國立中山大學 電機工程學系

碩士論文

自建構式一般化第二型模糊架構於神經模糊系統

A Self-Constructing General Type-2 Scheme for Neuro-Fuzzy System Modeling

研究生：鄭文豪 撰

指導教授：李錫智 教授

中華民國 九十八 年 六 月 三十 日

A Self-Constructing General Type-2 Scheme for Neuro-Fuzzy System Modeling

Directed by: Professor Shie-Jue Lee

Graduate Student: Wen-Hau Jeng

Department of Electrical Engineering

National Sun Yat-Sen University

Kaohsiung, Taiwan 80424, R.O.C.

致謝

時間過的真的很快，一下子，研究所兩年就到了尾聲。想到這兩年來的點點滴滴，雖然有時也會覺得辛苦，但大多是感到幸運，有這個機會再次回到校園來學習、充電。首先，要感謝的我指導教授 李錫智老師的指導。老師這兩年來不只是在學業上，也在人格教育上給了我很多的指導，我想這些想法與理念甚至是挫折中所得到的教訓，都可以讓我受用一生。

再來，我要感謝帶我的博班學長葉吉原學長。如果沒有你的耐心指導，我這篇論文應該還在天上，還沒掉下來。所以，吉原學長謝謝你，有了你提供的在學術上、論文撰寫上的支援，我才可以順利的做好我的事情。再來，我要感謝我的三位好同學李連旺、蔡憲奇、歐陽正。跟你們成為同學是一件幸運的事情，跟你們一起去修課更是讓我獲益良多。這說長不長說短不短的兩年，因為有了你們而增色了不少。再來也想要跟歐陽振森老師說聲謝謝，因為有您軟硬體上的幫助，使我可以站在巨人的肩膀上，論文一起步就已經完成了許多。其他還有許許多多需要感謝的好學長、好學姊、好學弟、好學妹…，這裡雖然沒有提到你們的名字，但與你們在中山生活的點滴已經成為我珍貴的回憶，謝謝你們給我這麼美好的兩年。

最後，我想要感謝我的父母，沒有你們的支持與鼓勵，我想我沒辦法一個人完成碩士的學業。爸、媽謝謝你們，這麼多年來，你們辛苦了。

鄭文豪 謹於

國立中山大學

中華民國 九十八 年 八 月 二十 號

目錄

摘要

本論文提供了一個自建構式第二型類神經模糊網路(type-2 fuzzy neural network)的系統模型。要建構一個第二型類神經模糊網路主要包含三個問題：降階(type reduction)、架構鑑別(structure identification)、參數鑑別(parameter identification)。在降階的問題上，使用α平面(α-planes)的概念將第二型模糊集合解構成區間第二型模糊集合(interval type-2 fuzzy set)，然後再對區間第二型模糊集合使用KM(Karnik-Mendel)演算法來降階。經過運算，每個α平面會得到一組上下限，將這些平面的上下限計算加權平均即為純量的輸出。這種方式總體計算量非常的大，所以我們提供了一個方法來降低計算量。在降階的基礎上，可以建立出第二型類神經模糊推論系統。對於架構鑑別，使用相似度可增加模糊分群法將資料分群；然後，一群就可以用其特徵化成一條模糊規則(fuzzy rule)。實驗模擬的結果顯示，本論文提供的方法可以在不影響準確度的情況下減少降階所需要的時間。

Abstract

We propose a self-constructing general type-2 fuzzy neural network for system modeling. The problems of constructing a general type-2 fuzzy neural network include type reduction, structure identification, and parameter identification. Regarding the type reduction, an α-planes strategy is used to decompose a type-2 fuzzy set into several interval type-2 fuzzy sets, and then apply the Karnik-Mendel algorithm to do type reduction to interval type-2 fuzzy sets. After getting both the lower and upper bound of the output for each α-plane, a crisp output value is obtained by the weighted average method. Since the amount of time required by this method is more demanding, an efficient strategy is proposed to solve this problem. Based on type reduction, a type-2 fuzzy neural network for fuzzy inference can be built. Regarding the structure identification, an incremental similarity-based fuzzy clustering method is used to partition the dataset into several clusters and a local regression model is obtained for each cluster, and then a type-2 fuzzy rule is extracted from each cluster. A hybrid learning algorithm which combines particle swarm optimization and recursive least squares estimator is adopted in the parameter identification to refine the antecedent and consequent parameters, respectively, of fuzzy rules. Simulation results show that our proposed method runs faster in type reduction without deterioration of the forecasting performance and the resulting networks obtained are robust against outliers.

.

圖形與表格目錄

第一章

導論

1.1 研究動機與文獻回顧

有別於統計、機率等硬計算，機器學習主要是模仿和學習人類的行為的一種軟計算。現今機器學習涵蓋的範圍十分廣泛，同時成功地應用在許多日常生活方面、學術研究及工程領域上。例如日本已經成功的將模糊系統運用在冷氣機、洗衣機等日常生活家電上；而機器學習中的許多分群分類演算法也在資料探勘(data mining)等領域中有許多重要的應用。

一般來說機器學習的主要目的在於找到適用的規則，使得這些被找到的規則可以詮釋有限取樣的實驗資料。在過去已經發展出許多受歡迎且具有公信力的學習機器，例如類神經網路(Neural Network)[4]、模糊邏輯(Fuzzy Logic)[34][14]和支撐向量機(Support Vector Machine)[12]等。這些學習機器的目標函數與網路架構並不相同。在此我們對於模糊邏輯特別有興趣，因為它的架構通常具有很清楚的物理意義。在本論文中我們將討論第二型模糊推論(type-2 fuzzy inference)的各項細節和推論方法。

一般提到模糊邏輯，所討論的都是第一型模糊(type-1 fuzzy)。有別於傳統的二元邏輯(binary logic)，第一型模糊所使用的模糊集合(fuzzy set)，是使用一個介於[0, 1]之間的程度值來替代單純的(0, 1)二元邏輯。但是，第一型模糊有時候並沒有辦法表示出某些不確定的情況[18][23][24][26][27]。對於此類問題可以以第二型模糊集合為基礎的第二型模糊邏輯來解決；因為第二型模糊集合擁有更加有彈性的歸屬函數值(membership value)—第二型模糊集合的歸屬函數值也是一個模糊集合[20][27]。由於第二型模糊集合的特色，不但使第二型模糊推論在設計上更有彈性，系統的錯誤容忍度也上升了。現今已經有許多第二型模糊邏輯的應用與研究，像是自動控制(automatic control)[5][19][31]、函數近似(function

approximation)[1][35]、資料分類(data classification)[22][36]、醫療應用(medical applications)[20][27]，但是絕大多數的第二型模糊使用的是區間第二型模糊(interval type-2 fuzzy set)。區間第二型模糊是第二型模糊的一種特例。之所以使用區間第二型模糊來替代第二型模糊的理由有以下兩點(1)比起區間第二型模糊，第二型模糊推論過於複雜，複雜到找不到一個適合的方式來做模糊推論(2)第二型模糊在降階(type reduction)計算時，需要極大量的時間。

Liu[20]針對第二型模糊提出了降階的方法。Liu 主要的想法是利用 α 平面(α-plane)將第二型模糊集合解構成區間第二型模糊集合，然後再使用KM(Karnik-Mendel)演算法[11][25][28]來對區間第二型模糊做降階。但是對於第二個問題(第二型模糊在降階計算時，需要極大量的時間)，我們依然覺得結果並不令人滿意；主要原因是因為如果使用大量的α平面來解構第二型模糊集合，則需要的計算量依然非常的大。Coupland and John[14]提供了一個極有效率的降階方法，不過該篇論文的模糊集合的歸屬函數(membership function)必須對稱，在使用上有所限制。Lucas 等人[21]提供了一個叫垂直切面(vertical slice)的解模糊方法，這個方法使用了完全不同的概念來做解模糊化。本論文基於 Liu 的方法提出一個改善的方法來減少降階運算所需要的時間。我們的想法是：α平面之間的降階結果應該存在某些關聯性，如果關聯性存在，則利用上一個α平面降階 KM 演算法的結果，當成這次 α 平面降階 KM 演算法的初始值。如此一來，就可以大幅窄化 KM 演算法的解空間，進而加速降階的速度。本論文會證明 α 平面之間的關連性的確存在。而利用這樣的關連性對第二型模糊做降階所做的改善將加速降階運算的速度，並且不會犧牲原先系統的準確度。

此外，架構鑑別與參數鑑別也有許多方法被提出。Wang[32]等人提供了一個適用於區間第二型模糊規則(interval type-2 fuzzy rule)的類神經網路架構；他們使用梯度法(gradient descent)來做參數鑑別。Hagras[6]修正了一些 Wang[32]中的式子。Lee[17]提供了一個適用於區間第二型模糊規則的遞迴式的類神經網路(recurrent neural network)架構，其歸屬函數使用的是非對稱性(asymmetric)的高斯

(Gaussian)函數；他們也同樣使用梯度法來做參數鑑別。他們宣稱非對稱性的高斯歸屬函數的效果比對稱性的高斯歸屬函數的效果好。Juang 和 Tsao 提供了一個適用於區間第二型模糊規則的自發展式(self-evolving)的類神經網路。他們使用了對稱性的高斯函數當歸屬函數；並且使用結合了梯度法和卡爾曼濾波法(reduced-order kalman filter)的混合式學習法。Castro 等人[2]提供了三種區間第二型模糊規則的網路架構；他們的歸屬函數有對稱性的也有非對稱性的。模擬實驗的結果顯示了越複雜的模型會有越好的準確度。

1.2 方法概述

在本論文中，我們透過了輸入—輸出的樣本，使用了類神經模糊的技術來建構第二型模糊規則。我們對訓練樣本使用了自建構式模糊分群法(self-constructing fuzzy clustering)[16]來分群；其原理是基於輸入樣本的相似度以及輸出樣本的相似度來作為分群的依據。自建構式模糊分群法的結果，每一群均一一對應形成一個第二型模糊規則。簡單的說，我們使用了自建構式模糊分群法來做架構鑑別(structure identification)。我們用了第二型模糊集合當模糊法則的前鑑部(antecedent)；選了 TSK(Takagi-Sugeno-Kang)模型當模糊法則的後鑑部(consequence)。模糊規則架構決定了之後，接著我們要決定參數鑑別(parameter identification)的方法。我們選用了混合式學習法(hybrid learning)來做參數鑑別；用粒子群最佳化(PSO, particle swarm optimization)[13]來最佳化前鑑部的參數，使用最小平方估算法(LSE, least squares estimator)[30]來最佳化後鑑部的參數。在第二型模糊推論的部份，假定一組輸入值進入第二型模糊推論系統，則每個輸入維度都會得到一個第一型模糊集合。然後將各自模糊規則的每個第一型模糊集合做 T-norm；T-norm 的作法細節參考 Fuzzy Sets and Fuzzy Logic[14]這本書的第 4.4、4.5 節。本論文採用 4.4 節，也就是乘法的作法。每個模糊規則的前鑑部做完 T-norm 的結果就是該模糊規則的啟動強度(firing strength)；其型態也是一個第一

型模糊集合。接下來，我們將每個模糊規則的啟動強度(它是一個一型模糊集合)做 α 截集(α-cut)，每個 α 截集會得到一個區間。將同一個 α 截集的所有模糊規則的啟動強度得到的區間和後鑑部—TSK 的結果，使用 KM 演算法做降階。降階的結果會得到一個區間，再把各 α 截集所得到的區間組合起來就可以得到一個第一型模糊集合。這個第一型模糊集合其實就是第二型模糊推論的輸出。如果要輸出一個純量，則對這個第一型模糊集合使用質心法來計算，就可以得到一個純量當做輸出。本論文在不同的 α 截集間使用 KM 演算法的部份做加速。

1.2 論文架構

本論文有七個章節。第一章為整篇論文作簡介。第二章介紹第二型模糊邏輯系統；第三章介紹我們如何為降階演算法做加速。第二第三章是本論文的核心。第四章我們提供了第二型模糊類神經網路的學習演算法。第五章我們針對整個系統給了一個例子。第六章討論實驗的結果。第七章為結論與未來方向。

第二章

第二型模糊邏輯

在本章節中，我們將一一介紹第二型模糊邏輯的基本元件。

2.1 第二型模糊集合

對於一般傳統的第一型模糊集合而言，其歸屬函數值是一個介於 0 到 1 的實數。但是有時我們需要更具有彈性的歸屬函數與歸屬函數值，這時候第一型模糊集合就無法滿足要求；我們這時候會使用 Zadeh[34]在 1975 年所提出來的第二型模糊集合來設計出較具彈性的歸屬函數。第二型模糊集合可以定義成：

$$\mu_{\tilde{A}} : X \to F([0,1]) \tag{2.1}$$

其中 $F([0,1])$ 是一個第一型模糊歸屬函數，其值域介於[0, 1]之間。圖 2.1 為第二型模糊歸屬函數的 $x=a$ 和 $x=b$ 的歸屬函數值的示意圖，圖 2.1 不同的輸入值 $x=a$ 和 $x=b$ 所得到的第二階歸屬函數的變異數是不相同的。如果 $F([0,1])$ 不是一個模糊集合而只是一個區間，則這個特別的第二型模糊集合我們稱之為區間第二型模糊集合；另外，如果 $F([0,1])$ 不是一個模糊集合而只是一個純量，則這個特別的第二型模糊集合就是一般的第一型模糊集合。

想要了解並運用第二型模糊集合，有兩個非常重要的概念來幫助我們，那就是 α 截集[14]和 α 平面[20]。透過 α 截集，可以把第二型模糊集合間的 T-nrom 給實做出來[14]。給定一個輸入值 $x=a$ 至一個第二型模糊集合，我們之前說過會得到一個第二型模糊集合的第二階歸屬函數，這個第二階歸屬函數是一個第一型模糊集合。將第二型模糊集合的第二階歸屬函數透過 α 截集切分成數個區間，再與另外一個第二型模糊集合的第二階歸屬函數 α 截集所形成的對應區間做 T-norm 運算。區間和區間 T-norm 的結果也是一個區間。這些做完 T-norm 所形成的區間依照 α 截集的順序再組合成為一個第一型模糊集合，這裡所得到的第一型模糊集合就是第二型模糊集合間在輸入值 $x=a$ 時的 T-nrom 結果。實做出第二型模糊集合間的 T-nrom，則模糊推論的各個模糊規則啟動強度的計算就沒問題了。利用 α 截集的觀念，也可以解決第二型模糊推論的另外一個難題，降階問題。如圖

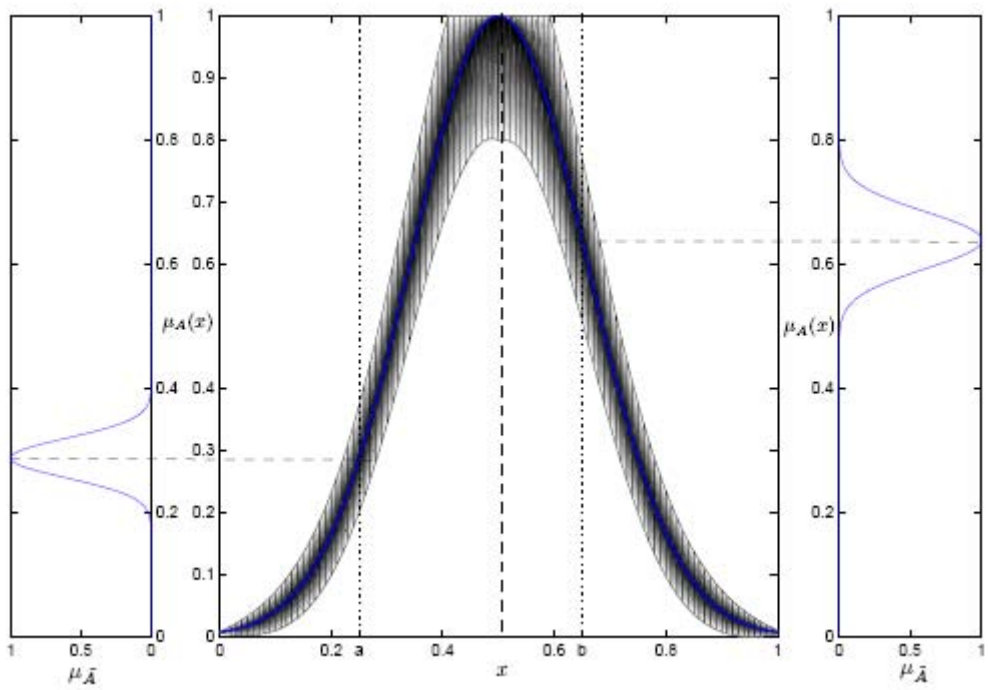

圖 2.1：第二型模糊集合示意圖

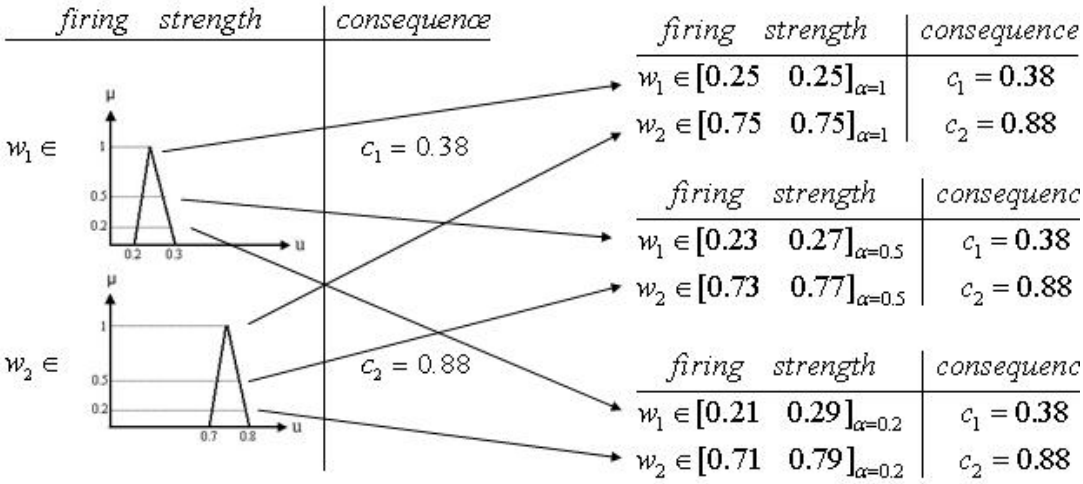

圖 2.2：啟動強度透過 α 截集分割圖

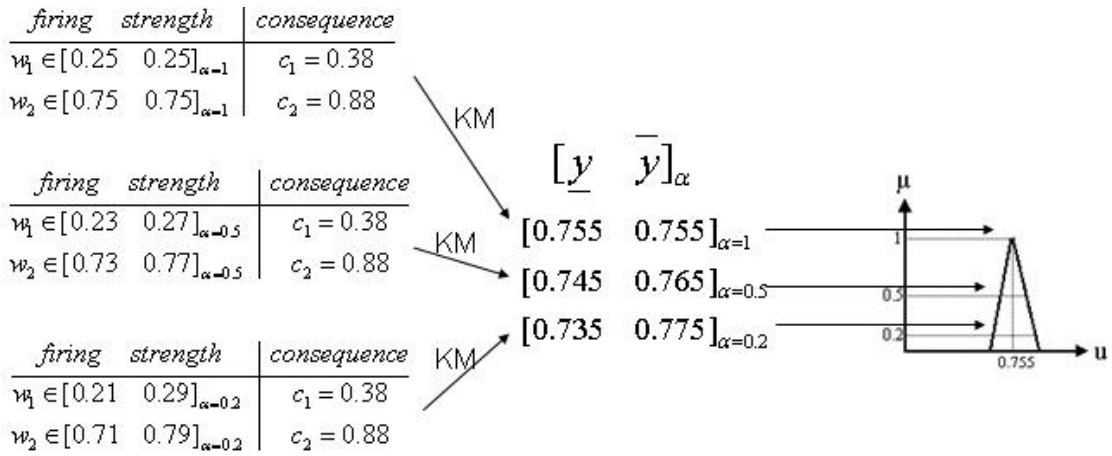

圖 2.3：第二型模糊推論的降階

2.2，將第二型模糊推論的模糊規則啟動強度(它是一個第一型模糊集合)透過 α 截集切分成數個區間，一個模糊規則同一個 α 會對應到一個區間。如圖 2.3，將同一個 α 所對應到的所有模糊規則的區間集合起來，將這些集合透過 KM 演算法可以做區間第二型模糊集合的降階，降階的結果會得到一個區間。這些區間是一一對應到一個 α 值，將這些區間在組合起來第二型模糊推論的降階就完成了。這種降階的過程與 Liu[20]所提出來的方法和想法是類似的，也就是換一個觀點來看的話，第二型模糊推論的降階是使用 α 平面來切割第二型模糊集合，其結果會跟使用 α 截集切割第二型模糊集合的第二階歸屬函數一樣。

我們使用數學式子並且舉一個例子來幫助我們了解 α 截集和 α 平面。給定一個第一型模糊集合，令它為 A。則 A 的 α 截集，表示為 ${}^{\alpha}A$ 為一個純量集合：

$$ {}^{\alpha}A = \{x \mid \mu_A(x) \geq \alpha\} \tag{2.2}$$

其中，α 是一個大於 0 小於等於 1 的實數。從式子 2.2 可以得知 ${}^{\alpha}A$ 是一個模糊集合 A 的歸屬函數值大於等於 α 的所有 x 值，而且 ${}^{\alpha}A$ 是一個區間，由 x 值所集合而成的區間。我們進一步的定義一個與 ${}^{\alpha}A$ 相關聯的模糊集合 ${}_{\alpha}A$：

$$ {}_{\alpha}A = \alpha \cdot {}^{\alpha}A \tag{2.3}$$

將所有 α 對應的模糊集合 ${}_{\alpha}A$ 聯集起來，可以得到原來的模糊集合 A：

$$ A = \bigcup_{\alpha \in [0,1]} {}_{\alpha}A \tag{2.4}$$

我們現在來舉一個實際例子，令一個第一型模糊集合 A：

$$A = \frac{0.3}{x_1} + \frac{0.5}{x_2} + \frac{1}{x_3}$$

我們試著對 A 取兩個 α 截集，這裡的例子取的是 $\alpha = 0.1$ 和 $\alpha = 0.3$：

$$^{0.1}A = \frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3}$$
$$^{0.3}A = \frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3}$$

接著，我們對 $^{0.1}A$ 和 $^{0.3}A$ 取對應的模糊集合 $_{0.1}A$ 和 $_{0.3}A$：

$$_{0.1}A = \frac{0.1}{x_1} + \frac{0.1}{x_2} + \frac{0.1}{x_3}$$
$$_{0.3}A = \frac{0.3}{x_1} + \frac{0.3}{x_2} + \frac{0.3}{x_3}$$

然後將所有的模糊集合 $_{\alpha}A$ 聯集起來：

$$_{0.1}A \cup _{0.3}A \cup _{0.5}A \cup _{1}A = \frac{0.3}{x_1} + \frac{0.5}{x_2} + \frac{1}{x_3} = A$$

就會得到原本的模糊集合 A。

我們另外給定一個第二型模糊集合 $\widetilde{A}$。定義 $\widetilde{A}(x)$ 為第二型模糊集合 $\widetilde{A}$ 的第二階歸屬函數值，所以我們知道 $\widetilde{A}(x)$ 為一個第一型模糊集合。既然 $\widetilde{A}(x)$ 是一個第一型模糊集合，則我們也可以對 $\widetilde{A}(x)$ 取 α 截集：

$$^{\alpha}\widetilde{A}(x) = \left\{(x,u) \mid \mu_{\widetilde{A}}(x,u) \geq \alpha\right\} \tag{2.5}$$

其中 α 是一個大於 0 小於等於 1 的實數。另外 u 是一個介於 0 到 1 之間的實數，它是第二型模糊集合 $\widetilde{A}$ 的第二階歸屬函數的定義域(domain)。接著我們對 $^{\alpha}\widetilde{A}(x)$ 取 α 對應的模糊集合 $_{\alpha}\widetilde{A}(x)$：

$$_{\alpha}\widetilde{A}(x) = \alpha \cdot {}^{\alpha}\widetilde{A}(x) \tag{2.6}$$

若我們將不同 x 值的區間 ${}^{\alpha}\widetilde{A}(x)$ 全部聯集起來，形成一個二維的平面，這個特別的平面我們稱之為 α 平面：

$$\widetilde{A}^{\alpha} = \bigcup_{x} {}^{\alpha}\widetilde{A}(x) \tag{2.7}$$

我們一樣對 $\widetilde{A}^{\alpha}$ 取 α 對應的模糊集合：

$$\widetilde{A}_{\alpha} = \alpha \cdot \widetilde{A}^{\alpha} \tag{2.8}$$

做到這，我們先整理一下。首先，$\widetilde{A}^{\alpha}$ 是一個區間第二型模糊集合。因為 $\widetilde{A}^{\alpha}$ 的第二階歸屬函數值均為 1。而 $\widetilde{A}_{\alpha}$ 也是一個區間第二型模糊集合，它的第二階歸屬函數值均為 α。既然 $\widetilde{A}_{\alpha}$ 是一個區間第二型模糊集合，則它就可以使用 KM 演算法[20]來進行降階，也就是 Liu 的觀點。當然 α 平面也與 α 截集具有相同的特性，那就是將所有的 $\widetilde{A}_{\alpha}$ 聯集起來，就可以得到原來的模糊集合 $\widetilde{A}$：

$$\widetilde{A} = \bigcup_{\alpha \in [0,1]} \widetilde{A}_{\alpha} \tag{2.9}$$

為方便理解，我們依然給一個例子來實際操作一下。給定一個第二型模糊集合 $\widetilde{A}$：

$$\widetilde{A} = \underbrace{\frac{0.3}{0.1} + \frac{1}{0.2} + \frac{0.7}{0.3}}_{x_1} + \underbrace{\frac{0.7}{0.4} + \frac{1}{0.5} + \frac{0.5}{0.6} + \frac{0.1}{0.7}}_{x_2} + \underbrace{\frac{0.8}{0.9} + \frac{1}{1}}_{x_3}$$

就這個例子，我們取 $\widetilde{A}$ 這個第二型模糊集合在 $x = x_1$、$x = x_2$、$x = x_3$ 時第二階歸屬函數：

$$\widetilde{A}(x_1) = \frac{0.3}{0.1} + \frac{1}{0.2} + \frac{0.7}{0.3}$$
$$\widetilde{A}(x_2) = \frac{0.7}{0.4} + \frac{1}{0.5} + \frac{0.5}{0.6} + \frac{0.1}{0.7}$$
$$\widetilde{A}(x_3) = \frac{0.8}{0.9} + \frac{1}{1}$$

$\widetilde{A}(x_1)$、$\widetilde{A}(x_2)$、$\widetilde{A}(x_3)$ 為第一型的模糊集合，也是同時也是 $\widetilde{A}$ 這個第二型模糊集

合在 $x = x_1$、$x = x_2$、$x = x_3$ 時第二階歸屬函數。接著，我們對 $\widetilde{A}(x_1)$、$\widetilde{A}(x_2)$、$\widetilde{A}(x_3)$ 取 α 截集，$\alpha = 0.1$。

$$
\begin{aligned}
{}^{0.1}\widetilde{A}(x_1) &= \frac{1}{0.1} + \frac{1}{0.2} + \frac{1}{0.3} \\
{}^{0.1}\widetilde{A}(x_2) &= \frac{1}{0.4} + \frac{1}{0.5} + \frac{1}{0.6} + \frac{1}{0.7} \\
{}^{0.1}\widetilde{A}(x_3) &= \frac{1}{0.9} + \frac{1}{1}
\end{aligned}
$$

接著我們對 ${}^{0.1}\widetilde{A}(x)$ 取對應的模糊集合 ${}_{0.1}\widetilde{A}(x)$：

$$
\begin{aligned}
{}_{0.1}\widetilde{A}(x_1) &= \frac{0.1}{0.1} + \frac{0.1}{0.2} + \frac{0.1}{0.3} \\
{}_{0.1}\widetilde{A}(x_2) &= \frac{0.1}{0.4} + \frac{0.1}{0.5} + \frac{0.1}{0.6} + \frac{0.1}{0.7} \\
{}_{0.1}\widetilde{A}(x_3) &= \frac{0.1}{0.9} + \frac{0.1}{1}
\end{aligned}
$$

$\alpha = 0.1$ 的 α 平面 $\widetilde{A}^{0.1}$：

$$
\widetilde{A}^{0.1} = \underbrace{\frac{1}{0.1} + \frac{1}{0.2} + \frac{1}{0.3}}_{x_1} + \underbrace{\frac{1}{0.4} + \frac{1}{0.5} + \frac{1}{0.6} + \frac{1}{0.7}}_{x_2} + \underbrace{\frac{1}{0.9} + \frac{1}{1}}_{x_3}
$$

有了這個實際的例子，就比較清楚的知道 α 截集和 α 平面的觀念了。

2.2 第二型模糊邏輯系統

第二型模糊邏輯系統包含下列幾個元件：模糊化(fuzzifier)、模糊規則庫(fuzzy rule base)、模糊推論引擎(fuzzy inference engine)和輸出流程(output processing)。其中，輸出流程包含了降階器(type-reducer)和解模糊化(defuzzifier)。整個第二型模糊邏輯系統運作的流程如圖 2.4。我們對每個元件一一簡述如下：

區塊 1 **模糊化**：對每個輸入進系統的純量，第二型的模糊化可以將純量轉換成模糊量。在這個步驟中，每一個輸入值，透過第二型模糊集合，都會得到一個第二階歸屬函數值，這個第二階歸屬函數值同時也是一個第一型模糊集合。我們令這個第二階歸屬函數值為 $\widetilde{A}_{i,j}(x_i)$，$i = 1,2,\cdots,n$ 且

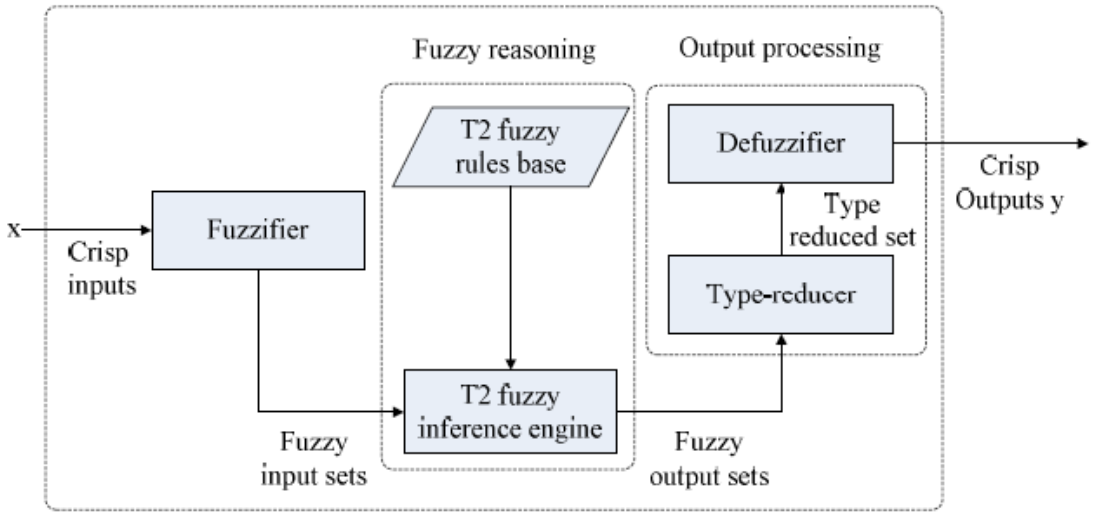

圖 2.4：第二型模糊邏輯系統

$j=1,2,\cdots,J$。其中n是單筆資料輸入的維度個數，J是第二型模糊法則的法則個數。我們可以將$\tilde{A}_{i,j}(x_i)$這個第一型模糊集合分割如下：

$$\tilde{A}_{i,j}(x_i)=_{\alpha_1}\tilde{A}_{i,j}(x_i)\cup_{\alpha_2}\tilde{A}_{i,j}(x_i)\cup\cdots\cup_{\alpha_M}\tilde{A}_{i,j}(x_i) \tag{2.10}$$

其中M為α截集的數量；換句話說，M為一個第一型模糊集合切割後的份數。令區間$[l_{i,j,\alpha},r_{i,j,\alpha}]$為$\tilde{A}_{i,j}(x_i)$的α截集，也就是說$l_{i,j,\alpha}=\min\left({}^{\alpha}\tilde{A}_{i,j}(x_i)\right)$、$r_{i,j,\alpha}=\max\left({}^{\alpha}\tilde{A}_{i,j}(x_i)\right)$。式子(2.10)就可以被改寫成：

$$\tilde{A}_{i,j}(x_i)=\alpha_1\cdot\left[l_{i,j,\alpha_1},r_{i,j,\alpha_1}\right]\cup\alpha_2\cdot\left[l_{i,j,\alpha_2},r_{i,j,\alpha_2}\right]\cup\cdots\cup\alpha_M\cdot\left[l_{i,j,\alpha_M},r_{i,j,\alpha_M}\right] \tag{2.11}$$

根據2.11式，我們可以知道我們可以藉由計算$l_{i,j,\alpha_1},l_{i,j,\alpha_2},\cdots,l_{i,j,\alpha_M},r_{i,j,\alpha_1},r_{i,j,\alpha_2},\cdots,r_{i,j,\alpha_M}$來表示$\tilde{A}_{i,j}(x_i)$這個第一型模糊集合。

區塊 2 **模糊推論**：模糊推論區塊包含了模糊規則庫和模糊推論引擎。模糊規則庫是一個包含了許多 IF-THEN 規則的集合。而模糊推論引擎則可以將區塊 1 模糊化的輸出搭配模糊規則庫做模糊推論。模糊推論的結果是每條模糊規則都會得到自己的啟動強度。要得到啟動強度，我們必須對同一條模糊法則不同輸入維度間的歸屬函數值做 T-norm 運算。一般來說，有兩種 T-nrom 運算可以做選擇，MIN 和 PRODUCT 兩種。在本論文中，我們選

用 PRODUCT 這種 T-nrom 來做運算，好得到各個模糊規則的啟動強度。所以，第 j 條模糊法則的啟動強度可以被表示成：

$$f_j = \tilde{A}_{1,j}(x_1) * \tilde{A}_{2,j}(x_2) * \cdots * \tilde{A}_{n,j}(x_n) \tag{2.12}$$

n 為輸入資料的維度個數。$\tilde{A}_{i,j}(x_i)$ 是一個第一型模糊集合，所以我們必須對兩個第一型模糊集合做 T-norm 運算才可以求出各模糊規則的啟動強度 f_j。根據[14]這本書中，對於兩模糊集合 A、B 做 PRODUCT 這種 T-nrom 運算的話，PRODUCT 運算 $*$ 定義為：

$$(A * B) = \bigcup_{\alpha \in [0,1]} \alpha (A * B) \tag{2.13}$$

到這邊，我們知道兩模糊集合做 PRODUCT 要用 2.13 式，而要得到各模糊規則的啟動強度要用 2.12 式，再搭配 2.11 式來重新表示每個 $A_{i,j}(x_i)$，則我們可以將 2.12 式做如下推導：

$$\begin{aligned} f_j &= \tilde{A}_{1,j}(x_1) * \tilde{A}_{2,j}(x_2) * \cdots * \tilde{A}_{n,j}(x_n) \\ &= \bigcup_{\alpha \in [0,1]} \alpha \left(\tilde{A}_{1,j}(x_1) * \tilde{A}_{2,j}(x_2) * \cdots * \tilde{A}_{n,j}(x_n) \right) \\ &= \bigcup_{\alpha \in [0,1]} \alpha \cdot \left[\prod_{i=1}^{n} l_{i,j,\alpha}, \prod_{i=1}^{n} r_{i,j,\alpha} \right] \\ &= \bigcup_{\alpha \in [0,1]} \alpha \cdot [\underline{f}_{j,\alpha}, \overline{f}_{j,\alpha}] \end{aligned} \tag{2.14}$$

其中，$l_{i,j,\alpha}$ 是第 j 條模糊規則第 i 個輸入維度的第二階歸屬函數在 α 值的 α 截集的區間下限；$r_{i,j,\alpha}$ 是第 j 條模糊規則第 i 個輸入維度的第二階歸屬函數在 α 值的 α 截集的區間下限。$\underline{f}_{j,\alpha}$ 和 $\overline{f}_{j,\alpha}$ 都是純量，它們分別代表第 j 條模糊規則的啟動強度在 α 值的 α 截集的區間的下限和上限。2.14 式告訴我們：想要對第二型模糊集合做模糊推論、想要得到模糊規則的啟動強度所需要的 T-nrom 運算需要使用 α 截集的概念，透過 $l_{i,j,\alpha}$、$r_{i,j,\alpha}$ 來完成實作。

區塊 3 **輸出流程**：模糊推論的輸出是各個第二型模糊法則的啟動強度，簡單的

說，就是一堆第一型的模糊集合。以本論文所使用的 TSK 模型而言，這些啟動強度要跟後鑑部做一個加權平均。由於第二型模糊推論的啟動強度是一個第一型模糊集合，所以，這裡會面臨一個第二型模糊集合的降階問題。而這個問題，我們使用降階器來做降階，輸出降階後的結果去做解模糊化完成最後的輸出。對區間第二型模糊推論的降階問題而言，已經存在一個準確且有效率的演算法，就是 KM[11][25]演算法。遺憾的是，KM 演算法並無法直接套用在第二型模糊推論的降階問題上。在這個部份，我們依然使用 α 截集的概念，將各模糊規則的啟動強度，也就是每個第一型模糊集合分解成多個區間。這麼一來就會把原來第二型模糊推論的降階問題轉換成多個區間第二型模糊推論的降階問題。使用 KM 演算法來得到每個區間第二型模糊推論的降階結果之後，再將這接結果組合起來就完成了第二型模糊推論的降階了。所以，我們可以知道第二型模糊推論的降階輸出值是一個第一型模糊集合，我們將之表示如下：

$$\bigcup_{\alpha\in[0,1]}\alpha\cdot[\underline{y}_{\alpha},\overline{y}_{\alpha}] \tag{2.15}$$

其中 $\underline{y}_{\alpha}$ 和 $\overline{y}_{\alpha}$ 是 KM 演算法在各 α 值的輸出，$\underline{y}_{\alpha}$ 是對應各 α 值的左端點，$\overline{y}_{\alpha}$ 是對應各 α 值的右端點。$\underline{y}_{\alpha}$ 與 $\overline{y}_{\alpha}$ 公式如下：

$$\underline{y}_{\alpha}=\frac{\sum_{j=1}^{L}b_j\overline{g}_{j,\alpha}+\sum_{j=L+1}^{J}b_j\underline{g}_{j,\alpha}}{\sum_{j=1}^{L}\overline{g}_{j,\alpha}+\sum_{j=L+1}^{J}\underline{g}_{j,\alpha}} \tag{2.16}$$

$$\overline{y}_{\alpha}=\frac{\sum_{j=1}^{R}b_j\underline{g}_{j,\alpha}+\sum_{j=R+1}^{J}b_j\overline{g}_{j,\alpha}}{\sum_{j=1}^{R}\underline{g}_{j,\alpha}+\sum_{j=R+1}^{J}\overline{g}_{j,\alpha}} \tag{2.17}$$

其中 L、R 是由 KM 演算法得到的數值，要注意的是不同的 α 值所算得的 L 值有可能相同也有可能不同；當然，不同的 α 值所算得的 R 值也是有可能相同也有可能不同。TSK 模型在使用 KM 演算法做降階運算時，需要先由小到大排序後鑑部 c_j [6]，每個模糊規則的啟動強度 $[\underline{f}_{j,\alpha},\overline{f}_{j,\alpha}]$ 也要跟

著原本對應的c_j一起改變排列順序;換句話說就是模糊規則的順序要依照c_j的大小，由小至大排列，然後才可以進行 KM 演算法的運算。所以，b_j是將各模糊規則的原來的後鑑部c_j重新排序過後的新的後鑑部，而$[\underline{g}_{j,\alpha}, \overline{g}_{j,\alpha}]$是$[\underline{f}_{j,\alpha}, \overline{f}_{j,\alpha}]$移動順序之後的啟動強度。所以，由 2.16 式和 2.17 式我們可以得到一個第一型模糊集合的輸出，也就是 2.15 式。至此，我們已經完成了降階器的輸出，TSK 模型的第二型模糊推論降階器的輸出是一個第一型模糊集合。如果我們想要得到的是一個純量的輸出，則我們必須進一步處理 2.15 式，最直觀的處理方式當然就是將 2.15 式把 α 值當作權重作一個加權平均作為純量輸出。這個步驟，就是解模糊化：

$$\hat{y} = \frac{\sum_{\alpha} \alpha \cdot \frac{\underline{y}_{\alpha} + \overline{y}_{\alpha}}{2}}{\sum_{\alpha} \alpha} \tag{2.18}$$

2.3 第二型模糊邏輯系統範例

為方便理解，這個小節我們給一個實際運算的例子來說明第二型模糊邏輯系統的運算流程。當然，我們使用的是 TSK 這種模糊推論的模型。假定系統有四條模糊規則，輸入的維度兩維x_1、x_2，輸出維度一維y。

$$\text{Rule 1：IF } x_1 \text{ is } \mu_{\tilde{A}_{1,1}(x_1)} \text{ and } x_2 \text{ is } \mu_{\tilde{A}_{2,1}(x_2)} \text{ THEN } y \text{ is } c_1$$
$$\text{Rule 2：IF } x_1 \text{ is } \mu_{\tilde{A}_{1,2}(x_1)} \text{ and } x_2 \text{ is } \mu_{\tilde{A}_{2,2}(x_2)} \text{ THEN } y \text{ is } c_2$$
$$\text{Rule 3：IF } x_1 \text{ is } \mu_{\tilde{A}_{1,3}(x_1)} \text{ and } x_2 \text{ is } \mu_{\tilde{A}_{2,3}(x_2)} \text{ THEN } y \text{ is } c_3$$
$$\text{Rule 4：IF } x_1 \text{ is } \mu_{\tilde{A}_{1,4}(x_1)} \text{ and } x_2 \text{ is } \mu_{\tilde{A}_{2,4}(x_2)} \text{ THEN } y \text{ is } c_4$$

其中前鑑部使用的是第二型模糊集合。為了減少計算量，我們的只用了三個參數來表示一個第二型模糊集合。這三個參數分別是：第一階歸屬函數(高斯函數，Gaussian)的中心值$m_{i,j}$、第一階歸屬函數的標準差$\sigma 1_{i,j}$、第二階歸屬函數(高斯函數) 的標準差$\sigma 2_{i,j}$，其中，i表示輸入的維度，j表示此參數屬於哪一條模糊

規則。第二階歸屬函數的中心值使用的是第一階歸屬函數的輸出值。所以前鑑部的參數如下：

$$\begin{bmatrix} m_{1,1} & \sigma 1_{1,1} & \sigma 2_{1,1} & m_{2,1} & \sigma 1_{2,1} & \sigma 2_{2,1} \\ m_{1,2} & \sigma 1_{1,2} & \sigma 2_{1,2} & m_{2,2} & \sigma 1_{2,2} & \sigma 2_{2,2} \\ m_{1,3} & \sigma 1_{1,3} & \sigma 2_{1,3} & m_{2,3} & \sigma 1_{2,3} & \sigma 2_{2,3} \\ m_{1,4} & \sigma 1_{1,4} & \sigma 2_{1,4} & m_{2,4} & \sigma 1_{2,4} & \sigma 2_{2,4} \end{bmatrix}$$

$$= \begin{bmatrix} -0.650 & 0.300 & 0.0300 & -0.760 & 0.206 & 0.0206 \\ 0.473 & 0.250 & 0.0250 & 0.633 & 0.215 & 0.0215 \\ -0.655 & 0.206 & 0.0206 & 0.655 & 0.135 & 0.0135 \\ 0.720 & 0.185 & 0.0185 & -0.830 & 0.128 & 0.0128 \end{bmatrix}$$

這個例子我們使用第零階的 TSK 模型(zero order TSK)，所以，我們只有四個後鑑部的參數：

$$c = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix} = [-0.2660 \quad -0.3533 \quad 0.3046 \quad 1.3992]$$

參數設定好之後，我們就可以實際來運算看看了。給定一組輸入 $\mathrm{x} = \begin{bmatrix} -0.85, & -0.88 \end{bmatrix}$和一個輸出 $y = -0.2660$。我們使用 PRODUCT 來當 T-norm，α 值取 1、0.6、0.2 三組來做 α 截集。各步驟運算細節如下：

模糊化：根據上節的描述，我們知道模糊化最終要得到的是所有 $[l_{i,j,\alpha}, r_{i,j,\alpha}]$ 的區間。i 表示輸入維度，以本例子而言 $i = 1,2$。j 表示第幾條模糊規則，所以 $j = 1,2,3,4$。α 在本例子是 1、0.6、0.2 三種可能。再得到 $[l_{i,j,\alpha}, r_{i,j,\alpha}]$ 之前必須先求出第二階歸屬函數的中心值，也就是第一階歸屬函數的輸出 $\mu_{i,j}$：

$$\mu_{1,1} = 0.6412 \quad \mu_{2,1} = 0.7118$$
$$\mu_{1,2} = 0.0000 \quad \mu_{2,2} = 0.0000$$
$$\mu_{1,3} = 0.4084 \quad \mu_{2,3} = 0.0000$$
$$\mu_{1,4} = 0.0000 \quad \mu_{2,4} = 0.8591$$

藉由 $\mu_{i,j}$ 就可以得到第二階歸屬函數 $[l_{i,j,\alpha}, r_{i,j,\alpha}]$，如表 2.1。需要注意的是：$[l_{i,j,\alpha}, r_{i,j,\alpha}]$ 也就是第二階歸屬函數值的計算上，我們有設定 $l_{i,j,\alpha}$ 和 $r_{i,j,\alpha}$ 的範圍 $0 \le l_{i,j,\alpha} \le r_{i,j,\alpha} \le 1$。若 $l_{i,j,\alpha}$ 和 $r_{i,j,\alpha}$ 小於 0 則為 0；若 $l_{i,j,\alpha}$ 和 $r_{i,j,\alpha}$ 大於 1 則為 1。

模糊推論：根據上節的描述我們知道模糊推論是要將第二階歸屬函數

$[l_{i,j,\alpha}, r_{i,j,\alpha}]$作 T-norm 運算，最後要得到每個模糊規則的啟動強度$[\underline{f}_{j,\alpha}, \overline{f}_{j,\alpha}]$。這個部份的運算要將表 2.1 的資料代入 2.14 式中，就可以算出$[\underline{f}_{j,\alpha}, \overline{f}_{j,\alpha}]$，我

表 2.1：第二階歸屬函數 α 截集實例

	Dimension = 1		Dimension = 2	
Rule 1	$[l_{1,1,1}, r_{1,1,1}]$	[0.6412,0.6412]	$[l_{2,1,1}, r_{2,1,1}]$	[0.7118,0.7118]
	$[l_{1,1,0.6}, r_{1,1,0.6}]$	[0.6197,0.6626]	$[l_{2,1,0.6}, r_{2,1,0.6}]$	[0.6971,0.7266]
	$[l_{1,1,0.2}, r_{1,1,0.2}]$	[0.6031,0.6792]	$[l_{2,1,0.2}, r_{2,1,0.2}]$	[0.6857,0.7380]
Rule 2	$[l_{1,2,1}, r_{1,2,1}]$	[0.0000,0.0000]	$[l_{2,2,1}, r_{2,2,1}]$	[0.0000,0.0000]
	$[l_{1,2,0.6}, r_{1,2,0.6}]$	[0.0000,0.0179]	$[l_{2,2,0.6}, r_{2,2,0.6}]$	[0.0000,0.0154]
	$[l_{1,2,0.2}, r_{1,2,0.2}]$	[0.0000,0.0317]	$[l_{2,2,0.2}, r_{2,2,0.2}]$	[0.0000,0.0273]
Rule 3	$[l_{1,3,1}, r_{1,3,1}]$	[0.4084,0.4084]	$[l_{2,3,1}, r_{2,3,1}]$	[0.0000,0.0000]
	$[l_{1,3,0.6}, r_{1,3,0.6}]$	[0.3937,0.4231]	$[l_{2,3,0.6}, r_{2,3,0.6}]$	[0.0000,0.0097]
	$[l_{1,3,0.2}, r_{1,3,0.2}]$	[0.3823,0.4346]	$[l_{2,3,0.2}, r_{2,3,0.2}]$	[0.0000,0.0172]
Rule 4	$[l_{1,4,1}, r_{1,4,1}]$	[0.0000,0.0000]	$[l_{2,4,1}, r_{2,4,1}]$	[0.8591,0.8591]
	$[l_{1,4,0.6}, r_{1,4,0.6}]$	[0.0000,0.0132]	$[l_{2,4,0.6}, r_{2,4,0.6}]$	[0.8499,0.8682]
	$[l_{1,4,0.2}, r_{1,4,0.2}]$	[0.0000,0.0235]	$[l_{2,4,0.2}, r_{2,4,0.2}]$	[0.8428,0.8753]

表 2.2：啟動強度實例

	Rule 1	Rule 2	Rule 3	Rule 4
$[\underline{f}_{j,1}, \overline{f}_{j,1}]$	[0.4564,0.4564]	[0.0000,0.0000]	[0.0000,0.0000]	[0.0000,0.0000]
$[\underline{f}_{j,0.6}, \overline{f}_{j,0.6}]$	[0.4320,0.4814]	[0.0000,0.0003]	[0.0000,0.0041]	[0.0000,0.0115]
$[\underline{f}_{j,0.2}, \overline{f}_{j,0.2}]$	[0.4136,0.5012]	[0.0000,0.0009]	[0.0000,0.0075]	[0.0000,0.0205]

們將計算的結果列於表 2.2。

輸出流程：這個步驟有兩個子步驟，就是降階和解模糊化。我們將利用剛剛得到的啟動強度$[\underline{f}_{j,\alpha}, \overline{f}_{j,\alpha}]$和後鑑部$c_j$作為輸入值，使用 KM 演算法來做降階。這裡我們使用$\alpha = 0.6$，也就是表 2.2 中$[\underline{f}_{j,0.6}, \overline{f}_{j,0.6}]$這組數據作為 KM 演算法的輸入值來實際演練 KM 演算法。

Step 1：將後鑑部c_j由小到大排序，在後鑑部c_j互相交換的同時，其對應的

啟動強度$[\underline{f}_{j,0.6}, \overline{f}_{j,0.6}]$也須一起交換。也就是說模糊規則的順序要依照$c_j$的大小，由小至大排列。表 2.3 列出了$c_j$、$[\underline{f}_{j,0.6}, \overline{f}_{j,0.6}]$、$b_j$、$[\underline{g}_{j,0.6}, \overline{g}_{j,0.6}]$

表 2.3：模糊規則排序示意表

original			reorder		
Rule	$[\underline{f}_{j,0.6}, \overline{f}_{j,0.6}]$	c_j	Rule	$[\underline{g}_{j,0.6}, \overline{g}_{j,0.6}]$	b_j
1	[0.4320,0.4814]	-0.2660	2	[0.0000,0.0003]	-0.3533
2	[0.0000,0.0003]	-0.3533	1	[0.4320,0.4814]	-0.2660
3	[0.0000,0.0041]	0.3046	3	[0.0000,0.0041]	0.3046
4	[0.0000,0.0115]	1.3992	4	[0.0000,0.0115]	1.3992

之間的關係。

Step 2：初始化 KM 演算法。每條模糊規則的$\underline{g}_{j,0.6}$和$\overline{g}_{j,0.6}$相加除以二當作 KM 演算法的初始值。

$$\hat{g}_{1,0.6} = \frac{\underline{g}_{1,0.6} + \overline{g}_{1,0.6}}{2} = \frac{0.0000 + 0.0003}{2} = 0.00015$$

$$\hat{g}_{2,0.6} = \frac{\underline{g}_{2,0.6} + \overline{g}_{2,0.6}}{2} = \frac{0.4320 + 0.4814}{2} = 0.4567$$

$$\hat{g}_{3,0.6} = \frac{\underline{g}_{3,0.6} + \overline{g}_{3,0.6}}{2} = \frac{0.0000 + 0.0041}{2} = 0.00205$$

$$\hat{g}_{4,0.6} = \frac{\underline{g}_{4,0.6} + \overline{g}_{4,0.6}}{2} = \frac{0.0000 + 0.0105}{2} = 0.00575$$

$$\Rightarrow \underline{y}_{0.6} = \underline{\hat{y}}_{0.6} = \frac{\hat{g}_{1,0.6} \times b_1 + \hat{g}_{2,0.6} \times b_2 + \hat{g}_{3,0.6} \times b_3 + \hat{g}_{4,0.6} \times b_4}{\hat{g}_{1,0.6} + \hat{g}_{2,0.6} + \hat{g}_{3,0.6} + \hat{g}_{4,0.6}} = -0.2429$$

Step 3：$\because -0.2660 \le -0.2429 \le -0.3046 \Leftrightarrow b_2 \le \underline{\hat{y}}_{0.6} \le b_3$

$\therefore L_{0.6} = 2$

Step 4：根據 KM 演算法的計算法則，因為$L_{0.6} = 2$

$$\Rightarrow \begin{matrix} \hat{g}_{1,0.6} = \overline{g}_{1,0.6} = 0.0003 \\ \hat{g}_{2,0.6} = \overline{g}_{2,0.6} = 0.4814 \\ \hat{g}_{3,0.6} = \underline{g}_{3,0.6} = 0.0000 \\ \hat{g}_{4,0.6} = \underline{g}_{4,0.6} = 0.0000 \end{matrix} \Rightarrow \underline{\hat{y}}_{0.6} = -0.2660$$

Step 5：$\because \underline{\hat{y}}_{0.6} \ne \underline{y}_{0.6}$，到 Step 6

Step 6：令 $\underline{y}_{0.6} = \underline{\hat{y}}_{0.6} = -0.2660$ ，回到 Step 3

Step 3：$\because -0.3533 \le -0.2660 \le -0.2660 \Leftrightarrow b_1 \le \underline{\hat{y}}_{0.6} \le b_2$

$\therefore L_{0.6} = 1$

Step 4：因為 $L_{0.6} = 1$

$$\Rightarrow \begin{matrix} \hat{g}_{1,0.6} = \overline{g}_{1,0.6} = 0.0003 \\ \hat{g}_{2,0.6} = \underline{g}_{2,0.6} = 0.4320 \\ \hat{g}_{3,0.6} = \underline{g}_{3,0.6} = 0.0000 \\ \hat{g}_{4,0.6} = \underline{g}_{4,0.6} = 0.0000 \end{matrix} \Rightarrow \underline{\hat{y}}_{0.6} = -0.2660$$

Step 5：$\because \underline{\hat{y}}_{0.6} = \underline{y}_{0.6}$ 達成收斂條件。$\therefore \underline{y}_{0.6} = -0.2660, L_{0.6} = 1$

重複上述 KM 演算法，直到所有的 $\underline{y}_\alpha$ 與 $\overline{y}_\alpha$ 均被計算出來為止。我們將 $\underline{y}_\alpha$ 與 $\overline{y}_\alpha$ 的計算結果列於表 2.4。

表 2.4：降階運算的結果

$[\underline{y}_1, \overline{y}_1]$	[−0.2660,−0.2660]
$[\underline{y}_{0.6}, \overline{y}_{0.6}]$	[−0.2660,−0.2181]
$[\underline{y}_{0.2}, \overline{y}_{0.2}]$	[−0.2662,−0.1789]

最後，我們將表 2.4，也就是降階之後的結果使用 2.18 式做解模糊化：

$$\hat{y} = \frac{\sum_\alpha \alpha \cdot \frac{\underline{y}_\alpha + \overline{y}_\alpha}{2}}{\sum_\alpha \alpha}$$

$$= \frac{1 \times \frac{-0.2660 + -0.2660}{2} + 0.6 \times \frac{-0.2660 + -0.2181}{2} + 0.2 \times \frac{-0.2662 + -0.1789}{2}}{1 + 0.6 + 0.2}$$

$$= -0.2532$$

第三章

加速的降階演算法

第三章我們已經介紹完模糊邏輯系統，其中模糊降階的部份，是使用 α 截集將啟動強度切割成數個區間，然後使用 KM 演算法去完成第二型模糊推論的降階。在這個部份，如果我們將啟動強度切割的非常細，也就是 α 截集取非常多個，這時候我們將面臨非常大的計算量。在這章，我們將提供一個降階加速的方法來減少運算所需的時間。

首先，我們快速的描述一下第二階模糊推論的降階流程。我們會拿到數個成對的啟動強度 $\bigcup_{\alpha}[\underline{f}_{j,\alpha},\overline{f}_{j,\alpha}]$ 和後鑑部 c_j ，按照 c_j 由小到大排序後得到 $\bigcup_{\alpha}[\underline{g}_{j,\alpha},\overline{g}_{j,\alpha}]$ 與 b_j 。接下來就把一組一組的 $[\underline{g}_{j,\alpha},\overline{g}_{j,\alpha}]$ 和 b_j 輸入 KM 作區間第二型模糊降階得到 $[\underline{y}_{\alpha},\overline{y}_{\alpha}]$，最後再把各 α 截集降階的結果組合起來 $\bigcup_{\alpha}[\underline{y}_{\alpha},\overline{y}_{\alpha}]$ 就可以完成第二階模糊推論的降階。假設我們將一個啟動強度切成 M 份 α 截集，則 KM 演算法就必須要運作 M 次。但是，如果啟動強度是一個具有凸性(convexity)的第一型模糊集合的話，則就可以使用我們的方法來加速運算。給定 $0<\alpha_M<\alpha_{M-1}<\cdots<\alpha_1\le 1$ 為要切的 α 截集，則我們要使用 KM 來計算 $[\underline{y}_{\alpha_2},\overline{y}_{\alpha_2}]$ 的時候，可以使用 $[\underline{y}_{\alpha_1},\overline{y}_{\alpha_1}]$ 來當 KM 演算法的初始值；以此類推，要算 $[\underline{y}_{\alpha_3},\overline{y}_{\alpha_3}]$ 的時候，可以使用 $[\underline{y}_{\alpha_2},\overline{y}_{\alpha_2}]$ 來當 KM 演算法的初始值。

為方便理解我們附上虛擬碼以及一個簡單的例子。虛擬碼就在後面兩頁，例子我們就沿用 2.3 節的例子。假定我們已經做出 $\underline{y}_1$ ，所以我們會知道 L_1 是多少，在我們的例子會算出 $L_1=1$，所以我們就會知道使用 $\underline{y}_1$ 來做初始值，初始的 $L_{0.6}$ 也是 1。所以我們就可以來計算 $\underline{y}_{0.6}$ 了：

Step 1：先將啟動強度 $\bigcup_{\alpha}[\underline{f}_{j,\alpha},\overline{f}_{j,\alpha}]$ 和後鑑部 c_j，按照 c_j 由小到大排序後得到 $\bigcup_{\alpha}[\underline{g}_{j,\alpha},\overline{g}_{j,\alpha}]$ 與 b_j ，這個部分我們從表 2.3 中取得。

Step 2：$L_1 = 1$(由計算 $\underline{y}_1$ 的過程中得知)，所以 $L_{0.6} = 1$

$$\Rightarrow \begin{matrix} \hat{g}_{1,0.6} = \overline{g}_{1,0.6} = 0.0003 \\ \hat{g}_{2,0.6} = \underline{g}_{2,0.6} = 0.4320 \\ \hat{g}_{3,0.6} = \underline{g}_{3,0.6} = 0.0000 \\ \hat{g}_{4,0.6} = \underline{g}_{4,0.6} = 0.0000 \end{matrix} \Rightarrow \underline{\hat{y}}_{0.6} = -0.2660$$

Step 3：$\because -0.3533 \le -0.2660 \le -0.2660 \Leftrightarrow b_1 \le \underline{\hat{y}}_{0.6} \le b_2$

$\therefore L_{0.6} = 1$，$L_{0.6}$ 的值沒有改變，所以收斂。$\therefore \underline{y}_{0.6} = -0.2660, L_{0.6} = 1$

這裡稍微與 2.3 節不同的是：我們判斷了 $L_{0.6}$ 的值沒有改變，就收斂了。這裡做這樣的判斷是合理的，因為當 $L_{0.6}$ 的值沒有改變 $\hat{g}_{j,0.6}$ 的值也不會改變，進而 $\underline{\hat{y}}_{0.6}$ 的值也不變。所以當判斷到 L_α 的值不變的時候，運算迴圈就可以停了。從這個例子，我們可以發現到計算的過程比 2.3 節少了一次迴圈的步驟(2.3 節例子的 step3、step4、step5、step6)，其原因其實就是因為我們使用了不同的初始值，導致計算的步驟減少了。我們把 2.3 節的例子用原本的方法計算所需的迴圈數，和使用加速運算所需要的迴圈數列於表 3.1。一般的情況來說，我們的方法和 Liu[20] 的方法(也就是 2.3 節的方法)比較起來，大約都可以快 50%的時間，實際的情況要視輸入的維度、α 截集的切割數、模糊規則的個數和所採用的模型有關。

表 3.1：KM 執行次數比較表

	Liu's		Ours	
	$\underline{y}_\alpha$	$\overline{y}_\alpha$	$\underline{y}_\alpha$	$\overline{y}_\alpha$
$\alpha = 1$	2	2	2	2
$\alpha = 0.6$	3	3	1	1
$\alpha = 0.2$	3	3	1	2
Total	8	8	4	5

Algorithm 1 Enhanced KM algorithm for computing $\underline{y}$

Input: The number of α-palnes, M. The firing strength $f = \{f_1, f_2, ..., f_j, ..., f_J\}$, where f_j is a type-1 fuzzy set and $j = 1, ..., J$. The consequent part $c = \{c_1, c_2, ..., c_j, ..., c_J\}$.

Break the α in M values, i.e., $\alpha = \{1, \frac{M-1}{M}, ..., \frac{2}{M}, \frac{1}{M}\}$.

Decompose the firing strength by using α-cut, i.e., $f = \bigcup_\alpha \ \{^\alpha f_1, ^\alpha f_2, ..., ^\alpha f_j, ..., ^\alpha f_J\} = \bigcup_\alpha \ \{[\underline{f}_{1,\alpha}, \ \overline{f}_{1,\alpha}], [\underline{f}_{2,\alpha}, \ \overline{f}_{2,\alpha}], ..., [\underline{f}_{j,\alpha}, \ \overline{f}_{j,\alpha}], ..., [\underline{f}_{J,\alpha}, \ \overline{f}_{J,\alpha}]\}$, where $^{j,\alpha}f$ is a α-cut for f_j, $\underline{f}_{j,\alpha}$ is a lower bound of $^\alpha f_j$, $\overline{f}_{j,\alpha}$ is an upper bound of $^\alpha f_j$, and $j = 1, ..., J$.

Sort c in ascending order and call the sorted c by $b = \{b_1, b_2, ..., b_j, ..., b_J\}$, where $b_1 < b_2 < ... < b_j < ... < b_J$.

Reorder the firing strength, f, with their respective c, and call them $g = \bigcup_\alpha \ \{^\alpha g_1, ^\alpha g_2, ..., ^\alpha g_j, ..., ^\alpha g_J\} = \bigcup_\alpha \ \{[\underline{g}_{1,\alpha}, \ \overline{g}_{1,\alpha}], [\underline{g}_{2,\alpha}, \ \overline{g}_{2,\alpha}], ..., [\underline{g}_{j,\alpha}, \ \overline{g}_{j,\alpha}], ..., [\underline{g}_{J,\alpha}, \ \overline{g}_{J,\alpha}]\}$.

for $i = 1$ to M **do**

  if $i = 1$ **then**

    Initialize $\hat{g}_{j,\alpha_i}$ by setting $\hat{g}_{j,\alpha_i} = \frac{\underline{g}_{j,\alpha_i} + \overline{g}_{j,\alpha_i}}{2}$, $j = 1, 2, ..., J$.

    Compute $y_{init} = \frac{\sum_{j=1}^{J} b_j \hat{g}_{j,\alpha_i}}{\sum_{j=1}^{J} \hat{g}_{j,\alpha_i}}$.

    $y_{\alpha_i} = y_{init}$.

    $L_i = J$ and $L_{i-1} = J$.

  else

    Set $\hat{g}^j_{\alpha_i} = \begin{cases} \overline{g}_{j,\alpha_i}, & j \le L_{i-1} \\ \underline{g}_{j,\alpha_i}, & j > L_{i-1} \end{cases}$.

    Compute $y_{\alpha_i} = \frac{\sum_{j=1}^{J} b_j \hat{g}_{j,\alpha_i}}{\sum_{j=1}^{J} \hat{g}_{j,\alpha_i}}$. // Use the results of the previous step with large α-plane to be the initial values.

  end if

  while true **do**

    for $k = \min(L_{i-1}, J-1)$ to 1 **do**

      If $b_k \le y_{\alpha_i} \le b_{k+1}$, set $L_i = k$ and break this for loop. // Find switch point.

    end for

    if $L_i \neq L_{i-1}$ **then**

      Set $\hat{g}_{j,\alpha_i} = \begin{cases} \overline{g}_{j,\alpha_i}, & j \le L_i \\ \underline{g}_{j,\alpha_i}, & j > L_i \end{cases}$.

      Compute $y'_{\alpha_i} = \frac{\sum_{j=1}^{J} b_j \hat{g}_{j,\alpha_i}}{\sum_{j=1}^{J} \hat{g}_{j,\alpha_i}}$.

    end if

    if $(y'_{\alpha_i} = y_{\alpha_i})$ or $(L_i = L_{i-1})$ **then**

      Stop while loop.

    else

      Set $y_{\alpha_i} = y'_{\alpha_i}$.

    end if

  end while

  Set $\underline{y}_{\alpha_i} = y_{\alpha_i}$.

end for

$\underline{y} = \bigcup_{i=1}^{M} \ \{\underline{y}_{\alpha_i}\}$.

Return: $\underline{y}$, the initial output y_{init}, the reordered consequent part b, and the reordered firing strength g.

Algorithm 2 Enhanced KM algorithm for computing $\overline{y}$

Input: The number of α-palnes, M. The initial output y_{init}. The reordered firing strength $g = \bigcup_{\alpha} \{[\underline{g}_{1,\alpha}, \overline{g}_{1,\alpha}], [\underline{g}_{2,\alpha}, \overline{g}_{2,\alpha}], ..., [\underline{g}_{j,\alpha}, \overline{g}_{j,\alpha}], ..., [\underline{g}_{J,\alpha}, \overline{g}_{J,\alpha}]\}$. The reordered consequent part $b = \{b_1, b_2, ..., b_j, ..., b_J\}$.

Break the α in M values, i.e., $\alpha = \{1, \frac{M-1}{M}, ..., \frac{2}{M}, \frac{1}{M}\}$.
for $i = 1$ to M **do**
  if $i = 1$ **then**
    Set $y_{\alpha_i} = y_{init}$.
    $R_i = 0$ and $R_{i-1} = 0$.
  else
    Set $\hat{g}^j_{\alpha_i} = \begin{cases} \underline{g}_{j,\alpha_i}, & j \leq R_{i-1} \\ \overline{g}_{j,\alpha_i}, & j > R_{i-1} \end{cases}$.
    Compute $y_{\alpha_i} = \frac{\sum_{j=1}^{J} b_j \hat{g}_{j,\alpha_i}}{\sum_{j=1}^{J} \hat{g}_{j,\alpha_i}}$. // Use the results of the previous step with large α-plane to be the initial values.
  end if
  while true **do**
    for $k = \max(R_{i-1}, 1)$ to $J - 1$ **do**
      If $b_k \leq y_{\alpha_i} \leq b_{k+1}$, set $R_i = k$ and break this for loop. // Find switch point.
    end for
    if $R_i \neq R_{i-1}$ **then**
      Set $\hat{g}_{j,\alpha_i} = \begin{cases} \underline{g}_{j,\alpha_i}, & j \leq R_i \\ \overline{g}_{j,\alpha_i}, & j > R_i \end{cases}$.
      Compute $y'_{\alpha_i} = \frac{\sum_{j=1}^{J} b_j \hat{g}_{j,\alpha_i}}{\sum_{j=1}^{J} \hat{g}_{j,\alpha_i}}$.
    end if
    if $(y'_{\alpha_i} = y_{\alpha_i})$ or $(R_i = R_{i-1})$ **then**
      Stop while loop.
    else
      Set $y_{\alpha_i} = y'_{\alpha_i}$.
    end if
  end while
  Set $\overline{y}_{\alpha_i} = y_{\alpha_i}$.
end for
$\overline{y} = \bigcup_{i=1}^{M} \{\overline{y}_{\alpha_i}\}$.

Return: $\overline{y}$.

知道了作法，也實作了例子，發現根據這個例子，用上個 α 截集 KM 演算法的輸出值當作下個 α 截集 KM 演算法的初始值，的確可以減少運算量。接著，我們來討論這種作法的在一般普遍的情形是不是也依然存在這樣的現象。答案是會的，只要每個模糊規則的啟動強度是一個凸模糊集合(convex fuzzy set)，則使用較大的 α 值的 α 截集 KM 演算法的輸出來當較小的 α 值的 α 截集 KM 演算法的初始值，會比原先所使用的初始值效率要好上許多。底下，我們將要用數學的方式來證明：用較大的 α 值的 α 截集 KM 演算法的輸出來當較小的 α 值的 α 截集 KM 演算法的初始值，的確可以窄化搜尋空間。

引理 1：給定一組資料 $x=\{x_1,x_2,\cdots,x_n\}$，以及一組對應的權重 $\{w_1,w_2,\cdots,w_n\}$，其中 $x_i\in R$ 、 $w_i\geq 0$ 。定義 $c=\dfrac{\sum_{i=1}^{n}x_iw_i}{\sum_{i=1}^{n}w_i}$，換句話說 c 是 x 的加權平均(質心)。試證明 $\sum_{j=1}^{n}(x_j-c)w_j=0$ 。

proof：

根據 c 的定義：$c=\dfrac{\sum_{i=1}^{n}x_iw_i}{\sum_{i=1}^{n}w_i}\Rightarrow c\sum_{i=1}^{n}w_i=\sum_{i=1}^{n}x_iw_i$

所以 $\sum_{j=1}^{n}(x_1-c)w_j=\sum_{j=1}^{n}x_jw_j-\sum_{j=1}^{n}cw_j=c\sum_{j=1}^{n}w_j-\sum_{j=1}^{n}cw_j=0\Rightarrow$ 引理 1 得證。

引理 1 其實只是說明了質心的其中一種物理性質：質心左邊的總力矩會等於質心右邊的總力矩。我們後面將會大量的使用到這樣的觀點和想法來得到我們的結論。

引理 2：給定一組資料 $x=\{x_1,x_2,\cdots,x_n\}$，以及一組對應的權重 $\{w_1,w_2,\cdots,w_n\}$，其

中 $x_i \in R$ 、 $w_i \geq 0$ 。定義 x 的加權平均 $c = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$ ；另外再給定一個 $c' \in R$ 。試證

明：如果 $\sum_{j=1}^{n}(x_j - c')w_j \leq 0$ ，則 $c \leq c'$ 。

proof：

跟據引理 1，我們已經知道 $\sum_{j=1}^{n}(x_j - c)w_j = 0$ ，則

$$\begin{aligned}\sum_{j=1}^{n}(x_j - c')w_j &= \sum_{j=1}^{n}(x_j - c')w_j - \sum_{j=1}^{n}(x_j - c)w_j \\ &= \sum_{j=1}^{n}[(x_j - c') - (x_j - c)]w_j \\ &= \sum_{j=1}^{n}(c - c')w_j \\ &= (c - c')\sum_{j=1}^{n} w_j \leq 0\end{aligned}$$

因為 $w_i \geq 0$ ，所以 $\sum_{j=1}^{n} w_j \geq 0$ 。

所以 $(c - c')\sum_{j=1}^{n} w_j \leq 0 \Rightarrow c - c' \leq 0$ ，引理 2 得證。

引理 2 在說明一個物理意義：如果支點 c' 的左邊力矩大於右邊的力矩，則這個系統的質心 c 的位置在於目前支點 c' 的左邊 $c \leq c'$。接下來我們陳述我們要證明時需使用到的條件與一些客觀的事實。

陳述 1：啟動強度是一個凸模糊集合。

陳述 2：根據陳述 1，給定 $0 < \alpha_2 < \alpha_1 \leq 1$。則對此兩個 α 值的 α 截集 $[\underline{g}_{\alpha_1}, \overline{g}_{\alpha_1}]$ 、

$[\underline{g}_{\alpha_2}, \overline{g}_{\alpha_2}]$ 會得到 $\underline{g}_{\alpha_2} \le \underline{g}_{\alpha_1} \le \overline{g}_{\alpha_1} \le \overline{g}_{\alpha_2}$ 的結果。

陳述 3：令 $\underline{y}_{\alpha_1}$ 是 α 值為 α_1 的 α 截集 KM 演算法的輸出的左端點。則

$$\underline{y}_{\alpha_1} = \frac{\sum_{j=1}^{L} b_j \overline{g}_{j,\alpha_1} + \sum_{j=L+1}^{J} b_j \underline{g}_{j,\alpha_1}}{\sum_{j=1}^{L} \overline{g}_{j,\alpha_1} + \sum_{j=L+1}^{J} \underline{g}_{j,\alpha_1}} \tag{3.1}$$

其中，L 是由 KM 演算法算得，b_j 與 $[\underline{g}_{j,\alpha_1}, \overline{g}_{j,\alpha_1}]$ 為從新排序過的後鑑部 c_j 與啟動強度 $[\underline{f}_{j,\alpha_1}, \overline{f}_{j,\alpha_1}]$。則將引理 1 使用在 3.1 式會得到：

$$\sum_{j=1}^{L} (b_j - \underline{y}_{\alpha_1}) \overline{g}_{j,\alpha_1} + \sum_{j=L+1}^{J} (b_j - \underline{y}_{\alpha_1}) \underline{g}_{j,\alpha_1} = 0 \tag{3.2}$$

陳述 4：對 $\underline{y}_{\alpha_1}$ 與 b_j 而言，我們從 KM 演算法中可以得知 $b_L \le \underline{y}_{\alpha 1} \le b_{L+1}$，所以我們可以推得 $b_i \le \underline{y}_{\alpha 1}$ 當 $i \le L$ 且 $\underline{y}_{\alpha 1} \le b_j$ 當 $j \ge L+1$，因為 b_j 是一個由小到大排序的數列。

我們令 $\underline{y}_{\alpha_2, initial}$ 作為 KM 計算 $\underline{y}_{\alpha_2}$ 的初始值。則

$$\underline{y}_{\alpha_2, initial} = \frac{\sum_{j=1}^{L_{\alpha_1}} b_j \overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J} b_j \underline{g}_{j,\alpha_2}}{\sum_{j=1}^{L_{\alpha_1}} \overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J} \underline{g}_{j,\alpha_2}} \tag{3.3}$$

L_{α_1} 為 $\alpha = \alpha_1$ 時 KM 演算法的計算結果。關於 $\underline{y}_{\alpha_2, initial}$，我們知道 $\underline{y}_{\alpha_2} \le \underline{y}_{\alpha_2, initial}$ 一定成立。因為 $\underline{y}_{\alpha_2}$ 是 $\alpha = \alpha_2$ 時 b_j 與 $[\underline{g}_{j,\alpha_2}, \overline{g}_{j,\alpha_2}]$ 任何組合所算出來質心的左端點，$\underline{y}_{\alpha_2, initial}$ 也是這些質心之一，所以 $\underline{y}_{\alpha_2} \le \underline{y}_{\alpha_2, initial}$ 成立。如果我們可以證明

$\underline{y}_{\alpha_2,initial} \le \underline{y}_{\alpha_1}$，我們就會得到 $\underline{y}_{\alpha_2} \le \underline{y}_{\alpha_2,initial} \le \underline{y}_{\alpha_1}$，如此一來，我們就證明了 $\underline{y}_{\alpha_2}$ 的解空間其實有一個上限，我們不必由原來的初始值整個解空間去做搜尋，而是從 $\underline{y}_{\alpha_2,initial}$ 開始找。所以，使用 $\underline{y}_{\alpha_2,initial}$ 來當初始值來替代原來的初始值，就可以加速整個運算。所以，我們必須要證明 $\underline{y}_{\alpha_2,initial} \le \underline{y}_{\alpha_1}$ 確實成立。很幸運的，這個關係是存在的，只要我們證明了 $\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2} \le 0$，再透過引理 2 就可以得到 $\underline{y}_{\alpha_2,initial} \le \underline{y}_{\alpha_1}$。以下我們就來完成最後的證明，證明 $\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2} \le 0$ 成立。

proof：

由陳述 3，3.2 式得知 $\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_1} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_1} = 0$

$$\begin{aligned}
&\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2} \\
&= \sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2} - \left(\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_1} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_1}\right) \\
&= \sum_{j=1}^{L_{\alpha_1}}\left((b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} - (b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_1}\right) + \sum_{j=L_{\alpha_1}+1}^{J}\left((b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2} - (b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_1}\right) \\
&= \sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})(\overline{g}_{j,\alpha_2} - \overline{g}_{j,\alpha_1}) + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})(\underline{g}_{j,\alpha_2} - \underline{g}_{j,\alpha_1})
\end{aligned} \tag{3.4}$$

由陳述 2 和陳述 4 分別可以得知 $(\overline{g}_{j,\alpha_2} - \overline{g}_{j,\alpha_1}) \ge 0$ 與 $(b_j - \underline{y}_{\alpha_1}) < 0$ 當 $j \le L_{\alpha_1}$

$$\Rightarrow \sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})(\overline{g}_{j,\alpha_2} - \overline{g}_{j,\alpha_1}) \le 0$$

由陳述 2 和陳述 4 分別可以得知 $(\underline{g}_{j,\alpha_2} - \underline{g}_{j,\alpha_1}) \le 0$ 與 $(b_j - \underline{y}_{\alpha_1}) > 0$ 當 $j \ge L_{\alpha_1} + 1$

$$\Rightarrow \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})(\underline{g}_{j,\alpha_2} - \underline{g}_{j,\alpha_1}) \le 0$$

所以，3.4 式可以得到下面結論：

$$\sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})\overline{g}_{j,\alpha_2} + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})\underline{g}_{j,\alpha_2}$$

$$= \sum_{j=1}^{L_{\alpha_1}}(b_j - \underline{y}_{\alpha_1})(\overline{g}_{j,\alpha_2} - \overline{g}_{j,\alpha_1}) + \sum_{j=L_{\alpha_1}+1}^{J}(b_j - \underline{y}_{\alpha_1})(\underline{g}_{j,\alpha_2} - \underline{g}_{j,\alpha_1}) \le 0$$

至此，我們證明了要計算 $\underline{y}_{\alpha_2}$ 透過存在 $\underline{y}_{\alpha_2} \le \underline{y}_{\alpha_2,initial} \le \underline{y}_{\alpha_1}$ 這樣的關係，使用 $\underline{y}_{\alpha_2,initial}$ 來當作 KM 找 $\underline{y}_{\alpha_2}$ 的初始值，就可以加速運算。而要計算 $\overline{y}_{\alpha_2}$ 也是使用 $\overline{y}_{\alpha_2,initial}$ 來做 KM 的初始值，因為 $\overline{y}_{\alpha_1} \le \overline{y}_{\alpha_2,initial} \le \overline{y}_{\alpha_2}$ 的關係亦存在，只要用類似的步驟就可以證明 $\overline{y}_{\alpha_1} \le \overline{y}_{\alpha_2,initial} \le \overline{y}_{\alpha_2}$ 不等式的確成立。

第四章

第二型模糊類神經學習演算法

在這一章中我們將描述第二型模糊類神經網路的架構，主要講述架構鑑別和參數鑑別的學習演算法。

4.1 第二型 TSK 模糊類神經網路

第二型 TSK 模糊類神經網路架構我們可以約略的把它分成四層。如圖 4.1 這四層分別是：第一層是模糊化層、第二層為交集(conjunction)層、第三層為降階層、第四層為輸出層。我們將依序介紹這四層網路。

第一層：這層我們要將輸入 $\{x_1, x_2, \cdots, x_n\}$ 做模糊化。這層一共有 J 個模糊規則，每條模糊規則包含 n 節點(node)，也就是每條 rule 我們都需要把 $\{x_1, x_2, \cdots, x_n\}$ 匯入一次。在第一層中 $Node(i, j)$ 代表的是一個第二型模糊集合 $\tilde{A}_{i,j}$，表示第 j 條模糊規則的第 i 個維度的特徵函數。我們定義第二型模糊集合 $\tilde{A}_{i,j}$ 的第一階歸屬函數 $\mu_{i,j}$：

$$\begin{aligned}\mu_{i,j}(x_i) &= gt1(x_i; m_{i,j}, \sigma 1_{i,j}) \\ &= \exp\left[-\left(\frac{x_i - m_{i,j}}{\sigma 1_{i,j}}\right)^2\right]\end{aligned} \tag{4.1}$$

其中 x_i 為單筆輸入資料的第 i 個維度值，$m_{i,j}$ 和 $\sigma 1_{i,j}$ 為第 j 條模糊規則的第 i 個維度的第一階歸屬函數(高斯函數)的中心點(mean)和標準差(deviation)。如果以 2.3 節的例子 $\mathbf{x} = [-0.85, -0.88]$、$m_{1,1} = -0.650$、$\sigma 1_{1,1} = 0.300$、$m_{2,1} = -0.760$、$\sigma 1_{2,1} = 0.206$ 則：

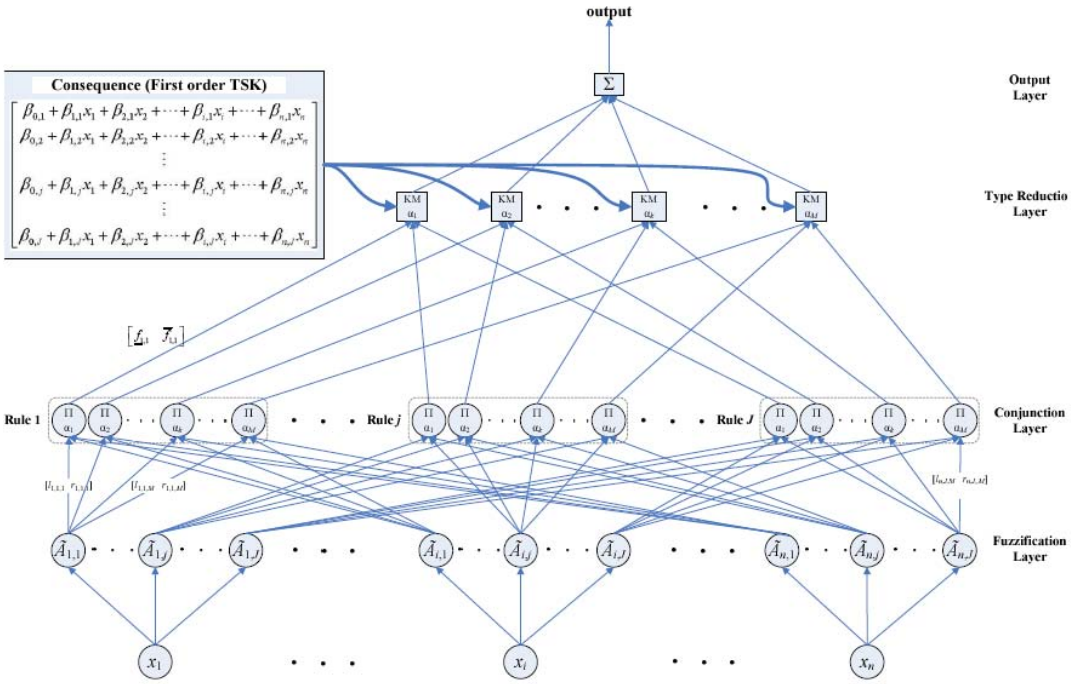

圖 4.1：第二型 TSK 模糊類神經網路架構

$$\mu_{1,1}(-0.85) = \exp\left[-\left(\frac{-0.85-(-0.65)}{0.3}\right)^2\right] = 0.6412$$

$$\mu_{2,1}(-0.88) = \exp\left[-\left(\frac{-0.88-(-0.76)}{0.206}\right)^2\right] = 0.7118$$

算完第一階歸屬函數之後，就可以來計算第二階歸屬函數了。第二階歸屬函數採用的也是高斯函數。事實上，第二階歸屬函數的標準差是可以根據第一階歸屬函數的不同或是輸入的 x_i 不同而不同。但是為了計算量的考量，我們第二階歸屬函數的標準差就採同一數值。我們令之為 $\sigma 2_{i,j}$。所以，第二階歸屬函數 $\mu_{\tilde{A}_{i,j}(x_i)}(u)$ 定義為：

$$\begin{aligned}\mu_{\tilde{A}_{i,j}(x_i)}(u) &= gt2(u;\mu_{i,j}(x_i),\sigma 2_{i,j}) \\ &= \exp\left[-\left(\frac{u-\mu_{i,j}(x_i)}{\sigma 2_{i,j}}\right)^2\right]\end{aligned} \tag{4.2}$$

其中$u \in [0,1]$，是第一階歸屬函數的值域(range)，同時也是第二階歸屬函數的定義域(domain)。$\mu_{i,j}(x_i)$為第 j 條模糊規則第 i 個維度的第一階歸屬函數值，也就是 4.1 式的輸出，這裡把這個數值拿來當第二階歸屬函數(高斯函數)的中心點。$\sigma 2_{i,j}$之前已經介紹過了，是第二階歸屬函數(高斯函數)的標準差。在之前的章節，我們已經有介紹過一個輸入進入第二型模糊集合會得到一個模糊集合的輸出，這個模糊集合我們將之表示為 $\widetilde{A}_{i,j}(x_i)$。這個模糊集合我們可以再用 α 截集將之解構成數個區間：

$$\begin{aligned}\widetilde{A}_{i,j}(x_i) &= \bigcup_{k=1}^{M} {}_{\alpha_k}\widetilde{A}_{i,j}(x_i) \\ &= \bigcup_{k=1}^{M} \alpha_k \cdot {}^{\alpha_k}\widetilde{A}_{i,j}(x_i) \\ &= \bigcup_{k=1}^{M} \alpha_k \cdot [l_{i,j,\alpha_k}, r_{i,j,\alpha_k}]\end{aligned} \tag{4.3}$$

其中 M 為 α 截集所切的份數。$\widetilde{A}_{i,j}(x_i)$這個第一型模糊集合$\alpha = \alpha_k$時取 α 截集會得到一個區間，l_{i,j,α_k}就是這個區間的區間下限、r_{i,j,α_k}就是區間上限。則l_{i,j,α_k}、r_{i,j,α_k}與$m_{i,j}$、$\sigma 1_{i,j}$、$\sigma 2_{i,j}$的關係式為：

$$\begin{aligned}l_{i,j,\alpha_k} &= \min({}^{\alpha_k}\widetilde{A}_{i,j}(x_i)) \\ &= \exp\left[-\left(\frac{x_i - m_{i,j}}{\sigma 1_{i,j}}\right)^2\right] - \sigma 2_{i,j}\sqrt{\ln\frac{1}{\alpha_k}}\end{aligned} \tag{4.4}$$

和

$$\begin{aligned}r_{i,j,\alpha_k} &= \max({}^{\alpha_k}\widetilde{A}_{i,j}(x_i)) \\ &= \exp\left[-\left(\frac{x_i - m_{i,j}}{\sigma 1_{i,j}}\right)^2\right] + \sigma 2_{i,j}\sqrt{\ln\frac{1}{\alpha_k}}\end{aligned} \tag{4.5}$$

所以，第一層的每個節點的輸出定義為 $o_{i,j,\alpha_k}^{(1)}$ ：

$$o_{i,j,\alpha_k}^{(1)} = [l_{i,j,\alpha_k}, r_{i,j,\alpha_k}] \tag{4.6}$$

以 2.3 節剛剛算過的例子 $\sigma 2_{1,1} = 0.0300$ ，我們來算算 $o_{1,1,1}^{(1)}$ 、 $o_{1,1,0.6}^{(1)}$ ：

$$\begin{aligned} o_{1,1,1}^{(1)} &= [l_{1,1,1}, r_{1,1,1}] \\ &= 0.6412 \pm 0.03 \times \sqrt{\ln \frac{1}{1}} \\ &= [0.6412, 0.6412] \\ o_{1,1,0.6}^{(1)} &= [l_{1,1,0.6}, r_{1,1,0.6}] \\ &= 0.6412 \pm 0.03 \times \sqrt{\ln \frac{1}{0.6}} \\ &= [0.6197, 0.6626] \end{aligned}$$

第二層：這層我們要把上層輸出，也就是每條模糊規則裡的各個維度得到的第一型模糊集合做 PRODUCT 運算，做完 PRODUCT 運算就可以得到每條模糊規則的啟動強度。啟動強度的計算方式為：

$$\begin{aligned} f_j &= \bigcup_{k=1}^{M} \alpha_k \cdot \left[\prod_{i=1}^{n} l_{i,j,\alpha_k}, \prod_{i=1}^{n} r_{i,j,\alpha_k} \right] \\ &= \bigcup_{k=1}^{M} \alpha_k \cdot [\underline{f}_{j,\alpha_k}, \overline{f}_{j,\alpha_k}] \end{aligned} \tag{4.7}$$

啟動強度 f_j 是一個第一型模糊集合。透過 4.7 式每個模糊規則的節點減少為 M 個，也就是說，每條模糊規則透過 PRODUCT 運算只剩下一個啟動強度，這個啟動強度我們使用 α 截集將之解構成 M 個區間。所以這層的輸出 $o_{j,\alpha_k}^{(2)}$ 為：

$$\begin{aligned} o_{j,\alpha_k}^{(2)} &= [\underline{f}_{j,\alpha_k}, \overline{f}_{j,\alpha_k}] \\ &= \prod_{i=1}^{n} o_{i,j,\alpha_k}^{(1)} \end{aligned} \tag{4.8}$$

其中 $j=1,2,\cdots,J$ 、 $k=1,2,\cdots,M$ 。我們接續前面的例子：

$$\begin{aligned} o_{1,1}^{(2)} &= [\underline{f}_{1,1}, \overline{f}_{1,1}] \\ &= [0.6412\times 0.7118, 0.6412\times 0.7118] \\ &= [0.4564, 0.4564] \end{aligned}$$

第三層：這層為降階層。只有 M 個節點，也就是 α 截集將啟動強度切成 M 份，我們就把這 M 份區間跟後鑑部做 M 次 KM 演算法來降階。後鑑部的部份，我們使用的是第一階 TSK 模型(first order TSK model)。所以後鑑部的 c_j 表示為：

$$c=\begin{bmatrix} c_1 \\ \vdots \\ c_j \\ \vdots \\ c_J \end{bmatrix} = \begin{bmatrix} \beta_{0,1}+\beta_{1,1}x_1+\cdots+\beta_{i,1}x_i+\cdots+\beta_{n,1}x_n \\ \vdots \\ \beta_{0,j}+\beta_{1,j}x_1+\cdots+\beta_{i,j}x_i+\cdots+\beta_{n,j}x_n \\ \vdots \\ \beta_{0,J}+\beta_{1,J}x_1+\cdots+\beta_{i,J}x_i+\cdots+\beta_{n,J}x_n \end{bmatrix} \tag{4.9}$$

其中 $\beta_{i,j}$ 為後鑑部的參數。由第二、三章知道，將第二層的輸出 $[\underline{f}_{j,\alpha_k}, \overline{f}_{j,\alpha_k}]$ 搭配後鑑部 c_j 再按照 c_j 大小做排序後，可以將 $[\underline{f}_{j,\alpha_k}, \overline{f}_{j,\alpha_k}]$ 、 c_j 轉換成 $[\underline{g}_{j,\alpha_k}, \overline{g}_{j,\alpha_k}]$ 、 b_j 。接著將 $[\underline{g}_{j,\alpha_k}, \overline{g}_{j,\alpha_k}]$ 、 b_j 輸入 KM 演算法就可以完成降階，得到 KM 演算法的輸出 $[\underline{y}_{\alpha_k}, \overline{y}_{\alpha_k}]$ 。所以第三層的輸出 $o_{\alpha_k}^{(3)}$ 是：

$$o_{\alpha_k}^{(3)} = \frac{\underline{y}_{\alpha_k} + \overline{y}_{\alpha_k}}{2} \tag{4.10}$$

接續前面的例子， $o_1^{(3)}$ ：

$$\begin{aligned} o_1^{(3)} &= \frac{-0.2660+(-0.2660)}{2} \\ &= -0.2660 \end{aligned}$$

第四層：第四層為輸出層。我們要將第三層得到的降階結果做解模糊化的運算，這個運算我們採用的是使用 α_k 作為權重，對所得到的模糊集合作加權平均。

$$o^{(4)} = \frac{\sum_{k=1}^{M} o_{\alpha_k}^{(3)} \cdot \alpha_k}{\sum_{k=1}^{M} \alpha_k} \tag{4.11}$$

這邊有一點需要注意，那就是如果輸入的資料是有正規化(normalized)過的資料，則這邊的輸出就需要做正規化還原(denormalized)的動作才可以得到實際的輸出。最後我們也接續前敘的例子算出最後的輸出：

$$\begin{aligned} o^{(4)} &= \frac{1\times(-0.2660)+0.6\times(-0.2420)+0.2\times(-0.2226)}{1+0.6+0.2} \\ &= -0.2532 \end{aligned}$$

4.2 架構鑑別與參數鑑別演算法

這個小節要討論架構鑑別(structure identification)與參數鑑別(parameter identification)演算法。我們使用了自建構式產生法(self-constructing rule generation)[16]、多重線性迴歸(multiple linear regression)[30]，來建立模糊規則，並將結果展開成第二型模糊邏輯系統，並決定系統各初始參數值。我們將建立第一階 TSK 模型的架構鑑別流程放到圖 4.2。

自建構式模糊分群(self-constructing fuzzy clustering)演算法最大的優點就是可以快速，且自動的分群分類。自建構式分群法只要將所有的資料讀入一次就可以將所有的資料分群或分類完畢，而且事先不需要先決定群數。缺點是會受到資料輸入的順序而影響到分群的結果。當一筆新的資料被讀進系統時，會根據輸入和之前資料的相似度與輸出和之前資料輸出的相似度來決定這筆資料是加入先前資料已經形成的群還是自己型成一個新群。基本的概念是這樣，更多的詳細資

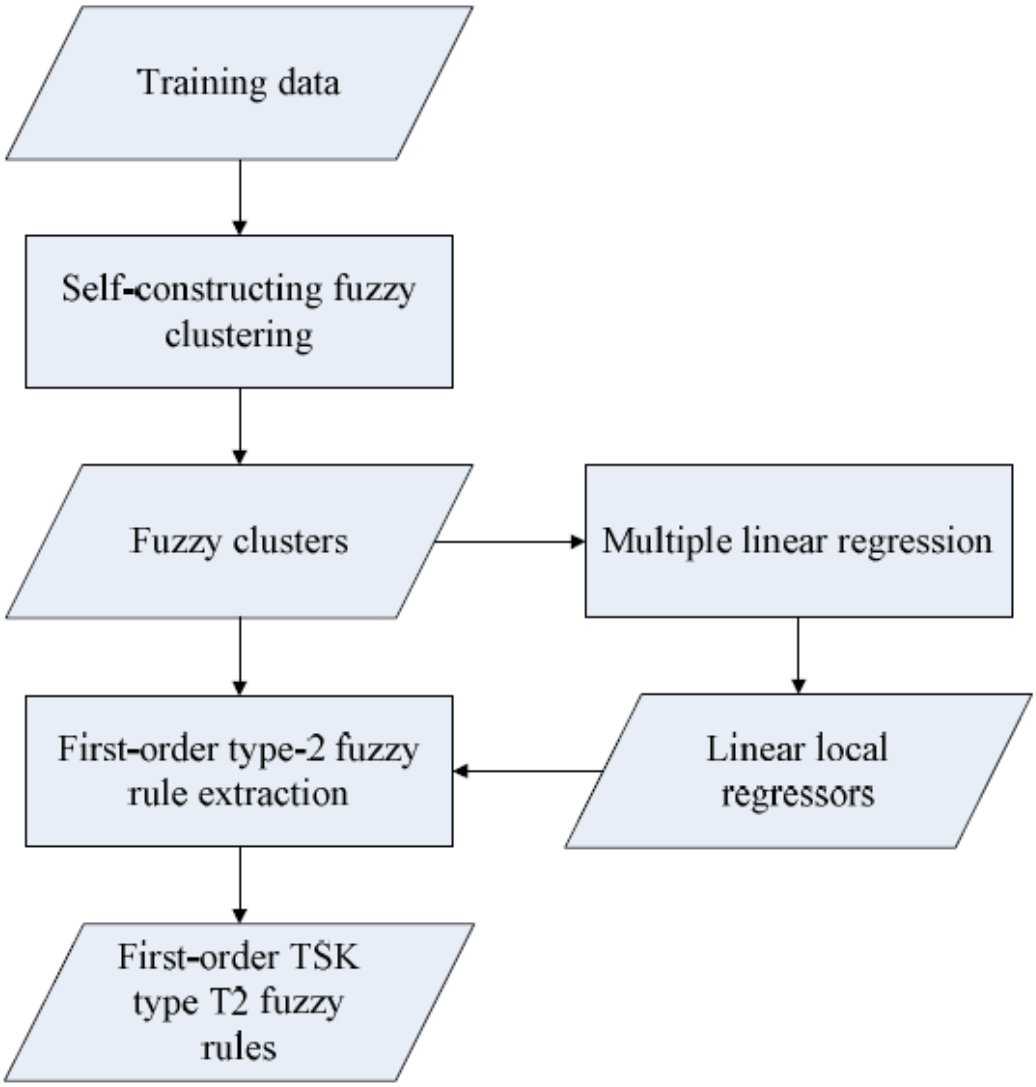

圖 4.2：架構鑑別流程

料可以在[16]裡面找的到。或是想要參考中文的資料，請找本實驗室已畢業的博班學長歐陽振森老師的博士論文，裡面也有相當完整的陳述與證明。假定自建構式模糊分群已經幫我們將資料分成 J 群，模糊邏輯系統中的後鑑部參數，我們會使用多重線性迴歸的方法配合自建構式模糊分群的結果來求得。我們會將每群的輸入與輸出當成是一個線性的關係，用最小平方估計法作一個輸入對輸出的線性迴歸，一共 J 群，所以要做 J 次。

當自建構式模糊分群和多重線性迴歸完成了它們的作業時，我們可以將這 J 群的資料展開成一個第二型模糊邏輯系統(第一階 TSK 模型)。這個 IF-THEN 系

統我們使用自建構式模糊分群的結果來做 IF 條件，也就是前鑑部的參數；使用多重線性迴歸的結果來做 THEN 的推論，也就是後鑑部的參數。第 j 條 IF-THEN 模糊規則表示如下：

$$\begin{aligned} &IF\ x_1 \text{ is } \mu_{\tilde{A}_{1,j}}(x_1) \text{ and} \cdots \text{and } x_i \text{ is } \mu_{\tilde{A}_{i,j}}(x_i) \text{ and} \cdots \text{and } x_n \text{ is } \mu_{\tilde{A}_{n,j}}(x_n) \\ &\text{THEN } y_j \text{ is } \beta_{0,j} + \beta_{1,j}x_1 + \cdots + \beta_{i,j}x_i + \cdots + \beta_{n,j}x_n \end{aligned} \tag{4.12}$$

其中 $\mathrm{x} = [x_1, \cdots, x_i, \cdots, x_n]^T$ 是輸入向量，y_j 是第 j 條模糊規則的輸出。$\beta_j = [\beta_{0,j}, \beta_{1,j}, \cdots, \beta_{i,j}, \cdots, \beta_{n,j}]^T$ 是第 j 條模糊規則後鑑部的參數。$\tilde{A}_{i,j}$ 是一個第二型模糊集合，是第 j 條模糊規則前鑑部的第 i 個特徵。$\tilde{A}_{i,j}$ 的第一階歸屬函數定義如下：

$$\mu_{i,j} = gt1(x_i; m_{i,j}, \sigma 1_{i,j}) \tag{4.13}$$

$m_{i,j}$ 來自於自建構式模糊分群第 j 群第 i 個維度的中心點，$\sigma 1_{i,j}$ 來自於自建構式模糊分群第 j 群第 i 個維度的標準差。$\tilde{A}_{i,j}$ 的第二階歸屬函數定義如下：

$$\mu_{\tilde{A}_{i,j}} = gt2(u; gt1(x_i; m_{i,j}, \sigma 1_{i,j}), \sigma 2_{i,j}) \tag{4.14}$$

其中 $u \in [0,1]$ 是第一階歸屬函數的值域，也同時是第二階歸屬函數的定義域。$\sigma 2_{i,j} = 0.1 \times \sigma 1_{i,j}$。如此一來，所有第二型模糊邏輯第一階 TSK 模型的所有參數都被決定，我們就是這樣子使用自建構式模糊分群和多重線性迴歸的結果來作為第二型模糊邏輯第一階 TSK 模型的初始參數值。這樣子決定參數的作法除了之前所提到的快速、不需事先決定模糊規則數等兩個優點以外，它也使系統再給定初始值的時候就有不錯的準確度，相當等於我們花了一點點時間不但決定了模糊規則數也給了系統一個還不錯的初始值，這樣可以減少後續學習演算法改善系統準確度所需要花的時間。

決定了系統的初始值以後，就要開始做參數鑑別。我們使用了混合式學習演算法(hybrid learning)，就是我們將系統中前鑑部和後鑑部的參數使用不同的學習演算法來做更新。這種方法可以窄化整個參數的解空間[7]，加快得到最佳解(系統的誤差最小)的速度。我們使用了粒子群最佳化(particle swarm optimization, PSO)[13]來最佳化前鑑部的參數；使用了最小平方估測法(least squares estimator)[30]來最佳化後鑑部。我們把參數鑑別的流程放在圖 4.3。關於粒子群最佳化演算法的相關細節，請參考[13]。

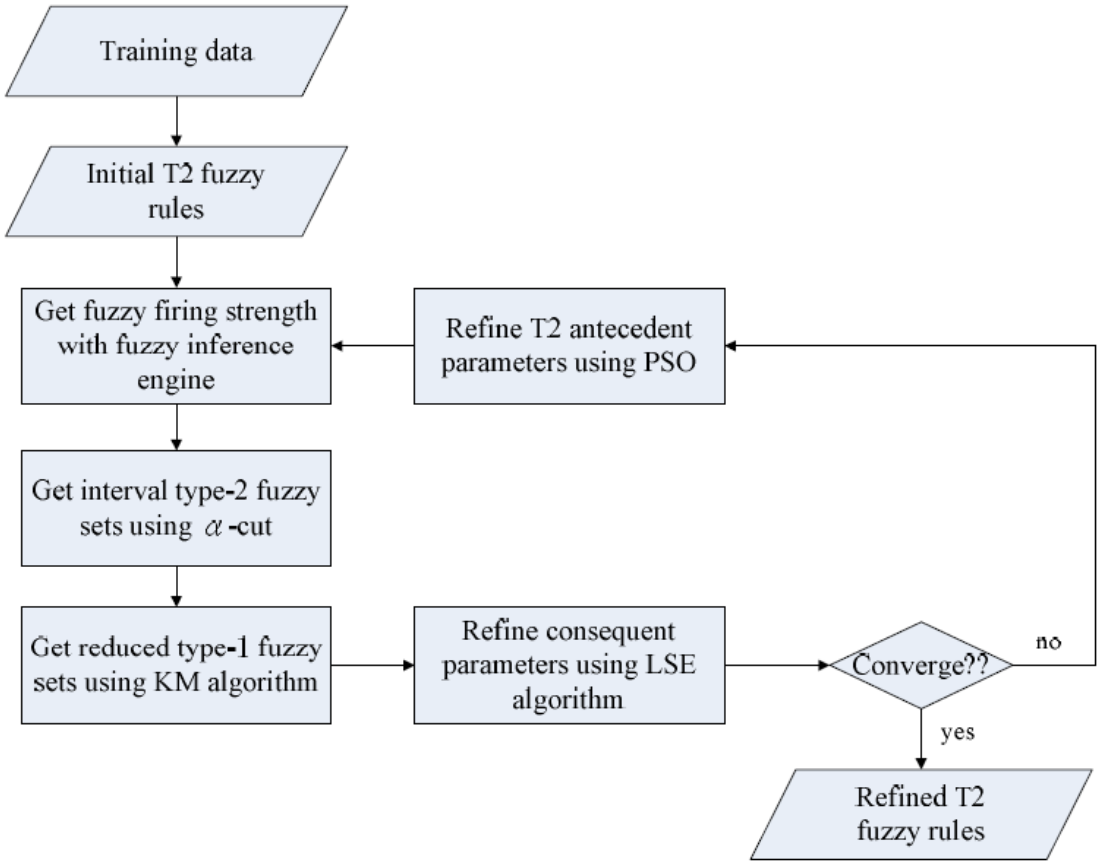

圖 4.3：參數鑑別流程

第五章

範例

這章我們操作一個實際的例子以方便我們了解架構鑑別與參數鑑別。我們假定有一組資料，這組資料有 10 個樣本，輸入維度是 2 維，輸出為 1 維。這組資料陳列如下：

$$X=\begin{bmatrix}\mathrm{x}_1\\ \mathrm{x}_2\\ \mathrm{x}_3\\ \mathrm{x}_4\\ \mathrm{x}_5\\ \mathrm{x}_6\\ \mathrm{x}_7\\ \mathrm{x}_8\\ \mathrm{x}_9\\ \mathrm{x}_{10}\end{bmatrix}=\begin{bmatrix}-0.85 & -0.88\\ -0.65 & -0.72\\ -0.45 & -0.68\\ 0.48 & 0.52\\ 0.32 & 0.63\\ 0.62 & 0.75\\ -0.58 & 0.63\\ -0.73 & 0.68\\ 0.78 & -0.81\\ 0.66 & -0.85\end{bmatrix},y=\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6\\ y_7\\ y_8\\ y_9\\ y_{10}\end{bmatrix}=\begin{bmatrix}-0.2260\\ -0.3250\\ -0.1710\\ 0.2299\\ 0.0940\\ 0.2718\\ 0.3087\\ 0.4499\\ -0.3420\\ -0.1978\end{bmatrix}$$

我們就用這組資料來做架構鑑別，做完接著做參數鑑別。基本的步驟如下：

步驟 1：架構鑑別

我們使用自建構式模糊分群演算法，設定參數 $\rho=0.001$、$\tau=0.12672$、$\sigma_0=[0.1,0.1]$，則我們會得到四群 C_1、C_2、C_3、C_4 我們將之表列於表 5.1。

表 5.1：例子分群列表

	member's	mean	Standard deviation	output
C_1	1、2、3	[-0.6500 -0.7600]	[0.3000 0.2058]	-0.2542
C_2	4、5、6	[0.4733 0.6330]	[0.2501 0.2150]	0.1986
C_3	7、8	[-0.6550 0.6550]	[0.2061 0.1354]	0.3793
C_4	9、10	[0.7200 -0.8300]	[0.1849 0.1283]	-0.2699

得到四個群之後，我們分別對這四群作多重線性迴歸來得到模糊邏輯系統後鑑部的參數。這些參數如下：

$$\begin{bmatrix}\hat{\beta}_1\\ \hat{\beta}_2\\ \hat{\beta}_3\\ \hat{\beta}_4\end{bmatrix}=\begin{bmatrix}-0.8760 & 1.1297 & -1.7845\\ 0.0229 & 0.6873 & -0.2363\\ -0.3078 & 0.8876 & 0.1615\\ 0.1033 & -1.0485 & -0.4600\end{bmatrix}$$

得到了自建構式模糊分群演算法和多重線性迴歸的結果之後，我們就可以利用這些資訊來展開成第二型模糊邏輯系統(第一階 TSK 模型)。四群就分別可以展開成四個模糊規則：

Rule1: $IF\ x_1$ is $gt2(u;gt1(x_1;-0.6500,0.3000),0.03000)$ and
x_2 is $gt2(u;gt1(x_2;-0.7600,0.2058),0.02058)$
THEN y_1 is $-0.8760+1.1297x_1-1.7845x_2$

Rule2: $IF\ x_1$ is $gt2(u;gt1(x_1;0.4733,0.2501),0.02501)$ and
x_2 is $gt2(u;gt1(x_2;0.6333,0.2150),0.02150)$
THEN y_2 is $0.0229+0.6873x_1-0.2363x_2$

Rule3: $IF\ x_1$ is $gt2(u;gt1(x_1;-0.6550,0.2061),0.02061)$ and
x_2 is $gt2(u;gt1(x_2;0.6550,0.1354),0.01354)$
THEN y_3 is $-0.3078-0.8876x_1+0.1615x_2$

Rule4: $IF\ x_1$ is $gt2(u;gt1(x_1;0.7200,0.1849),0.01849)$ and
x_2 is $gt2(u;gt1(x_2;-0.8300,0.1283),0.01283)$
THEN y_1 is $0.1033-1.0485x_1-0.4600x_2$

這裡的u是第一階歸屬函數的值域，同時也是第二階歸屬函數的定義域。

步驟 2：第二型模糊推論

根據剛剛得到的四個模糊規則及其初始參數，我們可以來做模糊推論。我們設定 α 截集的 α 分別為 1、0.6、0.2 三層。則按照第三章第四章的方法來做模糊推論，得到的系統輸出為：

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \\ \hat{y}_5 \\ \hat{y}_6 \\ \hat{y}_7 \\ \hat{y}_8 \\ \hat{y}_9 \\ \hat{y}_{10} \end{bmatrix} = \begin{bmatrix} -0.2352 \\ -0.3216 \\ -0.1688 \\ 0.2273 \\ 0.0903 \\ 0.2625 \\ 0.2910 \\ 0.4294 \\ -0.3297 \\ -0.4868 \end{bmatrix}$$

其 RMSE(root mean square error)為 0.0114。

步驟 3：參數鑑別

根據步驟 1 得到的前鑑部與後鑑部的參數，我們進行混合式學習法來找尋最佳參數。我們會使用粒子群最佳化來最佳化前鑑部的參數；使用了最小平方估測法來最佳化後鑑部。做完之後會得到下面的結果：

Rule1: $IF\ x_1$ is $gt2(u; gt1(x_1; -0.6138, 0.2780), 0.0863)$ and
x_2 is $gt2(u; gt1(x_2; -0.9131, 0.3529), 0.0110)$
THEN y_1 is $-0.5936 + 0.6448x_1 - 0.7770x_2$

Rule2: $IF\ x_1$ is $gt2(u; gt1(x_1; 0.5940, 0.3623), 0.0991)$ and
x_2 is $gt2(u; gt1(x_2; 0.5220, 0.3447), 0.0019)$
THEN y_2 is $-0.0142 + 0.6905x_1 - 0.1512x_2$

Rule3: $IF\ x_1$ is $gt2(u; gt1(x_1; -0.4480, 0.3068), 0.0447)$ and
x_2 is $gt2(u; gt1(x_2; 0.7551, 0.2337), 0.0389)$
THEN y_3 is $-0.3506 - 1.1075x_1 + 0.0829x_2$

Rule4: $IF\ x_1$ is $gt2(u; gt1(x_1; 0.6436, 0.8911), 0.0103)$ and
x_2 is $gt2(u; gt1(x_2; -0.8950, 0.1953), 0.0209)$
THEN y_1 is $0.3754 - 1.1775x_1 - 0.2229x_2$

這裡的 u 是第一階歸屬函數的值域，同時也是第二階歸屬函數的定義域。重複步驟 2，將最後的參數拿來做模糊推論得到的輸出為：

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \\ \hat{y}_5 \\ \hat{y}_6 \\ \hat{y}_7 \\ \hat{y}_8 \\ \hat{y}_9 \\ \hat{y}_{10} \end{bmatrix} = \begin{bmatrix} -0.22600 \\ -0.32500 \\ -0.17099 \\ 0.22990 \\ 0.09400 \\ 0.27180 \\ 0.30883 \\ 0.45008 \\ -0.34204 \\ -0.19784 \end{bmatrix}$$

其 RMSE 為 0.000072。所以，參數鑑定後，有提升系統的準確度。

第六章

實驗結果

為了印證系統的效能，我們用了兩組非線性的模擬資料，和三組實際資料。實際資料來自於 UCI 的機器學習資料庫(Machine Learning Databases)。第一組實驗我們比較了原本降階演算法和加速的降階演算法的效率。第二組實驗，我們實驗了第二型模糊推論的確有較高的錯誤容忍度。第三組實驗，我們拿實際的資料來看看高維度的資料實際運作的情況。PSO 所使用的參數是：粒子數為 5、最大的疊代(iteration)數為 30、PSO 的參數 $w=0.5$ 、 $c_1=1.5$ 、 $c_2=1.5$ 。

6.1 實驗 1

給定一個函數：

$$y=1.1\times(1-x+2x^2)\times e^{\frac{-x^2}{2}}, x\in[-5,5] \tag{6.1}$$

根據這個函數，我們取出我們第一組訓練樣本 DS-I。在 $[-5,5]$ 的區間取 250 組 (x,y) 數對，把 x 作為輸入，y 作為輸出。250 組樣本隨機取 200 組當做訓練樣本，剩下的 50 組就是測試樣本。訓練樣本與測試樣本沒有任何樣本是相同的，也就是說訓練樣本與測試樣本的交集是空集合。接著，固定自建構式模糊分群演算法的參數 $\rho=0.01$ 、 $\sigma_0=0.1$ ，我們調整自建構式模糊分群演算法第三個參數 τ 。當 $\tau=0.2,0.1,0.05,0.01,0.005$ 時，自建構式模糊分群演算法會將訓練樣本自動分類為 6、12、22、77、113 群。分群完畢後，就可以把它展開成第二型模糊推論的系統。在模糊推論的過程中，降階演算法的 α 截集的個數我們定為 $M=5$ ，所以我們設定 $\alpha\in\{1,0.8,0.6,0.4,0.2\}$ 。估測訓練樣本與測試樣本的準確度都使用最小平方誤差法(RMSE)來評量系統的準確度。 J 表示模糊法則的總數目，也就是系統總

共有幾條模糊法則，J 來自於自建構式模糊分群演算法的群數，$J \in \{6,12,22,77,113\}$。K 表示在降階、參數鑑別的過程中 KM 疊代的總次數。從表 6.1 可以觀察到：在系統準確度相同的情況下，將第二型模糊推論降階加速後，KM 演算法執行的總疊代數減少了許多，因為使用了較好的 KM 演算法起始值的緣故，減少了將近有 50%的計算量。表 6.2 我們比較了在模糊規則數不同時，參數鑑別的過程中，我們對降階演算法所做的加速有什麼影響。表 6.2 一樣是 $M = 5$、$\alpha \in \{1,0.8,0.6,0.4,0.2\}$ 的情況，$J \in \{6,12,22,77,113\}$。從表 6.2 可以觀察到，增加模糊規則數會減少 RMSE，也就是增加模糊規則數會增加系統的準確度。另外一個現象就是，當模糊規則數上升的時候，因為我們對降階演算法加速的關係，KM 演算法所計算的總疊代數增加的數目不多。

表 6.1：DS-I 做單次模糊推論時 KM 疊代次數比較表

	Liu's					
	Training			Testing		
J	Error	Time	K	Error	Time	K
6	0.1358	0.1656	4440	0.1392	0.0440	1130
12	0.1023	0.2073	4760	0.1095	0.0537	1190
22	0.1073	0.2696	4760	0.1142	0.0695	1190
77	0.1449	0.6722	4380	0.1471	0.1733	1110
113	0.1177	0.9310	4420	0.1222	0.2308	1120
	Ours					
	Training			Testing		
J	Error	Time	K	Error	Time	K
6	0.1358	0.1199	2211	0.1392	0.0291	550
12	0.1023	0.1278	2211	0.1095	0.0316	550
22	0.1073	0.1434	2211	0.1142	0.0352	550
77	0.1449	0.2432	2211	0.1471	0.0595	550
113	0.1177	0.3088	2211	0.1222	0.0779	550

表 6.2：DS-I 做參數鑑別時 KM 疊代次數比較表(J 變動)

	Liu's				
	Error		Refining time		
J	Training	Testing	Total	KM part	K
6	0.0013	0.0013	72.7031	51.2188	1806310
12	0.0016	0.0015	94.4687	67.6875	2096566
22	0.0012	0.0011	119.8281	85.7188	2143409
77	0.0002	0.0003	288.6719	198.0156	2300959
113	0.0002	0.0003	405.1250	272.1875	2423196
	Ours				
	Error		Refining time		
J	Training	Testing	Total	KM part	K
6	0.0015	0.0014	36.2188	26.4531	740845
12	0.0011	0.0012	42.8594	28.1719	812061
22	0.0011	0.0010	54.3750	32.6406	889174
77	0.0001	0.0002	120.2500	44.7656	1039848
113	0.0001	0.0001	167.6250	56.2031	1073908

表 6.2：DS-I 做參數鑑別時 KM 疊代次數比較表(M 變動)

	Liu's				
	Error		Refining time		
M	Training	Testing	Total	KM part	K
5	0.0013	0.0013	72.7031	51.2188	1806310
10	0.0010	0.0010	142.5156	107.6094	3725516
25	0.0016	0.0016	353.4688	278.3906	10209967
50	0.0013	0.0013	675.6094	533.6875	19632223
100	0.0015	0.0013	1301.0938	1038.1563	41370908
	Ours				
	Error		Refining time		
M	Training	Testing	Total	KM part	K
5	0.0015	0.0014	36.2188	26.4531	740845
10	0.0011	0.0011	58.0625	48.1875	1352823
25	0.0015	0.0016	113.5313	103.2188	3182214
50	0.0012	0.0011	222.7813	210.1094	6176023
100	0.0014	0.0014	440.8594	421.9531	12253609

表 6.3 我們比較了 $J=6$，但是 M 變動時，在參數鑑別時，降階演算法改善的效果。我們分別調整 $M \in \{5,10,25,50,100\}$，我們也可以從表 6.3 明顯發現 M 比較大的時候，要使用加速的降階演算法是比較好的。從表 6.3 也可以看出：增加 M ，

也就是增加 α 截集所要切的份量到某特定程度之後就不會再增加準確度了;相同的結論也出現在[20]。

6.2 實驗 2

這階段的實驗我們要驗證第二型模糊類神經網路的確具有較佳的錯誤容忍度。我們將輸出 y 保持不變，但輸入值我們加入中心點為零，標準差 0.1 的常態分布的雜訊。我們隨機將 DS-I 的訓練樣本隨機挑出 20%、30%、40%的點加入雜訊後形成新的 DS-II、DS-III、DS-IV 樣本。將 DS-I、DS-II、DS-III、DS-IV 輸入自建構式模糊分群演算法後，自建構式模糊分群演算法可以自動決定分群數：DS-I、DS-III 為 6 群，DS-II、DS-IV 為 5 群。自建構式模糊分群演算法的群數可以展開成為第二型模糊邏輯的規則數。我們將實驗結果整理於表 6.4 與圖 6.1，除了第二型模糊推論以外，我們也把第一型模糊推論與區間第二型模糊推論也加入比較。首先，我們可以發現我們的方法比起原本的降階法運算更省時，但卻不會影響系統的準確度。由表 6.4 或圖 6.1 也可以觀察到第二型模糊推論的確對雜訊比較不敏感，在雜訊比較高的環境，表對相對穩定

我們給定另外一個函數：

$$y = x_1^2 \times \sin(x_2\pi), x_1 \in [-1,1], x_2 \in [-1,1] \tag{6.2}$$

我們根據這個函數取 225 組 (x_1, x_2, y) 數對來作為 DS-V 的訓練樣本，(x_1, x_2) 為輸入，y 為輸出。再另外取 50 組作為測試樣本。這組資料訓練樣本就包含了測試樣本，也就是訓練樣本與測試樣本的交集為測試樣本。我們也在 DS-V 的訓練樣本中挑出 20%、40%加上中心點為零，標準差為 0.2 常態分佈的雜訊，成了 DS-VI、DS-VII。自建構式模糊分群演算法的參數為：$\rho = 0.0001$、$\tau = 0.4$、$\sigma_0 = [0.1, 0.1]^T$。我們將實驗的結果整理於表 6.5 和圖 6.2。我們依然可以發現：我們的加速法相對於原本的降階法會比較快速，且不會影響準確度；另外第二型模糊推論具有較好的錯誤容忍率。

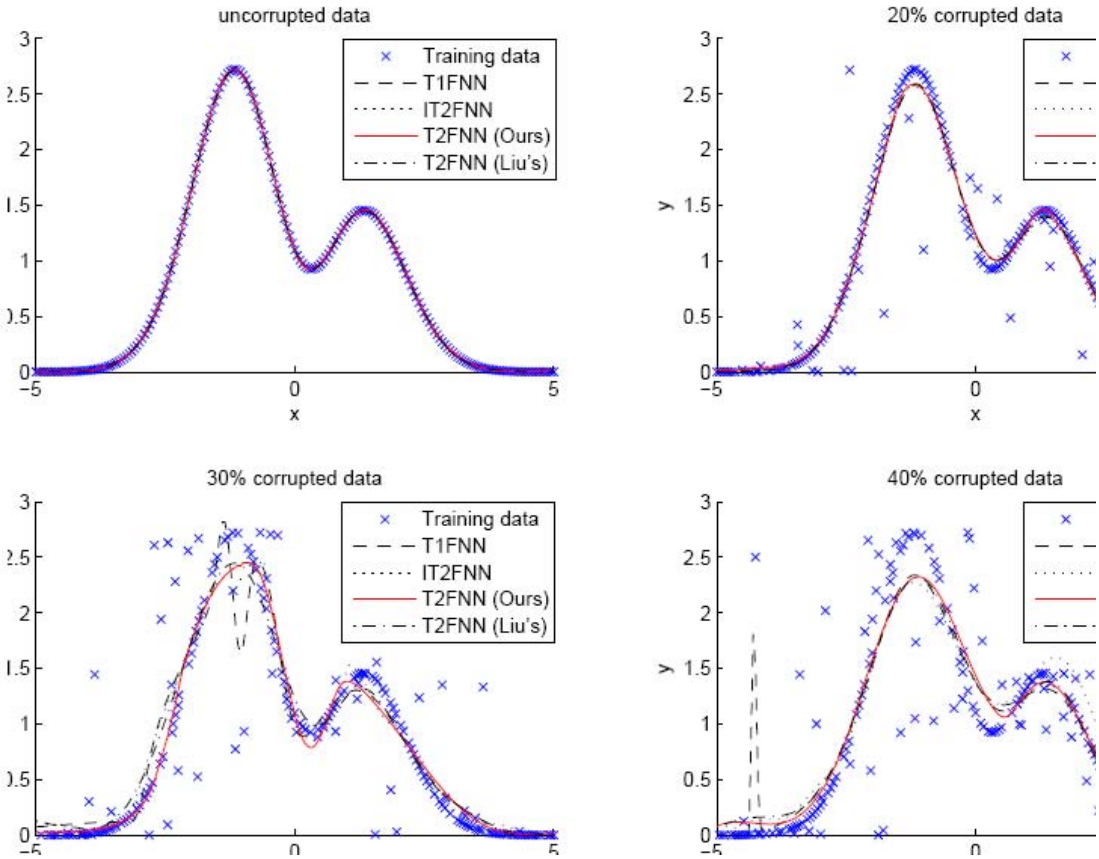

圖 6.1：實驗 2 四組資料的模擬結果

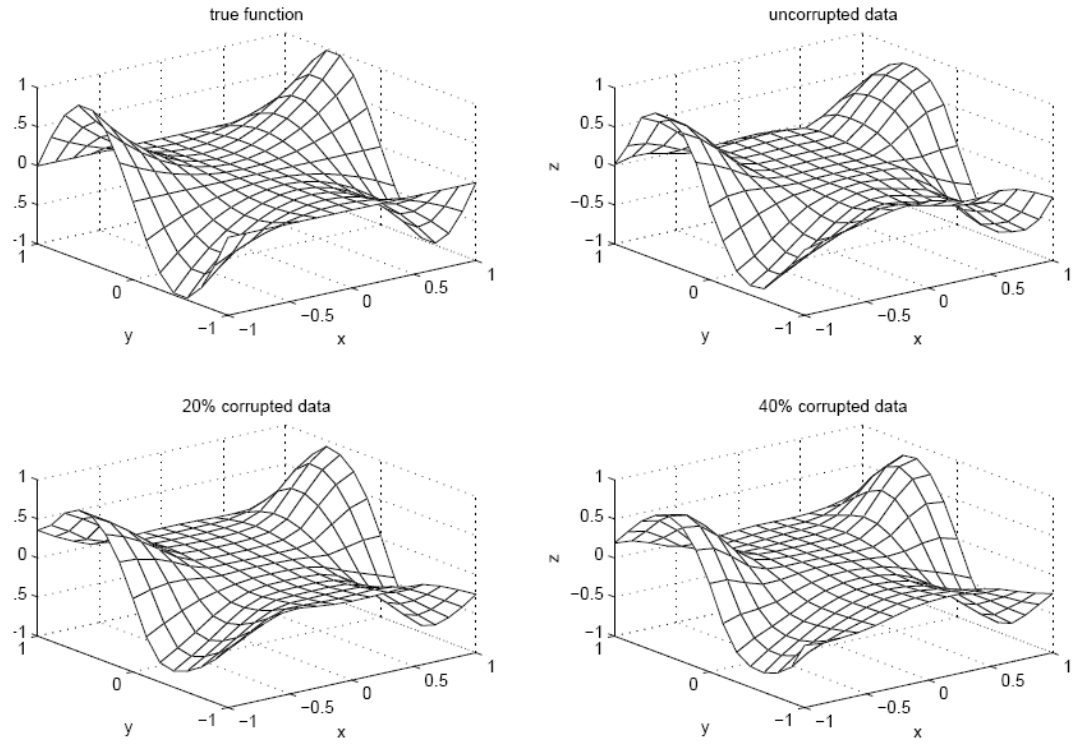

圖 6.2：實驗 2 三組資料的模擬結果

表 6.4：實驗 2 四組資料的模擬結果

Dataset	DS-I (uncorrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	6	0.0022	0.0022	5.8575
IT2FNN	6	0.0022	0.0022	11.7730
T2FNN (Liu's)	6	0.0013	0.0013	72.7031
T2FNN (Ours)	6	0.0015	0.0014	36.2188
Dataset	DS-II (20% corrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	5	0.0901	0.0230	5.2280
IT2FNN	5	0.0922	0.0228	10.8268
T2FNN (Liu's)	5	0.0901	0.0229	69.2969
T2FNN (Ours)	5	0.0899	0.0227	38.1875
Dataset	DS-III (30% corrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	6	0.1372	0.0872	5.7922
IT2FNN	6	0.1412	0.0598	11.6966
T2FNN (Liu's)	6	0.1446	0.0470	73.9219
T2FNN (Ours)	6	0.1406	0.0435	39.5000
Dataset	DS-IV (40% corrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	5	0.1470	0.1112	5.2121
IT2FNN	5	0.1616	0.0760	11.0156
T2FNN (Liu's)	5	0.1558	0.0625	70.6719
T2FNN (Ours)	5	0.1559	0.0611	37.6875

表 6.4：實驗 2 三組資料的模擬結果

Dataset	DS-V (uncorrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	6	0.0280	0.0280	6.5618
IT2FNN	6	0.0178	0.0186	12.6257
T2FNN (Liu's)	6	0.0199	0.0176	85.9844
T2FNN (Ours)	6	0.0193	0.0179	46.8594
Dataset	DS-VI (20% corrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	5	0.0492	0.0403	6.0877
IT2FNN	5	0.0506	0.0379	11.7611
T2FNN (Liu's)	5	0.0475	0.0293	79.5313
T2FNN (Ours)	5	0.0427	0.0298	44.7969
Dataset	DS-VII (40% corrupted data)			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	7	0.0685	0.0384	7.0394
IT2FNN	7	0.0689	0.0380	13.6490
T2FNN (Liu's)	7	0.0684	0.0359	90.1406
T2FNN (Ours)	7	0.0684	0.0350	48.3594

6.3 實驗 3

這階段我們取了三組實際的資料來跑實驗。資料的來源來自於 UCI 的機器學習資料庫，一共有三個 HOUSING、PYRIM、MPG。

HOUSING 這組資料是一組 13 個維度的輸入對應到一個維度的輸出。那 13 個維度是描述房子的狀態，而輸出就是房子的售價。HOUSING 一共有 506 筆資料，我們隨機取了 465 筆來當作訓練樣本，剩下的當測試樣本。自建構式模糊分群演算法的參數為：$\rho = 0.1^{13}$ 、$\tau = 0.3$ 、$\sigma_0 = [0.2, 0.2, \cdots, 0.2]^T$ 。實驗結果整理於表 6.6。我們在取訓練樣本與測試樣本時，是隨機取。為了降低離群值的影響，我們將原樣本重複取了十次，共取了十組訓練樣本與測試樣本。所以實驗也做了

十次，表 6.6 的結果是十次實驗的平均。由於建構式模糊分群演算法每次決定的模糊規則數並不相同，所以表 6.6 的模糊規則數並不是一個整數，那是作十次實驗，十次的模糊規則數的平均值。

PYRIM 這組資料是一組 26 個維度的輸入對應到一個維度的輸出。PYRIM 一共有 74 筆資料，我們隨機取了 67 筆來當作訓練樣本，剩下的當測試樣本。自建構式模糊分群演算法的參數為：$\rho = 0.1^{26}$ 、$\tau = 0.5$ 、$\sigma_0 = [0.3, 0.3, \cdots, 0.3]^T$ 。實驗結果整理於表 6.6。

MPG 這組資料是一組 7 個維度的輸入對應到一個維度的輸出。MPG 一共有 392 筆資料，我們隨機取了 353 筆來當作訓練樣本，剩下的當測試樣本。自建構式模糊分群演算法的參數為：$\rho = 0.1^{7}$ 、$\tau = 0.5$ 、$\sigma_0 = [0.2, 0.2, \cdots, 0.2]^T$ 。實驗結果整理於表 6.6。

表 6.6：實際資料實驗結果

Dataset	HOUSING			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	6.10	0.0921	0.1085	13.0199
IT2FNN	6.10	0.1076	0.1034	30.8832
T2FNN (Liu's)	6.10	0.1181	0.1015	214.1828
T2FNN (Ours)	6.10	0.1128	0.1016	137.2266
Dataset	PYRIM			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	2.00	0.0325	0.0632	2.8250
IT2FNN	2.00	0.0297	0.0685	4.6654
T2FNN (Liu's)	2.00	0.0146	0.0678	22.0703
T2FNN (Ours)	2.00	0.0163	0.0633	15.5625
Dataset	MPG			
Methods	Number of rules	Training error	Testing error	Refining time
T1FNN	2.70	0.1129	0.0955	6.2992
IT2FNN	2.70	0.1157	0.0920	14.4773
T2FNN (Liu's)	2.70	0.1168	0.0869	104.0984
T2FNN (Ours)	2.70	0.1149	0.0867	64.0593

第七章

結論與未來展望

在本論文中，我們將第二型模糊邏輯做了一個完整的介紹。包含了第二型模糊集合、第二型模糊推論，第二型模糊推論的降階。這個過程中，我們使用了 α 截集的概念來幫助了解第二型模糊的種種。其中對第二型模糊推論降階的加速是本論文最主要的貢獻。使用我們的降階演算法在大多數的情況下，只需要原來降階演算法三分之一的時間就可以完成降階。如果要計算整個模糊推論的時間，則使用我們的降階演算法，在大部分的情況下可以省下總時間 50%左右。而我們的方法加速了降階運算，但卻不會影響系統的準確度，系統的效能還是跟原本的一樣好。在實驗的過程中，發現了第二型模糊推論比起第一型模糊推論來，第二型模糊推論具有較佳的錯誤容忍度。但這個部份尚未有強力的證明，有待後續的研究來補強這部份的論點。本論文最後也提供了一個第二型模糊邏輯系統的架構鑑別與參數鑑別的方法。我們使用自建構式模糊分群演算法來作架構鑑別。這個方法的優點是：它只要將資料從頭到尾讀過一次，就可以將資料分類完畢，而且不需要事先決定群數，它會在運算的過程中自己決定。所以自建構式模糊分群演算法的優點就是快速、自動決定模糊規則數、不錯的準確度。利用自建構式模糊分群演算法的結果我們可以將之展開成一個第二型模糊邏輯系統，接著我們使用混合式學習演算法來做參數鑑別：用 PSO 來最佳化前鑑部的參數，使用 LSE 來最佳化後鑑部的參數。由於這樣可以窄化參數搜尋的解空間，且 PSO 和 LSE 也可以各自發揮自己的優勢。換句話說，在參數鑑別時，PSO 加 LSE 算是個稱的上快速且準確的方式。也許會有更好的方式，但 PSO 加 LSE 拿來做參數的鑑別已經是個相當不錯的選擇了。

參考文獻

[1] O. Castillo and P. Melin, "Comparison of hybrid intelligent systems, neural networks, and interval type-2 fuzzy logic for time series prediction," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 3086–3091, August 2007.

[2] J. R. Castro, O. Castillo, P. Melin, A. Rodríguez-Díaz, and L. G. Martinez, "Intelligent control using an interval type-2 fuzzy neural network with a hybrid learning algorithm," in *Proceedings of the International Conference on Fuzzy Systems*, pp. 893–900, June 2008.

[3] S. Coupland and R. John, "A fast geometric method for defuzzification of type-2 fuzzy sets," *IEEE Transaction on Fuzzy Systems*, vol. 16, no. 4, pp. 929–941, August 2008.

[4] M. T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*. THOMSON, 1996.

[5] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Transaction on Fuzzy Systems*, vol. 12, no. 4, pp. 524–539, August 2004.

[6] H. Hagras, "Comments on dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 5, pp. 1206–1209, October 2006.

[7] J.-S. R. Jang, C.-T. Sun, "Neuro-Fuzzy Modeling and Control" *Proceedings of the IEEE*, vol. 83, no. 3, March 1995.

[8] J.-S. R. Jang, C.-T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Pearson Education Taiwan Ltd., 2004.

[9] R. I. John, P. R. Innocent, and M. R. Barnes, "Neuro-fuzzy clustering of radiographic tibia image data using type-2 fuzzy sets," *Information Sciences*, vol. 125, no. 1-4, pp. 65–82, June 2000.

[10] C.-F. Juang and Y.-W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Transaction on Fuzzy Systems*, vol. 16, no. 6, pp. 1411–1424, December 2008.

[11] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, no. 1-4, pp. 195–220, February 2001.

[12] V. Kecman, *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. The MIT Press, March 2001.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[14] G. J. Klir and B. Yuan, *Fuzzy Set and Fuzzy logic*. Prentice Hall PTR, May 1995.

[15] L. D. Lascio, A. Gisolfi, and A. Nappi, "Medical differential diagnosis through type-2 fuzzy sets," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 371–376, May 2005.

[16] S. J. Lee and C. S. Ouyang, "A neuro-fuzzy system modeling with self-constructing rule generation and hybrid SVD-based learning," *IEEE Transaction on Fuzzy Systems*, vol. 11, no. 3, pp. 341–353, June 2003.

[17] C.-H. Lee, T.-W. Hu, C.-T. Lee, and Y. chia Lee, "A recurrent interval type-2 fuzzy neural network with asymmetric membership functions for nonlinear system identification," in *Proceedings of the International Conference on Fuzzy Systems*, pp. 1496–1502, June 2008.

[18] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Transaction on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, October 2000.

[19] F.-J. Lin and P.-H. Chou, "Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network," *IEEE Transaction on Industrial Electronics*, vol. 56, no. 1, pp. 178–193, January 2009.

[20] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 179, no. 9, pp. 2224–2236, April 2008.

[21] L. A. Lucas, T. M. Centeno, and M. R. Delgado, "General type-2 fuzzy inference systems: Analysis, design and computational aspects," in *Proceedings of the International Conference on Fuzzy Systems*, pp. 1–6, July 2007.

[22] L. A. Lucas, T. M. Centeno, and M. R. Delgado, "Land cover classification based on general type-2 fuzzy classifiers," *International Journal of Fuzzy Systems*, vol. 10, no. 3, pp. 207–216, September 2008.

[23] J. M. Mendel, *UNCERTAIN Rule-Based Fuzzy Logic Systems*. Prentice Hall PTR, January 2001.

[24] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic system," *IEEE Transaction on Fuzzy Systems*, vol. 12, no. 1, pp. 84–98, February 2004.

[25] J. M. Mendel, F. Liu, "Super-Exponential Convergence of The Karnik-Mendel Algorithms Used for Type-Reduction in Interval Type-2 Fuzzy Logic Systems", *IEEE International Conference on Fuzzy Systems,* pp. 1253-1260, January 2006.

[26] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," *Information Sciences*, vol. 177, no. 1, pp. 84–110, January 2007.

[27] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 20–29, February 2007.

[28] J. M. Mendel and F. Liu, "Super-exponential convergence of the karnik-mendel algorithms for computing the centroid of an interval type-2 fuzzy set," *IEEE Transaction on Fuzzy Systems*, vol. 15, no. 2, pp. 309–320, April 2007.

[29] H. B. Mitchell, "Pattern recognition using type-II fuzzy sets," *Information*

Sciences, vol. 170, no. 2-4, pp. 409–418, February 2005.

[30] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. Wiley-Interscience, April 2001.

[31] R. Sepúlveda, O. Castillo, P. Melin, A. Rodríguez-Díaz, and O. Montiel, "Experimental study of intelligent controllers under uncertainty using type-1 and type-2 fuzzy logic," *Information Sciences*, vol. 177, no. 10, pp. 2023–2048, May 2007.

[32] C.-H. Wang, C.-S. Cheng, and T.-T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 3, pp. 1462–1477, June 2004.

[33] D. Wu, J. M. Mendel, "Enhanced Karnik-Mendel Algorithms for Interval Type-2 Fuzzy Sets and Systems," *Proceedings of the NAFIPS*, pp. 184-189, June 2007.

[34] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-1," *Information Sciences*, vol. 8, pp. 199–249, January 1975.

[35] M. H. F. Zarandi, B. Rezaee, I. B. Turksen, and E. Neshat, "A type-2 fuzzy rulebased expert system model for stock price analysis," *Expert Systems with Applications*, vol. 36, no. 1, pp. 139–154, January 2009.

[36] J. Zeng and Z.-Q. Liu, "Type-2 fuzzy markov random fields and their application to handwritten chinese character recognition," *IEEE Transaction on Fuzzy Systems*, vol. 16, no. 3, pp. 747–760, June 2008.