模擬投資策略下之演算法比較-以時間序列預測為例

研究生　　林奇鋒
指導教授　　李俊賢

摘要

本篇論文主要探討不同演算法在時間序列上的預測效能，並透過模擬的投資策略去做效能測試。時間序列的預測是一個很廣泛且重要的議題。我們透過一般的複數神經模糊集系統(CNFS)，我們可得到一組複數型態的輸出，實數部分和虛數部分可針對不同目標做預測。模型設計上，前鑑部使用高斯型態的複數神經模糊集 (CFSs)，後鑑部則使用 Takagi-Sugeno 的線性函式，兩者使用 IF-THEN 規則連結。此外，為了最佳化模型的預測結果，我們前鑑部使用不同的演算法，粒子群演算法 (Particle Swarm Optimization)以及人工蜂群演算法(Artificial Bee Colony Optimization)進行參數優化，後鑑部則使用 Recursive Least Squares Estimator (RLSE)進行參數優化。最後我們將用三個實驗測試在不同的學習演算法下效能的差異性。

關鍵詞：complex neuro-fuzzy system (CNFS); complex fuzzy set (CFS); time-series forecasting

1. 前言

人工智慧在現實中有很多的領域，從18、19世紀就有許多相關的研究，但是由於硬體的設備問題，導致成果一直無法應用在一般的生活中。如今隨著硬體設備的發展，人工智慧已經對人類帶來不少的幫助，研究更是多不勝數，像是財經方面的預測[1]，圖像辨識方面[2][3]、遊戲方面[4]等。上述這些有一部分屬於時間序列的預測，時間序列的預測是一個很重要的議題，因為他在真實世界中的應用範圍非常廣。過去已經有很多學者提出不同的方法來針對時間序列的預測問題，像是ARIMA[5][6]、模糊理論、神經網路運算、神經模糊混合系統等。其中，最常被提出的就是神經模糊混合系統[7]-[10]。

神經模糊混合系統 (NFSs)是基於神經網路 (Neural Network)的延伸，所謂的神經網路是一種用多層神經元所串連起來的網路，透過層與層之間的資料傳輸來得到輸出，現今火紅的深度學習即是神經網路的一種，在此領域中已有不少的研究[2][3][21]。

而神經網路的延伸，神經模糊混合系統(NFSs)一直是被廣泛研究的模型，其中，類神經網路系統有所謂的 IF-THEN 規則，這些規則如同我們人類的經驗法則。通常可以將 IF-THEN 的規則結合模糊理論，使整體的架構更有彈性，我們稱之為神經模糊混合系統。如上述所提，神經模糊混合系統的特性使他對時間序列的預測有著不錯的效果。所以現在有關時間預測這方面的研究，大多採用類神經網路為模型架構。

本實驗是採用神經模糊混合系統的作法，將整體模型效仿 IF-THEN 規則，建造成多層神經元架構。在模型實作方面，我們將模糊

理論結合類神經網路系統，形成神經模糊混合系統，前鑑部使用高斯型態的球型複數神經模糊集，後鑑部則使用 Takagi-Sugeno 的線性函式[17]，前鑑部和後鑑部則透過 IF-THEN 規則結合。透過此模型和機器學習我們預期對時間序列的預測可以更加精準。

關於模糊集，在 1965 年由 Zadeh 學者，最先提出模糊集的概念[11]，使資料可以透過某個函式，得到介於 0 到 1 之間的歸屬程度 (Membership degree)。之後在 2002 年，有另一篇研究提出了複數模糊集 (CFSs)[12]的概念，歸屬程度可以透過函式得到一個複數型態的值，這使得歸屬程度可以呈現在一個半徑為 1 的複數單位圓盤中。這個概念使原本能表示的歸屬程度更加的豐富。一般來說我們可以透過複數型神經模糊集系統 (CNFS)[13][14]，得到一組複數型態的輸出，而實數部分和虛數部分可針對不同目標做預測，所以可針對兩個不同目標。目前兩個目標的預測已經有很多的研究產出[6][13][14]。

在本研究中，為了使資料可以有效的被應用，在資料前處理的部分，我們根據將原始資料的30個漲跌值作為特徵，並透過夏農資訊熵 (Shannon Entropy)[15]，去計算他們個別對目標的資訊貢獻量，除此之外，我們透過了多目標特徵選取的概念[16]，算出每個特徵對目標的有效資訊量，以此作為挑選訓練資料的依據。從資料中萃取出最有效的資料，除了降低模型的運算負擔，也能有效的提升預測的效能。最後，機器學習部分，我們使用知名的粒子群演算法 (PSO)[18]以及 Artificial Bee Colony Optimization (ABCO)[22] 兩種演算法個別結合廣為人知的 Recursive Least Square Estimator (RLSE)[19]進行參數優化，並將它們分別整合成三種方法我們稱之為 PSO-RLSE [20]和 ABCO-RLSE。我們將前鑑部和後鑑部參數藉由不同的演算法訓練，想透過 divide-and-conquer 原理，降低搜尋的維度，使模型更容易找到最佳解，提高整體的效能。

2. 研究方法

2.1 多目標特徵挑選

為了使得資料被有效的應用，本實驗將對資料做前處理，以降低運算成本及提升模型精準度。一開始原始資料標記為$\{x_i^{(j)}, i = 1,2,\dots,n, j = 1,2,\dots,N\}$，其中，$n$為每組資料的總筆數，$N$為資料的總組數。接著將所有原始資料做一次差分，公式如下:

$$x_{i+1}^{(j)} - x_i^{(j)} = d_i^{(j)}, i = 1,2,\dots,n-1 \quad (1)$$

其中，n為資料的總筆數，j為第j組數據。將差分過後的資料漲跌形成30個特徵值，第一組數據，擁有的特徵值標記為f_1到f_{30}，而第二組數據的特徵值標記為f_{31}到f_{60}，依此類推。透過多目標的特徵選取方法[16]，我們可以從特徵中取得訓練資料。多目標的特徵選取方法與資訊熵有關，一般來說，資訊熵 (Entropy)是指接收的每條消息中包含的資訊的平均量，若資訊的隨機性越高，則資訊熵值會越高。公式如下:

$$H(x) = \int p_d(x) log(\frac{1}{p_d(x)}) dx \quad (2)$$

其中，$H(x)$為x的期望值，$p_d(x)$為x的機率密度，但若x的機率密度大於 1，則log部分會是負數，會影響到整體的期望值，所以我們對公式做了一些更改，更改後的公式如下:

$$H(x) = \int p_d(x) log(\frac{\varphi_1}{p_d(x)}) dx \quad (3)$$

$$\varphi_1 = \max(1, \varepsilon + \max(p_d(\mathrm{x}))) \quad (4)$$

其中，$\varepsilon = 1 \times 10^{-10}$。

由於我們特徵的選擇是針對目標，所以我們透過資訊熵的概念，計算每個特徵與目標之間的影響資訊量，公式如下:

$$I_{x\to y} = I(X^+, y) \int_0^\infty p_d(x) dx + I(X^-, y) \int_{-\infty}^0 p_d(x) dx \quad (5)$$

其中，$I_{x\to y}$為特徵x對目標y的影響資訊量，$I(X^+, y)$為特徵x為正數以及目標y的互資訊，$I(X^-, y)$為特徵x為負數以及目標y的互資訊，公式如下:

$$I(X^+, y) = H(Y) - H\big(Y(X^+)\big) \quad (6)$$

$$I(X^-, y) = H(Y) - H\big(Y(X^-)\big) \quad (7)$$

其中，$H(Y)$為目標的期望值，$H\big(Y(X^+)\big)$為特徵x為正數時所對應的目標y的期望值，$H\big(Y(X^-)\big)$為特徵x為負數時所對應的目標y的期望值。公式如下:

$$H(Y(X^+)) = \iint p_d(y(x^+))p_d(x^+)\, log\left(\frac{\varphi_2}{p_d(y(x^+))}\right) dydx^+ \quad (8)$$

$$\varphi_2 = max(1, \varepsilon + max\left(max\big(p_d(y)\big), max\left(p_d\big(y(x^+)\big)\right)\right)) \quad (9)$$

$$H(Y(X^-)) = \iint p_d(y(x^-))p_d(x^-)\, log\left(\frac{\varphi_3}{p_d(y(x^-))}\right) dydx^- \quad (10)$$

$$\varphi_3 = max(1, \varepsilon + max\left(max\big(p_d(y)\big), max\left(p_d\big(y(x^-)\big)\right)\right)) \quad (11)$$

其中，$\varepsilon = 1 \times 10^{-10}$，$p_d(x^+)$為特徵x為正數時的機率密度，$p_d(y(x^+))$為特徵x為正數時所對應的目標$y$的機率密度，$p_d(x^-)$為特徵x為負數時的機率密度，$p_d(y(x^-))$為特徵x為負數時所對應的目標$y$的機率密度。

透過上述影響資訊量的公式，可以得到每個特徵對每個目標的影響資訊量，此時就能依據這些影響資訊量做多目標的特徵選取，步驟如下:

Step 1: 算出第i個特徵對第j個目標的selection gain 標記為$g(f_i \to t_j)$，其中，f_i為第i個特徵變數，t_j為第j個目標變數。selection gain 公式如下:

$$g(f_i \to t_j) = I_{f_i \to t_j} - R_{f_i \to SP^{(j)}} \quad (12)$$

其中，$I_{f_i \to t_j}$為f_i對t_j的影響資訊量，$R_{f_i \to SP^{(j)}}$為f_i對$SP^{(j)}$中已存在特徵的冗餘資訊量，$SP^{(j)}$為第j個已選特徵池。冗餘資訊量公式如下:

$$R_{f_1 \to SP^{(j)}} = \frac{1}{2|SP|}\sum_{k=1}^{|SP|}\{I_{f_i \to s_k} + I_{s_k \to f_i}\} \quad (13)$$

其中，$|SP|$代表$SP^{(j)}$內的特徵個數，$I_{f_i \to s_k}$為f_i對$SP^{(j)}$內的第k個特徵變數的影響資訊量，$I_{s_k \to f_i}$為$SP^{(j)}$內的第k個特徵變數對f_i的影響資訊量。經過上述計算若$g(f_i \to t_j)$大於 0，則將特徵f_i加入第j個已選特徵池$SP^{(j)}$中。

Step 2: 無論重疊與否，將所有已選特徵池$\{SP^{(j)}, j = 1,2, \dots, |TS|\}$中出現過的特徵變數記錄下來，儲存成$\Omega$，$\Omega = \{\phi_k, k = 1,2, \dots, |\Omega|\}$，其中，$\phi_k$是$\Omega$中第$k$個特徵變數。計算每個特徵出現在所有 SP 的次數，標記為$N_{OL}(\phi_k), k = 1,2, \dots, |\Omega|$。

Step 3: 透過$N_{OL}(\phi_k)$即可計算覆蓋率$\omega(\phi_k)$，公式如下:

$$\omega(\phi_k) = \frac{N_{OL}(\phi_k)}{|TS|}, k = 1,2, \dots, |\Omega| \quad (14)$$

計算$\{\omega(\phi_k), \mathrm{k} = 1,2, \dots, |\Omega|\}$的平均標記為$\overline{\omega}$。

Step 4: 累加每個 SP 裡，特徵的 selection gain:

$$g_{sum}(\phi_k) = \sum_{j=1}^{|TS|} g(\phi_k \to t_j), k = 1,2, \dots, |\Omega| \quad (15)$$

計算$\{g_{sum}(\phi_k), \mathrm{k} = 1,2, \dots, |\Omega|\}$的平均標記為$\overline{g}_{sum}$。

Step 5: 根據累加後的資訊增益量g_{sum}和覆蓋率ω，計算出特徵的有效貢獻量ρ:

$$\rho(\phi_k) = \omega(\phi_k) * g_{sum}(\phi_k), k = 1,2, \dots, |\Omega| \quad (16)$$

Step 6: 測試Ω中所有的特徵變數，若$\rho(\phi_k) > \overline{\omega} \cdot \overline{g}_{sum}$，則將$n_{tmp}$累加。

Step 7: 設定上下界，標記為n_U和n_L，透過上下界找出n_{FP}，，n_{FP}表示最後選取的特徵數目。本研究中所有實驗n_U皆設定為 4，n_L皆設定為 2。若n_{tmp}介於上下界間，則將n_{FP}設定成n_{tmp};若n_{tmp}小於下界，則將n_{FP}設定成n_L;若n_{tmp}大於上界則將n_{FP}設定成n_U。

Step8 :將$\{\rho(\phi_k), \mathrm{k} = 1,2, \dots, |\Omega|\}$排序，並選取前$\mathrm{n}_{FP}$個特徵變數加入最後的特徵池(FP)中，當作多目標的特徵挑選結果。

2.2 結構化學習

結構化學習是為了將訓練資料可以更有邏輯的應用到模型建造中。經過多目標特徵選取後可以得到一些被挑選過的特徵，我們把這些特徵當作模型的訓練資料，標記為$\vec{h} = [h_1 \quad h_2 \quad \dots \quad h_M]^T$，$M$為輸入維度的數量。在本研究中，這些不同輸入維度的訓練資料，用 MATLAB 裡所提供的 subclust()函式做分群，每一個輸入維度係數皆用 0.3。並將分群後的群中心配合每個維度的標準差形成高斯型態的模糊集，高斯函數的公式如下:

$$gaussmf(h; c, \sigma) = \exp(\frac{-(h-c)^2}{2\sigma^2}) \quad (17)$$

其中，h為輸入變數，c和 σ為群中心和標準差的參數。基於各個輸入維度的模糊集，我們可以得到K個前鑑部，如下:

Premise k :

IF $x_1 = A_1^{(k)}(h_1)\ and\ x_2 = A_2^{(k)}(h_2) \dots and\ x_M = \mathrm{A}_\mathrm{M}^{(k)}(h_M)$ (18)

其中，x_j為第j個輸入的語意變數;$A_j^{(k)}(\cdot)$為第k個前鑑部中第j個輸入語意變數的模糊集; h_j為第j個輸入變數，$j = 1,2, \dots, M$。

為了模型的運算效率，我們透過啟動強度來篩選前鑑部，降低前鑑部的數量。步驟如下:

Step 1: 從各個輸入維度與模糊集可以得到每個前艦部的啟動強度，以第k個前鑑部為例，啟動強度如下:

$$\beta_i^{(\mathrm{k})} = \prod_{j=1}^{M} A_j^{(k)}(h_j(i)) \quad (19)$$

其中，$h_j(i)$為第j個輸入維度的第i筆資料; $A_j^{(k)}(\cdot)$為第k個前鑑部中第j個輸入維度的模糊集。

Step 2: 將每個前鑑部的啟動強度累加標記為$\beta_{\mathrm{sum}}^{(\mathrm{k})}$，公式如下:

$$\beta_{sum}^{(\mathrm{k})} = \sum_{i=1}^{n} \beta_i^{(\mathrm{k})} \quad (20)$$

其中，n為資料總筆數。計算$\{\beta_{sum}^{(\mathrm{k})}, \mathrm{k} = 1,2, \dots, \mathrm{K}\}$ 平均值標記為$\bar{\beta}_{sum}$，標準差標記為β_{sum}^{std}。

Step 3: 查看每個前鑑部，若$\beta_i^{(\mathrm{k})} > \bar{\beta}_{sum} \cdot \beta_{sum}^{std}$，則將$b_{tmp}$累加。設定上下界，標記為$b_U$和$b_L$，透過上下界找出$b_{FP}$，$b_{FP}$表示最後選取的前鑑部數目。本研究中所有實驗$\mathrm{b}_U$皆設定為 15，$b_L$皆設定為 4。若$b_{tmp}$介於上下界間，則將$b_{FP}$設定成$b_{tmp}$;若$b_{tmp}$小於下界，則將$b_{FP}$設定成$b_L$;若$b_{tmp}$大於上界則將$b_{FP}$設定成$b_U$。

Step 4: 將$\{\beta^{(\mathrm{k})}, k = 1,2, \dots, K\}$排序，並保留前$b_{FP}$個前鑑部，當作之後模型運作的前鑑部層神經元。

假設目前訓練資料集合，標記為$TD = \left\{(\vec{x}, \vec{t})^{(i)}, i = 1,2, \dots, n\right\}$，$n$為資料總筆數，本模型使用 IF-THEN 規則，後鑑部個數 Q 與前鑑部個數相同，而每個後鑑部結構為 T-S function，T-S function 公式如下:

$$y^{(q)} = a_0^{(q)} + \sum_{i=1}^{M} a_i^{(q)} h_i \quad (21)$$

其中，$\{a_0^{(\mathrm{q})}, a_i^{(\mathrm{q})}, i = 1,2, \dots, M\}$是第$q$個後鑑部的參數，$h_i$是第$i$個維度的輸入。

2.3 模型輸入與輸出

本實驗的模型為一個六層的神經網路。訓練資料集合標記為$TD = \{(\vec{x}_i, \vec{t}_i), i = 1,2, \dots, n\}$，$n$為資料總筆數，$\vec{x}_i$是$M \times 1$的輸入向量，$M$為輸入維度數量;$\vec{t}_i$為$\mathrm{N} \times 1$的目標向量，N為複數型態目標的數量。透過模型可以得到輸出$\{\vec{y}_i, i = 1,2, \dots, n\}$。

Layer 1: 這層稱之為輸入層，是將原始資料做差分後，並透過多目標特徵選取，將最後挑出的特徵當作訓練資料，我們將時間序列第t個點的輸入向量標記為:

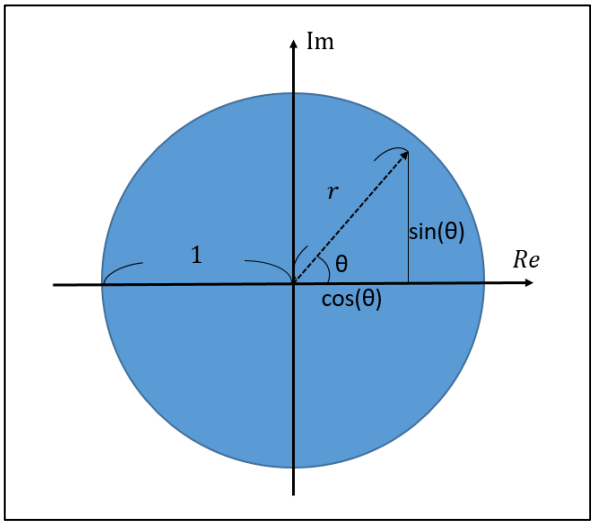
$$\vec{h}(\mathrm{t}) = [h_1(\mathrm{t})\ h_2(\mathrm{t}) \dots h_\mathrm{M}(\mathrm{t})]^\mathrm{T} \quad (22)$$

Layer 2: 這層為複數模糊集合層，透過前面結構學習的分群，可在不同維度上建構

數個模糊集，而每個不同維度的輸入都可經由模糊集得到歸屬程度。傳統高斯模糊集只能得到一組實數型的歸屬程度，而透過複數模糊集合可得到多組複數型態的歸屬程度，且會落在半徑為 1 的複數單位圓盤圖(1)內。不同的歸屬程度可以給不同的模型輸出做應用，以達到多目標預測的效果，透過複數模糊集的概念，從中可得到歸屬程度$\mu_s(h)$，公式如下:

$$\begin{aligned}\mu_s(h) &= r(h)\exp\big(j\omega_s(h)\big)\\ &= Re(\mu_s(h)) + j\,\mathrm{Im}\big(\mu_s(h)\big)\\ &= r(h)\cos\big(\omega_s(h)\big) + jr(h)\sin(\omega_s(h))\end{aligned} \quad (23)$$

其中，$r(h)$為高斯函數(17); $j=\sqrt{-1}$; $\omega_s(h)$為$\frac{dr}{\mathrm{dx}}$。



圖一、複數單位圓盤 (圖片來源：本研究)

Layer 3: 此層稱之為前鑑部層，經過前面的結構學習，我們可以篩選出K個對模型較有用的前鑑部，標記為:

Premise k :

IF $x_1 = A_1^{(k)}(h_1)\ and\ x_2 = A_2^{(k)}(h_2)\ \ldots and\ x_M = \mathrm{A}_\mathrm{M}^{(k)}(h_M)$ (24)

其中，x_j為第j個輸入的語意變數;$A_j^{(k)}(\cdot)$為第k個前鑑部中第j個輸入語意變數的模糊集; h_j為第j個輸入變數，$j=1,2,\ldots,M$。將每個輸入維度的歸屬程度相乘後，可得到每個前鑑部的啟動強度:

$$\beta^{(\mathrm{k})} = \prod_{j=1}^{M} A_j^{(k)}(h_j) \quad (25)$$

Layer 4: 此層稱之為正規化層，我們針對K個前鑑部的輸出作正規化，公式如下:

$$\gamma^{(k)} = \frac{\beta^{(k)}}{\sum_{k=1}^{K}\beta^{(k)}} \quad (26)$$

Layer 5:此層稱之為後鑑部層，經過此層的運算可以得到Q個模型輸出，Q與前鑑部個數K相同，輸出公式如下:

$$\hat{y}^{(q)} = \gamma^{(\mathrm{q})}(a_0^{(\mathrm{q})} + \sum_{i=1}^{M} a_i^{(q)} h_i) \quad (27)$$

其中，$\{a_0^{(\mathrm{q})}, a_i^{(\mathrm{q})}, i=1,2,\ldots,M\}$是第$q$個後鑑部的參數。

Layer 6:此層稱之為輸出層，將上一層得到的Q個模型輸出結合，即為我們的模型輸出:

$$\bar{y} = \sum_{q=1}^{Q}\hat{y}^{(q)} \quad (28)$$

2.4 參數學習

根據 divide-and-conquer 的概念，我們將使用不同的機器學習演算法，對各層的參數作優化，以便更容易找到最佳解。本實驗中我們使用了粒子群演算法以及人工蜂群演算法個別當作前鑑部的參數學習演算法，後鑑部參數則使用了 RLSE 做學習，演算法內容如下:

2.4.1 粒子群演算法 (PSO)

對於前鑑部的參數優化，我們使用知名的 PSO 演算法[18]學習，其原理類似鳥群在尋找食物，每回合透過自身的最佳位置和全群最佳位置調節速度，特性為收斂快速，演算法公式如下:

$$\vec{V}_i(t+1) = \omega\vec{V}_i(t) + c_1\xi_1(\overrightarrow{Pbest_i}(t) - \vec{P}_i(t)) + c_2\xi_2(\overrightarrow{Gbest}(t) - \vec{P}_i(t)) \quad (29)$$

$$\vec{P}_i(t+1) = \vec{P}_i(t) + \vec{V}_i(t+1) \quad (30)$$

其中，$\vec{P}_i(t)$為第t回合時第i個粒子的位置，$\vec{V}_i(t)$為第t回合時第i個粒子的速度，$\overrightarrow{Pbest_i}(t)$為第$t$回合時第$i$個粒子的最好位置，$\overrightarrow{Gbest}(t)$為第$t$回合時全部粒子中最好的位置，ω、$C_1$、$C_2$為 PSO 的參數，$\xi_1$、$\xi_2$為介於 0 到 1 的隨機數。在本實驗中，粒子的位置代表前鑑部的參數，其中包含了每個維度的分群中心、標準差以及λ_1、λ_2。

2.4.2 人工蜂群演算法 (ABCO)

此演算法總共有三種蜜蜂，包括工蜂、觀察蜂、偵查蜂，但與真實的蜜蜂找尋食物無直接相關，步驟如下:

Step 1: 隨機形成工蜂位置，並更新工蜂位置，公式如下:

$$v_i^j = x_i^j + rand[-1,1](x_i^j - x_k^j) \quad (31)$$

其中，v_i^j為第i只工蜂在移動後的第j個維度; x_i^j為第i只工蜂移動前的第j個維度; x_k^j為其他隨機蜜蜂的第j個維度的值。

Step 2: 使用輪盤法選一個位置，輪盤機率公式如下:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (32)$$

其中，p_i為機率; fit_i為適應值，本實驗將成本的倒數視為適應值; SN為蜜蜂的總數目

Step 3: 所有的觀察蜂在剛剛被選中的位置附近搜尋，公式如下:

$$v_i^j = x_i^j + rand[-1,1](x_i^j - x_{selected}^j) \quad (33)$$

其中，v_i^j為第i只工蜂在移動後的第j個維度; x_i^j為第i只工蜂移動前的第j個維度; $x_{selected}^j$為被選中蜜蜂的第j個維度的值。

Step 4: 若該工蜂已經達到限制回合，都未更新則派出偵查蜂取代，偵查蜂位置產生公式如下:

$$x_i^j = x_{min}^j + rand[0,1](x_{max}^j - x_{min}^j) \quad (34)$$

其中，x_i^j為第i只工蜂的第j個維度; x_{max}^j為所有工蜂第j個維度的最大值; x_{min}^j為所有工蜂第j個維度的最小值。

Step 5: 重複 step 2~ step 4，直到反覆運算結束。

2.4.3 Recursive Least Square Estimator (RLSE)

在本實驗中 RLSE[19]是用來更新後鑑部參數，一般來說 LSE 問題可以被指定成一個線性的模型，如下:

$$y = f_1(u)\theta_1 + f_2(u)\theta_2 + \cdots + f_m(u)\theta_m + \varepsilon \quad (35)$$

其中，y 是目標;u 是模型的輸出;$\{f_i(.), i = 1,2,\dots,m\}$是 u 已知的方程式;$\{\theta_i, i = 1,2,\dots,m\}$是我們估計的未知參數，$\varepsilon$則是整個模型的誤差 LSE 的問題也可以被寫成矩陣的方式表達，如下:

$$Y = \mathbf{C}\theta + \varepsilon \quad (36a)$$

其中:

$$\mathbf{C} = \begin{bmatrix} f_1(u_1) & f_2(u_1) & \cdots & f_m(u_N) \\ f_1(u_2) & f_2(u_2) & \cdots & f_m(u_N) \\ \vdots & \vdots & \cdots & \vdots \\ f_1(u_N) & f_2(u_N) & \cdots & f_m(u_N) \end{bmatrix} \quad (36b)$$

$$\theta = [\theta_1, \theta_2, \dots, \theta_m]^T \quad (36c)$$

$$Y = [y_1, y_2, \dots, y_N]^T \quad (36d)$$

$$\varepsilon = [\epsilon_1 \quad \epsilon_2 \quad \cdots \quad \epsilon_N]^T \quad (36e)$$

$\mathbf{C}$是輸入的矩陣，θ是我們估計的未知參數矩陣，Y是目標矩陣，ε是誤差的向量。要優化θ，可透過 RLSE 的等式[19]運算:

$$P_{k+1} = P_k - \frac{P_k b_{k+1}(b_{k+1})^T P_k}{1+(b_{k+1})^T P_k b_{k+1}} \quad (37a)$$

$$\theta_{k+1} = \theta_k + P_{k+1} b_{k+1}(y_{k+1} - (b_{k+1})^T \theta_k) \quad (37b)$$

其中$k = 1,2,\dots,(n-1)$，$[(b_k)^T, y_k]$是[C, Y]的第k行，再開始 RLSE 演算法時，我們會設定θ_0為 0，P_0則設定為$\alpha\mathbf{I}$，$\mathbf{I}$為單位矩陣。我們 cost function 使用 RMSE，定義如下:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\vec{e}_i)(\vec{e}_i)'}{n}} \quad (38)$$

其中，$(\cdot)'$為複數的共軛運算; $\vec{e}_i = \vec{t}_i - \vec{y}_i$，其中$\vec{t}_i$為目標向量，$\vec{y}_i$為模型輸出向量。整個機器學習的的運作流程如下:

Step 1: 準備訓練資料。

Step 2: 用前鑑部演算法的粒子位置計算啟動強度。

Step 3: 用 RLSE 更新後鑑部參數，RLSE 算式中的b_{k+1}和θ_k向量如下:

$$b_{k+1} = [\boldsymbol{\phi}^{(1)}\ \boldsymbol{\phi}^{(2)} \dots \boldsymbol{\phi}^{(Q)}] \quad (39a)$$

$$\theta_k = \left[\ {}^{1}\tau(k)\ {}^{2}\tau(k) \dots {}^{Q}\tau(k)\ \right]^T \quad (39b)$$

$$^{i}\tau(k) = \left[\ {}^{i}a_0(k)\ {}^{i}a_1(k) \dots {}^{i}a_M(k)\ \right]^T \quad (39c)$$

其中$k = 1,2, \dots, (\text{n} - 1), \text{i} = 1,2, \dots, \text{K}$。

Step 4: 更新完所有參數後，計算出模型的輸出。

Step 5: 計算 RMSE。

Step 6: 對所有粒子重複 Step2~Step5，直到回合結束。

2.5 模擬投資

為了評估模型是否賺錢，使用 RMSE 是不夠的，因為從 RMSE 中無法看出是否有賺錢，只能從中瞭解模型的配市率，而配市率高並不代表利潤高，因此本實驗將預測出來的值配合策略，進一步決定要買進或是賣出，進而算出模型所能賺的利潤，買進與賣出公式如下:

買進: $forecast_{close}(t) > open(t)$ (40a)

賣出: $forecast_{close}(t) < open(t)$ (40b)

其中，$forecast_{close}(t)$為模型的輸出，意即預測t日的收盤價格;$open(t)$為t日實際的開盤價格。若預測的收盤價高於實際開盤價，代表必須買進，等今日結束後才會賺錢; 若預測的收盤價低於實際開盤價，代表模型預測今天會跌，所以要儘快賣出。

計算利潤的方式，則透過實際的收盤價與開盤價去做運算，公式如下:

$$\text{Profits} = \sum_{t_b=1}^{p}\big(close(t_b) - open(t_b)\big) + \sum_{t_s=1}^{q}\big(open(t_s) - close(t_s)\big) \quad (41)$$

其中，Profits為利潤，p為策略為買的總天數; q為策略為賣的總天數; $close(t)$代表第t天的收盤價。

透過上述公式，我們可以獲得整個模型的所帶來的利潤值，並大致模擬出此模型運用到真實世界的效果。本研究將會在每個實驗中秀出利潤值與其他參數。

3. 預期成果

預期透過三個實驗發現，本論文提出的模型CFNS有多目標預測的能力。而透過多目標特徵挑選，可以根據不同的資料去萃取原始資料中有用的資料，並且控制進入模型的資料大小。結構化學習的部分，可以將輸入的資料自動地根據資料做調整，在面臨不同的資料可以自己生成不同的結構。從實驗中預期一次對兩個目標做時間序列的預測，各個目標的效果不亞於其他論文所提出的方法，甚至更好。證明不同資料都能在此模型中，被有效的預測。也代表此研究中的兩種演算法配合RLSE的混合方法，有著一定的水準。最後，在模擬投資方面，我們希望本實驗可以透過模型和投資策略的結合，比其他模型賺取更多的利潤。

4. 預期貢獻

證明演算法的結合是可行的，並從實驗中分析出演算法的優劣勢，探討造成差異原因，而本篇論文所使用的IF-THEN規則類神經網路，前鑑部 (IF-part)和後鑑部 (THEN-part)各數相同，未來可使用不同個數，如同神經元之間的連接，增加模型的彈性，使其

可以增添更多的隱藏層。在本實驗中，全新設計的計算利潤公式，可以為未來虛擬投資的比較定下基礎，除了模型誤差的指標外，可透過利潤公式，更加貼近實際操作情況。

5. 預期貢獻

[1]. A. J. Patton, “A review of copula models for economic time series,” Journal of Multivariate Analysis, vol. 110, pp. 4-18, 2012.

[2]. J. Zbontar and Y. LeCun, “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches,” Journal of Machine Learning Research, vol. 17, pp. 1-32, April 2016.

[3]. Z. Feng, L. Jin, D. Tao and S. Huang “DLANet A Manifold Learning based Discriminative Feature Learning Network for Scene Classification,” Neurocomputing, vol. 157, pp. 11-21, 2015.

[4]. D. Perez et al., “The 2014 General Video Game Playing Competition,” IEEE Transactions on computational intelligence and AI in games, in press, 2014.

[5]. C. Li and J.-W. Hu, “A new ARIMA-based neuro-fuzzy approach and swarm intelligence for time series forecasting,” Engineering Applications of Artificial Intelligence, vol. 25, pp. 295-308, 2012.

[6]. C. Li and T.-W. Chiang, ”Complex neurofuzzy ARIMA forecasting—a new approach using complex fuzzy sets,” IEEE Transtraction on fuzzy systems. vol. 21.no. 3, pp.567-584, June 2013.

[7]. L. J. Herrera, H. Pomares, I. Rojas, A. Guillen, J. Gonzalez, M. Awad and A. Herrera, ”Multigrid-based fuzzy systems for time series prediction:CATS competition,” Neurocomputing, vol.70, pp. 2410-2425,2007.

[8]. I. Sugiarto and S. Natarajan, “Parameter estimation using least square method for MIMO Takagi-Sugeno neuro-fuzzy in time series forecasting,” J. Tek. Elektro, vol. 7(2), pp. 82-87, 2007.

[9]. M. Z.-Kermani and M. Teshnehlab, ”Using adaptive neuro-fuzzy inference system for hydrological time series prediction.” Appl. Soft Comput, vol. 8,pp. 928-936, 2008.

[10]. H. J. Rong, N. Sundararajan,G. B. Huang and P. Saratchandran, “Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction,” Fuzzy Set Syst, vol.157, pp. 1260-1275, 2006.

[11]. Zadeh, “Fuzzy sets”, Inf. Control , vol. 8, pp. 338-353, 1965.

[12]. D. Ramot, R. Milo, M. Friedman, and A, Kandel, “Complex fuzzy sets,” IEEE Trans. Fuzzy syst., vol . 10, no.2, pp. 171-186, Apr. 2002

[13]. C. Li, T.-W. Chiang. J.-W. Hu, and T. Wu, “Complex neuro-fuzzy intelligent approach to function approximation,” in Proc. 2010 3rd Int.Workshop Adv. Comput. Intell., pp. 151-156, 2010.

[14]. C. Li and T.-W. Chiang, “Complex fuzzy computing to time series prediction—A multi-swarm PSO learning approach,” Lect. Notes Artif. Intell., vol. 6592, pp. 242-251, 2011.

[15]. E. C. Shannon, ”A mathematical theory of communication,” Bell System Technical Journal, vol. 27, pp. 379-423, 1948.

[16]. C. Li , ”Multi-target feature selection,” unpublished, 2017.

[17]. T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applocations to modeling and control,” IEEE Trans. Systems, Man, and Cybernetics, vol. SMC-15, no. 1, pp. 116-132, 1985.

[18]. J. Kennedy and R. Eberhart, “Particle swarm optimization,” Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.

[19]. J. S. R. Jang, C. T. Sun and E. Mizutani, “Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence prentice hall,” Upper Saddle River, 1997.

[20]. C. Li and T.-W. Chiang, “Complex neuro-fuzzy self-learning approach to function approximation,” Lect. Notes Artif. Intell., vol. 5991, pp. 289-299, 2010.

[21]. H. Greenspan, B. van Ginneken and R. M. Summers, “Guest Editorial Deep Learning in Medical Imaging Overview and Future Promise of an Exciting New Technique,” IEEE Transactions on medical imaging, vol. 35, no. 5, May 2016.

[22]. D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization artificial bee colony algorithm,” J Glob. Optim., vol. 39, pp. 459-471, 2007.