DOI:10.6814/NCCU201900955

表六，各模型、預測區間長度的夏普比例

	5	10	15	20	25	30	35	40	45	50	55	60
Nor_KNN	-0.224942	0.156937	0.189061	0.099887	-0.553388	-0.544144	0.459278	-0.030167	0.086873	0.410970	0.324085	0.321217
Nor_LSTM	2.767165	2.241030	1.736243	1.316182	-0.048223	1.144066	0.509878	0.541085	0.920447	0.365649	0.223823	0.983118
EMD_KNN	2.703711	2.107093	1.255126	1.292870	0.102872	-0.454900	0.082593	-0.114843	-0.408942	-0.164811	-0.857700	-0.343770
EMD_LSTM	3.216705	3.332844	1.905412	1.737510	0.577104	-0.138860	0.309104	0.286392	-0.341337	-0.045232	-0.796450	-0.081775

圖十四，各模型、預測區間長度的夏普比率圖

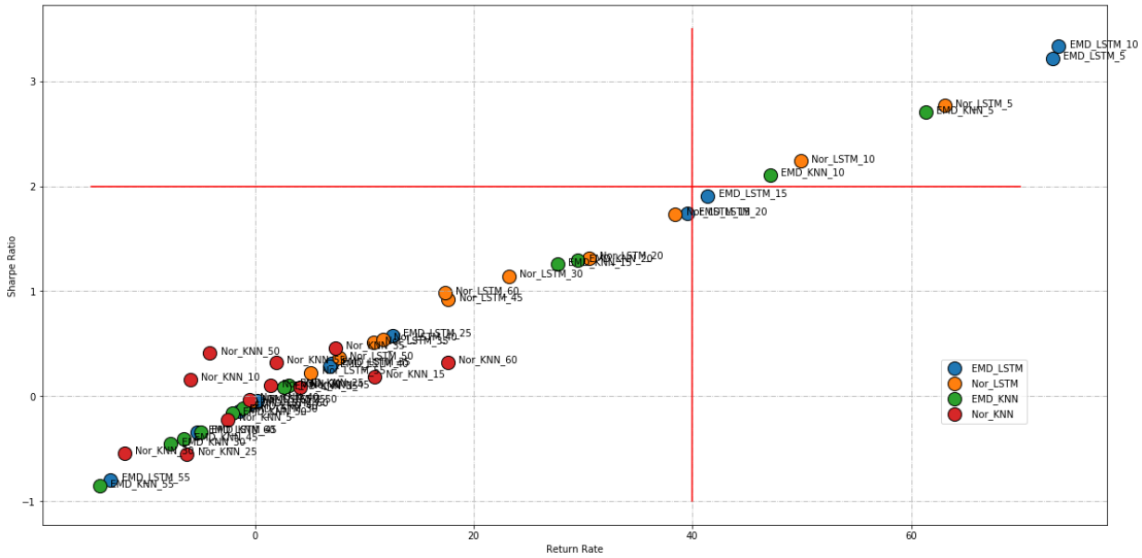


如表六、圖十四所示，以夏普比例來看仍是 EMD_LSTM 為最佳策略，在中短期預測區間(5~25)區間，EMD_LSTM、Nor_LSTM、EMD_KNN 都有夏普比例大於 1 的表現，而在短期預測區間(5~15)，EMD_LSTM、Nor_LSTM、EMD_KNN 更是有大於 2 以上的表現，其中 EMD_LSTM 有高達 3 以上的水準，由夏普比例分析我們可以得知 EMD 分解對於中、短預測區間的準確率、夏普比率都有顯著的幫助，但在長期(30~60)使用 EMD 分解的夏普比率反而不如使用 Z 值標準化，反映出模型的預測效能也在中短期的預測區間長度較好，而 Nor_KNN 在所有預測區間之下都沒有達到 1 以上，但 Nor_LSTM 同樣使用 Z 值標準化資料處理在中、短期區間依然有不錯的績效，證明 LSTM 在處理時間序列資料上優於 KNN 算法。

DOI:10.6814/NCCU201900955

以下我們同時觀察夏普比例以及交易總報酬率(最後餘額/原始投資)

圖十五，夏普比率以及交易總報酬率



圖十五中，橫軸是報酬率;縱軸是夏普比例，水平的紅線表示夏普比例 2 的分水嶺，垂直的紅線表示半年累積 40 萬報酬以上的分水嶺，在圖中越往右上角接近的位置表示該交易策略有高報酬率同時具有高的夏普比例代表在追求高報酬率的同時還可以將風險維持在可接受的範圍，圖十五所呈現的結果大致上和前面分析的相同，即短、中期的 EMD_LSTM、Nor_LSTM、EMD_KNN，有較好的績效，其中又以 5、10 兩個預測區間最為優秀，都保持著 2 以上的夏普比率同時在半年左右的時間有 40 倍以上的報酬率，表示短預測區間長度的模型除了有較高的準確率以及因為短預測區間台指期貨的變動有限，報酬率的波動不會太大，所以形成了高報酬率、高夏普比率以模型預測為基礎的當沖交易策略。

DOI:10.6814/NCCU201900955

五、結論與展望

5.1 結論

在本實驗中，我們嘗試結合專門用來處理序列相關資料的人工智慧深度學習 LSTM 神經網路以及專門用來將非平穩性序列資料拆解成數個有頻率意義信號的 EMD 分解，藉由交叉比對實驗證明我們使用的 EMD_LSTM 不管在模型的趨勢預測效能方面，或是實際依照模型結果進行當沖交易的交易績效都比傳統的機器學習方法 KNN 以及 Z 值標準化都有較佳的效果，以下我們整理從本實驗中獲得的幾個重要結論:

I. LSTM 神經網路在相同的資料處理方式下效能明顯優於 KNN 演算法 :

在實驗中我們交叉比較深度學習演算法(LSTM)與傳統機器學習演算法(KNN)，分別使用 EMD 分解與傳統 Z 值標準化處理後的資料做訓練後的模型效能，實驗證明在相同的資料處理流程下，LSTM 的效能整體會優於 KNN，實際將模型用於台灣指數期貨交易時 LSTM 也能得到較高報酬以及夏普比率的結果，因此在本實驗中我們可以得到 EMD-LSTM 是相對較有效可以用來預測期貨指數走勢以及當沖交易的方法之結論。

II. EMD 分解能有效提升 KNN、LSTM 模型對於未來趨勢預測的準確度:

從圖九、十、十一我們可以觀察到使用 EMD 分解的 LSTM 模型效能大致上都明顯優於傳統Z值標準化，但值得注意的是在使用 KNN 的狀況下使用 EMD 分解僅在短期預測時效能會優於 LSTM，也許和 LSTM 具有長期記憶功能有相關。

DOI:10.6814/NCCU201900955

III. 預測區間越長、模型效能、交易績效越差

在交叉比較 LSTM 與 KNN 與兩種資料處理方式的同時，我們也詳細的觀察設定不同長度的預測區間對於模型預測效能的影響，最後發現大致上呈現越長的預測區間模型的效能越差的趨勢，同時也有較差的夏普比率。

IV. EMD 分解搭配 LSTM 神經網路能實際在市場上獲利且有相對較高的績效

實際依照模型預測出的結果在測試資料集內進行交易回測，並以交易的報酬率、夏普比率指標來衡量模型交易的績效，最後得到短預測區間的 EMD-LSTM 能穩定獲利，有最好的交易績效之結果。

5.2 展望

本文研究將人工智慧用於預測金融時間序列趨勢，並與傳統的機器學習方法做比較，最後實際用於台灣指數期貨當沖交易，也就是將 LSTM 神經網路結合 EMD 分解預測未來趨勢的漲、跌以及交易，經過實驗證實取得不錯的成果。

然而由於運算資源的限制，本次研究僅使用近年較具代表性的 LSTM 神經網路與傳統上具有代表性的 KNN 演算法做比較，但近年來也發展出許多可以做金融時間序列趨勢預測的機器學習模型如 SVM、XGBoost 等，在本次研究的基礎下未來希望能加入更多具代表性的機器學習模型做比較。此外同樣受限於運算資源的限制，本次研究將資料的頻率限制在分鐘，在往後的研究中期望可以將資料的頻率進一步提升到 Tick 等級，也就是逐筆交易的紀錄，在使用逐筆交易的情況下，由於資料量龐大我們需要更強大的運算資源，以及結構更龐大的神經網路，但如果能在 Tick 等級的資料達到相同不錯的效果，我們就能以速度的優勢賺取更高的超額報酬，對於將人工智慧落實於金融市場交易的理想，也能更接近一步。

DOI:10.6814/NCCU201900955

參考文獻

[1] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi.(1994) “Learning long-term dependencies with gradient descent is difficult.” Neural Networks, IEEE Transactions on 5.2 (1994): 157-166.

[2] Cover, T.；P. Hart(1967) .“Nearest neighbor pattern classification.” in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967. doi: 10.1109/TIT.1967.1053964

[3] Cox, D.R. (1958). “The Regression Analysis of Binary Sequences.” Journal of the Royal Statistical Society: Series B, 20, 215-242.

[4] Diederik, Kingma & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[5] Doering , Jonathan & Fairbank, Michael & Markose, Sheri. (2017). “Convolutional neural networks applied to high-frequency market microstructure forecasting.” 31-36. 10.1109/CEEC.2017.8101595.

[6] Fisher, R.A. (1936). “The Use of Multiple Measurements in Taxonomic Problems.” Annals of Eugenics, 7, 179-188.

[7] Gode, D. K., & Sunder, S. (1993). Alloca;ve efficiency of markets with zero-intelligence traders: Market as a par;al subs;tute for individual ra;onality. Journal of poli;cal economy, 101(1), 119–137

[8]Hochreiter, Sepp, and Jürgen Schmidhuber(1997). “Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

[9] Huang, Norden E.；Zheng Shen；Steven R. Long3(1998). “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis” 454Proc. R. Soc. Lond. A

[10] Krizhevsky, Alex & Sutskever, Ilya & E. Hinton, Geoffrey. (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” Neural Information Processing Systems. 25. 10.1145/3065386.

[11] Le, Quoc V. Navdeep Jaitly, Geoffrey E. Hinton(2015). “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units”.arXiv:1504.00941v2 [cs.NE] 7 Apr 2015

[12] Li, Edwin (2018). “LSTM Neural Network Models for Market Movement Prediction” (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-231627

[13] Lipton, Zachary C. John Berkowitz, Charles Elkan(2015). “A Critical Review of

DOI:10.6814/NCCU201900955

Recurrent Neural Networks for Sequence Learning." arXiv:1506.00019v4 [cs.LG] 17 Oct 2015

[14] Loffe, Sergey. Christian Szegedy(2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." arXiv:1502.03167v3 [cs.LG] 2 Mar 2015

[15] McCulloch, Warren S.；Walter Pitts(1943). "A logical calculus of the ideas immanent in nervous activity." Bulletin of Mathematical Biology, 52, 99-115.

[16] Navon, Ariel Yosi Keller(Nov 2017). "Financial Time Series Prediction using Deep Learning." arXiv:1711.04174v1 [eess.SP] 11 Nov 2017

[17]Rosenblatt, F(1958). "The perceptron: A probabilistic model for information storage and organization in the brain." _Psychological Review_ 65 (6):386-408.

[18] Rumelhart, David E Geoffrey E. Hinton, Ronald J. Williams(1986). "Learning representations by back-propagating errors" . Nature. 323 (6088): 533–536. doi:10.1038/323533a0. ISSN 1476-4687.

[19] SUBHA, M.V & Nambi, S.T.. (2012). "Classification of stock index movement using k-nearest neighbours (k-NN) algorithm." WSEAS Transactions on Information Science and Applications. 9. 261-270.

[20] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov(2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

[21] Teixeira, LA & Oliveira, A.L. (2010). "A method for automatic stock trading combining technical analysis and nearest neighbor classification." Expert Syst. Appl., 37, 6885-6890.

[22]Williams, R. J. (1989). "Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27". Boston: Northeastern University, College of Computer Science.

[23]Zhang, Boning. (2018). Foreign exchange rates forecasting with an EMD-LSTM neural networks model. Journal of Physics: Conference Series. 1053. 012005. 10.1088/1742-6596/1053/1/012005.

[24] Zheng, Huiting & Yuan, Jiabin & Chen, Long. (2017). "Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation." Energies. 10. 1168. 10.3390/en10081168.

DOI:10.6814/NCCU201900955