

CTRL+ALT+DEL YOUR PROBLEMS WITH CODE

掌握代碼的藝術

The code is life. Never mess with the code.

張智硯

目 錄

序	vi
1 L<small>A</small>T<small>E</small>X 自學筆記	1
1.1 L<small>A</small>T<small>E</small>X 簡介	1
1.1.1 版本發展	1
1.1.2 為何要使用 L <small>A</small> T <small>E</small> X	2
1.1.3 學習 L <small>A</small> T <small>E</small> X 的免費資源	2
1.2 文本設定	4
1.2.1 首頁	4
1.2.2 章節	5
1.2.3 目錄	6
1.2.4 換行分頁	7
1.2.5 插入編號	8
1.2.6 字體	12
1.3 數學公式	16
1.4 表格	18
1.5 結論	21
2 函數計算與繪製	23
2.1 函數繪圖	23
2.2 Project : Comparison Theorem	41
2.3 結論	45
3 分配函數、亂數、抽樣分配的程式與繪圖	47
3.1 分配函數	47
3.1.1 連續型機率分配	47
3.1.2 間斷型機率分配	51
3.2 隨機亂數與相關圖形	53
3.3 抽樣分配實作	57
3.3.1 中央極限定理 CLT 應用於 Beta 分配	57
3.3.2 極小值幾何分配	58

3.4 專題—數字抽取	59
4 機器學習中的迴歸模型與公式原理	61
4.1 復歸 (Regression)	61
4.2 簡單線性迴歸 (Simple Linear Regression)	62
4.2.1 實作—以簡單線性迴歸進行二元分類	63
4.3 加廣型迴歸模型 (Augmented Regression Model)	66
4.3.1 實作—以加廣型迴歸進行二元分類	67
4.4 邏輯回歸 (Logistic Regression)	71
4.4.1 實作—以邏輯斯迴歸模型進行三元分類	72
4.5 結語	73
5 監督式學習：判別分析與 KNN 分類	75
5.1 線性判別式分析 (LDA)	75
5.2 二次判別式分析 (QDA)	77
5.3 K-最近鄰居法 (KNN)	78
5.4 LDA, QDA 與 KNN 學習器評比	81
5.4.1 相同共變異矩陣的比較	81
5.4.2 相異共變異矩陣的比較	84
5.4.3 環形資料集的比較	88
5.5 結論	89
6 淺度機器學習：類神經網路	91
6.1 前饋式 (Feedforward) 類神經網路的原理	91
6.2 前饋式類神經網路的實做與評比	93
6.2.1 兩截式機械手臂 (2-Joint Robot Arm)	93
6.2.2 圖形辨識	100
6.3 結語	102
7 蒙地卡羅模擬實驗：學習器的評比	103
7.1 學習器的回顧	103
7.2 分析學習器在不同情境下的性能	104
7.2.1 具有相同共變異矩陣的資料	104
7.2.2 具有相異共變異矩陣的資料	106
7.3 結論	108

圖 目 錄

1.1	首頁	5
1.2	章節	6
1.3	目錄	7
1.4	itemize	9
1.5	itemize 變換標示	10
1.6	enumerate	10
1.7	enumerate 變換編號格式	12
1.8	修改中文字體	15
1.9	不同自由度底下的 T 分配圖型	16
2.1	$f(x) = \sin(x) + \cos(x)$	23
2.2	$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	25
2.3	$f(x) = \sqrt[3]{\frac{4-x^3}{1+x^2}}$	27
2.4	$f(x) = \frac{1}{x}$	29
2.5	$f(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-1)^2}{8}}$	31
2.6	$f(x) = \sqrt[3]{x^2}$	32
2.7	$f(x) = 2x^3 - x^4$	34
2.8	$f(x) = \frac{\ln x}{x^3}$	36
2.9	$f(x) = 3, 1 \leq x \leq 5$	37
2.10	$x^2 + y^2 = 1$	39
2.11	square of side 1	40
2.12	Project : Comparison Theorem Part a	42
2.13	Project : Comparison Theorem Part c	43
3.1	不同自由度下的卡方分配圖	48
3.2	不同自由度下的 t 分配圖	49

3.3 不同參數下的 <i>Beta</i> 分配圖	50
3.4 不同參數下的 <i>F</i> 分配圖	51
3.5 不同參數下的 <i>Poisson</i> 分配圖	52
3.6 不同參數下的超幾何分配圖	53
3.7 不同抽樣數量的亂數常態直方圖	54
3.8 樣本數 30 的亂數常態分配相關圖	55
3.9 樣本數 1000 的亂數常態分配相關圖	55
3.10 不同抽樣數的亂數 <i>t</i> 分配直方圖	56
3.11 樣本數 30 的亂數 <i>t</i> 分配相關圖	56
3.12 樣本數 1000 的亂數 <i>t</i> 分配相關圖	57
3.13 <i>Beta</i> 平均樣本抽樣數為 30 下的中央極限定理結果	58
3.14 <i>Beta</i> 平均樣本抽樣數為 100 下的中央極限定理結果	58
3.15 極小值幾何分配抽樣數為 1000 下結果	59
3.16 專題	60
4.1 簡單線性迴歸—兩群資料距離較近	63
4.2 簡單線性迴歸—兩群資料距離較遠	64
4.3 簡單線性—兩群資料變異程度較小	65
4.4 簡單線性—兩群資料變異程度較大	65
4.5 簡單線性—兩群資料變異程度及樣本數相異	66
4.6 簡單線性與加廣型線性的比較—兩群資料距離較近	68
4.7 簡單線性與加廣型線性的比較—兩群資料距離較遠	68
4.8 簡單線性與加廣型線性的比較—兩群資料變異程度較小	69
4.9 簡單線性與加廣型線性的比較—兩群資料變異程度較大	70
4.10 簡單線性與加廣型線性的比較—兩群資料變異程度及樣本數相異	70
4.11 邏輯函數	71
4.12 三群散佈圖	72
4.13 以邏輯斯迴歸模型進行三群分類	73
5.1 LDA 應用於兩群資料的分類狀況	77
5.2 QDA 應用於兩群資料的分類狀況	78
5.3 多群資料中不同 <i>K</i> 值的 KNN 判別邊界	80
5.4 四種學習器在兩共變異矩陣相同資料的分群狀況	82

5.5 四種學習器在三共變異矩陣相同資料的分群狀況	83
5.6 四種學習器在兩共變異矩陣相異資料的分群狀況	85
5.7 四種學習器在三共變異矩陣相異資料的分群狀況	87
5.8 四種學習器在環形資料的分群狀況	88
6.1 一個隱藏層的前饋式類神經網路	92
6.2 兩截式機械手臂 (引用自 MathWorks : Modeling Inverse Kinematics in a Robotic Arm)	93
6.3 兩截式機械手臂所及範圍及訓練資料	95
6.4 兩層的前饋式神經網路架構。資料來源：“淺度機器學習：類神經網路”。汪群超。	95
6.5 總樣本數 $n = 45$ 且一個隱藏層 (含 10 個神經元) 的預測能力	96
6.6 當總樣本數量較小時不同隱藏層神經元數之間的 ANN 模型準確度	97
6.7 當總樣本數量較大時不同隱藏層神經元數之間的 ANN 模型準確度	98
6.8 均勻樣本與不均勻樣本使用於 ANN 模型的學習結果	100
6.9 手寫數字	101
6.10 MLPClassifier 於一個隱藏層 (30 個神經) 的測試資料的預測能力以混淆矩陣表示	101
7.1 相同共變異矩陣的兩群資料	104
7.2 相同共變異矩陣的三群資料	105
7.3 相異共變異矩陣的兩群資料	106
7.4 相異共變異矩陣的三群資料	107
7.5 雙環資料集	108

表 目 錄

1.1	英文字族	13
1.2	字體形狀	13
1.3	字體形狀	14
1.4	字體大小	14
5.1	四種學習器在兩共變異矩陣相同資料的學習狀況	82
5.2	四種學習器在三共變異矩陣相同資料的學習狀況	84
5.3	共變異矩陣相同資料的比較	84
5.4	四種學習器在兩共變異矩陣相異資料的學習狀況	86
5.5	四種學習器在兩共變異矩陣相異資料的學習狀況	86
5.6	共變異矩陣相異資料的比較	88
5.7	四種學習器在環形資料的學習狀況	89
6.1	四種神經元數量在總樣本數相對小 ($n = 45$) 時的訓練與測試 SSE	96
6.2	四種神經元數量在總樣本數相對大 ($n = 564$) 時的訓練與測試 SSE	99
6.3	四種神經元數量在 MLPClassifier 於一個隱藏層的訓練資料及測試資料正確率	102
7.1	七種學習器在兩共變異矩陣相同資料的學習誤判率	105
7.2	七種學習器在三共變異矩陣相同資料的學習誤判率	105
7.3	七種學習器在兩共變異矩陣相異資料的學習誤判率	106
7.4	七種學習器在三共變異矩陣相異資料的學習誤判率	107
7.5	七種學習器在雙環資料集的學習誤判率	108

序

這本書是關於 L^AT_EX 學習、函數繪圖、機器學習和深度學習的指南。本書將以 L^AT_EX 為此本書的文字編譯器，並以 python 為主要的實作語言。我們將首先介紹 L^AT_EX 的基本語法和排版技巧，並舉例說明如何使用 python 繪製函數圖像。

接下來，我們將深入探討機器學習和深度學習的原理和技術，並提供實用的 python 程式碼供讀者運用。最後，我將包含對於機器學習和深度學習的學習器進行評比與比較。

之所以寫這本書，是因為我認為函數繪圖、機器學習和深度學習是資料科學和人工智慧領域中非常重要的工具和技術。我們相信，本書將有助於幫助讀者更好地了解和掌握這些工具和技術，從而更好地解決資料分析和人工智慧的實際問題。本書旨在提供實用的知識和技巧，讓讀者能夠從中受益。我希望本書能成為讀者學習和實踐的重要參考。

製作此本書不僅是對於 L^AT_EX 、函數繪圖、機器學習和深度學習等領域的研究與總結，同時也是我們在碩士論文過程中對於這些技術的實踐與驗證。在寫作過程中，我們不斷地總結和整理所學知識，並深入探討各種技巧和方法的實現細節。在寫完本書後，我們相信它將會是一本寶貴的參考書，並且對於碩士論文的過程有很大的幫助。

張智硯
2023 年 1 月於台北大學

第 1 章

LATEX 自學筆記

此文主要介紹如何使用 LATEX 排版文章，所用的 LATEX 樣版是從汪群超老師的講義以及網路上的開放資源搜集而成。大部分學術文章都使用 LATEX 排版，而碩士論文則有人會使用 Microsoft Word 編寫。但是 Word 在撰寫文章時需要自行控制間距以及段落，導致使用者需要額外花費時間排版。故以下將逐一介紹 LATEX 的功能及語法。

1.1 LATEX 簡介

TEX 是在 1980 年代由 Donald E. Knuth (高德納)¹開發出來的一種文件排版系統，可以用來編輯要出版的研究論文、書籍等文件，甚至也可以拿來製作簡報用的投影片。就電腦軟體來說已經是古董級的產品，其在市場中的存活時間大約 40 年，可說是相當長壽。

1.1.1 版本發展

(a) TEX 與 LATEX

最初的 TEX 之指令較為複雜，因此在 1984 年來自斯坦福的 Leslie Lamport²將 Scribe 的易用性，轉移到 Tex 身上，最終發明了 LATEX 。TEX 與 LATEX 編譯出來的皆是 DVI 檔，可以轉為 PS 或者 PDF 進行印刷。

¹電腦科學家，1974 年圖靈獎得主。

²計算機科學家，於 2013 年獲得圖靈獎。

(b) PdfTeX

pdfTeX 可直接將原始檔編譯並且直接輸出 PDF 檔案和直接取得超連結、目錄和文件資訊。

(c) cwTeX

cwTeX 由吳聰敏教授等人編寫，目的是為了支援中文。cwTeX 的功能是將中文字轉換 TeX 格式。再交由 \LaTeX 排版。在一般的電腦術語中，cwTeX 稱為前階處理程式 (preprocessor)。

(d) XeTeX

XeTeX 是支援現代字型的 TeX 排版引擎，其作者及維護者為 Jonathan Kew。XeTeX 最初只是為 Mac OS 所開發，但它現在在各主要平台上都可以運作。它最大的進步為使用者可使用自己作業系統中所安裝的字型，不需另外安裝套件。

1.1.2 為何要使用 \LaTeX

許多教授、學者在發表專業文章時都是選用 \LaTeX 而非 Microsoft Word，其原因透過網上搜尋就會有大量文章在說明，我將主要的原因列於下方：

- \LaTeX 編輯的文件通常都會比用 Microsoft Word 編輯的文件美觀一點，尤其是在像資訊科學、數學、工程學等需要呈現數學公式的文件。
- \LaTeX 在各種平台上都可以使用，包含 Linux、Mac OS、Windows 等。
- \LaTeX 提供數百種套件 (package) 供應多樣化的使用，從呈現複雜演算法、西洋棋譜、樂譜等，都有合適的套件可以使用。要使用 Word 編輯這些文件就相對困難需多。
- \LaTeX 使用 \LaTeX 不須煩惱文件最後的樣子，如同汪老師課堂上常說的專家模式已經包辦了大部分的排版，使用者只需專注在文章的內容。

1.1.3 學習 \LaTeX 的免費資源

拜網路發展所賜，現今只要你有心想學習，網上就有很多免費的資源提供參考，而 \LaTeX 已是發展很久的技術，所以存有相當多的教材，我逐一在底下介

紹：

(a) 《Xe^LT_EX Tutorial》

此教材為國立臺北大學統計系的 Chun-Chao Wang 汪群超老師編寫，該教材正是我第一次學習 *T_EX* 的資源，可以在 GOOGLE 中搜尋老師的名字，馬上就能找到老師的個人網站開啟學習之路。需要特別注意的地方在於，裏頭很多套件及字型都有 Windows 及 Mac OS 兩種不同寫法，所以在使用時請根據您的作業系統來執行。

(b) 《大家來學^LA^TE_X》

此書作者為 Edward G.J. Lee 李果正，目前這份教材以電子書方式存放於 Github，這本書相當相當適合初學者使用，裏頭有完整介紹一本書該有的架構，並且需要你注意的點。

(c) 《cw^TE_X 排版系統第三版》

此書作者為吳聰敏、吳聰慧教授所編寫，目前這份教材已經絕版了，不再販售，但網路上還存有 PDF 版的電子課本，可自行搜尋使用。這份教材雖然是針對 cw^TE_X 所編寫，但其本體還是 *T_EX* 語言，所以我們還是可以從這本書學習到很多知識點，我最喜歡這本書的地方在於它有很多的範例在書上，並且都有搭配著語法，旁邊還會有貼心的註解。

(d) 《^LA^TE_X 入門》

此書作者為中國的劉海洋老師所編寫，劉老師認為 *L^AT_EX* 就只是一樣工具，使用者沒有必要去買一本厚厚像磚頭的書，因為最後有很大的機率只是供著。該書雖然書名是入門，但我認為還是蠻深入的，需要特別注意的地方為中國的字型及編碼可能與台灣不同，需要小心使用。

(e) 《Overleaf》

Overleaf 是一個非常有名的線上 *L^AT_EX* 編輯、協作平台有了 Overleaf，你無須在電腦上安裝 *T_EX* 環境和 *T_EX* 編輯器，在網站上即可以完成撰寫、編譯和 PDF 導出。該平台的 **Templates** 介面對使用者相當友善，裏頭有大量的 *L^AT_EX* 範本，可供直接下載修改使用。

1.2 文本設定

在製作一份文件時，要先根據你想要製作的文件類型進行文稿類別設定，因為不同的文稿類別會影響文章的整體排版。大部分的文章都以 article 或 book 文稿類文稿類排版而成。學術論文以 article 排版，教科書以 book 排版。跟我們最有關係的碩士論文之排版用任何一種文稿皆可以。文稿一開始通常先排版標題、作者名稱、日期。之後接著目錄然後才進入正文。

1.2.1 首頁

多數文檔類型提供了簡單界面，標準的類型只提供了四種命令 (title, author, thanks, date)。實際標題的排版由命令 \maketitle 來進行排版。具體的布局由使用的文檔類型決定。

《代碼》

```

1 \documentclass[12pt, a4paper]{article}
2 %documentclass{} 選擇文檔類型，12字型大小，a4paper紙張大小。
3 \title{Machine Learning for Trading}
4 %title{} 輸入文章標題。
5 \author{Ryan Babaie \\ Neil Hardy}
6 %author{} 輸入作者名稱。
7 \date{December 30, 2015}
8 %date{} 輸入日期。
9 \begin{document}
10 %begin{document} 開始內文。
11 \maketitle
12 %\maketitle 生成標題。
13 \end{document}
14 %begin{document} 結束內文。
15 }
```

Code 1.1: 製作首頁

《結果》

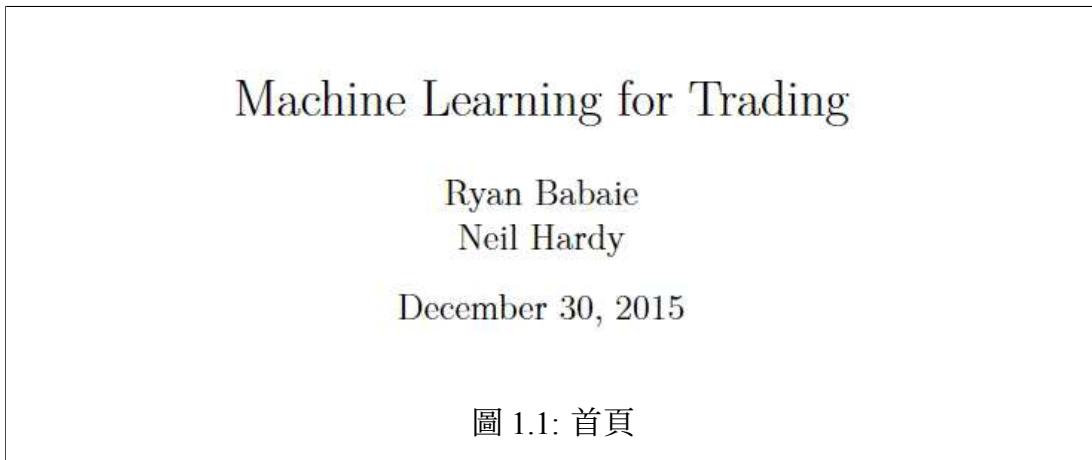


圖 1.1: 首頁

- 1 行：若文檔類型選擇 book 或 report 文稿類別排版時，題目將獨占一頁；但在 article 文稿類別中，題目之後即緊接著排版正文(或摘要)，題目並未獨占一頁。
- 7 行：date{} 中若輸入 today 則會出現當天日期。
- 11 行：maketitle 指令下出後才會出現標題、作者等指令，否則無法正常顯示。這道指令通常是緊接在 \begin{document} 指令之後。

1.2.2 章節

在 *LaTeX* 中，與文章層次有關的，就是章節與列表。不同的文檔類中的章節不一樣。我們常用的 article 文檔類的章節層次有：section, subsection, subsubsection, paragraph, subparagraph。

《代碼》

```
1 \begin{document}
2 \maketitle
3 \section{Python and Basic Interfaces}
4   \subsection{Dataframe}
5     \subsubsection{Reading CSVs into Dataframes}
6     \subsubsection{Plotting}
```

```
7   \subsubsection{Issues}
8 \section{Essential Economics}
9 \section{Machine Learning Algorithms}
10 \section{Statistical Analysis of Time Series Data}
11 }
```

Code 1.2: 章節

《結果》

```
1 Python and Basic Interfaces
  1.1 Dataframe
    1.1.1 Reading CSVs into Dataframes
    1.1.2 Plotting
    1.1.3 Issues
  2 Essential Economics
  3 Machine Learning Algorithms
  4 Statistical Analysis of Time Series Data
```

圖 1.2: 章節

1.2.3 目錄

LaTeX 提供了自動生成目錄的命令，只要在最後一個 `section` 後面加上語法 `\tableofcontents`，目錄神奇的出現，非常簡單直接。

《代碼》

```
1 \begin{document}
2 \maketitle
3 \section{Python and Basic Interfaces}
4   \subsection{Dataframe}
5     \subsubsection{Reading CSVs into Dataframes}
6     \subsubsection{Plotting}
```

```

7   \subsubsection{Issues}
8 \section{Essential Economics}
9 \section{Machine Learning Algorithms}
10 \section{Statistical Analysis of Time Series Data}
11 \tableofcontents %新增此行
12 }
```

Code 1.3: 目錄

《結果》

Contents

1	Python and Basic Interfaces	1
1.1	Dataframe	1
1.1.1	Reading CSVs into Dataframes	1
1.1.2	Plotting	1
1.1.3	Issues	1
2	Essential Economics	1
3	Machine Learning Algorithms	1
4	Statistical Analysis of Time Series Data	1

圖 1.3: 目錄

1.2.4 換行分頁

文件在排版時往往要求每一行具有相同的長度，為了對整段的文檔進行優化，將插入必要的換行和空格。一般情況下每個文件段的首行不縮排，接著每行開始所排，整篇文章左側呈現鋸齒狀，段與段之間沒有多餘的空格。

以下為換行命令：

1. \\ : 換行，放在每行句子結尾。
2. * : 強制換行後，禁止分頁。
3. \\[offset] : 換行，並且與下一行的行間距為原來行間距 +offset。
4. \\newline : 與\\相同。

5. `\linebreak[number]`：這條命令讓系統中斷當前行並將當前行已有文字拉長直至頁邊。如果使用了 `number` 可選參數，則這條命令就變成了一個換行請求，換不換行由系統決定。`number` 的值只能從 0 到 4，值越大代表換換行的意願越強烈。

以下為分頁命令：

1. `\newpage`：命令結束當前頁，開始新的一頁。
2. `\`：該命令如同 `\newpage`，新增新的一頁，此外，還可以清除浮動體³，這個作用是很大的。T_EX 在處理浮動問題上，採用隊列的方式，若是一個浮動體出問題便會影響後續浮動體處理。這時就需要使用 命令來解決此類問題。
3. `\pagebreak[number]`：這條命令讓系統從文本當前位置結束當前頁。如果給出了可選參數 `number`，則此命令變成了一個請求，如何處理由系統決定。`number` 的值只能從 0 到 4，值越大代表換行的意願越強烈。

以下為段落命令：

1. `\indent`：這條命令產生一塊水平空白區域，其寬度等於段落的縮排距離值。
2. `\par`：這條命令與一個空行的效果相同。

1.2.5 插入編號

1. `{itemize}`：此命令對文本進行簡單的排列，不是採用序號，而是實心圓點符號。這個命令需要和 `\item` 配合使用。

《代碼》

```

1 \begin{itemize}
2   \item List entries start with the \verb|\item| command.
3   \item Individual entries are indicated with
4   a black dot, a so-called bullet.
```

³T_EX 環境中浮動體預設為圖片及表格

```
5 \end{itemize}
```

Code 1.4: itemize

《結果》

Lists are easy to create:

- List entries start with the `\item` command.
- Individual entries are indicated with a black dot, a so-called bullet.
- The text in the entries may be of any length.

圖 1.4: itemize

2. `{itemize}` 圖標變換：如果 `item` 不想使用預設的黑點，可在 `item` 後方加上 [符號]，`item` 將會進行變化。

《代碼》

```
1 \begin{itemize}
2   \item This is my first point
3   \item Another point I want to make
4   \item[!] A point to exclaim something!
5   \item[$\blacksquare$] Make the point fair and square.
6   \item[NOTE] This entry has no bullet
7   \item[] A blank label?
8 \end{itemize}
```

Code 1.5: itemize 變換標示

《結果》

Change the labels using `\item[label text]` in an `itemize` environment

- This is my first point
- Another point I want to make
 - ! A point to exclaim something!
- Make the point fair and square.

圖 1.5: itemize 變換標示

3. `{enumerate}`：此命令與 `{itemize}` 差異在 `{enumerate}` 可使文本進行有順序的排列，默認是採用數字 1,2,3……進行排列。

《代碼》

```
1 Numbered (ordered) lists are easy to create:  
2 \begin{enumerate}  
3   \item Items are numbered automatically.  
4   \item The numbers start at 1 with each use of the  
5     \texttt{enumerate} environment.  
6   \item Another entry in the list  
7 \end{enumerate}
```

Code 1.6: itemize

《結果》

Numbered (ordered) lists are easy to create:

1. Items are numbered automatically.
2. The numbers start at 1 with each use of the `enumerate` environment.
3. Another entry in the list

圖 1.6: enumerate

3. `{enumerate}` 變換編號格式：與原先的差異在於想使用其他編號格式前，需先下指令 `\usepackage{enumerate}`，此為包含 `enumerate` 的宏包，接著在 `\begin{enumerate}` 後方加上 [符號]。

《代碼》

```
1 \usepackage{enumerate}
2 %\usepackage{enumerate}需要放在\begin{document}前面。
3 \begin{document}
4 \begin{enumerate}[(a)]
5 %編號格式採用小寫英文字母
6 \item Ducati Scrambler 1100 Sport Pro ABS
7 \item KTM Duke 1290 Super R ABS
8 \item Kawasaki ZH2 ABS
9 \item YAMAHA Serow 250
10 \end{enumerate}
11 \begin{enumerate}[(A)]
12 %編號格式採用大寫英文字母
13 \item Ducati Scrambler 1100 Sport Pro ABS
14 \item KTM Duke 1290 Super R ABS
15 \item Kawasaki ZH2 ABS
16 \item YAMAHA Serow 250
17 \end{enumerate}
18 \begin{enumerate}[(i)]
19 %編號格式採用羅馬數字
20 \item Ducati Scrambler 1100 Sport Pro ABS
21 \item KTM Duke 1290 Super R ABS
22 \item Kawasaki ZH2 ABS
23 \item YAMAHA Serow 250
24 \end{enumerate}
25 \begin{enumerate}[Motor(1):]
26 %編號格式採用自定義文字 + 數字
27 \item Ducati Scrambler 1100 Sport Pro ABS
28 \item KTM Duke 1290 Super R ABS
29 \item Kawasaki ZH2 ABS
30 \item YAMAHA Serow 250
```

```
31 \end{enumerate}  
32 \end{document}
```

Code 1.7: enumerate 變換編號格式

《結果》

```
(a) Ducati Scrambler 1100 Sport Pro ABS  
(b) KTM Duke 1290 Super R ABS  
(c) Kawasaki ZH2 ABS  
(d) YAMAHA Serow 250  
(A) Ducati Scrambler 1100 Sport Pro ABS  
(B) KTM Duke 1290 Super R ABS  
(C) Kawasaki ZH2 ABS  
(D) YAMAHA Serow 250  
(i) Ducati Scrambler 1100 Sport Pro ABS  
(ii) KTM Duke 1290 Super R ABS  
(iii) Kawasaki ZH2 ABS  
(iv) YAMAHA Serow 250  
Motor(1): Ducati Scrambler 1100 Sport Pro ABS  
Motor(2): KTM Duke 1290 Super R ABS  
Motor(3): Kawasaki ZH2 ABS  
Motor(4): YAMAHA Serow 250
```

圖 1.7: enumerate 變換編號格式

1.2.6 字體

本節只討論行文中的字體使用，數學環境內字體使用寫在下一章。在論文中我們使用最多的是一個字體的四種性質：字族 (font family)，字體形狀 (font shape)，字體系列 (font series)，和字體大小 (size)。

1.2.6.1 英文字體

LATEX 環境中提供帶參數命令及字體聲明兩種字體修改的命令，前者用於少量字體更換，後者用於環境中整體字體更換。

1. 字族

預設的字族有三種：羅馬字族 (roman family)，無襯線字族 (sans serif family) 和打字機字族 (typewriter family)，其中無襯線字族也叫等線字族，打字機字族稱為等寬字族，其命令顯示如下：

表 1.1: 英文字族

字族	帶參數命令	聲明命令	效果
羅馬	<code>\textrm{text}</code>	<code>\rmfamily</code>	Roman font family
無襯字	<code>\textsf{text}</code>	<code>\sffamily</code>	Sans serif font family
打字機	<code>\texttt{text}</code>	<code>\ttfamily</code>	Typewriter font family

2. 字體形狀

預設的字體形狀有四種：直立形狀 (upright shape)，義大利形狀 (italic shape)，傾斜形狀 (slanted shape)，小型大寫形狀 (small capitals shape)，其對應的命令及顯示如下：

表 1.2: 字體形狀

字體形狀	帶參數命令	聲明命令	效果
直立	<code>\textup{text}</code>	<code>\upshape</code>	Upright shape
義大利	<code>\textit{text}</code>	<code>\itshape</code>	<i>Italic shape</i>
傾斜	<code>\textsl{text}</code>	<code>\slshape</code>	<i>Slanted shape</i>
小型大寫	<code>\textsc{text}</code>	<code>\scshape</code>	small capitals

- 論文中正文模式使用直立字型。
- 斜體通常指的是義大利形狀，類似於手寫體，數學公式中的字體一般也使用義大利形狀字體。
- 並非所有的字族都有這麼多字體形狀，擁有小型大寫的比較少。

3. 字體系列

預設的字體系列有兩種，中等 (medium) 和加寬加粗 (bold extend)，其對應的命令及顯示如下：

表 1.3: 字體形狀

字體系列	帶參數命令	聲明命令	效果
中等	\textmd{text}	\mdseries	Medium series
加寬加粗	\textbf{text}	\bfseries	Bold extend series

4. 字體大小

在 L^AT_EX 修改字體的大小有好幾種，第一種：直接在 [documentclass] 中設定，第二種：使用 fontsize 設定，第三種：使用字體尺寸命令，其對應的命令及顯示如下 (本文章字體設定為 12pt)：

表 1.4: 字體大小

字體尺寸命令	排版效果	字體尺寸命令	排版效果
\tiny	Taiwan	\large	Taiwan
\scriptsize	Taiwan	\Large	Taiwan
\footnotesize	Taiwan	\LARGE	Taiwan
\small	Taiwan	\huge	Taiwan
\normalsize	Taiwan	\Huge	Taiwan

1.2.6.2 中文字體

若是在 L^AT_EX 中希望使用中文，可以使用\usepackage{xeCJK}。xeCJK 這個 package 可用於顯示中文、日文及韓文。

設定中文字體方式

《代碼》

```
1 \documentclass[12pt, a4paper]{article}
```

```
2 \usepackage{xeCJK}
3 \setCJKmainfont{TW-Kai}
4 \newCJKfontfamily{\cy}{cwTeXYen}
5 \newCJKfontfamily{\cm}{cwTeXMing}
6 \newCJKfontfamily{\ck}{cwTeXKai}
7 \newCJKfontfamily{\ch}{cwTeXHeiBold}
8 \newCJKfontfamily{\cf}{cwTeXFangSong}
9 \newCJKfontfamily{\wmh}{WenQuanYi Micro Hei}
10 \newCJKfontfamily{\bh}{BabelStone Han}
11 \newCJKfontfamily{\nss}{Noto Serif CJK TC}
12 \newCJKfontfamily{\nsf}{cwTeXYen}
13 \begin{document}
14 {\cy 我是張智硯}
15 {\cm 我是張智硯}
16 {\ck 我是張智硯}
17 {\ch 我是張智硯}
18 {\cf 我是張智硯}
19 {\wmh 我是張智硯}
20 {\bh 我是張智硯}
21 {\nss 我是張智硯}
22 {\nsf 我是張智硯}
```

Code 1.8: 修改中文字體

《結果》

我是張智硯
我是張智硯
我是張智硯
我是張智硯
我是張智硯
我是張智硯
我是張智硯
我是張智硯
我是張智硯

圖 1.8: 修改中文字體

1.3 數學公式

我認為數學公式是最強的功能，同時也是最複雜的部分，我在下方提供幾個例子：

《常態分配》

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (1.1)$$

《常態分配-動差母函數》

$$M_X(t) = \exp\left(\mu t + \sigma^2 \frac{t^2}{2}\right) \quad (1.2)$$

《Student's t-distribution》

$$f(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\nu\pi} \Gamma(\nu/2) (1+x^2/\nu)^{(\nu+1)/2}} \quad (1.3)$$

圖 1.9 為不同自由度底下的 T 分配函數圖

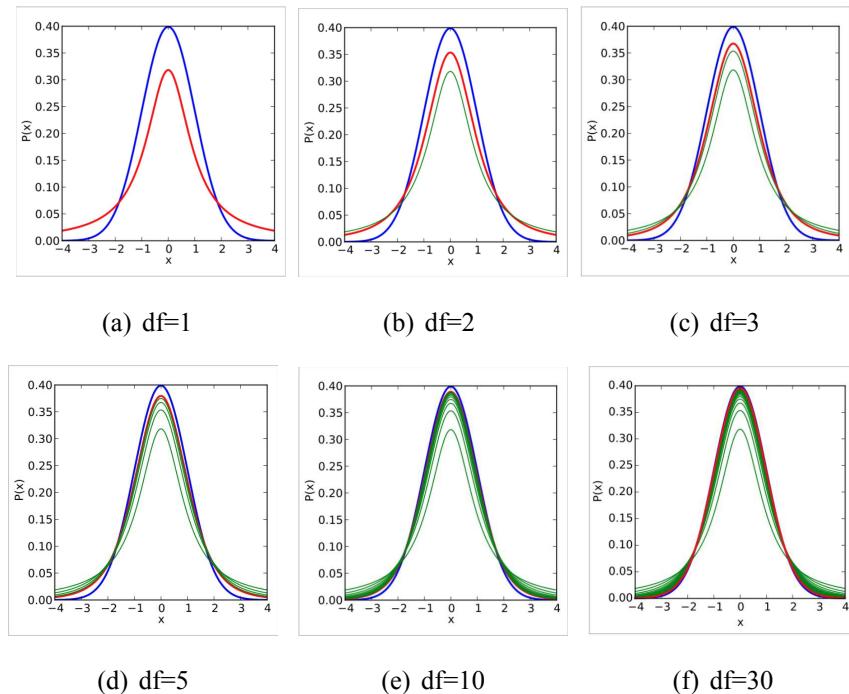


圖 1.9: 不同自由度底下的 T 分配圖型

《二元常態分配》

$$f(x, y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)} \left[\left(\frac{x-\mu_X}{\sigma_X}\right)^2 - 2\rho\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right) + \left(\frac{y-\mu_Y}{\sigma_Y}\right)^2 \right]} \quad (1.4)$$

《向量》

$$\vec{v} + \vec{w} = (v_1 + w_1)\vec{e}_1 + (v_2 + w_2)\vec{e}_2 + \cdots + (v_n + w_n)\vec{e}_n \quad (1.5)$$

《積分、極限、求和、乘積》

$$\lim_{x \rightarrow \infty} x_{22}^2 - \int_1^5 x dx + \sum_{n=1}^{20} n^2 = \prod_{j=1}^3 y_j + \lim_{x \rightarrow -2} \frac{x-2}{x} \quad (1.6)$$

《公式組合》

$$D(x) = \begin{cases} \lim_{x \rightarrow 0} \frac{a^x}{b+c}, & x < 3 \\ \pi, & x = 3 \\ \int_a^{3b} x_{ij} + e^2 dx, & x > 3 \end{cases} \quad (1.7)$$

《矩陣一》

$$A_{2 \times 2} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (1.8)$$

《矩陣二》

$$\begin{pmatrix} 1 & 0 & & 0 \\ 0 & 1 & & \\ & & 1 & 0 \\ 0 & & 0 & 1 \end{pmatrix} \quad (1.9)$$

《矩陣三》

$$A_{m \times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad (1.10)$$

《矩陣四》

$$A_{3 \times 3} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad (1.11)$$

《矩陣五》

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{T}_3(\phi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (1.12)$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{T}_2(\theta) \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \quad (1.13)$$

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \mathbf{T}_1(\psi) \begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} \quad (1.14)$$

《矩陣六》

$$\left[\begin{array}{cc|cc|cc} 168 & 368 & 26 & 368 & 26 & 168 \\ 368 & 1026 & 70 & 1026 & 70 & 368 \\ \hline 26 & 70 & 5 & 70 & 5 & 26 \\ 368 & 1026 & 70 & 1026 & 70 & 368 \\ \hline 26 & 70 & 5 & 70 & 5 & 26 \\ 168 & 368 & 26 & 368 & 26 & 168 \end{array} \right] \quad (1.15)$$

1.4 表格

表格在文章中扮演很重要的角色，掌握了表格就能使你的文章更有條理

《表格一》

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)
(4,1)	(4,2)	(4,3)

《表格二》

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures.
Tuesday	9C	19C	Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.
Wednesday	10C	21C	Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.

《表格三》

multi col row		multi col			multi row
		(2,3)	(2,4)	(2,5)	
multi row	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)

《表格四》

B		multi col			multi row
		(2,3)	(2,4)	(2,5)	
multi row	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)

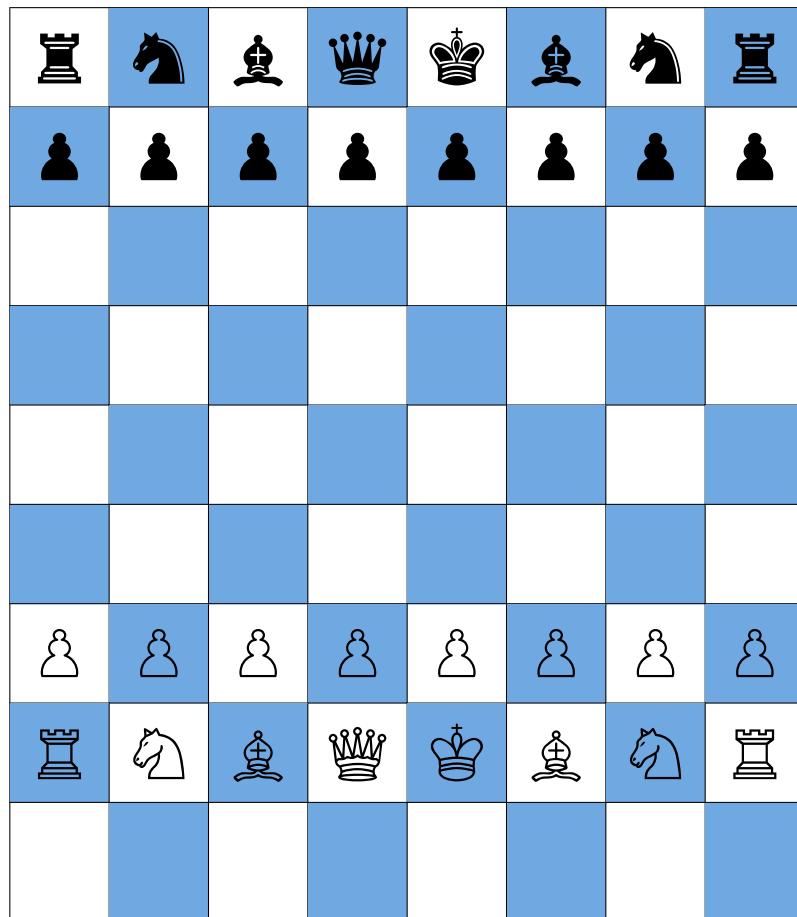
《表格五》

First column	Second column	Third column
1	A	E
2	B	F
3	C	G
4	D	H

《表格六》

	First column	Second column
First line	A	E
Second line	B	F
Third line	C	G
Fourth line	D	H

《表格七》



Chess	King	Queen	Rook	Bishop	Knight	Pawn
White	♔	♕	♖	♗	♘	♙
Black	♚	♛	♜	♝	♞	♟

1.5 結論

這份 \LaTeX 文件是課堂上要求的第一份作業，而我的標題則是打“自學筆記”，因為我確實花了很多的時間在查閱相關書籍還有文章，從一開始連標題都打不出來，到能簡陋完成 20 頁的 PDF，中間最大的學習就是要動手做。 \TeX 系統中的套件有太多種了，根本沒有學完的一天，因此我希望能夠掌握碩士論文會用到的套件、圖型、公式等就足夠了。

第 2 章

函數計算與繪製

此篇《函數計算與繪製》，為統計應用數學與計算課程的第二份作業，主要是紀錄如何使用 Python 編程撰寫數學函數並以視覺化方法呈現，其有關計算的套件為 Numpy；而函數圖形的繪製主要在練習 matplotlib 套件並熟悉其繪圖指令與技巧，最終再將圖片存檔為 eps 格式。

2.1 函數繪圖

(1) Sketch the graph of $f(x) = \sin(x) + \cos(x)$

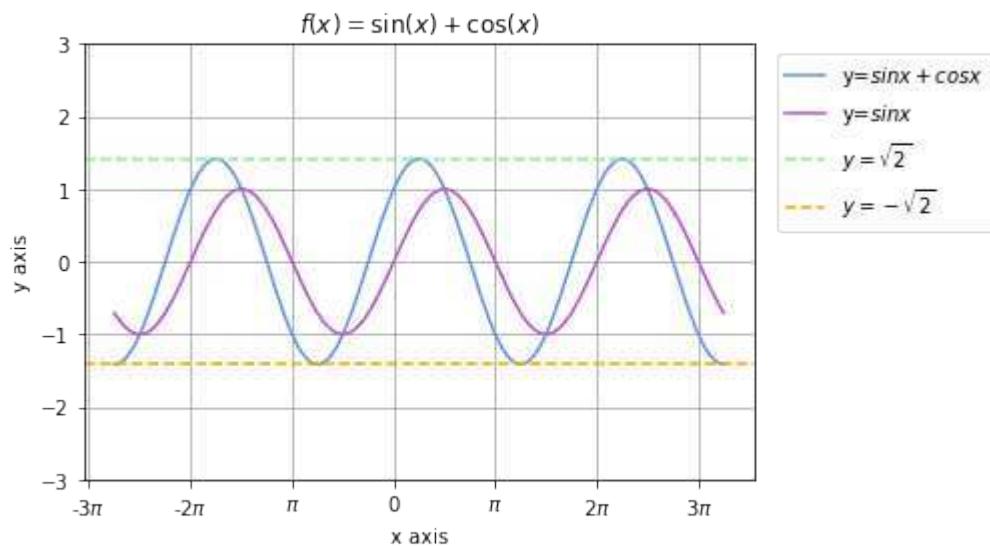


圖 2.1: $f(x) = \sin(x) + \cos(x)$

- $f(x) = \sin(x) + \cos(x) = \sqrt{2} \sin(x + \frac{\pi}{4})$ ，表示兩個周期相同的三角函數相加仍可形成三角函數，屬於函數的疊合。
- 三角函數經過疊合後，其函數最大值和最小值從 $(1, -1) \Rightarrow (\sqrt{2}, -\sqrt{2})$ 。
- $f(x) = \sqrt{2} \sin(x + \frac{\pi}{4})$ 是 $f(x) = \sin(x)$ 圖形沿著 X 軸左移 $\frac{\pi}{4}$ ，再垂直放大 $\sqrt{2}$ 倍。
- 函數週期維持 2π 。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(-2.75*np.pi, 3.25*np.pi, 200)
5 y1 = np.sin(x) + np.cos(x)
6 y2 = np.sin(x)
7 fig, ax = plt.subplots(1)
8 ax.plot(x, y1, color="#6495ED",
9 label = 'y=$sinx+cosx$')
10 ax.plot(x, y2, color="#BA55D3",
11 label = 'y=$sinx$')
12 plt.axhline(y = 2**0.5, color = '#90EE90', linestyle =
13 '--',
14 label = "$y=\sqrt{2}$")
15 plt.axhline(y = -(2**0.5), color = '#FFA500', linestyle =
16 '--',
17 label = "$y=-\sqrt{2}$")
18 ax.set_xticks(np.array([-3, -2, -1, 0, 1, 2, 3])*np.pi)
19 ax.set_xticklabels(['-3$\pi$', 
20 '-2$\pi$', '$\pi$', '0', '$\pi$', '2$\pi$', '3$\pi$'],
21 \fontsize=10 )
22 plt.title(r'$f(x)=\sin(x)+\cos(x)$')

```

Code 2.1: $f(x) = \sin(x) + \cos(x)$

- 7 行：此圖使用 pyplot 的 subplots 子圖功能進行繪圖，subplots 可進行多圖繪製；但若只有一張圖，則指令前方定義的 fig 和 ax 屬於同一對象。
- 18 行：若使用 Plot 預設的軸刻度，可能產生過小的間距使得軸刻度太過擁擠，或是無法表達該函數的特性，而 ax.set_xticks() 指令可自行設定 X 軸的刻度。此題在繪製三角函數時正好可以使得 X 軸都為 π 的倍數。
- 19 行：ax.set_xticklabels() 可以為 ax.set_xticks() 的自定義刻度命名，兩者相互配合。
- 25 行：plt.legend() 可以在圖中顯示 label，指令中的參數座標 (1.02, 1) 則為 label 放置的位置。

(2) Sketch the graph of $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

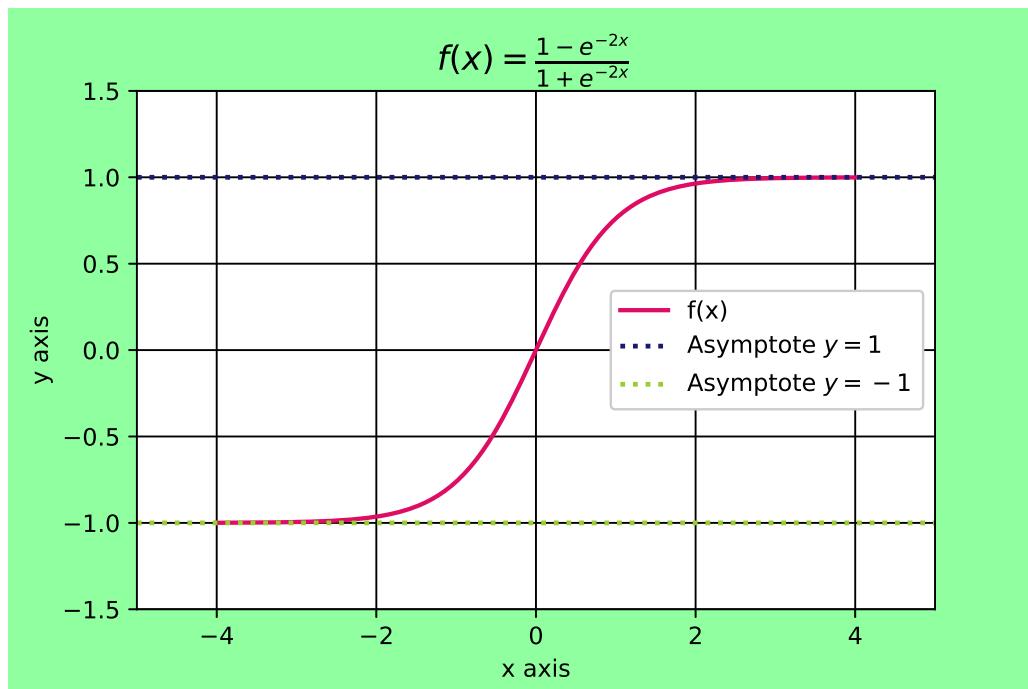


圖 2.2: $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

- 定義域： \mathbb{R} 。

- 值域： $\{f \in \mathbb{R} \mid 1 < f < -1\}$ 。
- 可以發現函數值包含於它的上下界，因此可知 $y = 1$ 和 $y = -1$ 兩者為 $f(x)$ 的漸近線。
- 此函數屬於兩種指數函數相除所形成，其圖型對稱於原點，屬於奇函數。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(-4,4,200)
5 y = np.divide(1-np.exp(-2*x),1+np.exp(-2*x))
6 fig, ax = plt.subplots(1)
7 fig.patch.set_facecolor('xkcd:mint green')
8 ax.plot(x, y, color="#8B0000", linewidth=1.8)
9 plt.axhline(y = 1, color = '#191970', linestyle = ':',
10 label ="Asymptote $y=1$", linewidth=2)
11 plt.axhline(y = -1, color = '#9ACD32', linestyle = ':',
12 label ="Asymptote $y=-1$", linewidth=2)
13 ax.set_xlim([-5, 5])
14 ax.set_ylim([-1.5, 1.5])
15 plt.xlabel('x axis')
16 plt.ylabel('y axis')
17 ax.grid(True)
18 plt.legend()
19 plt.title(r'$f(x)=\frac{1-e^{-2x}}{1+e^{-2x}}$')
20 plt.show()

```

Code 2.2: $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

- 4 行：此函數相除部分調用 Numpy 中的 `divide(x, y)` 功能。
- 6 行：設定畫布背景顏色為 XCKD 系列的 “mint green”。

- 7-10 行：使用 `plt.axhline` 繪製水平漸近線。

(3) Sketch the graph of $f(x) = \sqrt[3]{\frac{4-x^3}{1+x^2}}$

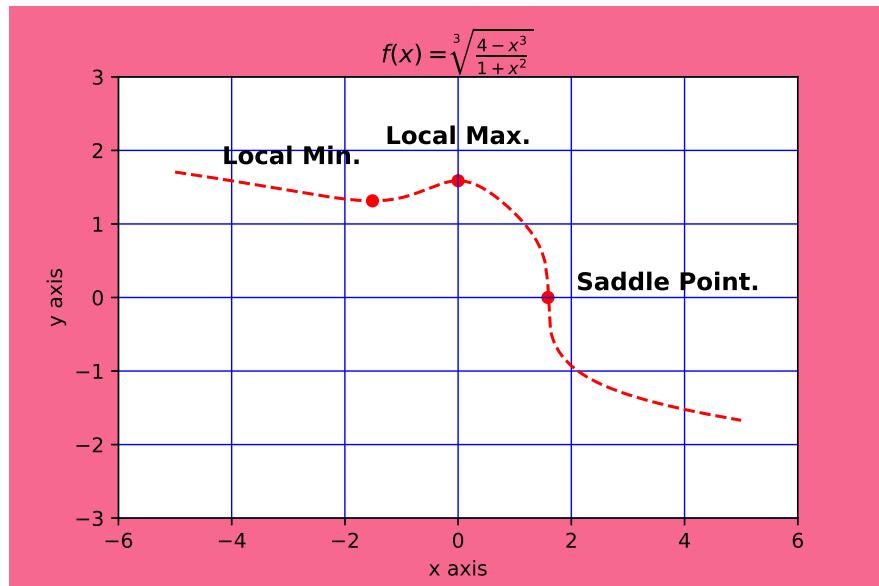


圖 2.3: $f(x) = \sqrt[3]{\frac{4-x^3}{1+x^2}}$

- 函數 $f(x) = \sqrt[3]{\frac{4-x^3}{1+x^2}}$ 之定義域及值域皆屬於實數，故無漸近線。
- 函數 $f(x)$ 經過一階導數判斷後可求出兩個極值。相對極小值 $(-1.51, 1.31)$ 和相對極大值 $(0, \sqrt[3]{4})$ 。
- 函數 $f(x)$ 經過二階導數判斷有一個反曲點 (Saddle Point) 在 $(\sqrt[3]{4}, 0)$ 。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(-5,5,200)
5 y = np.cbrt(np.divide(4-x**3,1+x**2))
6 fig, ax = plt.subplots(1)
7 ax.plot(x, y, linestyle = '--', color = 'r', alpha=0.9)
8 plt.scatter(-1.51274,1.31407, s=30, c='r')

```

```

9 plt.scatter(0, 4**((1/3)), s=30, c='r')
10 plt.scatter(4**((1/3)), 0, s=30, c='r')
11 plt.text(-1.51274-0.2, 1.31407+0.5, 'Local Min.', ha='right',
12 weight='heavy', fontsize=12)
13 plt.text(0, 4**((1/3))+0.5, 'Local Max.', ha='center',
14 weight='heavy', fontsize=12)
15 plt.text(4**((1/3))+0.5, 0.1, 'Saddle Point.', ha='left',
16 weight='heavy', fontsize=12)
17 ax.grid(visible = True, color='b', linestyle='--')
18 plt.title(r'$f(x)=\sqrt[3]{\frac{4-x^3}{1+x^2}}$')
19 plt.show()

```

Code 2.3: $f(x) = \left(\frac{4-x^3}{1+x^2}\right)^{\frac{1}{3}}$

- 4 與 11 行：繪製此函數須注意 X 軸範圍不能抓太大，否則圖形變得過於扁平，將無法清楚顯示兩個極值點和一個反曲點。
- 5 行：`np.cbrt()` 是快速尋找立方根的方式，它與其它的語法差異在於可以輸入負數，並且顯示正確結果。
- 8-10 行：利用 `plt.scatter(x, y, s, c)` 將所需要的點獨立標示出來，`xy` (座標)、`s` (圓點大小)、`c` (顏色)。
- 15-20 行：利用 `plt.text(x, y, name, ha, weight)` 繪製座標中的文字，其中 `ha` 為對齊參數 `horizontalalignment` 縮寫，可以控制文字位置，而 `weight` 控制文字粗細。

(4) Sketch the graph of $f(x) = \frac{1}{x}$

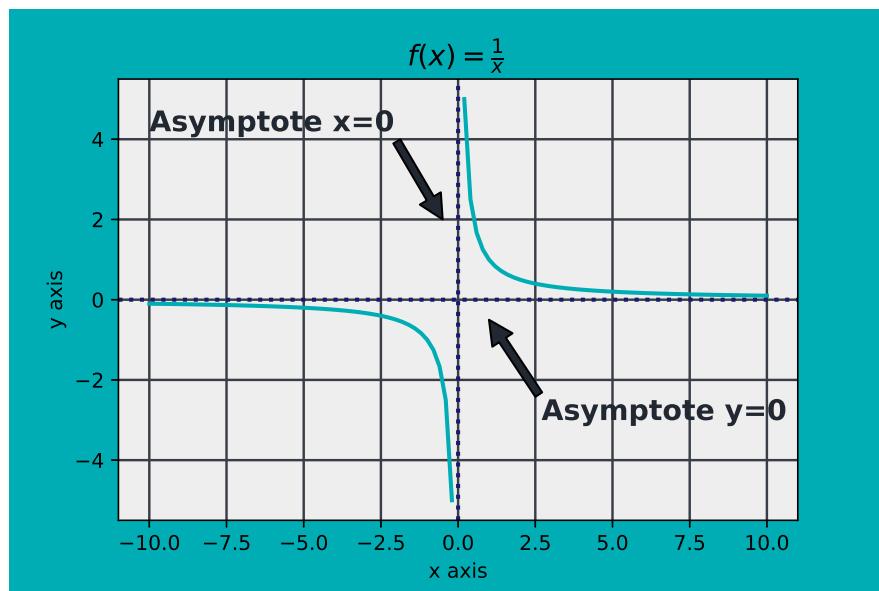


圖 2.4: $f(x) = \frac{1}{x}$

- 定義域： $(-\infty, 0) \cup (0, \infty)$ 。
- 值域： $\{f \in \mathbb{R} \mid f < 0 \text{ or } f > 0\}$ 。
- 根據定義域及值域可以發現圖形有兩條漸近線， $x = 0$ 與 $y = 0$ 。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x):
5     with np.errstate(divide='ignore', invalid='ignore'):
6         return 1/x
7 x=np.linspace(-10,10,101)
8 y=f(x)
9 plt.plot(x, y, linewidth=2, color ="#00ADB5")
10 ax = plt.gca()
11 ax.patch.set_facecolor('#EEEEEE')
12 plt.axhline(y = 0, color = '#191970', linestyle = ' : ',
```

```

13 label ="Asymptote $y=1$", linewidth=2)
14 plt.axvline(x = 0, color = '#191970', linestyle = ':',
15 label ="Asymptote $y=1$", linewidth=2)
16 plt.annotate('Asymptote x=0', xy=(-0.5,2), xytext=(-10,4),
17             color='#222831', weight='heavy', fontsize=14,
18             arrowprops={'facecolor': '#222831'})
19 plt.annotate('Asymptote y=0', xy=(1,-0.5), xytext=(2.7,-3),
20             color='#222831', weight='heavy', fontsize=14,
21             arrowprops={'facecolor': '#222831'})
22 plt.xlabel('x axis')
23 plt.ylabel('y axis')
24 plt.grid(visible = True, color='#393E46', linestyle='--',
25 linewidth=1.2)
26 plt.title(r'$f(x)=\frac{1}{x}$')
27 plt.show()

```

Code 2.4: $f(x) = \frac{1}{x}$

- 此題有關於顏色的部分都使用 16 進位制的顏色代碼，優點為可以精確顯示指定顏色。
- 4 行：此題使用 `def` (定義) 函數的方式來繪製，其中 `np.errstate()` 可以將函數中 `invalid` 值除去，因函數 $f(x) = \frac{1}{x}$ 牽涉到 $\frac{1}{0}$ 的無限值，但我們並不希望此無限值出現在圖中，故將其省略。
- 16-21 行：`plt.annotate()` 用於在圖形上給數據添加文本註釋，並且支持帶箭頭的劃線工具。此題我用此方法來標示兩條漸近線。

(5) Sketch the graph of $f(x) = \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{8}}$

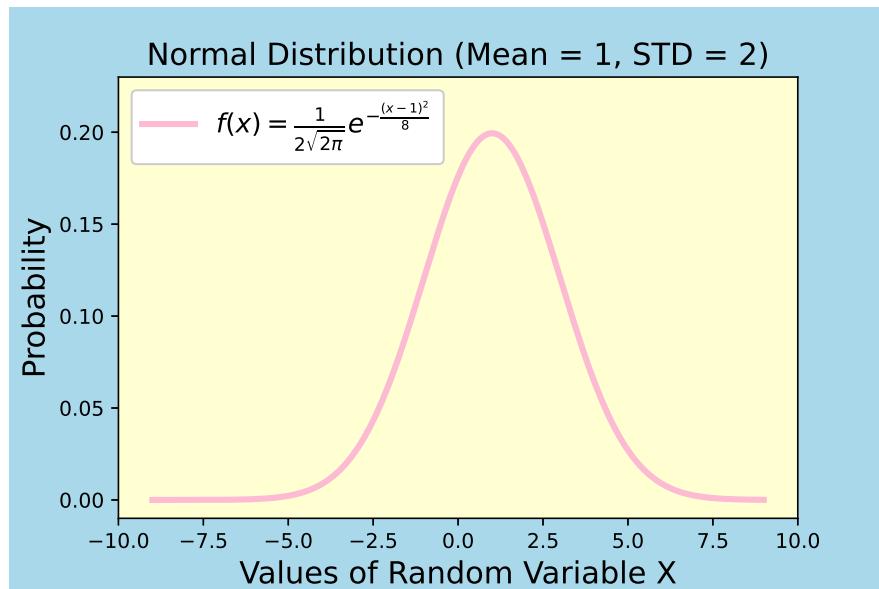


圖 2.5: $f(x) = \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{8}}$

函數 $f(x) = \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{8}}$ 可以看做常態機率密度函數來繪製，其平均數 = 1 且標準差 = 2。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy import stats
4
5 mu = 1
6 std = 2
7 snd = stats.norm(mu, std)
8 x = np.linspace(-9, 9, 200)
9 fig=plt.figure()
10 fx_name = r'$f(x)=\frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{8}}$'
11 plt.plot(x, snd.pdf(x), label=fx_name, color="#FCBAD3",
12 linewidth=3)
13

```

```

14 plt.legend(loc='upper left', fontsize=13)
15 plt.title('Normal Distribution (Mean = 1, STD = 2)', 
16 fontsize='15')
17 plt.xlabel('Values of Random Variable X', fontsize='15')
18 plt.ylabel('Probability', fontsize='15')
19 plt.show()

```

Code 2.5: $f(x) = \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{8}}$

- 3 行：此函數屬於常態機率密度函數，所以從 `scipy` 調用 `stats` 功能。`stats` 裡有許多有關統計的套件。
- 5-7 行：`stats.norm(mu, std)` 是在建構密度函數，裏頭的參數為函數的平均數及標準差，此方法可以省略自行建構函數 $f(x)$ 的動作，很有效率。

(6) Sketch the graph of $f(x) = \sqrt[3]{x^2}$

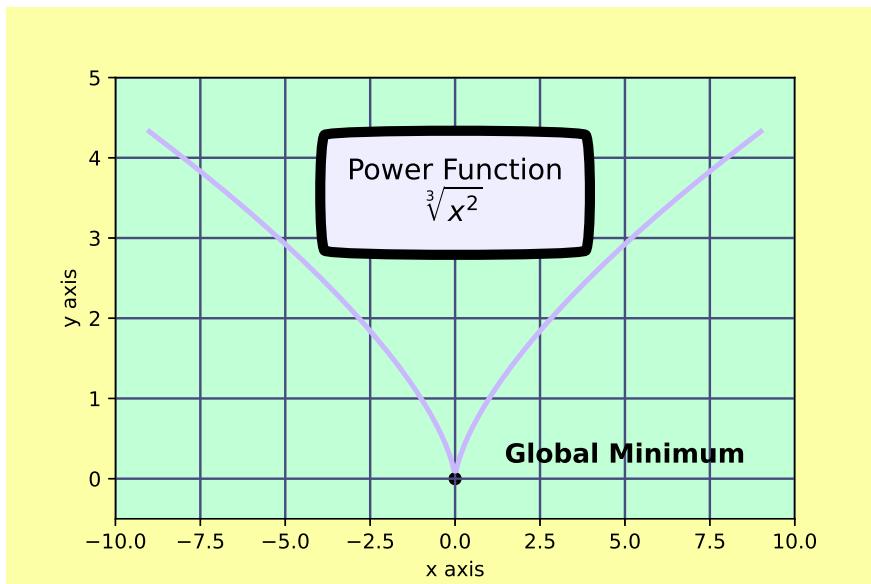


圖 2.6: $f(x) = \sqrt[3]{x^2}$

- 函數 $f(x) = \sqrt[3]{x^2}$ 屬於 Power Function，對稱軸為 Y 軸。
- 函數值域： $\{f \in \mathbb{R} \mid f \geq 0\}$ ，有全域極小值於 $(0, 0)$ 。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x):
5     return np.cbrt(x**2)
6 x=np.linspace(-9,9,300)
7 y=f(x)
8 fig=plt.figure()
9 plt.plot(x, y, linewidth=2.5, c='#CAB8FF')
10 fig.patch.set_facecolor('#FCFFA6')
11 ax = plt.gca()
12 ax.patch.set_facecolor('#C1FFD7')
13 plt.scatter(0,0, s=30, c='black')
14 bbox_props = dict(boxstyle='round4', pad=1, rounding_size=.2,
15 ec='black', fc='#EEEEFF', lw=5)
16 plt.text(0, 3.2, 'Power Function\n$\sqrt[3]{x^2}$',
17 fontsize=14, color='black', ha='center', bbox=bbox_props)
18 plt.text(5,0.2,'Global Minimum', ha='center',
19 weight='heavy', fontsize=13)
20 plt.show()

```

$$\text{Code 2.6: } f(x) = (x^2)^{\frac{1}{3}}$$

14-15 行：字典屬性 bbox 主要就是在用不同的框，把 text 框起來用一系列屬性來定義外型。`dict(boxstyle, pad, rounding_size)`，`boxstyle` 控制外型、`pad` 控制整體大小、`rounding_size` 控制外型曲度。

(7) Sketch the graph of $f(x) = 2x^3 - x^4$

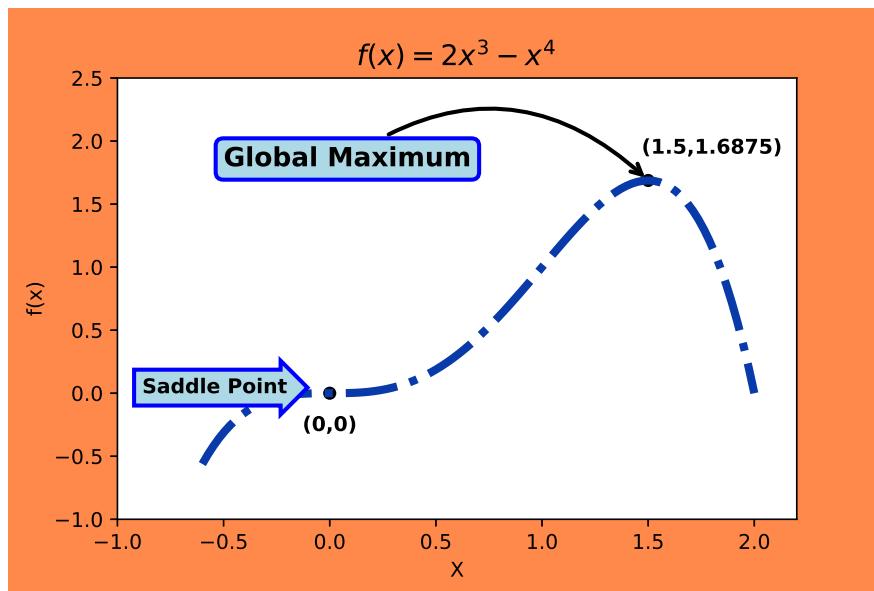


圖 2.7: $f(x) = 2x^3 - x^4$

- 函數 $f(x) = 2x^3 - x^4$ 屬於一多項式 (polynomial)。
- 函數 $f(x) = 2x^3 - x^4$ 經過一階導數判斷後有極大值 $(\frac{3}{2}, \frac{27}{16})$ 。
- 函數 $f(x) = 2x^3 - x^4$ 經過二階導數判斷後有反曲點 $(0, 0)$ 。

《Python Code》

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 xmin, xmax = -0.6, 2
5 x = np.linspace(xmin, xmax, 100)
6 p = [-1, 2, 0, 0, 0]
7 f = lambda x : np.polyval(p, x)
8 fig = plt.figure()
9 ax = plt.gca() # get current axis
10 ax.plot(x, f(x), linewidth = 4, color ='#083AA9',
11 linestyle ='-.')
12 fig.patch.set_facecolor('#FF884B')

```

```

13 ax = plt.gca()
14 ax.set_xlabel('X'), ax.set_ylabel('f(x)')
15 ax.set_title('$f(x)=2x^3-x^4$', fontsize =14)
16 bbox_props1= dict(boxstyle='round',pad=0.3 , ec='b' , lw=2 ,
17 fc='lightblue')
18 plt.annotate('Global maximum' , xy=(3/2, 27/16) ,
19 xytext=(-0.5, 1.8) , arrowprops=dict(arrowstyle='->' , lw=2 ,
20 connectionstyle='arc3 , rad=-0.4') , fontsize=13 ,
21 weight='heavy' , bbox=bbox_props1)
22 bbox_props2= dict(boxstyle='rarrow',pad=0.3 , ec='b' , lw=2 ,
23 fc='lightblue')
24 plt.text(-0.2, 0, 'Saddle Point' , bbox=bbox_props ,
25 ha='right' , weight='heavy')
26 plt.text(0, -0.3, '(0,0)' , ha='center' , weight=
27 'heavy' , fontsize=12)
28 plt.text(1.8, 1.9, '(1.5,1.6875)' , ha='center' ,
29 weight='heavy' , fontsize=10)
30 plt.scatter(0,0, s=30, c='black')
31 plt.scatter(1.5,27/16, s=30, c='black')
32 plt.show()

```

Code 2.7: $f(x) = 2x^3 - x^4$

- 6 與 7 行：函數 $f(x) = 2x^3 - x^4$ 屬於一多項式 (polynomial)，所以可以使用指令 `np.polyval(p, x)`，參數 p 帶入多項式的係數串列，x 帶入指定的數值。
- 18 行：此部分的 `plt.annotate` 符號使用箭頭功能，其相關指令為字典 `arrowprops = dict(arrowstyle = '->.....')`。

(8) Sketch the graph of $f(x) = \frac{\ln x}{x^3}$

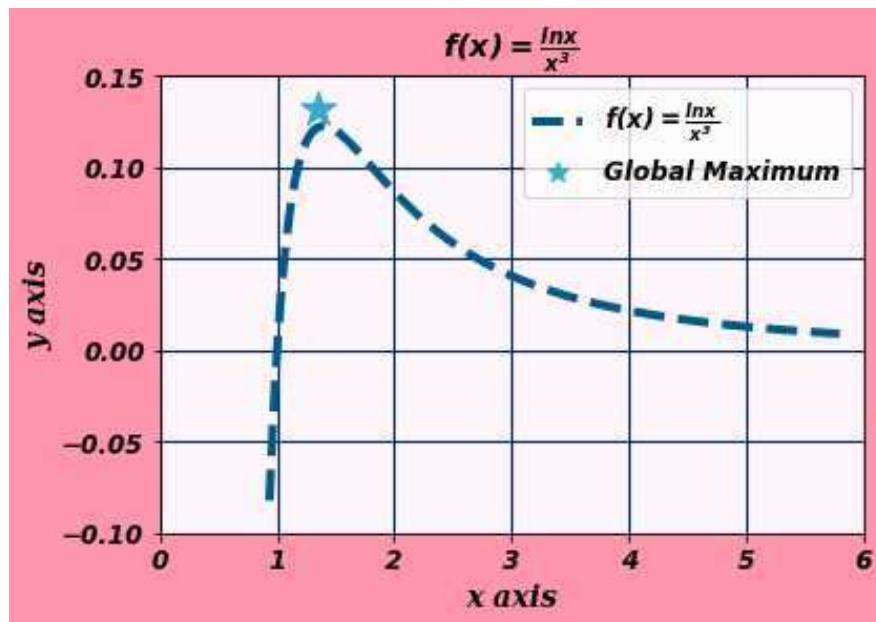


圖 2.8: $f(x) = \frac{\ln x}{x^3}$

- Domain : $\{x \in \mathbb{R} \mid x > 0\}$
- Range : $\{f \in \mathbb{R} \mid f \leq \frac{1}{3e}\}$
- Global maximum : 函數經過一階導數判斷後有 $\max \left\{ \frac{\ln x}{x^3} \right\} = \frac{1}{3e}$ 在 $x = \sqrt[3]{e}$
- 根據定義域及值域可以發現圖形有兩條漸近線 $x = 0$ 與 $y = 0$ 。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.lines as mlines
4
5 x = np.linspace(0.935, 5.8, 200)
6 y = np.divide(np.log(x), x**3)
7 fig, ax = plt.subplots(1)
8 fx_name = r'$f(x)=\frac{\ln x}{x^3}$'
9 line1, =ax.plot(x, y, color='#005A8D',

```

```

10 linewidth = 4, linestyle ='--',label=fx_name)
11 plt.title("Interactive Plot", labelparams)
12 bbox_props1= dict(boxstyle='round',pad=0.3', ec='b', lw=2,
13 fc='lightblue')
14 plt.title(r'$f(x)=\frac{\ln x}{x^3}$')
15 blue_star = mlines.Line2D([], [], color='#39A9CB',
16 marker='*', linestyle='None', markersize=10,
17 label='Global Maximum')
18 plt.scatter(1.35,0.1316, s=300, c='#39A9CB',marker="*")
19 plt.show()

```

Code 2.8: $f(x) = \frac{\ln x}{x^3}$

- 此圖形我認為不好畫，因為如果想把 Y 軸漸近線畫出來，會導致左方線段過於下面，使整體比例跑掉。
- 5-7 行：`plt.rcParams()` 是自定義樣式控制，它可以控制 matplotlib 中幾乎每個屬性的預設值，而我只想更改圖中的字形，所以先定義第 4 行的 `fontparams`，再將其 update 回 `plt.rcParams()` 即可。

(9) Sketch the graph of $f(x) = 3, 1 \leq x \leq 5$

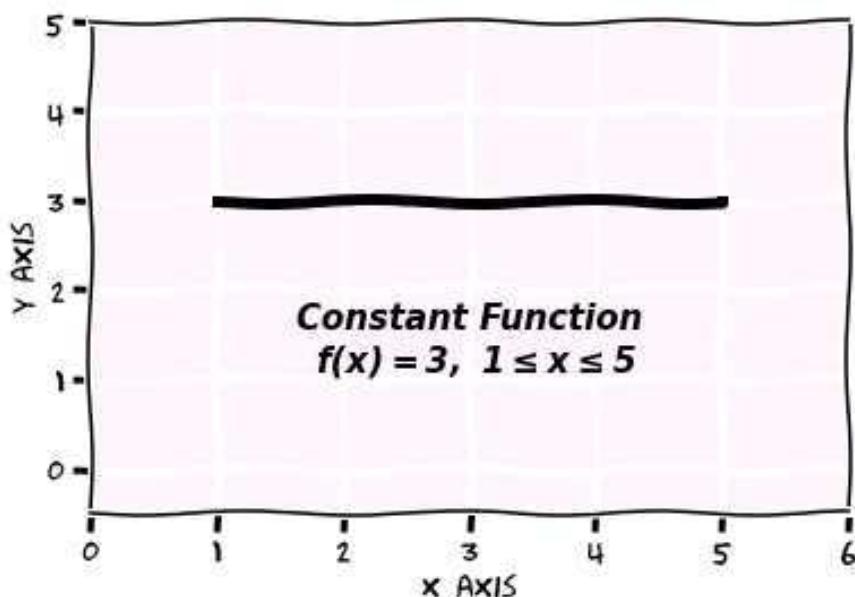


圖 2.9: $f(x) = 3, 1 \leq x \leq 5$

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x=np.linspace(1,5,200)
5 with plt.xkcd():
6     plt.plot(x, 3*(x**0), label ='Line1',
7               color ='black', linestyle ="-", linewidth=5)
8 ax = plt.gca()
9 bbox_props =dict(boxstyle='round4',pad=1,rounding_size=0.2',
10 ec='black', fc='#EEEEFF', lw=5)
11 plt.xticks(np.array([-1.5, -1.0 ,-0.5 ,0 , 0.5, 1.0 ,1.5]))
12 plt.yticks(np.array([-1.5, -1.0 ,-0.5 ,0 , 0.5, 1.0 ,1.5]))
13 plt.fill_between(x, y, facecolor ='#be003f')
14 ax.patch.set_facecolor('white')
15 plt.axis("equal")
16 phi = np.linspace(0, 2*np.pi, 200)
17 r = 1
18 x = r*np.cos(phi)
19 y = r*np.sin(phi)
20 plt.plot(x,y,linewidth=2.5,color ='#be003f')
21 ax = plt.gca()
22 plt.text(0,-0.2, 'Japan', ha='center',weight='heavy'
23 ,fontsize=90)
24 plt.title(r'$x^2+y^2=1$')
25 plt.text(3, 1.1, 'Constant Function\n $f(x)=3,\backslash 1\leq x\backslash leq5$', fontsize=15, color='black', ha='center')
26 plt.show()

```

Code 2.9: $f(x) = 3, 1 \leq x \leq 5$

此圖使用 xkcd¹模組來繪圖，matplotlib 的 xkcd 模式可以自動將圖形轉為手繪卡通圖的風格，所以專業文章可能用不上。

¹xkcd 是由蘭德爾·門羅創作的網絡漫畫。作者對其的描述是「關於浪漫、諷刺、數學和語言的網絡漫畫」，曾獲得網絡漫畫家選擇獎等獎項。

(10) Sketch the graph of $x^2 + y^2 = 1$

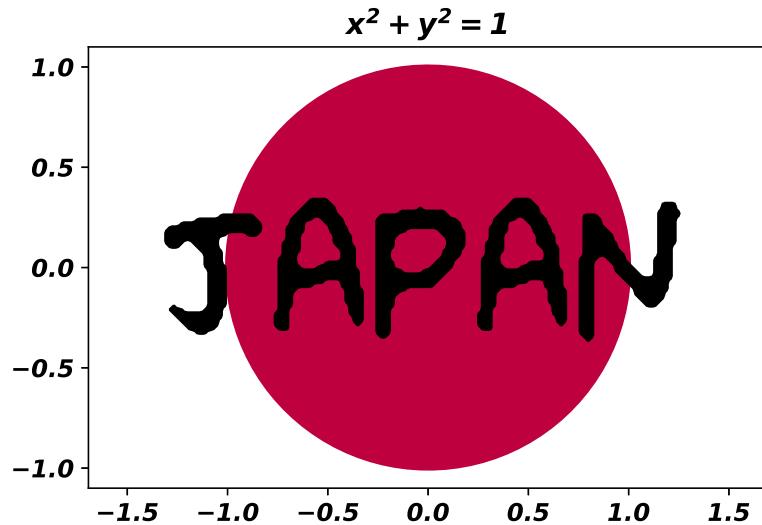


圖 2.10: $x^2 + y^2 = 1$

$x^2 + y^2 = 1$ 為中心點在 $(0, 0)$ 且半徑為 $r = 1$ 的圓。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 phi = np.linspace(0, 2*np.pi, 200)
5 r = 1
6 x = r*np.cos(phi)
7 y = r*np.sin(phi)
8 plt.plot(x,y,linewidth=2.5,color="#be003f")
9 ax = plt.gca()
10 plt.text(0,-0.2, 'Japan', ha='center',weight='heavy'
11 ,fontsize=90)
12 plt.xticks(np.array([-1.5, -1.0 ,-0.5 ,0 , 0.5, 1.0 ,1.5]))
13 plt.yticks(np.array([-1.5, -1.0 ,-0.5 ,0 , 0.5, 1.0 ,1.5]))
14 plt.fill_between(x, y, facecolor='#be003f')
15 ax.patch.set_facecolor('white')
```

```

16 plt.axis("equal")
17 phi = np.linspace(0, 2*np.pi, 200)
18 r = 1
19 x = r*np.cos(phi)
20 y = r*np.sin(phi)
21 plt.plot(x,y,linewidth=2.5,color="#be003f")
22 ax = plt.gca()
23 plt.text(0,-0.2, 'Japan', ha='center',weight='heavy'
24 ,fontsize=90)
25 plt.title(r'$x^2+y^2=1$')
26 plt.show()

```

Code 2.10: $x^2 + y^2 = 1$

- 此圖的繪製方法為將單位圓的公式轉換成正餘弦函數來表達。
- 4 行：定義 phi 為 $0 \sim 2\pi$ 的值。
- 6 行：將 x 以餘弦函數表達。
- 7 行：將 y 以正弦函數表達。

(11) Sketch the graph of a square of side 1

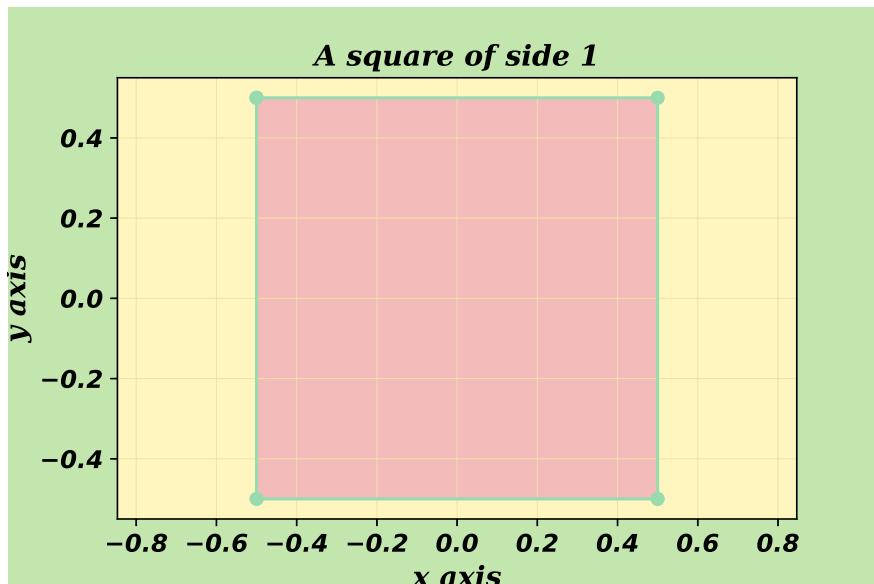


圖 2.11: square of side 1

《Python Code》

```

1 import matplotlib.pyplot as plt
2
3 #data
4 x = [-0.5, -0.5, 0.5, 0.5, -0.5]
5 y = [0.5, -0.5, -0.5, 0.5, 0.5]
6 fig = plt.figure()
7 ax = plt.gca()
8 plt.rcParams.update(fontparams)
9 plt.xlabel('x axis', labelparams)
10 plt.ylabel('y axis', labelparams)
11 plt.plot(x, y, 'o-', c='#97DBAE')
12 plt.fill_between(x, y, facecolor='#F4BBBB')
13 plt.axis("equal")
14 plt.title(r'A square of side 1', labelparams)
15 plt.show()

```

Code 2.11: square of side 1

- 4 與 5 行：手動輸入 x 值和對應的 y 值。
- 11 行：`plt.plot()` 中的參數 '`o-`' 可以將繪製的四個點連線並且標示實心點。
- 12 行：`plt.fill_between (x, y)` 可以將界於 x 和 y 內的圖形塗色，也就是正方體本身。

2.2 Project : Comparison Theorem

$$\text{let } S_n = \sum_{k=1}^n \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

a. Verify that $\lim_{n \rightarrow \infty} S_n$ diverges.

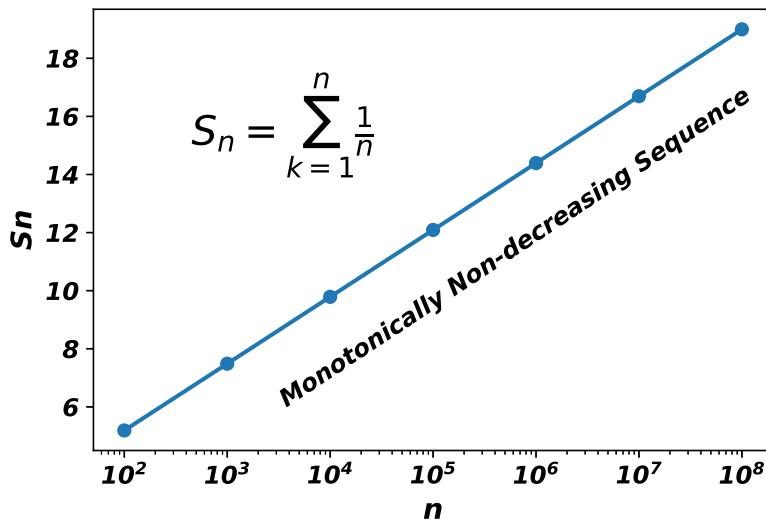


圖 2.12: Project : Comparison Theorem Part a

可以看出當 n 很大時，級數和 S_n 並未收斂。因此可以驗證 S_n 屬於發散級數。

《Python Code》

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 n = np.logspace(2.0, 8.0, num=7)
5 print(n)
6 S_n = []
7 for i in range(7):
8     S_n.append(np.sum(1/np.arange(1,n[i]+1)))
9 print(S_n)
10 ax = plt.gca()
11 # Set x logarithmic
12 ax.set_xscale('log')
13 plt.plot(n,S_n,'-o', linewidth=2)
14 plt.xlabel('n',labelparams)
15 plt.ylabel('Sn',labelparams)
16 plt.text(10**3.5,15, r'$\sum_{k=1}^n \frac{1}{k}$',
17 ha='center', weight='heavy', fontsize=20)

```

```

18 plt.text(10**5.8, 6, 'Monotonically Non-decreasing Sequence',
19 ha='center', weight='heavy', rotation=33.5, fontsize=12)
20 plt.show()

```

Code 2.12: Project : Comparison Theorem Part a

4-9 行為數值計算的程式碼。

b. Let γ_n denote the sum of the shaded areas. Show that $\gamma_n = S_n - \ln(n + 1)$.*proof :*

$$\begin{aligned}
\gamma_n &= \sum_{k=1}^n \int_{x=k}^{k+1} \left(\frac{1}{k} - \frac{1}{x}\right) dx \\
&= \sum_{k=1}^n \left(\int_k^{k+1} \frac{1}{k} dx - \int_k^{k+1} \frac{1}{x} dx \right) \\
&= \sum_{k=1}^n \left(\frac{k+1}{k} - 1 - (\ln|k+1| - \ln|k|) \right) \\
&= \sum_{k=1}^n \left(\frac{1}{k} + \ln|k| - \ln|k+1| \right) \\
&= \sum_{k=1}^n \frac{1}{k} + \sum_{k=1}^n (\ln|k| - \ln|k+1|) \\
&= S_n + \ln 1 - \ln 2 + \ln 2 - \ln 3 \dots \ln(n) - \ln(n+1) \\
&= S_n - \ln(n+1)
\end{aligned}$$

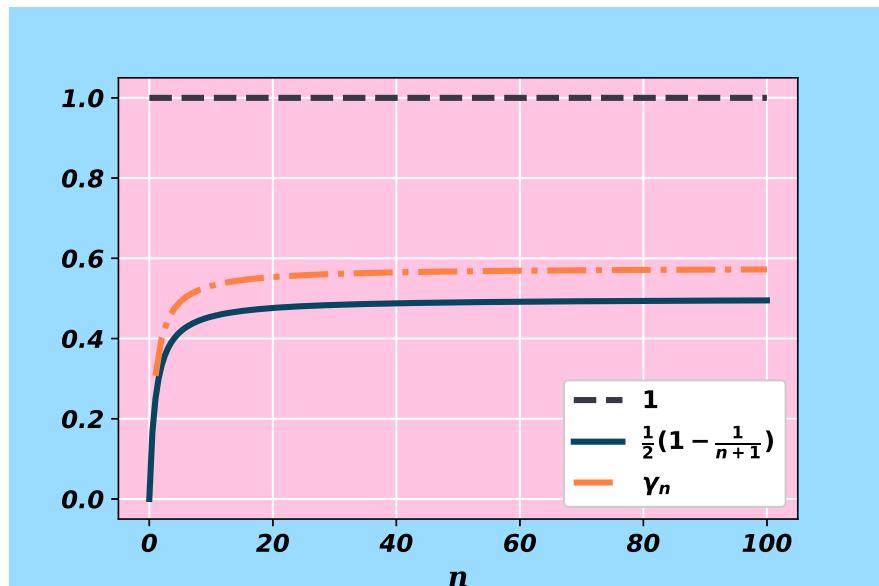
c. Verify that $\frac{1}{2}(1 - \frac{1}{n+1}) < \gamma_n < 1$.

圖 2.13: Project : Comparison Theorem Part c

《Python Code》

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 n = np.arange(1, 101, 1)
5 S_n=[]
6 print(n)
7 for i in range(100):
8     S_n.append(np.sum(1/np.arange(1,n[i]+1))-np.log(n[i]+1))
9 x1 = np.linspace(0, 100,200)
10 y1 = x1**0
11 x2 = np.linspace(0, 100,200)
12 y2 = 0.5*(1-np.divide(1,x2+1))
13 x3 = n
14 y3 = S_n
15 fig = plt.figure()
16 ax = plt.gca()
17 plt.plot(x1, y1, '--',color="#3A3845",
18 label=r'$1$',
19 linewidth=3)
20 plt.plot(x2, y2, '-.',color="#064663",label=r'$\frac{1}{n+1}$',
21 (1-frac{1}{n+1}),'-.',color="#FF8243",
22 label=r'$\gamma_n$',
23 linewidth=3)
24 plt.xlabel('n',labelparams)
25 plt.legend()
26 plt.grid(visible = True, color='w', linestyle='--',
27 linewidth=1.0)
28 plt.show()
```

Code 2.13: Project : Comparison Theorem Part c

2.3 結論

要完成此篇《函數計算與繪製》除了需要一點編程能力，還需要一定的數學分析能力，因為做函數圖形最重要的是先將該函數的特性清楚表達出來，接著才是運用 matplotlib 來美化函數圖。而最難的部分則是 Project，因為它不是簡單給定 $f(x)$ 函數就能畫得出來，我反而花了很多時間在想如何有效率的把數值呈現，後來發現用 Numpy 的套件來運算速度很快，所以熟悉 Numpy 相當重要。

第 3 章

分配函數、亂數、抽樣分配的程式與繪圖

本文主要是紀錄統計學中常見的分配及其實作，其中包括函數圖形的繪製、亂數生成、抽樣分配方法與檢驗，而這些內容是在幫助我們將統計理論與實踐結合，我認為相當有趣。

3.1 分配函數

在統計學中描述所有樣本空間中的樣本點 (sample point) 或是特定種類之事件所發生之機率的函數即稱為機率分配。以下將從連續型的機率分配開始介紹。

3.1.1 連續型機率分配

i. 卡方分配 (*chi-square distribution*)

若連續隨機變數 X 的機率密度函數為

$$f(x) = \frac{1}{2^{\frac{k}{2}} \Gamma\left(\frac{k}{2}\right)} x^{\frac{k}{2}-1} e^{-\frac{x}{2}}, \quad x > 0$$

則稱 X 服從卡方分配，其分配參數為 K ，可表示為 $X \sim \chi^2(k)$ 。

卡方分配是 Gamma 分配的一個特例。若 $X \sim \chi^2(k)$ ，則 $X \sim \text{Gamma}(\alpha = k/2, \beta = 2)$ ，其中參數

(a) α 稱為形狀參數，決定 Gamma 分配的形狀， α 越大則分配越接近對稱分配。

(b) β 稱為尺度參數，決定 Gamma 分配的分散程度， β 越大則分配越分散。

可知卡方分配的圖形分散程度固定，因 β 已被固定為 2。

另，卡方分配的參數 K (自由度) 必須為正整數。圖 3.1 介紹不同自由度下的卡方分配圖形。

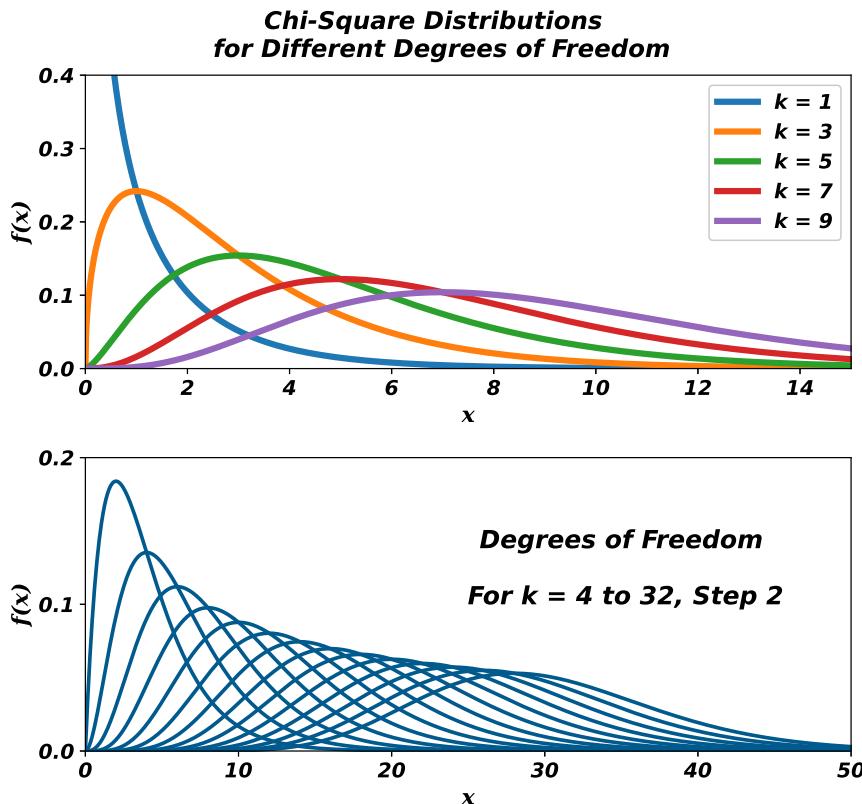


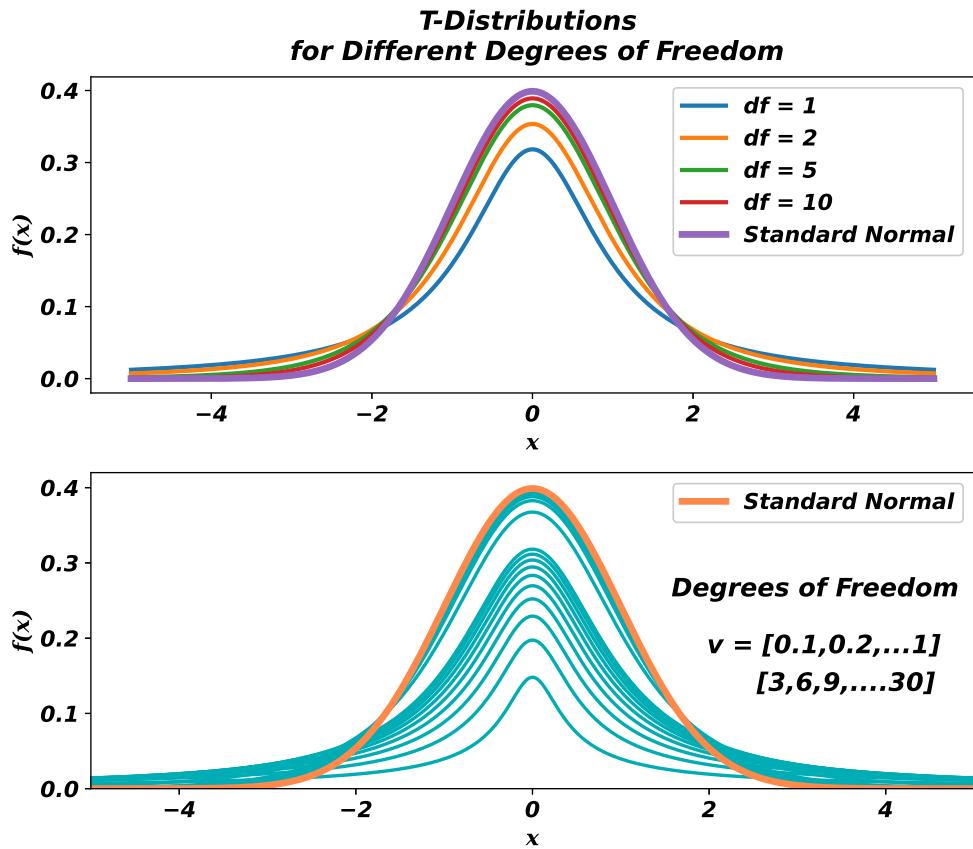
圖 3.1: 不同自由度下的卡方分配圖

ii. t 分配 (t distribution)

若連續隨機變數 X 的機率密度函數為

$$f(x) = \frac{\Gamma((\nu + 1)/2)}{\sqrt{\nu\pi} \Gamma(\nu/2) (1 + x^2/\nu)^{(\nu+1)/2}}, \quad -\infty < x < \infty$$

則稱 X 服從 t 分配，其分配參數為 v ，可表示為 $X \sim t(v)$ 。當 t 分配的自由度 v 越大時，其分配越接近標準常態分配。圖 3.2 介紹不同自由度下的 t 分配圖。

圖 3.2: 不同自由度下的 t 分佈圖

iii. Beta 分配 (Beta distribution)

若連續隨機變數 X 的機率密度函數為

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \text{ where } B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, x \in [0, 1]$$

則稱 X 服從 Beta 分配，其分配參數為 α 與 β ，可表示為 $X \sim Beta(\alpha, \beta)$ 。

因為 Beta 的值域，其主要應用是描述介於 $[0, 1]$ 的事物，例如機率與財務相關的折現因子等。

圖 3.3，畫出了給定不同 α 與 β 下的 Beta 分配之 pdf。可以發現：

- (a) 若 $\alpha > \beta$ ，則分配有左長尾。
- (b) 若 $\alpha < \beta$ ，則分配有右長尾。
- (c) 若 $\alpha = \beta$ ，則分配為對稱。

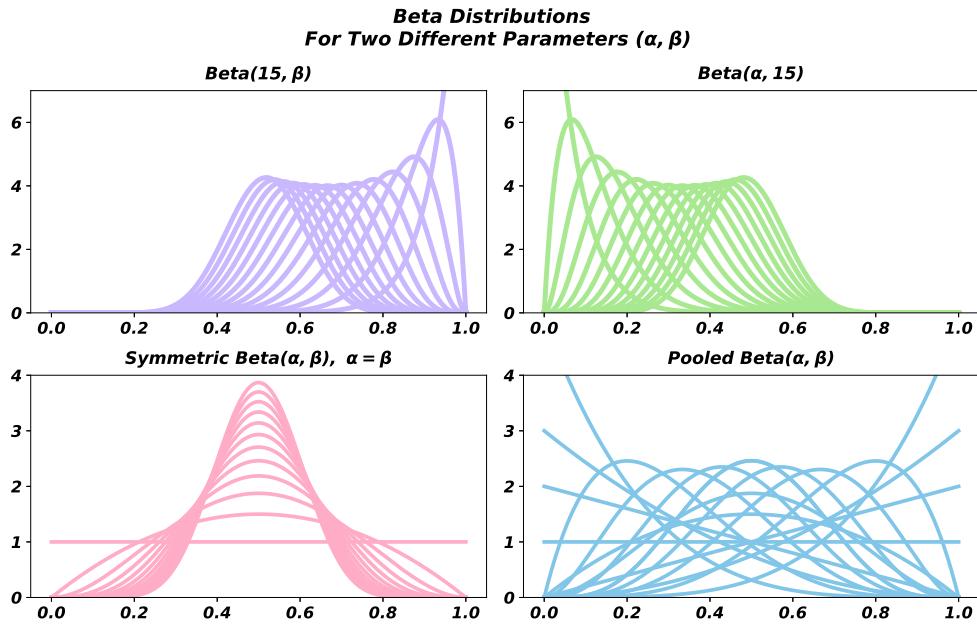


圖 3.3: 不同參數下的 Beta 分配圖

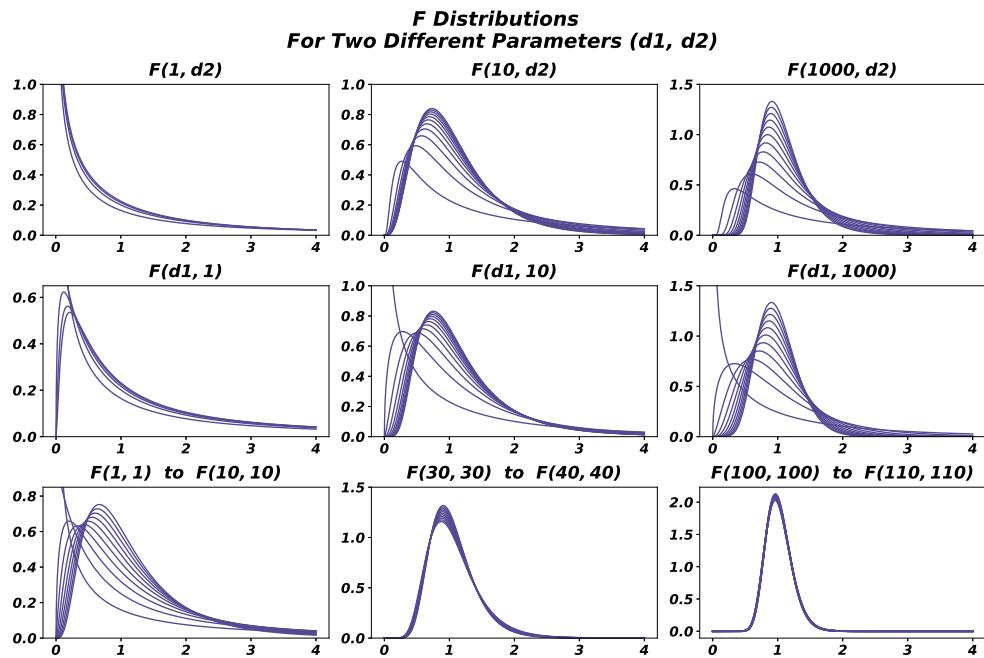
iv. F 分配 (*F distribution*)

若連續隨機變數 X 的機率密度函數為

$$f(x) = \frac{1}{B\left(\frac{d_1}{2}, \frac{d_2}{2}\right)} \left(\frac{d_1}{d_2}\right)^{d_1/2} x^{d_1/2-1} \left(1 + \frac{d_1}{d_2} x\right)^{-(d_1+d_2)/2}, \quad x > 0$$

則稱 X 服從 F 分配，其分配參數為 d_1 與 d_2 ，可表示為 $X \sim F(d_1, d_2)$ 。

F 分配為兩獨立卡方分配的比值，其自由度 d_1 與 d_2 分別為兩卡方分配的自由度。圖 3.4 繪出了不同自由度的 F 分配 pdf，隨著 (d_1, d_2) 越大，pdf 越接近對稱分配。

圖 3.4: 不同參數下的 F 分配圖

3.1.2 間斷型機率分配

i *Poisson* 分配 (*Poisson distribution*)

若離散隨機變數 X 的機率密度函數為

$$f(x) = \frac{e^{-\lambda} \lambda^k}{k!}, \quad x = 0, 1, 2, \dots, \infty$$

則稱 X 服從 *Poisson* 分配，其分配參數為 λ ，可表示為 $X \sim Poisson(\lambda)$ 。

Poisson 分配的主要應用是在單位時間或空間內，發生的事件次數，例如一小時內的來客數、一個月內的交通事故次數等。

根據圖 3.5 中分配的外型，*Poisson* 分配具右長尾，但當 λ 很大時分配接近鐘型分配。

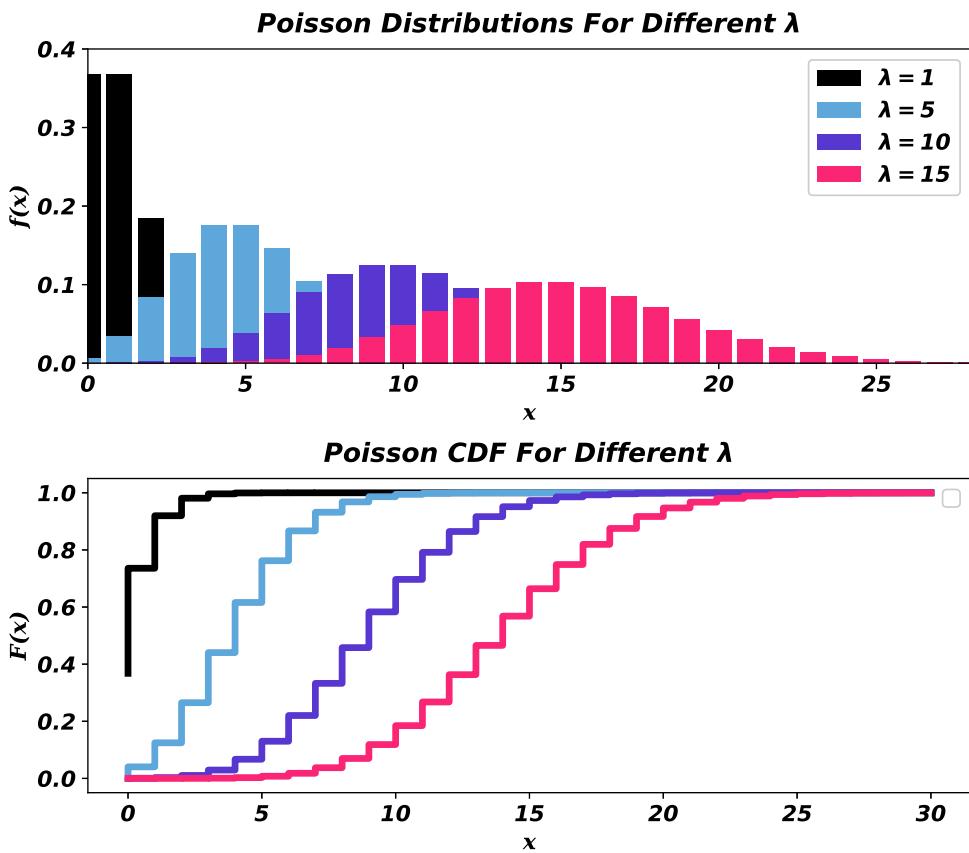


圖 3.5: 不同參數下的 Poisson 分配圖

ii 超幾何分配 (*Hypergeometric distribution*)

若離散隨機變數 X 的機率密度函數為

$$f(x) = \frac{\binom{K}{x} \binom{N-K}{n-x}}{\binom{N}{n}}, \quad x = 0, 1, 2, \dots, n, \quad n \leq K$$

則稱 X 服從超幾何分配，其分配參數為 N 、 K 與 n ，可以表示為 $X \sim Hyper(N, K, n)$ 。

若一隨機試驗中，母體數為 N ，其中有 K 個指定種類的物件。若抽取 n 個物件，令 X 表示指定物件的各數

- (a) 若抽樣方式為抽出不放回，則 $X \sim Hyper(N, K, n)$ 。
- (b) 若抽樣方式為抽出放回，則 $X \sim Binomial(n, p = K/N)$ 。

超幾何分配可視為有限母體下且抽出不放回的 Bernoulli 試驗之和。其分配

外型如圖 3.6，可以看出當 N 固定， k 值越大則分配越靠右側。也就是平均數越大。

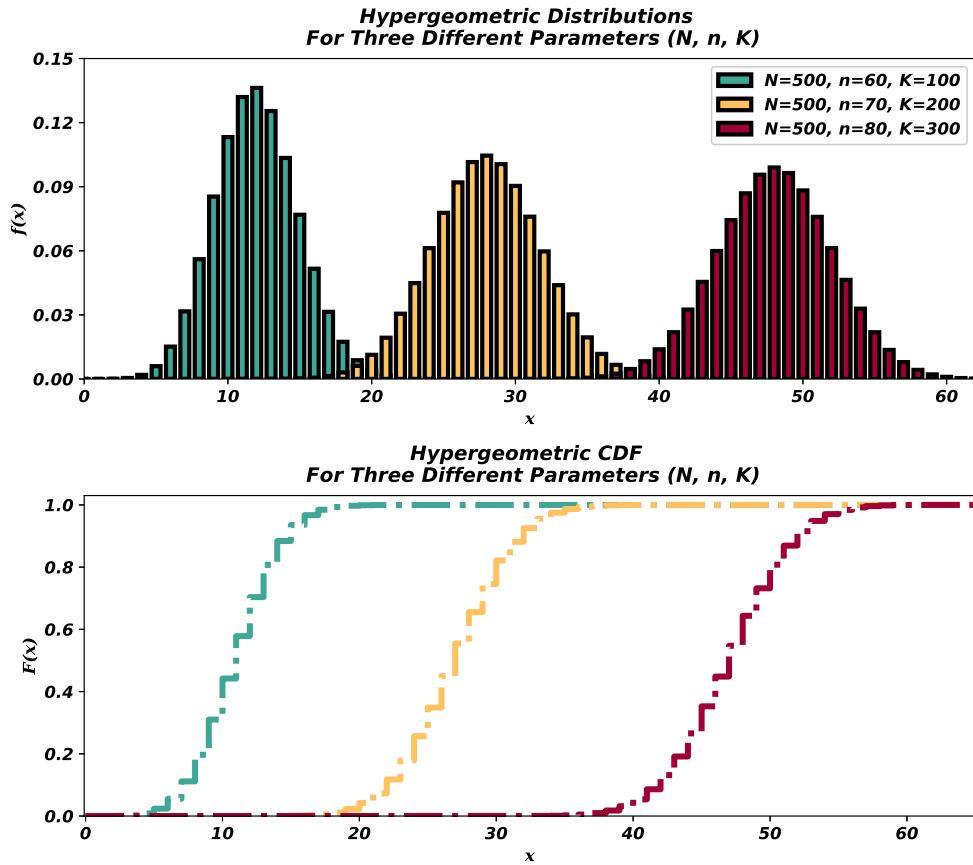


圖 3.6: 不同參數下的超幾何分配圖

3.2 隨機亂數與相關圖形

本節使用 `scipy` 套件生成具有特定分布的隨機亂數，其數據生成的理論是根據 Mersenne Twister (梅森旋轉演算法) 作為核心的產生器，是現存最廣泛被驗證的隨機數產生器之一。

1. 常態分配下的亂數

本小節主要在探討常態分配下的亂數生成 (平均數為 0 且標準差為 1)。圖 3.7 為不同抽樣數量的亂數常態直方圖，可以看出當樣本數量較少時，生成的數據集可能更具有隨機性，但是可能沒有足夠的數據來描述數據分佈。反之樣本數量較多時，生成的數據集可能更具有代表性，能夠更

好地描述數據分佈。

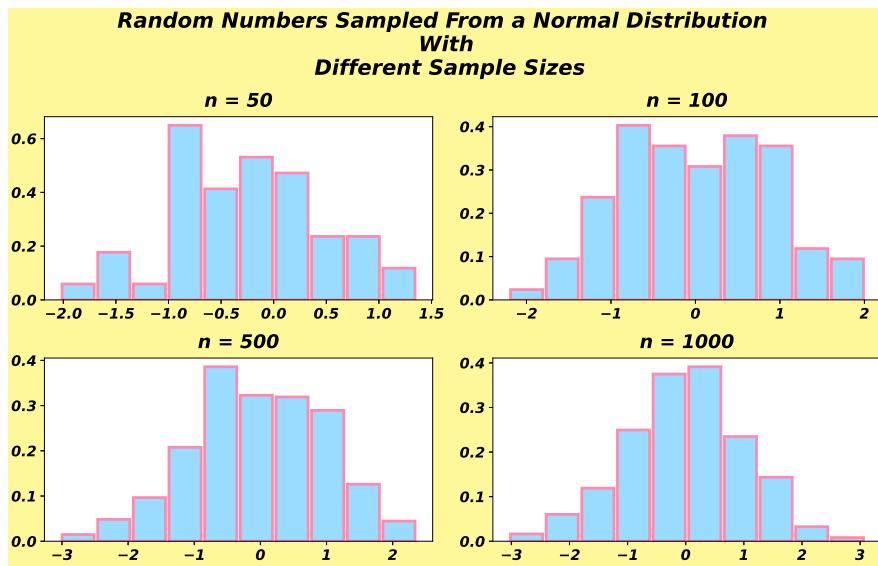


圖 3.7: 不同抽樣數量的亂數常態直方圖

圖 3.8 為樣本數 30 的亂數常態分配相關圖，包含直方圖、盒式圖、Q-Q 圖及 CDF 圖。當抽樣數為 30 時，可以看作小樣本的情況。透過直方圖，可以發現分配的外型並不為鐘型並且平均數也不在 0 的位置，而盒式圖也可以觀察出平均數的問題。接著，透過 Q-Q 圖可以發現樣本點沒有完全貼合於 45 度的直線上，所以由此推斷此抽樣分配並不完全為常態分配。最後經驗 CDF 圖也不完全貼合於經驗 CDF 圖，表示抽樣分配與常態分配有差距。

圖 3.9 將以上小樣本的狀況改變為抽樣數為 1000 的大樣本，試圖改善小樣本抽樣分配與真實常態分配不匹配的結果。最後根據圖 3.9 的四種常態分配相關圖，可以發現每種分析結果都符合預期。也就是大樣本下的抽樣分配比小樣本抽樣分配更接近真實的常態分配。

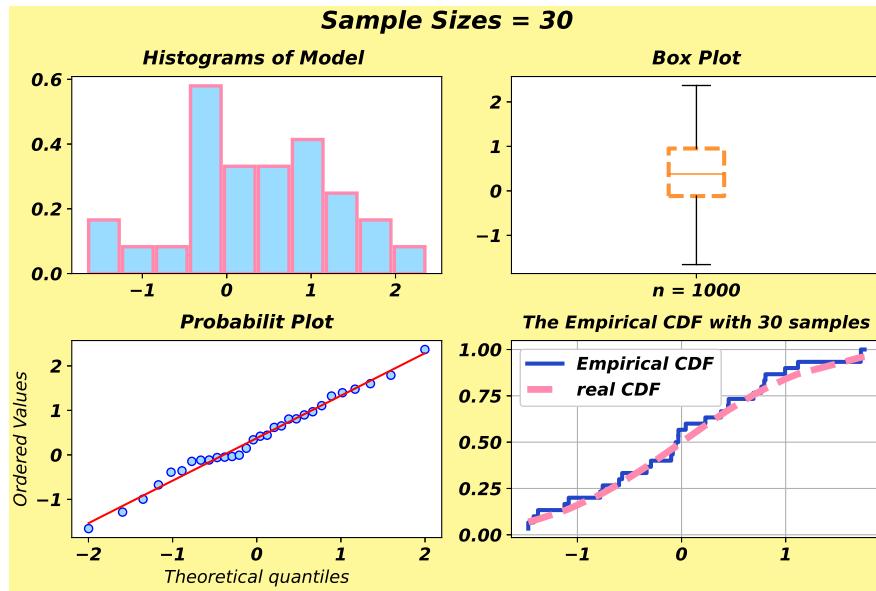


圖 3.8: 樣本數 30 的亂數常態分配相關圖

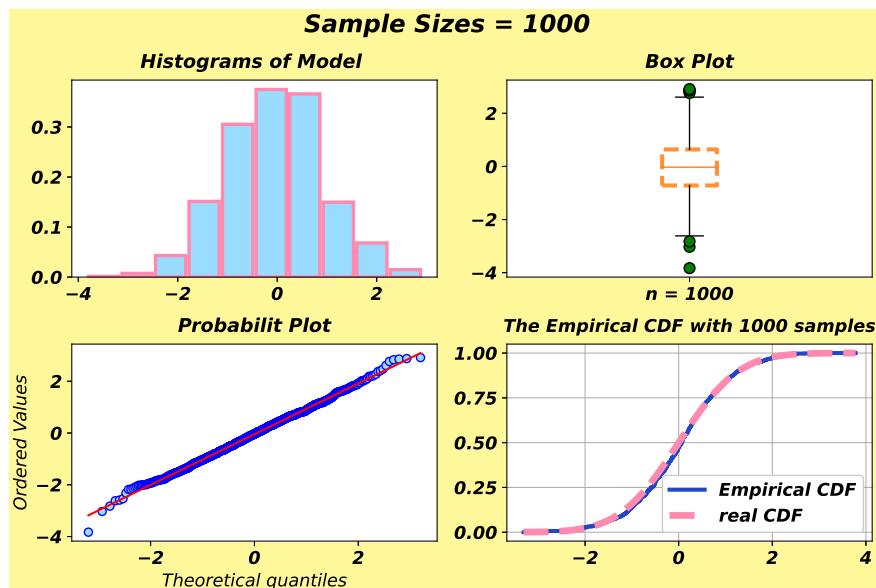


圖 3.9: 樣本數 1000 的亂數常態分配相關圖

2. t 分配下的亂數

本小節主要在探討 t 分配下的亂數生成。圖 3.10 為不同抽樣數量的亂數 t 分配直方圖，可以看出當樣本數量較少時分配形狀不完整。但樣本數量增加後， t 分配外觀愈來愈完整。並且，當樣本數達到 1000 時外觀近似常態分配。

根據以上的結果以及初步判斷後，將樣本數為 30 以及樣本數為 1000 時

的狀況分別討論，如圖 3.11 與圖 3.12。可以發現大樣本時的 t 抽樣分配比小樣本時還接近真實分配。

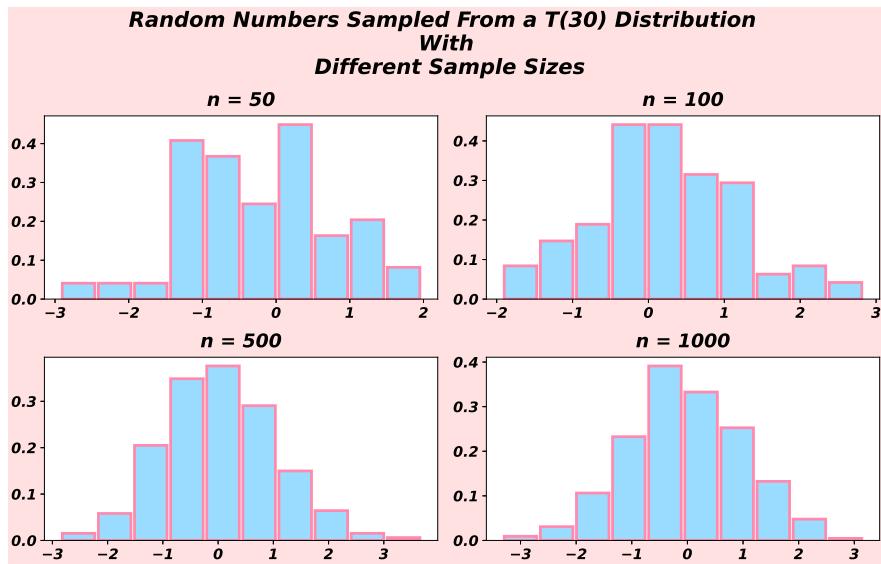


圖 3.10: 不同抽樣數的亂數 t 分配直方圖

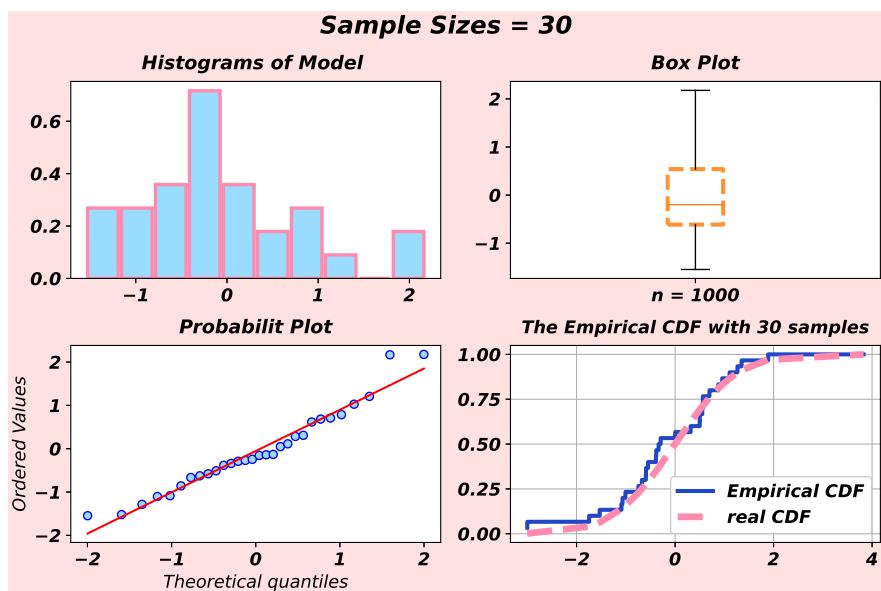
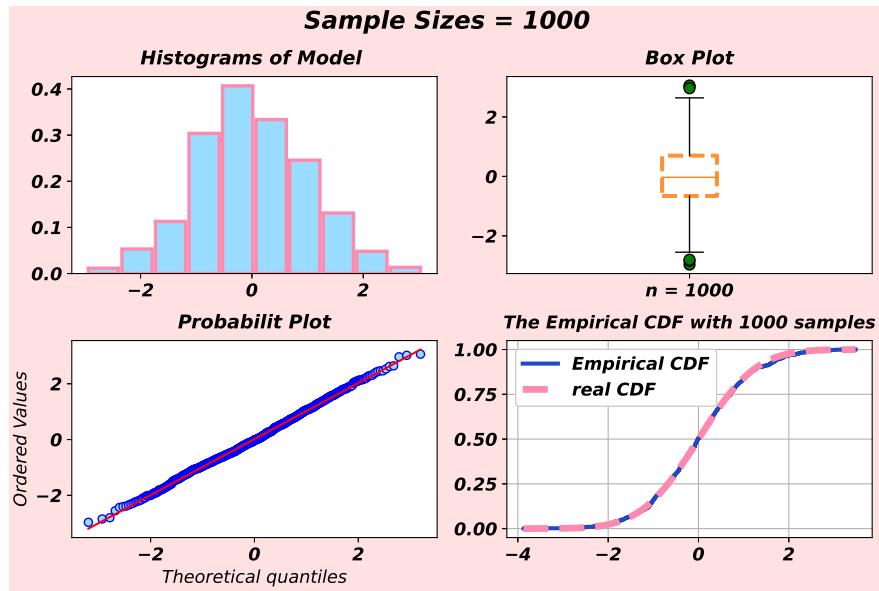


圖 3.11: 樣本數 30 的亂數 t 分配相關圖

圖 3.12: 樣本數 1000 的亂數 t 分配相關圖

3.3 抽樣分配實作

3.3.1 中央極限定理 CLT 應用於 Beta 分配

使用中央極限定理近似 Beta 分佈的常用做法是採用大量獨立同分佈的 Beta 分佈樣本的平均值來近似標準正態分佈。這可以通過以下步驟來實現：

1. 構造 n 個獨立同分佈的 $\text{Beta}(\alpha, \beta)$ 隨機變量 X_1, X_2, \dots, X_n
2. 計算平均值 $Y = (X_1 + X_2 + \dots + X_n)/n$
3. 通過中央極限定理可知 Y 的分佈接近於 $N(\mu, \sigma^2)$ ，其中 $\mu = \alpha/(\alpha + \beta)$, $\sigma^2 = \mu(1 - \mu)/n$

今給定 $\{X_i\}_{i=1}^n \sim \text{Beta}(9, 2)$ ，則根據中央極限定理可知 $\bar{x} \xrightarrow{A} N(9/11, \frac{3/242}{n})$
由於中央極限定理指的是當 n 很大時，隨機變量的平均的分佈近似於常態分佈，因此，當 n 越大時，近似的精度就越高。這就是為什麼圖 3.14 Beta 平均樣本抽樣數為 100 時的外型比圖 3.13 平均樣本抽樣數為 30 還更趨近常態分配。

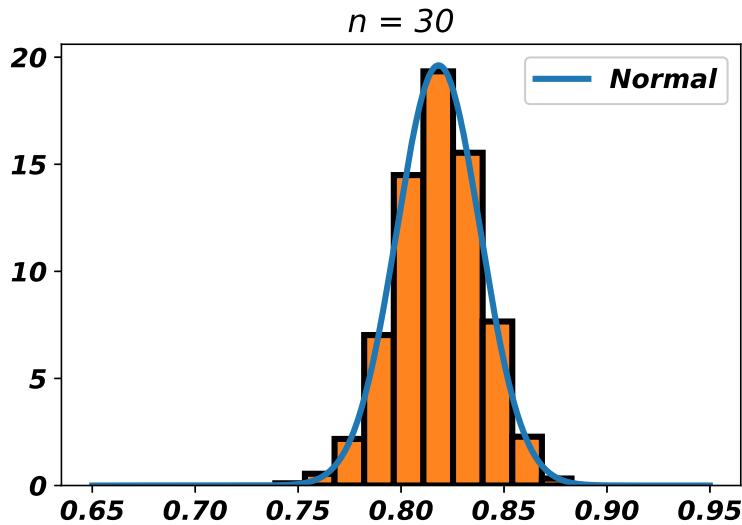


圖 3.13: Beta 平均樣本抽樣數為 30 下的中央極限定理結果

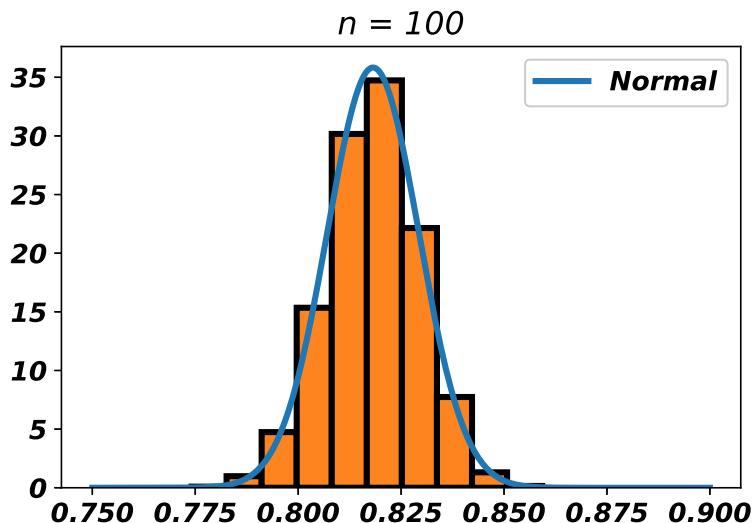


圖 3.14: Beta 平均樣本抽樣數為 100 下的中央極限定理結果

3.3.2 極小值幾何分配

如果有兩個離散隨機變量 X_1 和 X_2 ，分別服從平均為 $1/p_1$ 與 $1/p_2$ 的幾何分配，那麼它們的極小值 $\text{Min}(X_1, X_2)$ 依然服從幾何分配。這是因為幾何分配是一種離散型分佈，它描述的是第一次成功之前經過的嘗試次數，而不是累計成功次數。因此，對於兩個獨立的幾何分配變量，它們第一次成功的時

間點是獨立的，而第一次成功發生在兩個變量中最早的那個時間點恰好就是 $\text{Min}(X_1, X_2)$ 。極小值是由兩個獨立的幾何分配變量中第一次成功發生的時間點確定的，所以 $\text{Min}(X_1, X_2)$ 仍然服從平均為 $1 - (1 - p_1)(1 - p_2)$ 的幾何分配。並且，任意個幾何分配的變數的極小仍然服從幾何分配。

今給定 $X_1 \sim \text{Geo}(p_1), X_2 \sim \text{Geo}(p_2)$ ，則

$$\text{Min}(X_1, X_2) \sim \text{Geo}(1 - (1 - p_1)(1 - p_2))$$

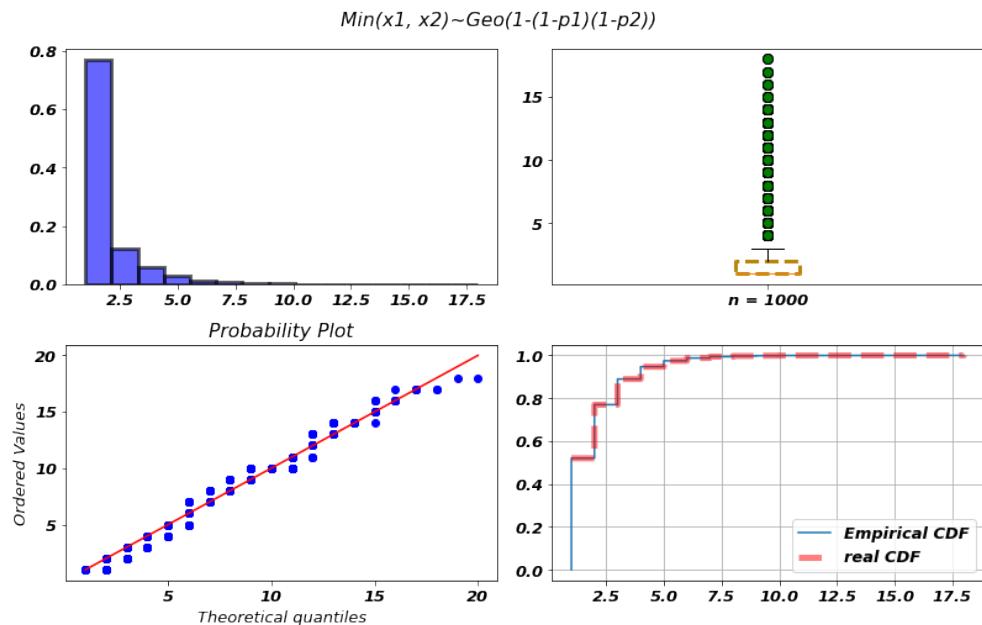


圖 3.15: 極小值幾何分配抽樣數為 1000 下結果

極小值幾何分配經過 1000 次的抽樣後，相關分析結果如圖 3.15。透過直方圖確認分配形狀確實為右偏的幾何分配，而 Q-Q 圖上的點沒有很貼合於 45 度線，因此 Q-Q 無法驗證數據是否確實遵循理論分配。然而，經驗累積機率密度函數圖與極小值幾何分配的真實累積機率密度函數非常相似。也就是抽樣分配的分佈與理論分配的差異很小。

3.4 專題一數字抽取

給定四個數字 (2, 4, 9, 12)。從這四個數字中隨機抽取四個數字 (取後放回) 計算其平均數並估計出這些機率值。

計算這四個數字 (2, 4, 9, 12) 的平均值可以使用 numpy 中的 mean 函數。抽樣部分可以通過使用 numpy.random.choice() 函數從給定數字 (2, 4, 9, 12) 中模擬大量隨機抽取 4 個數字 (取後放回)，並計算這 4 個數字的平均值來實現。其最終的抽樣結果與估計的機率值如圖 3.16。

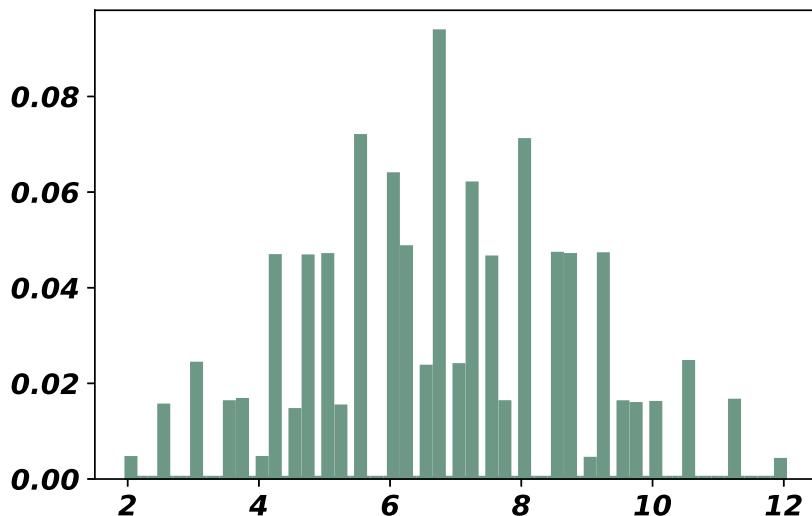


圖 3.16: 專題

《Python Code》

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 sample = [2, 4, 9, 12]
4 sampling = np.random.choice(sample , size = (100000,4))
5 y = np.mean(sampling , axis = 1)
6 weights = np.ones_like(y) / len(y)
7 plt.hist(y ,bins = 100 , weights = weights)
```

第 4 章

機器學習中的迴歸模型與公式原理

本文主要介紹簡單線性迴歸模型、加廣型迴歸模型與邏輯斯迴歸模型背後的數學原理，並透過 Python 進行簡易的分類器實作。前兩種迴歸模型分別使用五種相同的資料集進行分類，目的為比較兩種分類器的分類正確率及使用的時機；而邏輯斯迴歸模型則是進行了三元分類的實作，學習程式如何將資料集分隔成三區。

4.1 回歸 (Regression)

迴歸是來自統計學的方法，在機器學習領域中同樣重要。統計學中，它主要用於預測數值型目標變量，並研究其與其他變量之間的關係。在機器學習中，迴歸可以用來預測數值型目標變量，並用於預測分析、多元分析等方面，根據不同的目的性質和應用場景而產生不同的發展視角。一般的狀況來說，迴歸與分類是兩種不同的預測方式，迴歸屬於連續型的模型，也就是說迴歸一般不會用在分類問題上，但如果仍要使用則可以透過以下兩種方式

- (1) 如果輸出資料屬於類別資料時，例如，資料間有群組 A 及群組 B，可以透過調整的方式，將輸入資料屬於群組 A 及群組 B 改寫為 $Y = 0$ 與 $Y = 1$ 輸出。因此，類別資料量化後的問題便可加入到線性迴歸中使用。
- (2) 使用邏輯斯迴歸使得迴歸可以用來處理二元或多元分類問題。

4.2 簡單線性迴歸 (Simple Linear Regression)

假設共有 N 筆已知的輸入及輸出資料， $[x_1(i), x_2(i)]$ 與 $y(i)$ ，我們可以使用一個線性方程式

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \quad (4.1)$$

來表達因變量 (Y) 和一個或多個自變量 (X) 之間的關係。而迴歸係數 $\beta_0, \beta_1, \beta_2$ 可以透過最小平方法求得最佳解

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (4.2)$$

其中

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix}, \quad X = \begin{bmatrix} 1 & X_1(1) & X_2(1) \\ 1 & X_1(2) & X_2(2) \\ \vdots & \vdots & \vdots \\ 1 & X_1(N) & X_2(N) \end{bmatrix}, \quad y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad (4.3)$$

分別代表迴歸模型的參數估計、輸入及輸出資料矩陣。其中假設式 (4.2) 中的正規方程式 ($X^T X$) 的反矩陣存在，而每個輸出值 $y(i)$ 為類別資料 0 或 1。當迴歸模型的參數估計 $\hat{\beta}$ 完成估計時，代表此模型建構完成，接下來就能加入新的資料進行預測。規則如下：

給定一組新的資料 $x = (x_1, x_2)$ ，根據迴歸模型 (4.1) 其輸出的配適值為：

$$\hat{y} = x^T \hat{\beta}$$

然而，在迴歸模型下的配適值 \hat{y} 為一組連續型的數值，不一定為 0 或 1，所以若想使用迴歸來進行分類，則必須將配適值用以下模型改寫：

$$G = \begin{cases} GroupA & \text{if } \hat{y} \leq 0.5 \\ GroupB & \text{if } \hat{y} > 0.5 \end{cases}$$

上述規則將二維平面以 $\hat{y} = x^T \hat{\beta} = 0.5$ 分割為兩群，分別以 Group A 和 Group B 表示。

4.2.1 實作一以簡單線性迴歸進行二元分類

(a) 兩群資料—距離較近

圖 (4.1) 中兩群資料模擬自獨立雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3.5 \\ 2 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}$$

$$n_1 = 200, n_2 = 200$$

將兩群資料模擬的較為靠近且變異程度適中，可看出交界處的兩群多處重疊在一起，導致簡單線性迴歸線不易將兩群分隔開並分群，但是其餘資料則是很良好的被分群。

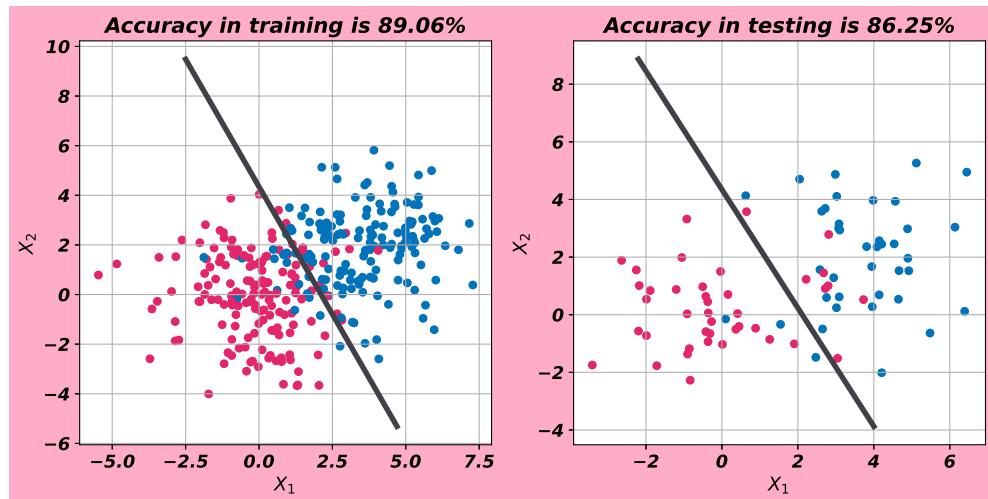


圖 4.1: 簡單線性迴歸—兩群資料距離較近

(b) 兩群資料—距離較遠

圖 (4.2) 兩群資料模擬自獨立雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}$$

$$n_1 = 200, n_2 = 200$$

此兩群模擬資料的變異程度與圖 (4.1) 中所使用的資料相同，為一差別為兩群平均數差異較大，導致其分隔的較遠。這也意味著重疊的部分會減少

許多，使得簡單線性迴歸線可以有效的將兩群分開，若參考圖 (4.2) 中左側的訓練資料，可以發現其建模正確率高達 96.56%。

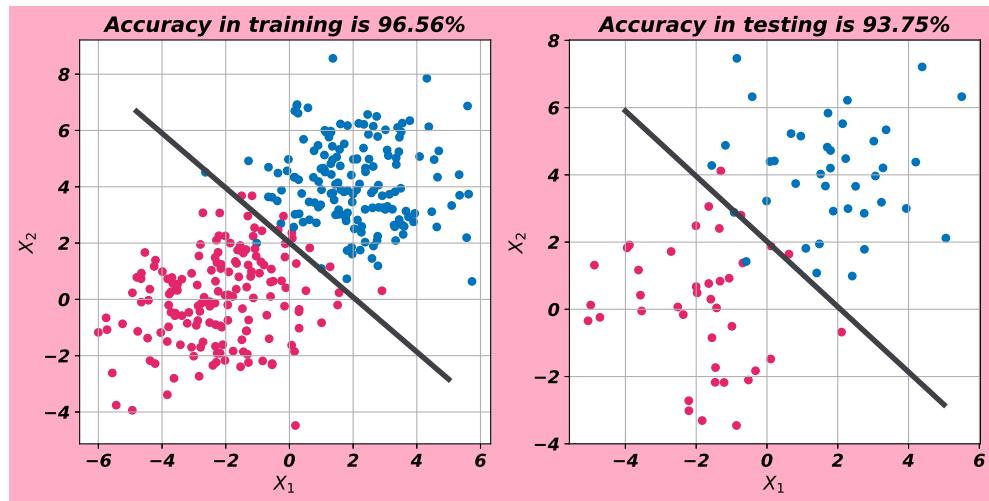


圖 4.2: 簡單線性迴歸—兩群資料距離較遠

(c) 兩群資料—變異程度較小

圖 (4.3) 兩群資料模擬自獨立雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$$

$$n_1 = 200, n_2 = 200$$

根據獨立雙變量常態分配的特性，若母體的變異程度較小，則其樣本大多落於資料的中心，且分部的範圍也較小，但這些特性也可能造成模擬出來的資料多處重疊在一起，使得簡單線性迴歸線無法有效的將兩群分開。

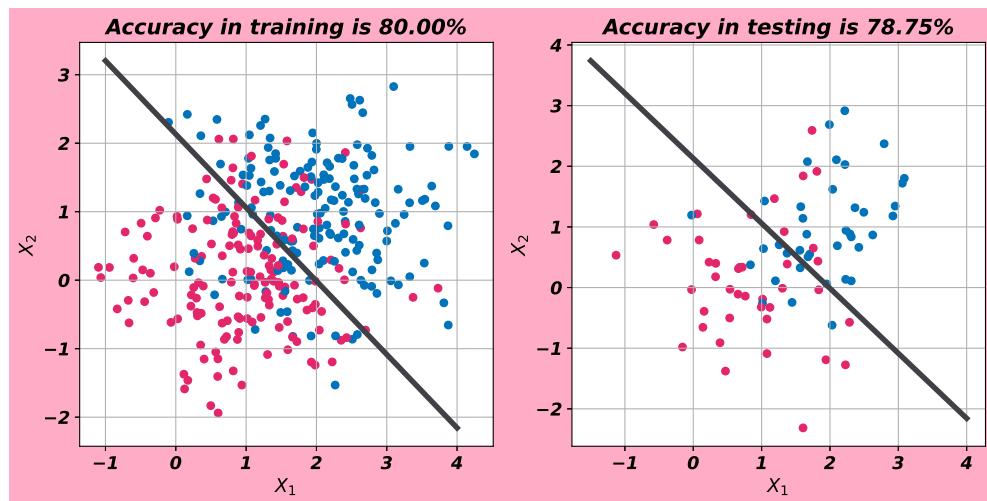


圖 4.3: 簡單線性一兩群資料變異程度較小

(d) 兩群資料—變異程度較大

圖 (4.4) 兩群資料模擬自獨立雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 10 & 0 \\ 0 & 8 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 10 & 0 \\ 0 & 8 \end{bmatrix}$$

$$n_1 = 200, n_2 = 200$$

根據獨立雙變量常態分配的特性，若母體的變異程度較大，則其樣本的分散程度也較大，使得兩群資料在重合處並無高度重疊，且因為資料較分散的特性，比較於圖 (4.3) 中變異程度較小的資料，可以有更良好的分群。

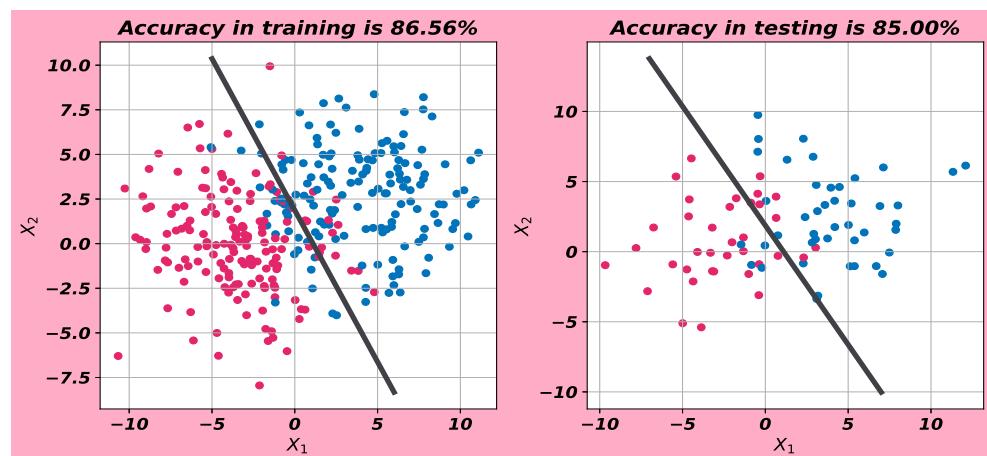


圖 4.4: 簡單線性一兩群資料變異程度較大

(e) 兩群資料—變異程度及樣本數相異

圖 (4.5) 兩群資料模擬自獨立雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix}$$

$$n_1 = 100, n_2 = 200$$

可以明顯看出兩群資料的分散狀況不同，母體變異數小的樣本 (左側) 分布較為密集，母體變異數大的樣本 (右側) 分布較為分散。而簡單線性迴歸線雖然已能將兩群分隔開，但是在此使用曲線可以更好。

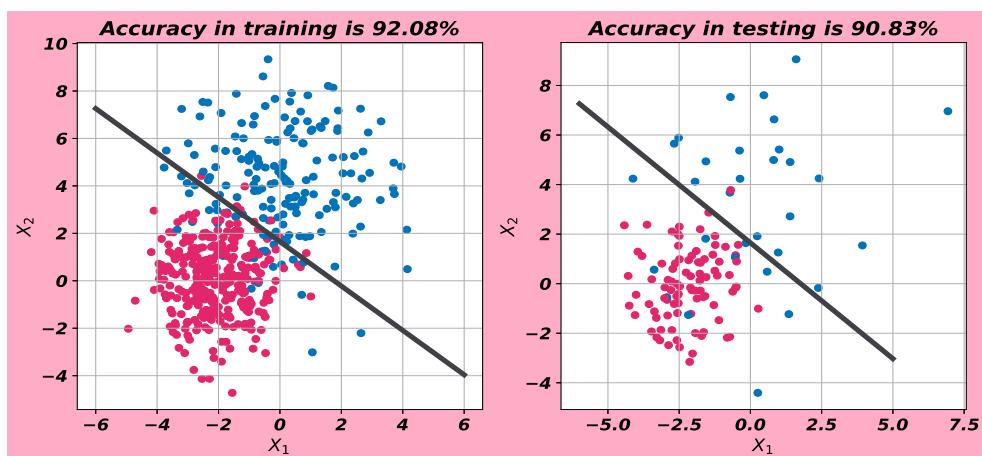


圖 4.5: 簡單線性—兩群資料變異程度及樣本數相異

4.3 加廣型迴歸模型 (Augmented Regression Model)

加廣型迴歸模型就是以簡單線性模型為基礎下，為降低訓練誤差，將其只能使用直線來分類的性質改為曲線，因非線性的分界線，也許能提供更適切的分隔效果。

假設輸入的變數為 X_1, X_2 ，則 (X_1, X_2) 所有可能的值涵蓋二維空間，若將此兩個變數擴展為 $X_1, X_2, X_1X_2, X_1^2, X_2^2$ 五個變數，同樣利用迴歸模式與最小平方法建立一條分界線，並將此分界線映射回原本的二維平面，會是一條

曲線。值得注意的是，這五個變數所形成的線性模式並不構成五維空間，原因為變數之間並不獨立，因此加廣型迴歸模型可寫成

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2 \quad (4.4)$$

其參數的估計與群組判別的方式與線性模式相同，唯一需要變動的部分為矩陣 X ，即

$$X = \begin{bmatrix} 1 & X_1(1) & X_2(1) & X_1(1)X_2(1) & X_1^2(1) & X_2^2(1) \\ 1 & X_1(2) & X_2(2) & X_1(2)X_2(2) & X_1^2(2) & X_2^2(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_1(N) & X_2(N) & X_1(N)X_2(N) & X_1^2(N) & X_2^2(N) \end{bmatrix} \quad (4.5)$$

接著參數 $\hat{\beta}$ 以最小平方法估計， $\hat{\beta} = (X^T X)^{-1} X^T y$ ，同簡單線性迴歸模式，最後式 (4.4) 的加廣型迴歸模型的分界線表示為集合

$$\{(X_1, X_2) \mid \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_1 X_2 + \hat{\beta}_4 X_1^2 + \hat{\beta}_5 X_2^2 = 0.5\}$$

4.3.1 實作一—以加廣型迴歸進行二元分類

以下實作所使用的資料同以簡單線性迴歸進行二元分類所用的資料集。

(a) 兩群資料—距離較近

觀察圖 (4.6) 可以發現使用加廣型迴歸線的分類狀況只比使用簡單線性迴歸線好一些，其原因為兩群資料使用相同變異程度的母體生成，導致其資料距離較近時，交界處並不明顯，進而使得加廣型迴歸線無法發揮其曲線長處。

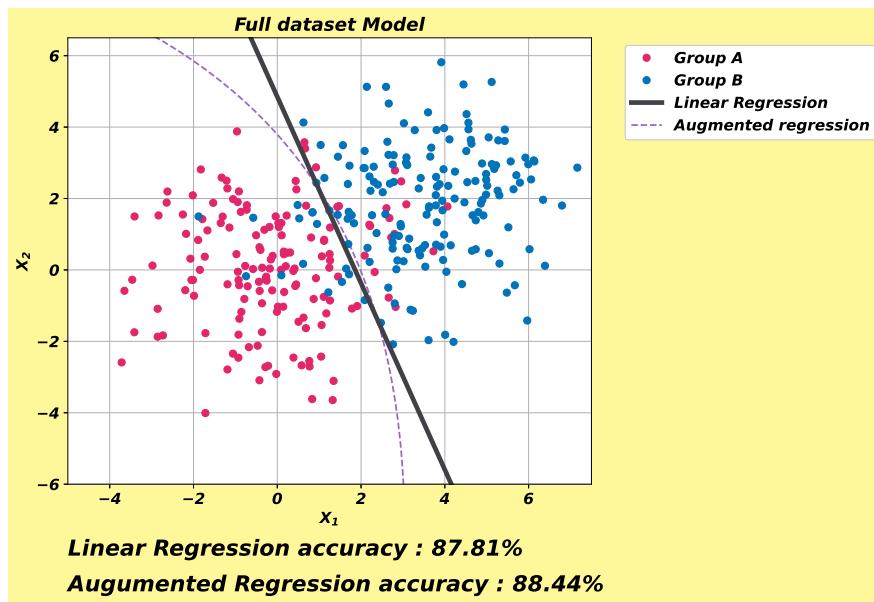


圖 4.6: 簡單線性與加廣型線性的比較—兩群資料距離較近

(b) 兩群資料—距離較遠

觀察圖 (4.7) 可以發現使用加廣型迴歸線的分類狀況比使用簡單線性迴歸線好，因其曲線的特性能繞過兩群中間較為交錯的部分。但如果今天兩群資料變得更為分散，則使用簡單線性模型進行分類就已足夠。

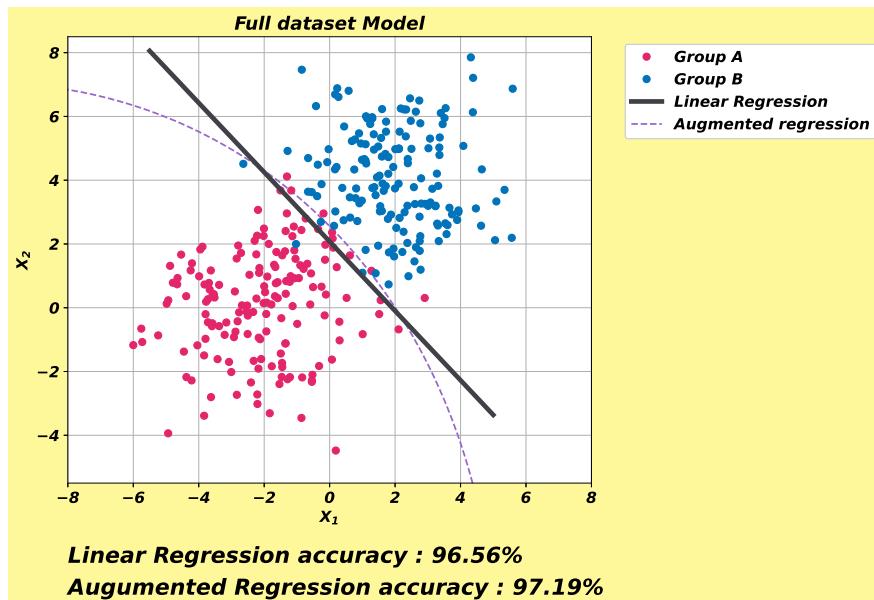


圖 4.7: 簡單線性與加廣型線性的比較—兩群資料距離較遠

(c) 兩群資料—變異程度較小

觀察圖 (4.8) 可以發現加廣型迴歸線與簡單線性迴歸線的形狀相當接近，原因推測為當兩群的交錯過於複雜時，加廣型迴歸線無法有效透過高維度的投影至二維平面，且其模型可能與簡單線性迴歸模型接近，以至於形狀近乎直線。

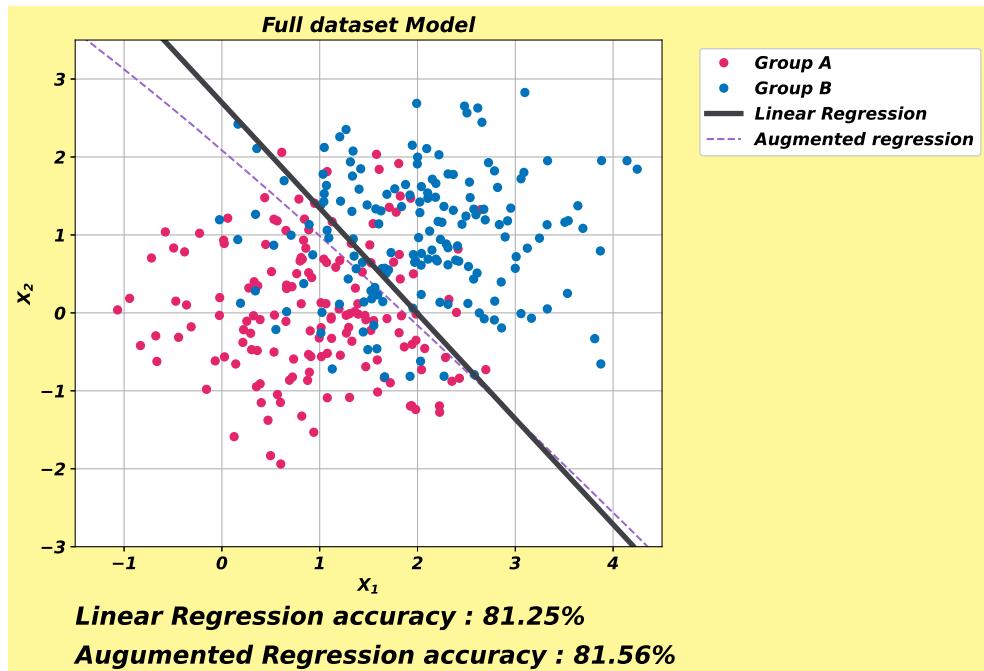


圖 4.8: 簡單線性與加廣型線性的比較—兩群資料變異程度較小

(d) 兩群資料—變異程度較大

觀察圖 (4.9) 可以發現使用加廣型迴歸線的分類狀況比使用簡單線性迴歸線好，因為當資料的變異程度較大時，兩群資料的外層會變得稀疏，進而使得分界變得清楚。

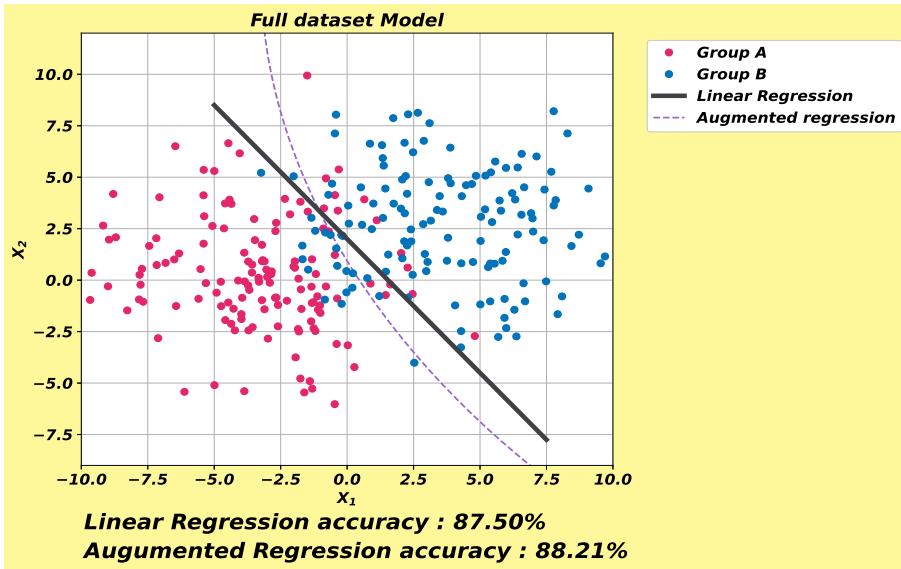


圖 4.9: 簡單線性與加廣型線性的比較—兩群資料變異程度較大

(e) 兩群資料—兩群資料變異程度及樣本數相異

觀察圖 (4.10) 可以發現加廣型迴歸線相當的彎曲，原因為群組 A 的變異相對小，加廣型迴歸線能幾乎將其圈住，此種變異程度相異的資料很適合使用加廣型迴歸模型進行分類。

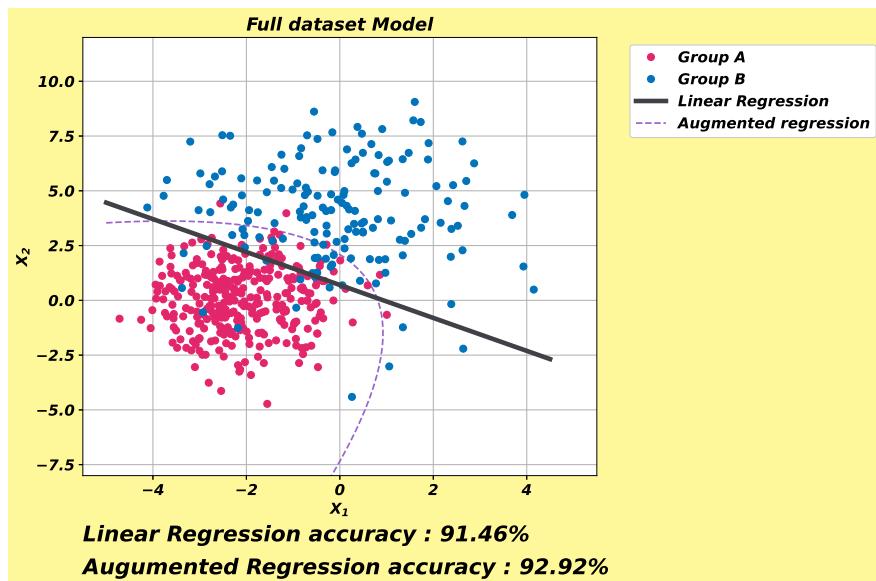


圖 4.10: 簡單線性與加廣型線性的比較—兩群資料變異程度及樣本數相異

4.4 邏輯回歸 (Logistic Regression)

邏輯回歸是處理二元分類的方式，屬於一種線性分類器。在線性迴歸中我們直接用特徵對目標建立回歸方程式；而邏輯回歸不是擬合一條直線或曲線，而是使用邏輯函數 (logistic function) 來壓縮 0 到 1 之間的線性方程的輸出。邏輯函數定義為：

$$\text{logistic}(\eta) = \frac{1}{1 + \exp(-\eta)} \quad (4.6)$$

並且其圖形為：

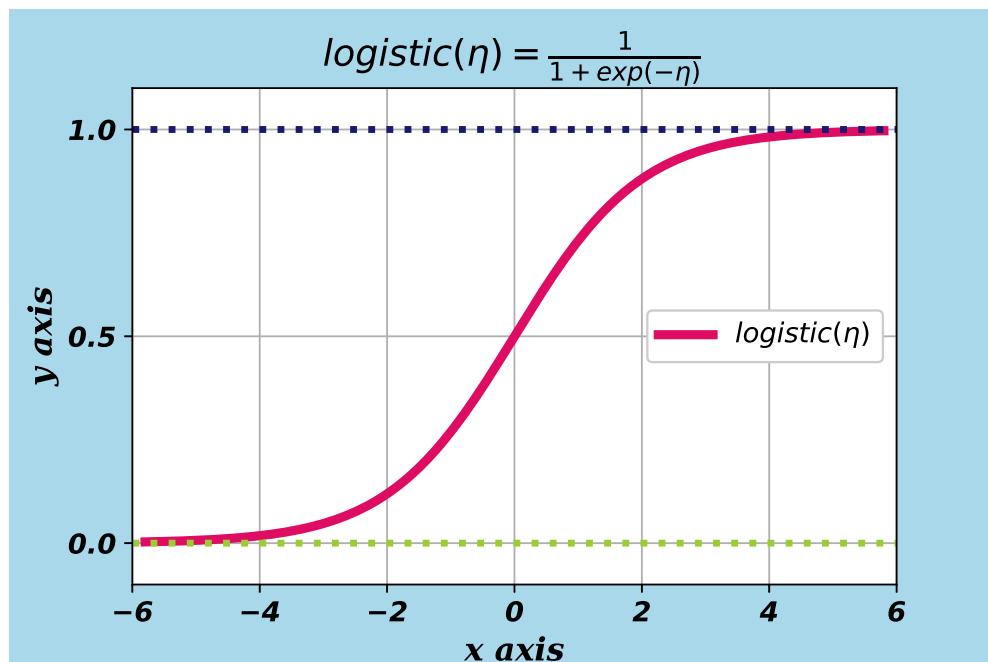


圖 4.11: 邏輯函數

從線性回歸到邏輯回歸的步驟很簡單。在線性回歸模型中，我們用線性方程對結果和特徵之間的關係進行了建模：

$$\hat{y} = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n \quad (4.7)$$

對於分類，我們更喜歡介於 0 與 1 之間的概率，因此將線性方程 (4.7) 的右側包裝到邏輯函數中：

$$P(X) = \frac{1}{1 + \exp(\beta_0 + \beta_1 X_1(i) + \dots + \beta_n X_n(i))} \quad (4.8)$$

這會強制輸出僅介於 0 到 1 之間的機率值，大於 50% 的機率會被預測為 1，小於 50% 的機率會被預測為 0，這就達到了分類的目的。

4.4.1 實作一以邏輯斯迴歸模型進行三元分類

實作所使用的資料集模擬自獨立雙變量常態的母體，在樣本數固定下，生成三群變異程度不同的資料，而資料的輸出值屬於類別資料，分別為 0, 1, 2。從圖 (4.12) 中可以看出群 2 為變異最大的群，次中為群 1，最後群 0 則為變異最小的群。

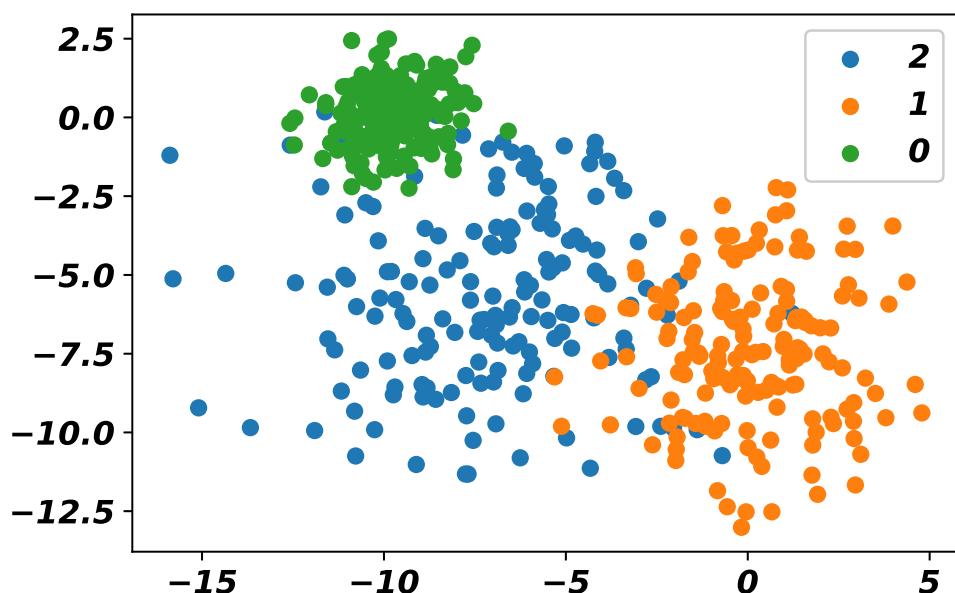


圖 4.12: 三群散佈圖

接著，運用邏輯斯迴歸模型將圖 (4.12) 分隔成三塊，並將其重新上色。觀察圖形後，我認為資料被分類的很好，並且根據計算，其分類正確率達 93.57%。這是因為邏輯斯迴歸模型本就是被設計在處理類別資料，而簡單線性模型與加廣型線性模型則是被重新定義才成為分類器，對於兩群以上的資料集，線性模型將會難以進行分類，並且效率也不及邏輯斯迴歸模型。

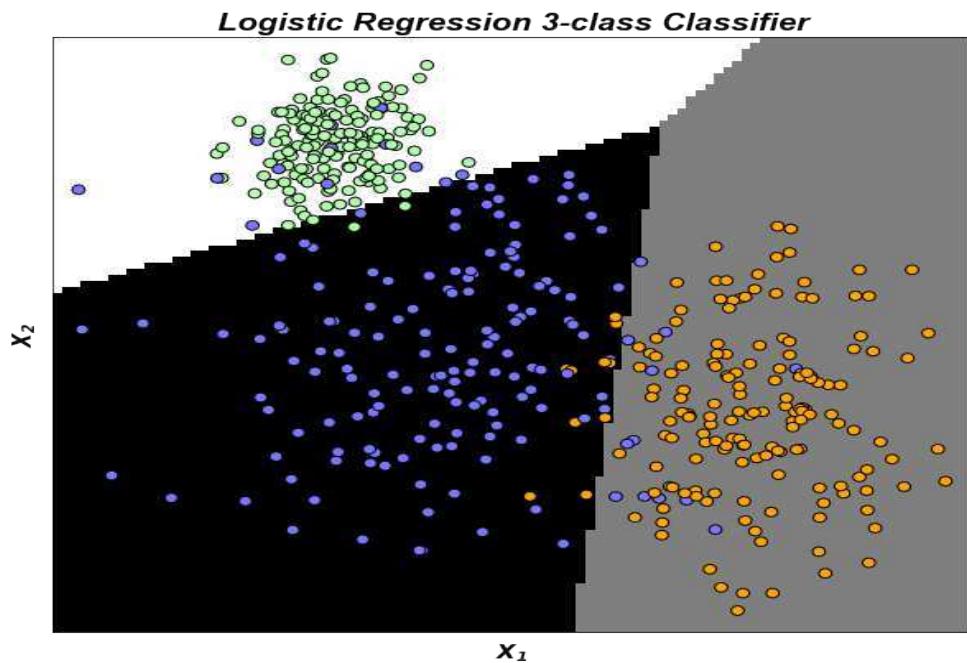


圖 4.13: 以邏輯斯迴歸模型進行三群分類

4.5 結語

初學機器學習，從線性迴歸來切入確實能使我很快地適應新的領域，即使它不太適合拿來做分類，但這讓我想去理解更多的分類器，不同狀況應該選用哪些模型能夠有更低的建模錯誤率是我之後要學習的。至於軟體方面，除了熟練掌握基礎的 Numpy 之外，在機器學習領域中，還可以更多地研究 scikit-learn 這個非常強大的套件。它提供了許多機器學習算法的實現，如線性回歸、邏輯回歸、決策樹、支持向量機等，並提供了許多工具來幫助我們評估模型的性能，如交叉驗證、網格搜索等。使用 scikit-learn 可以更快速地實現機器學習項目，並且具有良好的可擴展性和可維護性。

第 5 章

監督式學習：判別分析與 KNN 分類

監督學習是目前被廣泛應用的一種學習方式，它具有完整的訓練數據，而且每條訓練數據都有相應的標籤。在這樣的學習過程中，機器通過訓練集的數據訓練出模型，在未來投入新的數據時，機器便可根據這個模型對新的數據進行預測。貝氏定理 (Bayes Theorem) 是統計學角度中對分類問題的有效解答，簡要說就是利用事先先驗機率和證據來更新後驗機率。本文將從貝氏定理開始切入監督學習。

5.1 線性判別式分析 (LDA)

線性判別分析 (Linear Discriminant Analysis) 是一種監督式的線性分類算法，它的基本思想是將數據投影到低維空間後，使得同一類數據盡可能接近，不同類數據盡可能疏遠。在對新樣本進行分類時，將其投影到同樣的低維空間上，再根據投影點的位置來確定新樣本的類別。並且 LDA 可從貝氏理論來解釋與證明，此節先從貝氏判定準則開始介紹。

貝氏判定準則：為最小化總體風險，需要在每個樣本上選擇能使得條件風險 $R(G | x)$ 為最小的類別標記，即

$$G(X) = \arg \min_k R(G = k | X = x) \quad (5.1)$$

其中 G 被稱為貝氏最佳分類器。

當學習目標是最小化分類錯誤率時，條件風險為

$$R(G = k \mid X = x) = 1 - P(G = k \mid X = x) \quad (5.2)$$

其中 $P(G \mid X)$ 屬於後驗機率。

於是貝氏最佳分類器改寫為

$$G(X) = \arg \max_k P(G = k \mid X = x) \quad (5.3)$$

再利用貝氏定理得到

$$G(X) = \arg \max_k P(X \mid G = K)P(G = K) \quad (5.4)$$

其中 $P(X \mid G = K)$ 表示第 K 組資料發生的機率密度函數，而 $P(G = K)$ 代表群組 K 發生的機率。

根據貝氏判定準則，對每一個樣本，我們需要選擇使得 $P(X \mid G)P(G)$ 機率最大的類別。所以機器在此的任務就是根據訓練數據去估計機率 $P(G = k \mid X = x)$ 。而線性判別分析 (LDA) 是在貝氏判定準則下，對群組的機率密度函數 $f_k(x)$ 進行多元常態分佈的假設，寫成

$$f_k(x) = \frac{\exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right)}{\sqrt{(2\pi)^p |\Sigma_k|}} \quad (5.5)$$

假設只有兩個群組， k 以及 l 群組，當兩群組間具有相同共變異矩陣 $\sum_k = \sum_l \forall k$ 以及多元常態分佈的假設時，便可導出線性判別分析 (LDA)。我們可以使用 log-odds 比較兩個群組的狀況：

$$\begin{aligned} \ln \frac{P(G = k \mid X = x)}{P(G = l \mid X = x)} &= \ln \frac{f_k(x)}{f_l(x)} + \ln \frac{P(G = k)}{P(G = l)} \\ &= \ln \frac{P(G = k)}{P(G = l)} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l) \end{aligned} \quad (5.6)$$

藉由 log-odds 大於 0 或小於 0 來判斷資料屬於哪一群組的可能性較大，這樣就可以實現分類。若令 log-odds 為 0，則得到一組線性方程式，也就是群組的分界線。如果共變異矩陣 Σ 是 $\sigma^2 I$ 且類別的先驗機率相等，從式 (5.6) 可以看出線性判別函數為

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln P(G = k) \quad (5.7)$$

圖 (5.1) 即以 sklearn 實作 LDA。左圖顯示兩個二元常態分佈，有相同的共變異數矩陣和不同的均值，而右圖顯示了兩個二元常態分佈，有相異的共變異數矩陣和不同的均值。兩圖皆畫出了類別之間的擬合後 LDA 判別邊界，使用黑線表達。若先不使用誤判率結果來評斷模型的好壞，可以透過觀察知道左圖 (共變異數相同) 的分類狀況明顯好於右圖 (共變異數相異)，也就是群組的分佈形式對 LDA 的分類正確率有很大影響。

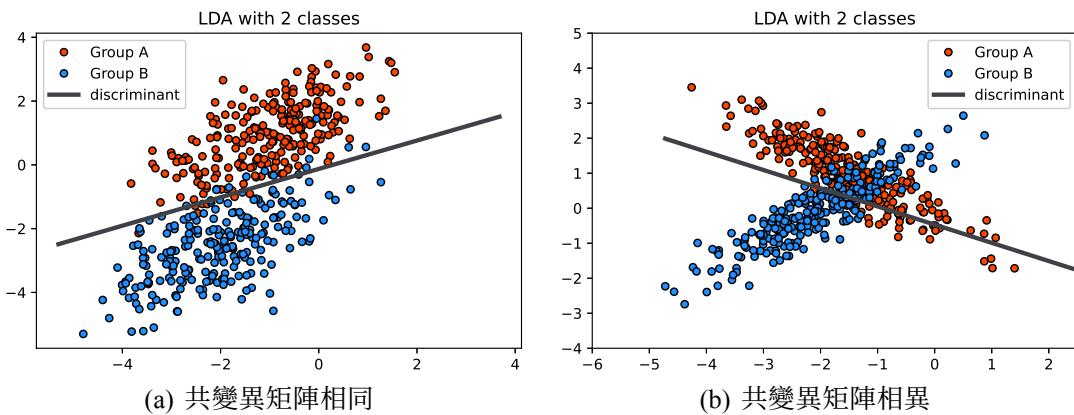


圖 5.1: LDA 應用於兩群資料的分類狀況

5.2 二次判別式分析 (QDA)

如前段所述，群組間的線性分界線來自群組的共變異數矩陣相同的假設。如果拿掉這個假設，令各群組的共變異數矩陣不同時，式 (5.6) 中 x 的二次項會被保留下來，於是我們會得到平方判別函數

$$\delta_k(x) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \ln P(G = K) \quad (5.8)$$

稱為二次 (Quadratic) 的原因來自切割群組 k 與群組 l 間的分界線不是直線，而是二次式的曲線，這條分界線寫為：

$$\{x | \delta_k(x) = \delta_l(x)\} \quad (5.9)$$

圖 (5.2) 中的兩種資料分布狀況與圖 (5.1) 相同，皆畫出了兩類別之間，由於 x 的二次函數近似出的判別邊界，使用黑線表達。若先不使用誤判率結果來評斷模型的好壞，可以透過觀察知道右圖結果明顯好於圖 (5.1) 中同資料

(共變異數相異) 的分類狀況，可能原因為 QDA 使用的是一個二次判別邊界，它可以比線性方法更有效處理複雜的問題。

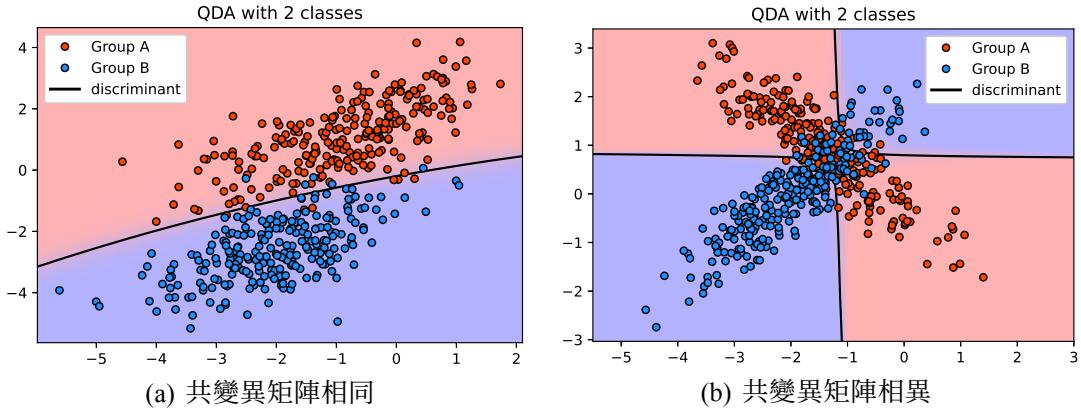


圖 5.2: QDA 應用於兩群資料的分類狀況

5.3 K-最近鄰居法 (KNN)

對於處理分類的問題，我們最熟悉的工具是採用最小平方法的迴歸模型，但此方法並沒有充分利用資料本身的訊息，譬如資料的變異性。若希望資料本身能提供更多的訊息，做為新資料所屬群組的判別依據，將問題以「機率」的角度來考量會是一個妥善的方法。

「假設 Y 及 $f(X)$ 分別代表輸出變數與輸出預測值，其中 $X \in R^p$ 表示有 p 個輸入變數。倘若我們的目標是輸出值與預測值間誤差愈小愈好，可以假設輸入變數 X 與輸出變數 Y 的聯合機率密度函數 $P(X, Y)$ 為已知，則此問題可以寫成」

$$\min_{f(X)} E_{XY}[(Y - f(X))^2] \quad (5.10)$$

其中

$$\begin{aligned} E_{XY}[(Y - f(X))^2] &= \int \int (Y - f(X))^2 P(X, Y) dXdY \\ &= \int \int (Y - f(X))^2 P(Y | X) P(X) dXdY \\ &= E_X E_{Y|X}[(Y - f(X))^2 | X] \end{aligned} \quad (5.11)$$

也就是找一個輸入與輸出變數間的關係式 $f(\cdot)$ ，使得真正的輸出值 Y 與其預測值 $f(X)$ 間的誤差的平方期望值越小越好，這個方法稱為「最小均方誤差

(Minimum Mean Squared Error, MMSE)」。在已知樣本值 $X = x$ 的條件下，它的最佳解如

$$y = \hat{f}(x) = E_{Y|X}(Y | X = x) \quad (5.12)$$

其中 X, Y 代表輸入與輸出變數， x 與 y 表示輸入值及輸出的擬合值。式 (5.11) 說明當輸入值為 x 時，最佳的輸出預測值為輸出變數的「條件式均值」。

因為期望值代表的是理論值，所以在實務上一般都利用平均值來估計 (5.12) 這類期望值。譬如以下樣本平均數

$$\hat{y} = Ave(y_i | X = x) \quad (5.13)$$

是個優良的估計式。這個估計式解讀為「將輸入資料為 x 的所有資料，找出對應的所有輸出值 y_i 取平均」。但是此方法在實際應用時會遇到困難，理由是已知的多變量連續型資料，剛好等於 x 的單點機率為零，因此實務上不可行。

將式 (5.13) 稍作修改後，下面這個輸出預測值的估計式解決了這些困擾。

$$\hat{y} = Ave(y_i | x_i \in N_k(x)) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \quad (5.14)$$

式 (5.14) 的解讀為：從已知的資料中找到 K 個最靠近 x 的資料，將這些鄰近的 K 筆資料所對應的 y 值平均起來作為「條件式均值」的估計，這個方法叫做 K -Nearest-Neighbor method。但若輸出變數 Y 的群組屬性為類別資料時，式 (5.12) 的 MMSE 最佳解問題可以寫成

$$\min_{g(X)} E_{XG}[L(G, g(X))] \quad (5.15)$$

由於是類別資料的關係，其輸出群組變數改寫為 G ，預測群組寫成 $g(X)$ ，兩者的誤差以 $L(G, g(X))$ 「損失函數」取代原先的平方差。當 $L(G, g(X))$ 定義為

$$G = \begin{cases} 0 & \text{if } G = g(x) \\ 1 & \text{if } G \neq g(x) \end{cases}$$

式 (5.12) 的最佳解則為

$$\hat{g}(X) = g_k \text{ if } P(g_k | X = x) = \min_{g \in G} P(g | X = x) \quad (5.16)$$

這個結果說明：當輸入值為 x 時，其所屬群組的 MMSE 預測為

「在所有的群組中，群組機率密度函數在 x 處的值為最大者」

它的想法與式 (5.3) 貝氏最佳分類器相同，都使用到「後驗機率」。也就是給定輸入變數 $X = x$ ， Y (或 G) 值的可能性（機率）。

圖 5.3 直觀地說明了 KNN 以及 K 對決策邊界的影響。 K 值表示最近鄰居的數量，該值是分類器的核心決定因素，因為 K 決定有多少鄰居影響分類。圖 5.3 (a) 中，繪製了來自包含三個群的多群資料散布圖。以紅色、藍色與綠色描繪的群是分開的，紅色和其它兩群之間有明顯的區別，藍色、綠色之間的邊界區域有一些重疊。

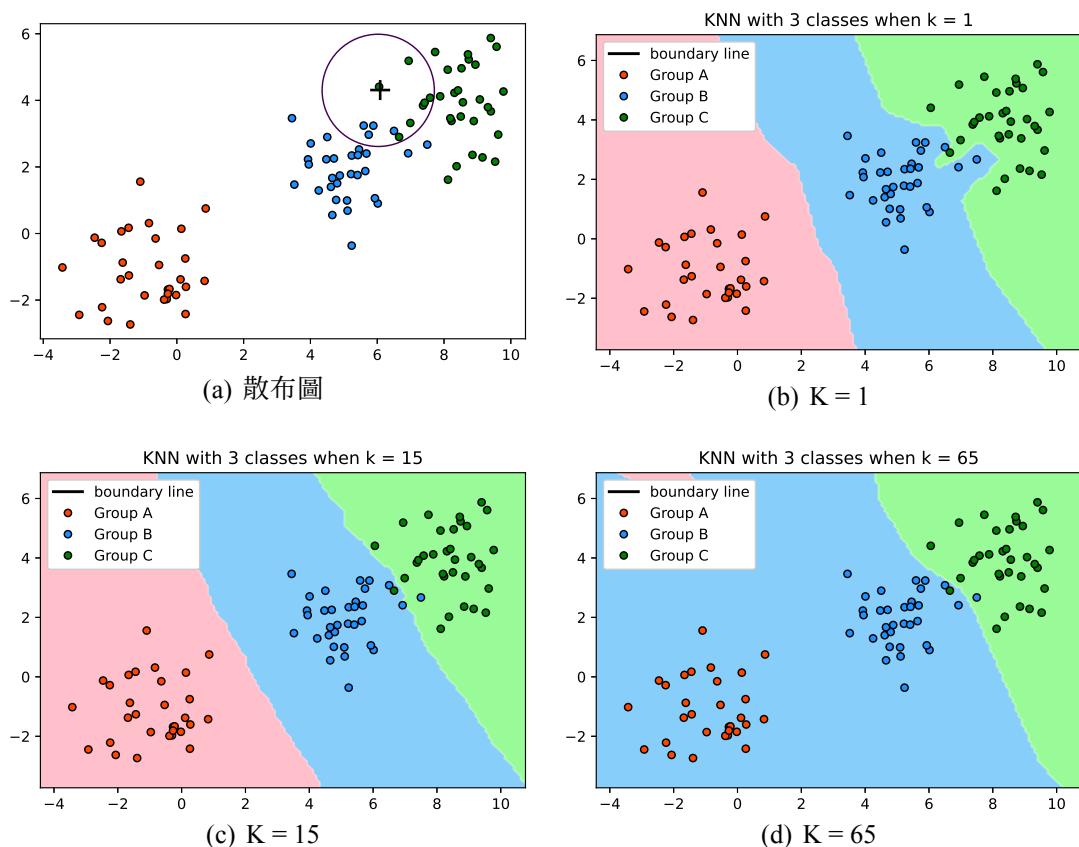


圖 5.3: 多群資料中不同 K 值的 KNN 判別邊界

散布圖中特別標註了十字形數據點，目的是識別該數據點屬於哪個類。採取的方法為繪製一個圓圍繞在此數據點外，計算圓圈內的點數。圍繞這個十字總計有 11 個點，其中有 7 個綠色點和 4 個藍色點。因此，如果選擇 $K = 11$ ，則十字形數據點屬於綠色和藍色的機率分別為 $7/11$ 和 $4/11$ 。因為屬於綠色的

機率比較高，所以此點會被塗為綠色。

如圖 5.3 所示，選擇的 K 值顯著改變預測結果。例如， $K = 1$ 的決策邊界把大家都分類的很好，因為它只與一個鄰居做比較，造就 1-NN 成為最複雜的 KNN 分類器，擬合最好，偏差最小。當 $K = 15$ 時，與 $K = 1$ 時活潑的決策邊界相比，它變得更加平滑，而且分別有一個藍色點和綠色點被分類錯誤。至於 $K = 65$ 的決策邊界甚至變成了一條線，因為它找了過多的鄰居，導致預測精度顯著降低。

5.4 LDA, QDA 與 KNN 學習器評比

本章節的目的在於探討 LDA, QDA 與 KNN 分類器，在使用相同的模擬資料集下，此三者各自的學習狀況為何？接著再探討三者間的差異，設法找出其背後影響預測能力的原因。而評估標準使用 bootstrapping 方式，重複抽樣 100 次。基於 100 個 bootstrap 數據集，計算訓練誤差與測試誤差的平均比例。

5.4.1 相同共變異矩陣的比較

共變異矩陣相同會使得群跟群的外型相似，如圖 5.4 以資料假設來自兩個雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mu_2 = \begin{bmatrix} -2 \\ -2.5 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1.2 & 0.8 \\ 0.8 & 1.2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1.2 & 0.8 \\ 0.8 & 1.2 \end{bmatrix}$$

$$n_1 = 300, n_2 = 300$$

利用 LDA, QDA, 5-NN 與 15-NN 學習器進行分群的結果。

觀察來自兩群資料的散布圖。以紅色和藍色描繪的群是分開的，兩群之間有明顯的區別，它們之間的邊界區域僅有一些重疊。因為變異數和樣本數的取值相同，所以兩群看起來十分相似，與理論相呼應。

表 5.1: 四種學習器在兩共變異矩陣相同資料的學習狀況

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
兩群-共變異矩陣相同	2.51%	2.53%	2.66%	2.76%	1.95%	3.11%	2.60%	2.71%

首先比較 LDA 模型與 QDA 模型，根據表 6.1，兩者的測試誤差分別為 2.53% 和 2.76%，表現結果相近，因為資料很明顯是線性可分的，並且它們的平均數差異大，重疊部分較少。

再根據表 6.1，兩種 KNN 分類器的測試誤差都比 LDA 模型高，所以可以合理推斷當真正的決策邊界是線性時，LDA 方法往往會表現良好。

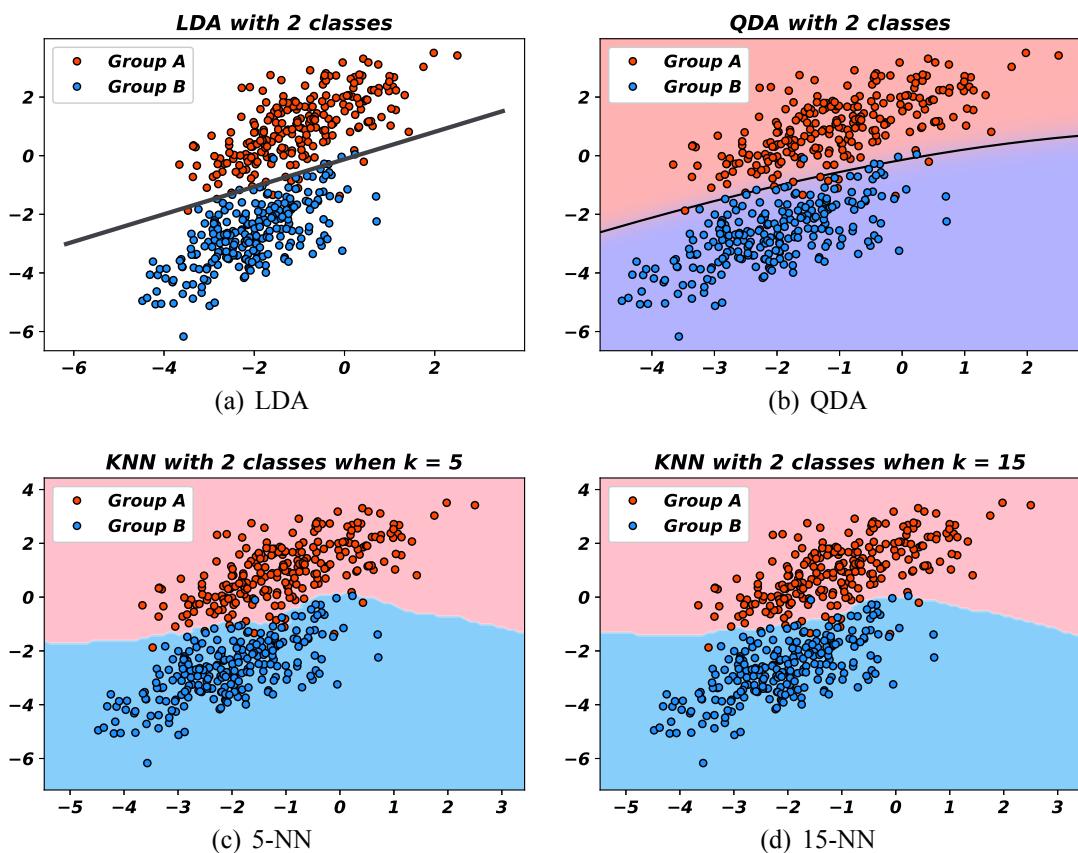


圖 5.4: 四種學習器在兩共變異矩陣相同資料的分群狀況

若將以上有明顯線性決策邊界推廣至多群，如圖 5.5 以資料假設來自三個雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4.5 \\ 6 \end{bmatrix}, \mu_3 = \begin{bmatrix} 8 \\ -1 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

$$n_1 = 250, n_2 = 250, n_3 = 250$$

利用 LDA, QDA, 5-NN 與 15-NN 學習器進行分群的結果。

觀察來自三群資料的散布圖。以紅色、藍色與綠色描繪的群是分開的，三群之間有明顯的區別，僅在邊界區域有一些重疊。因為變異數和樣本數的取值相同，所以三群看起來也十分相似。

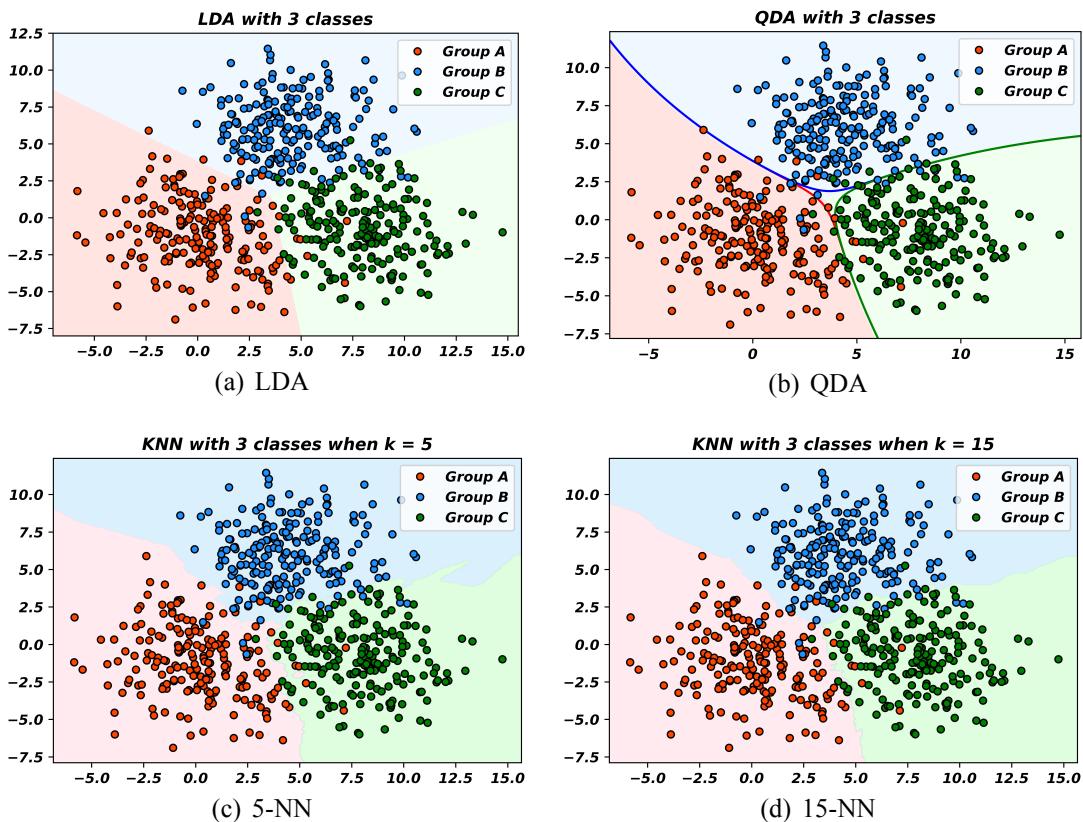


圖 5.5: 四種學習器在三共變異矩陣相同資料的分群狀況

首先比較 LDA 模型與 QDA 模型，根據表 6.2，兩者的測試誤差分別為 6.04% 和 5.89%，表現結果都非常良好。特別是 LDA 模型，因為此三群資料特別設定相同共變異數矩陣，符合 LDA 模型的基本假設，因此它的表現並不遜色於 QDA 模型。

再根據表 6.2，15-NN 模型的測試誤差與 LDA, QDA 模型相近。值得注意的是當 $K = 5$ 時，模型的訓練誤差與測試誤差差異較大，可能是受 K 值較小時的特性影響。

表 5.2: 四種學習器在三共變異矩陣相同資料的學習狀況

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
三群-共變異矩陣相同	5.80%	6.04%	5.85%	5.89%	5.03%	7.47%	5.35%	6.07%

將以上兩種相同共變異矩陣在四種學習器的誤差統整起來，如表 6.3，可以明顯看到 LDA 和 QDA 兩種判別分析的預測誤差分別小於兩種 KNN 分類器。所以可以合理推斷，「判別分析」，使用在相同共變異矩陣且平均數沒有過於靠近的資料上，分群效果較好。

表 5.3: 共變異矩陣相同資料的比較

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
兩群-共變異矩陣相同	2.51%	2.53%	2.66%	2.76%	1.95%	3.11%	2.60%	2.71%
三群-共變異矩陣相同	5.80%	6.04%	5.85%	5.89%	5.03%	7.47%	5.35%	6.07%

5.4.2 相異共變異矩陣的比較

共變異矩陣相異會使得群跟群的外型對稱或相異，如圖 5.6 以資料假設來自兩個雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} -1.5 \\ 1 \end{bmatrix}, \mu_2 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1.0 & -0.9 \\ -0.9 & 1.0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}$$

$$n_1 = 300, n_2 = 300$$

利用 LDA, QDA, 5-NN 與 15-NN 學習器進行分群的結果。

觀察來自兩群資料的散布圖。因變異數取值異號，平均數相等，所以紅色與藍色描繪的群是對稱的，只在交叉點高度重合。

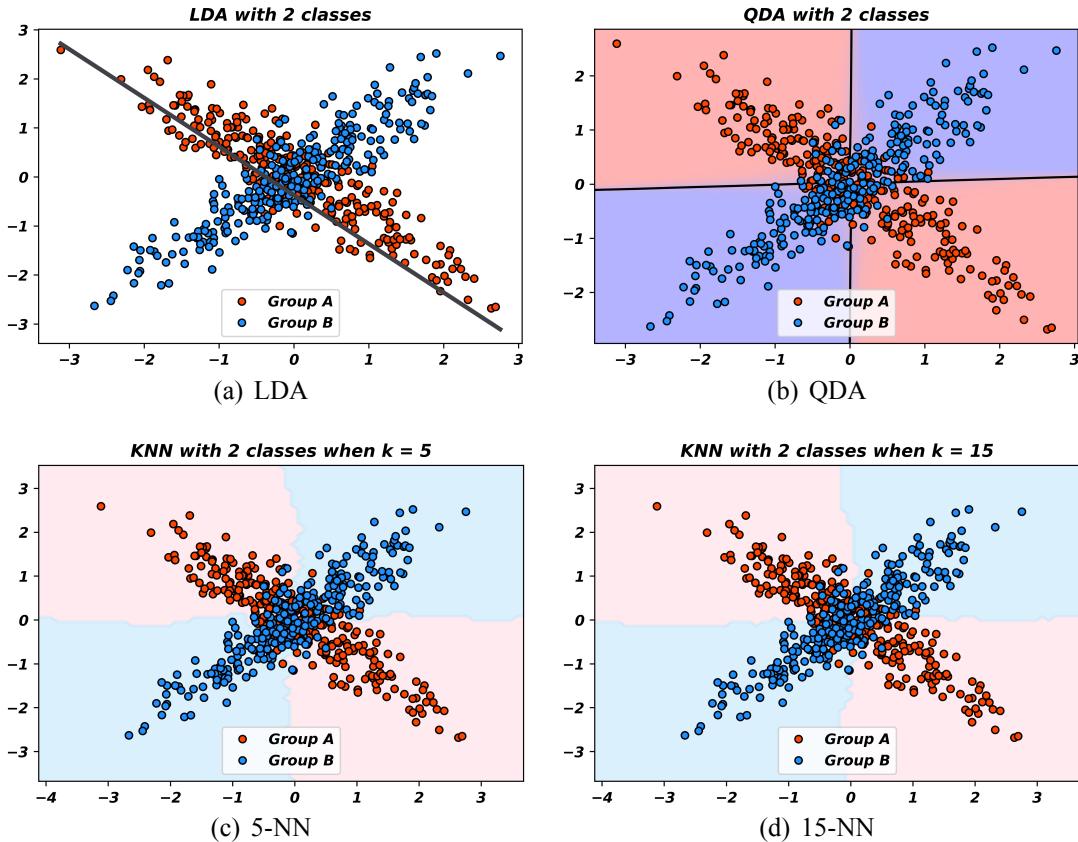


圖 5.6: 四種學習器在兩共變異矩陣相異資料的分群狀況

首先比較 LDA 模型與 QDA 模型，根據表 5.4，兩者的測試誤差分別為 54.34% 和 15.71%，QDA 模型的分類效果明顯比 LDA 模型好。因為此兩群資料設定相異共變異矩陣，除了違反 LDA 模型的基本假設，它的決策邊界還是兩條二次曲線。這些原因都大幅降低 LDA 模型的準確率。

再根據表 5.4，兩個 KNN 分類器的測試誤差與 LDA, QDA 模型相近，這代表 KNN 分類器也適合使用在決策邊界為曲線的情況。值得注意的是當 $K = 5$ 時，模型的訓練誤差與測試誤差差異較大，可能是受 K 值較小時的特性影響。

表 5.4: 四種學習器在兩共變異矩陣相異資料的學習狀況

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
兩群-共變異矩陣相異	48.49%	54.34%	16.04%	15.71%	12.21%	16.94%	15.22%	17.46%

若將以上有明顯非線性決策邊界推廣至多群，如圖 5.7 以資料假設來自三個雙變量常態的母體，其平均數、共變異矩陣與樣本數分別為：

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \mu_3 = \begin{bmatrix} -0.5 \\ -3 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1.0 & -0.9 \\ -0.9 & 1.0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}$$

$$n_1 = 250, n_2 = 250, n_3 = 250$$

利用 LDA, QDA, 5-NN 與 15-NN 學習器進行分群的結果。

觀察來自三群資料的散布圖。以紅色與綠色描繪的群是分開的，而藍色與其它兩群有兩塊重疊處，三群之間有明顯的區別。因為紅色與綠色描繪的群變異數和樣本數的取值相同，所以看起來也十分相似。藍色群的外型則對稱紅色與綠色群。

表 5.5: 四種學習器在兩共變異矩陣相異資料的學習狀況

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
三群-共變異矩陣相異	25.86%	26.31%	25.92%	26.18%	9.5%	13.32%	11.09%	12.79%

首先比較 LDA 模型與 QDA 模型，根據表 5.5，兩者的測試誤差分別為 26.31% 和 26.18%，分類表現差異不大。因為此三群資料設定兩同一異的共變異矩陣，並且因為位置的刻意安排，它的真實決策邊界是不規則。這些原因都大

幅降低 LDA 模型與 QDA 模型的準確率，

比較圖 5.7 中 QDA 模型與 KNN 分類器的決策邊界，它們都將資料切割成七個區域，藍色三塊，而綠色與紅色各兩塊。但 KNN 分類器將資料切割得更佳細膩，使得誤判率比 QDA 模型低。再根據表 5.5，兩個 KNN 分類器的測試誤差都比 LDA, QDA 模型還低許多。這代表 KNN 分類器適合使用在決策邊界為不規則的情況。

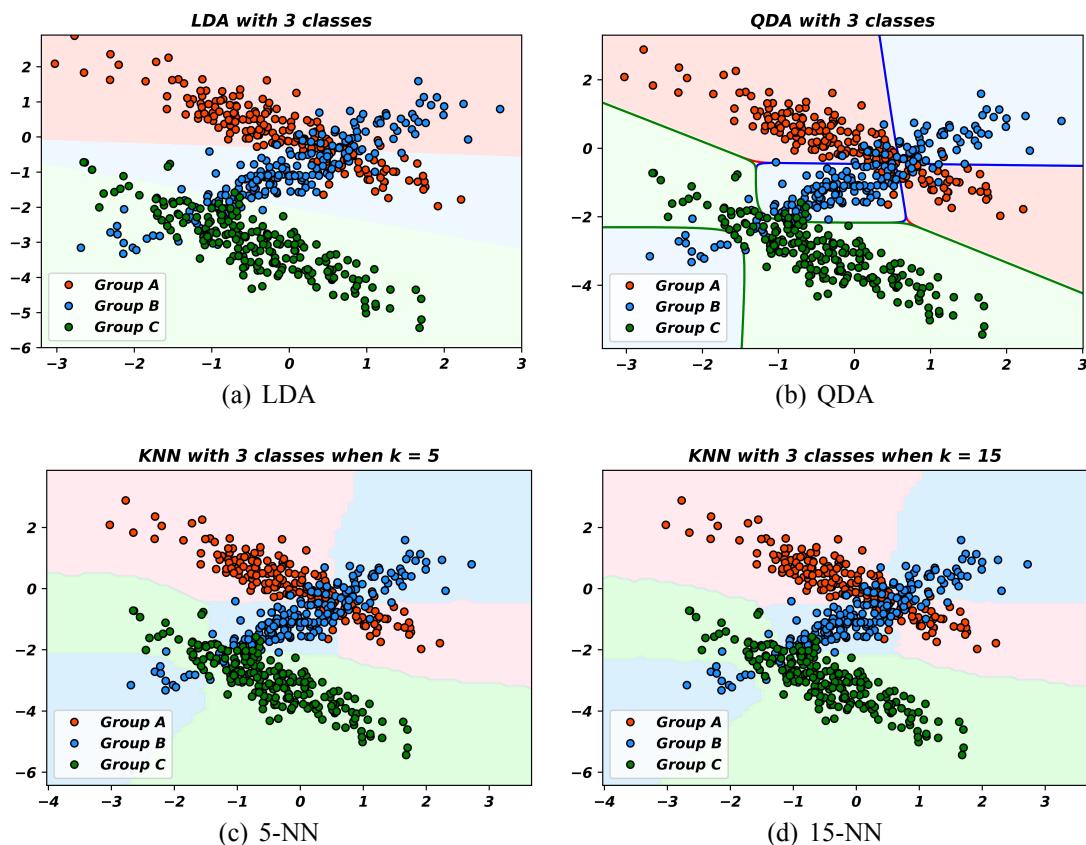


圖 5.7: 四種學習器在三共變異矩陣相異資料的分群狀況

最後，將以上兩種相異共變異矩陣在四種學習器的誤差統整起來，如表 5.6，可以明顯看到 LDA 模型的預測誤差大於其他三種分類器，因為資料的決策邊界非線性。QDA 模型在兩群資料時為表現最好的模型，其預測誤差只有 15.71%，但在決策邊界較複雜的三群資料時預測誤差卻又大幅增加。所以可以合理推斷，QDA 模型使用在資料的真實決策邊界接近二次曲線時，分群效果較好。對於更複雜的問題，如圖 5.7 應採用 KNN 分類器。

表 5.6: 共變異矩陣相異資料的比較

資料集	LDA		QDA		5-NN		15-NN	
	訓練	測試	訓練	測試	訓練	測試	訓練	測試
兩群-共變異矩陣相異	48.49%	54.34%	16.04%	15.71%	12.21%	16.94%	15.22%	17.46%
三群-共變異矩陣相異	25.86%	26.31%	25.92%	26.18%	9.5%	13.32%	11.09%	12.79%

5.4.3 環形資料集的比較

如圖 5.8 所示，此節使用的資料外型為兩個同心環，我使用 sklearn 中的 make_circles 套件來生成這些帶有標籤的數據。資料本身並不服從多元常態分配，它們僅是為了幫助可視化和理解分類器行為的玩具數據集。

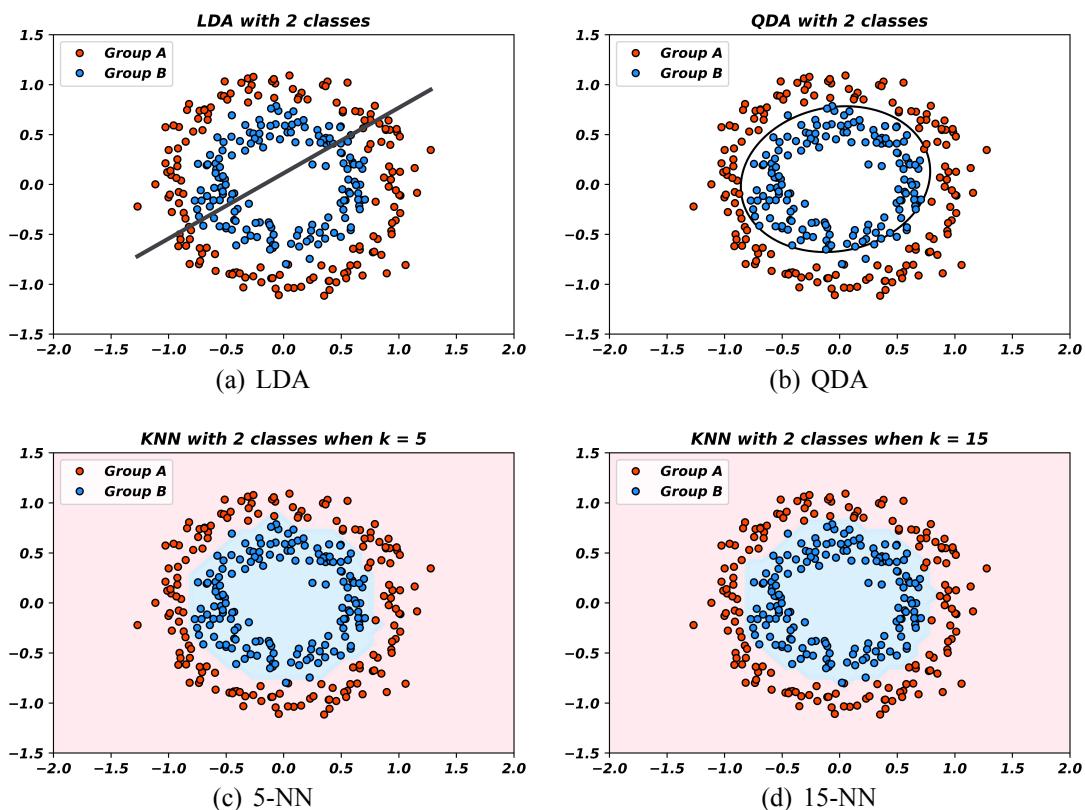


圖 5.8: 四種學習器在環形資料的分群狀況

LDA 模型在環形資料裡可能已不具分類的能力，如表 5.7，得知它的測試誤差高達 56.08%。而 QDA 模型與兩種 KNN 分類器所產生的決策邊界很相似，只是 KNN 分類器對異常點較為敏感，所以呈現不規則橢圓狀。

表 5.7: 四種學習器在環形資料的學習狀況

	LDA		QDA		5-NN		15-NN	
資料集	訓練	測試	訓練	測試	訓練	測試	訓練	測試
環形	47.74%	56.08%	3.13%	4.39%	1.01%	1.82%	1.41%	1.72%

5.5 結論

本文前段主要介紹判別分析與 KNN 分類的理論。中後段的重心放在資料模擬與分類器的表現評比，透過實作我給出了以下結論：

- 當真正的決策邊界是線性的，LDA 模型往往會表現良好。
- 當邊界「適度」非線性時，QDA 模型可能會給出比 LDA 模型更好的結果。
- 對於更複雜的決策邊界，KNN 此種不明確假設數據分佈的參數方法，分類結果更佳。但是需要特別考慮 K 值，因為它影響 KNN 分類的準確率。

目前 KNN 方法只討論 5-NN 與 15-NN，若再增加 K 的取值方法，譬如交叉驗證 (Cross-Validation)，會使得本文更加完整。

第 6 章

淺度機器學習：類神經網路

本文主要介紹前饋式類神經網路，是人工神經網路的一種。前饋神經網路採用單向且多層結構。每一層包含若干個神經元，各神經元可以接收前一層神經元訊息，進行分析處理後傳遞給下一層。最前面的層叫做輸入層，最後一層叫做輸出層，其他的中間層叫做隱藏層。隱藏層可以是一層或是多層。神經網路主要針對輸入層與輸出層資料，以不同的演算方式，建立兩者間映射的相對關係。因此，可將其視為一種數理迴歸方法。以下將從類神經網路的原理開始介紹。

6.1 前饋式 (Feedforward) 類神經網路的原理

圖 6.1¹展示擁有一個隱藏層的前饋式類神經網路，其中左邊輸入層 (input layer) 有 p 個變數個數 (圖中 $p = 14$)，中間隱藏層 (hidden layer) 有 q 個神經元 (圖中 $q = 10$)，以及最右邊的輸出層 (output layer)，共有 r 個輸出變數 (圖中 $r = 3$)。值得注意的是，當隱藏層的神經元數量 q 越大，則輸出與輸入之間的關係越複雜，從數學觀點來看，也就是網路中的非線性化程度越高。

¹資料來源：“淺度機器學習：類神經網路”。汪群超。

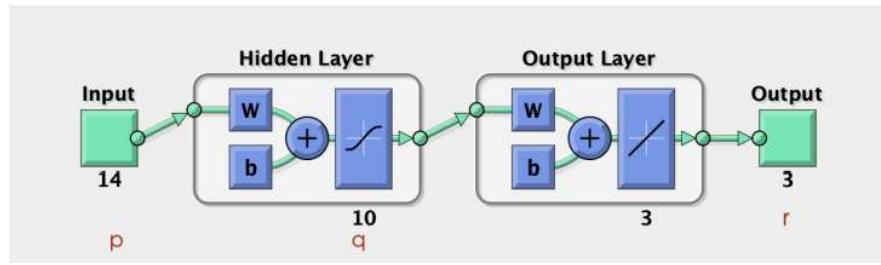


圖 6.1: 一個隱藏層的前饋式類神經網路

假設輸入層的 p 個變數表示為 x_1, x_2, \dots, x_p , 輸出層的 r 個變數表示為 $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_r$ ，則前饋式類神經網路的輸出與輸入間的數學關係寫成

$$\hat{y}_k(x, w) = \sum_{j=1}^q w_{kj}^{(2)} h \left(\sum_{i=1}^p w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)}, \quad 1 \leq k \leq r \quad (6.1)$$

- 上標 (1) 與 (2) 代表該參數作用於網路的第 1 層 (隱藏層) 還是第 2 層 (輸出層)。
- 參數 $w_{ji}^{(1)}$ 與 $w_{kj}^{(2)}$ 為權重係數。
- 參數 $w_{j0}^{(1)}$ 與 $w_{k0}^{(2)}$ 為偏移量。

其中 $h(\cdot)$ 為非線性的激勵函數 (Activation Function)，在此寫為

$$h(z) = c_1 \frac{1 - e^{-c_2 z}}{1 + e^{-c_2 z}}, \quad 1 \leq h(z) \leq 1 \quad (6.2)$$

函數 $h(z)$ 的外型為 S 型函數，稱為 (Sigmoid Function)。它作用於神經網路中的隱藏層，負責將進入神經元的訊息轉換為新的非線性訊號，再傳遞給輸出層。而函數 $h(z)$ 也可用於輸出層來增加整體複雜度。但在此先假設我們的任務是預測介於 $(-\infty, \infty)$ 的數值，那麼在輸出層中則使用線性函數 $y = x$ 。

當類神經網路在已知輸入與輸出資料 $x_i(n)$ 與 $y_k(n)$ 時，會逐一調整神經元的權重 $(w_{ji}^{(1)}, w_{kj}^{(2)})$ 與偏移量 $(w_{j0}^{(1)}, w_{k0}^{(2)})$ ，試著找到最佳的權重設定，直到真實資料 $y_k(n)$ 與類神經網路輸出資料 \hat{y}_k 的誤差為最小。假設共有 N 筆資料，則以上類神經網路的學習和訓練過程，可以寫成多變量函數的最小值問題，即

$$\min_{\Omega} e(\Omega) \quad (6.3)$$

其中誤差函數

$$\begin{aligned} (\Omega) &= \sum_{n=1}^N \sum_{k=1}^r (y_k(n) - \hat{y}_k(n))^2 \\ &= \sum_{n=1}^N \sum_{k=1}^r \left(y_k(n) - \sum_{j=1}^q w_{kj}^{(2)} h \left(\sum_{i=1}^p w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)^2 \end{aligned} \quad (6.4)$$

其中參數 $\Omega = \{w_{ji}^{(1)}, w_{kj}^{(2)}, w_{j0}^{(1)}, w_{k0}^{(2)}\}_{i=1,2,\dots,p; j=1,2,\dots,q; k=1,2,\dots,r}$, 共有 $pq + qr + q + r$

個參數組合。當式 (6.4) 中的參數組合愈多，則類神經網路將會愈複雜，可將其視為非線性程度很高的函數。因此類神經網路能將輸入 (X) 與輸出 (Y) 的關係配適的比單純線性函數更佳。

6.2 前饋式類神經網路的實做與評比

6.2.1 兩截式機械手臂 (2-Joint Robot Arm)

圖 6.2 為簡易的兩截式機械手臂，手臂長度分別為 L_1, L_2 。兩截手臂依據兩個角度 θ_1, θ_2 的改變，可以使得手臂尖端的位置 (x, y) 涵蓋如圖 6.3 的陰影範圍。

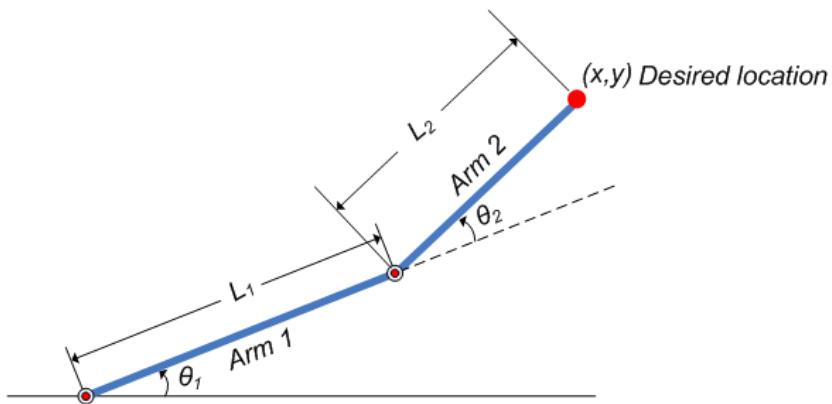


圖 6.2: 兩截式機械手臂 (引用自 MathWorks : Modeling Inverse Kinematics in a Robotic Arm)

在兩截式機械手臂中，給定關節的角度，則正向運動方程式 (式 6.5) 能給出手臂尖端的位置。但現在我們所關注的是給定 (x, y) 座標後，機械手臂是否能計

算出兩個關節的角度 (θ_1, θ_2) ，使得手臂尖端可以移動到 (x, y) ，這便是逆向運動方程式 (式 6.6) 的計算。

Forward Kinematics :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \quad (6.5)$$

Inverse Kinematics :

$$\begin{bmatrix} \theta_2 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \cos^{-1} \left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right) \\ \tan^{-1} \left(\frac{y}{x} \right) - \tan^{-1} \left(\frac{l_2 \sin(\theta_2)}{l_1 + l_2 \cos(\theta_2)} \right) \end{bmatrix} \quad (6.6)$$

對於兩截式機械手臂這樣簡單的結構，逆向運動方程式可以藉由數學推演。但是，當維度提高或手臂關節增加 (例如：在 3 維的立體空間中運行 n 截機械臂)，推演逆向運動方程式變得相當困難。「在此，我們捨棄式 (6.6) 的解析解不用，轉而藉由類神經網路的學習方式，在機器手臂能到達的範圍內找一些位置當作訓練資料 (如圖 (6.3) 的 + 字位置)，讓類神經網路透過這些訓練資料找到輸出角度與輸入位置的關係，再來看看學習過後，當面對新的位置資料 (x, y) 時，是否能準確輸出如式 (6.6) 正確的角度 θ_1, θ_2 ?」²

²資料來源：“淺度機器學習：類神經網路”。汪群超。

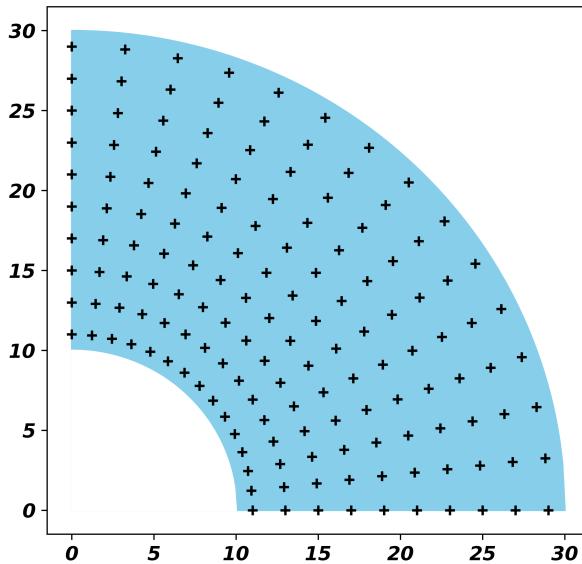


圖 6.3: 兩截式機械手臂所及範圍及訓練資料

本節實作皆採用兩層式的前饋式架構，根據以上機械手臂的原理，神經網路的已知輸入為 (x, y) 座標，輸出為機械手臂兩個關節角度 (θ_1, θ_2) ，屬連續型輸出變數。架構如圖 6.4，其中類神經網路之演算法採用 BFGS 演算法，中間隱藏層與輸出層的激勵函數分別為 TanSig 及 PureLin，其中 TanSig 也叫做雙曲正切函數，如同式 (6.2)。在此先嘗試設定隱藏層為 10 個神經元，了解初始模型的學習情況後，接著可以再加入更多神經元在隱藏層，並觀察輸出的擬合值與真實值之間的差異。直到神經網路的輸出能達到我們所要求的結果。

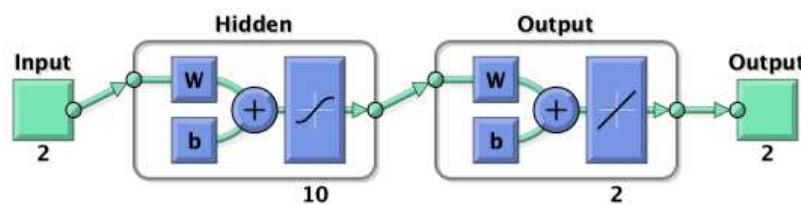


圖 6.4: 兩層的前饋式神經網路架構。資料來源：“淺度機器學習：類神經網路”。汪群超。

圖 6.5，使用均勻分配在二維平面生成亂數，並經過適當剪裁滿足機械手臂的活動範圍。此範圍中共有 45 個樣本，其中 70% 作為訓練資料，30% 作為訓練資料。訓練資料用來訓練 ANN 模型，接著以測試資料檢驗訓練結果。在圖 6.4 的架構下 (隱藏層含 10 個神經元)，此 ANN 模型最終計算出來的測試組

Sum of Squared Error (SSE) 為 0.017024，本章將其作為評估模型預測準確性的指標。SSE 的值越接近於 0 越好，它代表模型的預測值與真實值之間的差距平方總合。另外一種評估方式為，觀察測試資料（○符號）與預測資料（* 符號）的相對位置，如果兩者幾乎貼合在一起，那將代表模型的預測能力很好。若相對遠離，則可能需要重新訓練模型，或是調整模型參數。

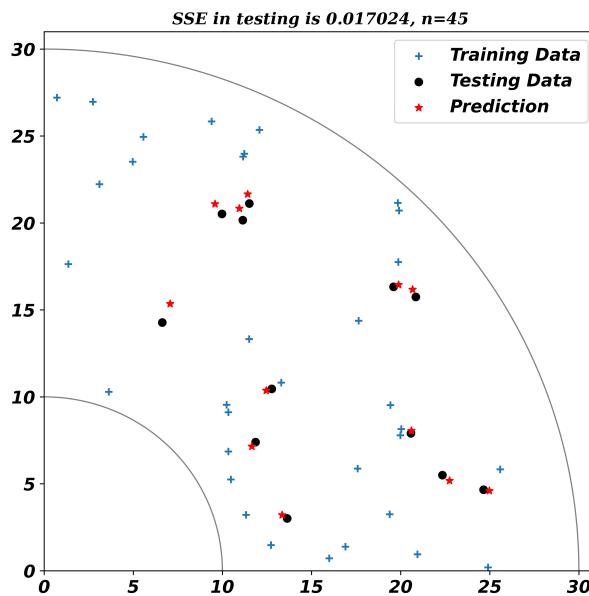


圖 6.5: 總樣本數 $n = 45$ 且一個隱藏層 (含 10 個神經元) 的預測能力

當隱藏層中的神經元數持續增加時，模型的複雜度與訓練時間會逐漸提升。但是模型的表現能力是否因此變得更好則需要討論。所以，接下來探討的是，當樣本數相對小 ($n = 45$) 時，將隱藏層含 10 個神經元的模型推廣至更多神經元的模型。

表 6.1: 四種神經元數量在總樣本數相對小 ($n = 45$) 時的訓練與測試 SSE

	10		30		50		80	
資料集	訓練	測試	訓練	測試	訓練	測試	訓練	測試
總樣本數 ($n = 45$)	0.0094	0.0170	0.0089	0.2417	0.0091	0.2561	0.0098	0.3115

根據圖 6.6，四個不同隱藏層神經元數 (10, 30, 50 ,80) 下的預測情況都不太理想，因為測試資料 (○符號) 與預測資料 (* 符號) 的相對位置都有明顯差異，只有當神經元數為 10 時好一些。而表 6.1 為四種情況下 ANN 模型的訓練群與測試群 SSE。可以發現測試群 SSE 隨著神經元數提升而變大，也就是模型的預測值與真實值之間的差距變遠。特別注意的是，訓練群 SSE 都保持在較低的水準，也就是模型把訓練資料訓練得相當好。

綜合以上的觀察結果。在總樣本數量較小的情況下，隱藏層中的神經元數增加反而降低模型整體的表現能力。雖然訓練資料匹配優良，但測試群的匹配卻不佳。由統計學的觀點，此種狀況稱為過度擬合 (Overfitting)。

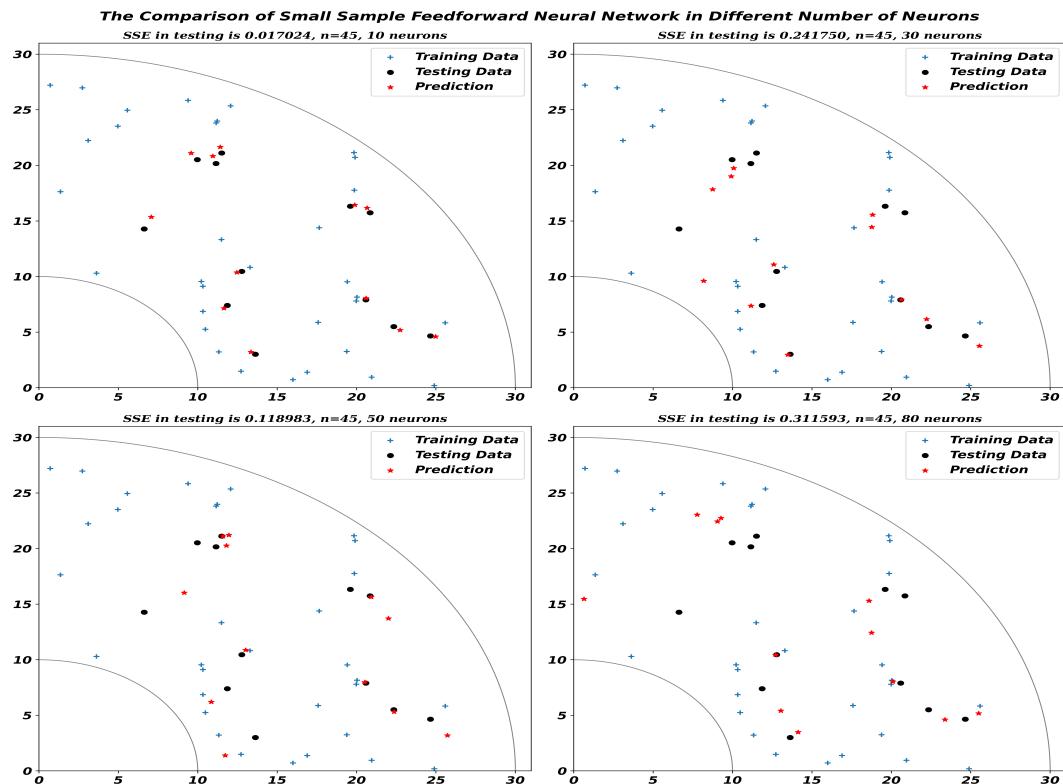


圖 6.6: 當總樣本數量較小時不同隱藏層神經元數之間的 ANN 模型準確度

圖 6.6 的實作在樣本數及神經元數上的設定比較罕見，因為正常情況下神經元數並不會設定的比樣本數還大，除了耗費時間外還有可能造成反效果。所以在此將均勻樣本總數提高至 $n = 564$ ，隱藏層神經元數設定 (10, 30, 50 ,80)，重新進行 ANN 模型訓練與測試。

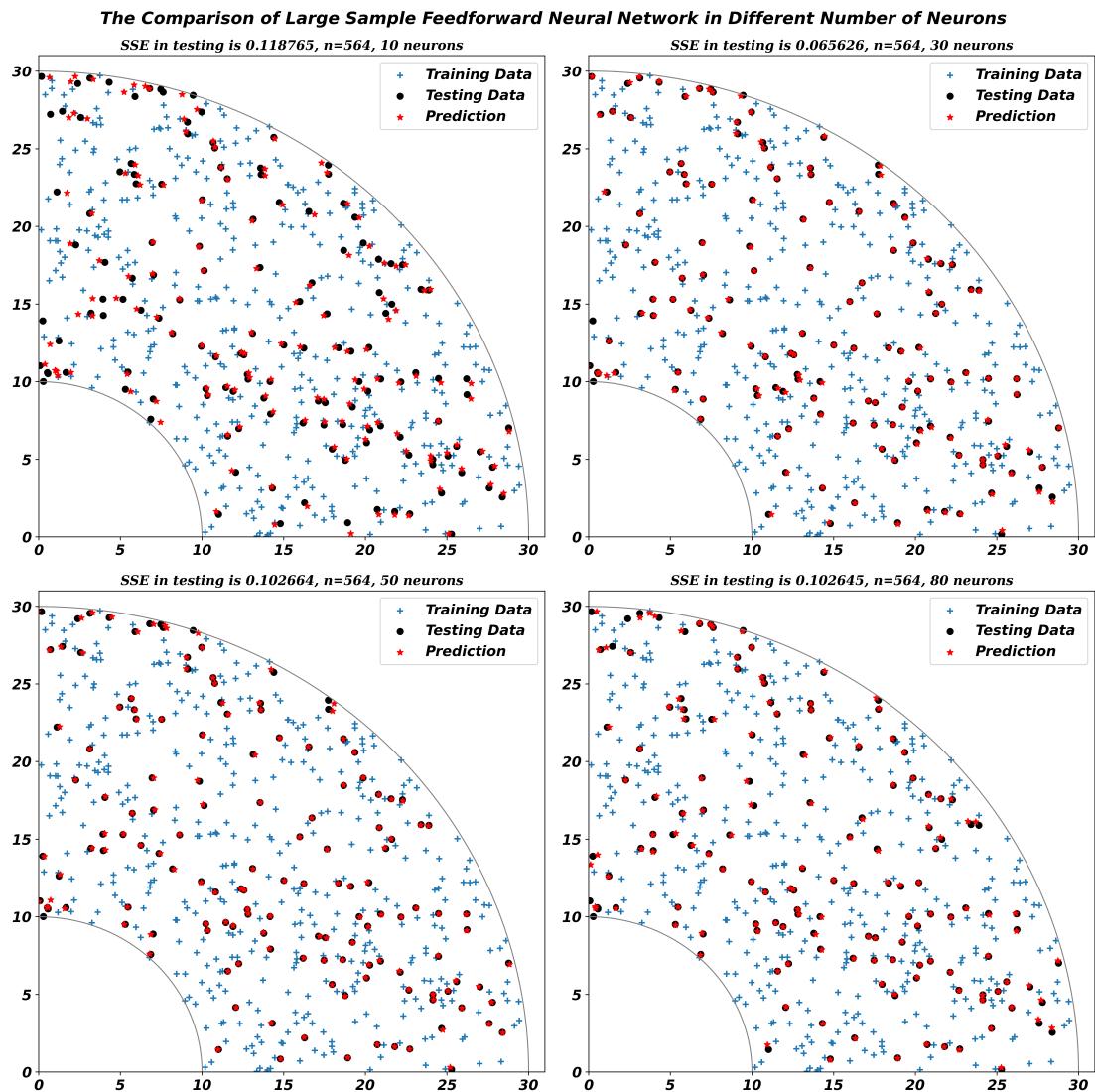


圖 6.7: 當總樣本數量較大時不同隱藏層神經元數之間的 ANN 模型準確度

圖 6.7 為總樣本數提高後的模型學習結果，可以發現除了隱藏層神經元數為 10 時，預測資料 (* 符號) 都相當靠近或貼合於測試資料 (○ 符號)。其餘三種狀況與圖 6.6 小樣本時的學習狀況相比，優良許多。而表 6.2 為四種情況下 ANN 模型的訓練群與測試群 SSE。可以發現訓練群 SSE 在神經元數為 50 與 80 時控制在 0.0099 不再降低。而在神經元數為 30 時，雖然訓練誤差不是最低，但其測試群 SSE 為 0.0656，與其他狀況相比下為最低的誤差。

綜合以上觀察結果。當總樣本數量提升到 ($n = 564$) 時，隱藏層只設定 10 個神經元可能會使得模型不夠複雜，導致其擁有最高的訓練群與測試群 SSE。特別注意的是，與 30 個神經元相比，當神經元數達到 50 與 80 時，兩者模型

的訓練群 SSE 都下降到 0.0099 不變。但測試群 SSE 却提高至 0.1026，比 30 個神經元時還高。因此，可以合理推斷在模型架構不變的情況下，對於樣本數 ($n = 564$)，設定 30 個神經元較為合適。因為再提高神經元數雖然能提升模型對於訓練樣本的能力，但卻無法優化預測的精準度。

表 6.2: 四種神經元數量在總樣本數相對大 ($n = 564$) 時的訓練與測試 SSE

神經元數	10		30		50		80	
資料集	訓練	測試	訓練	測試	訓練	測試	訓練	測試
總樣本數 ($n = 564$)	0.2250	0.1187	0.0105	0.0656	0.0099	0.1026	0.0099	0.1026

在進行神經網路模型訓練時，擁有正確質量和數量的數據集（樣本）對於獲得準確的結果非常重要。沒有代表性的樣本用於訓練可能無法實現良好的模型性能。因此，接下來我想討論樣本在「品質」上的改變是否會影響 ANN 模型的學習狀況。

於圖 6.8 的左側和右側分別展示使用均勻樣本與不均勻樣本用於 ANN 模型的學習狀況與測試群 SSE。在此我設定雙方都是在同樣的神經網路架構下進行建模，樣本數為 ($n = 150$)，隱藏層神經元數為 20。觀察兩邊圖形後發現預測資料 (* 符號) 於測試資料 (○ 符號) 的貼合程度差異不大，皆有預測資料偏離測試資料的情況。但是均勻樣本下的測試群 SSE 為 0.0077 明顯低於不均勻樣本測試群 SSE。而圖 6.8 下方展示訓練過程中 SSE 下降的趨勢，可以看出使用均勻樣本的 SSE 下降速率比使用不均勻樣本的還快，但是最終 Epochs 次數，使用均勻樣本的次數幾乎為使用不均勻樣本的兩倍。

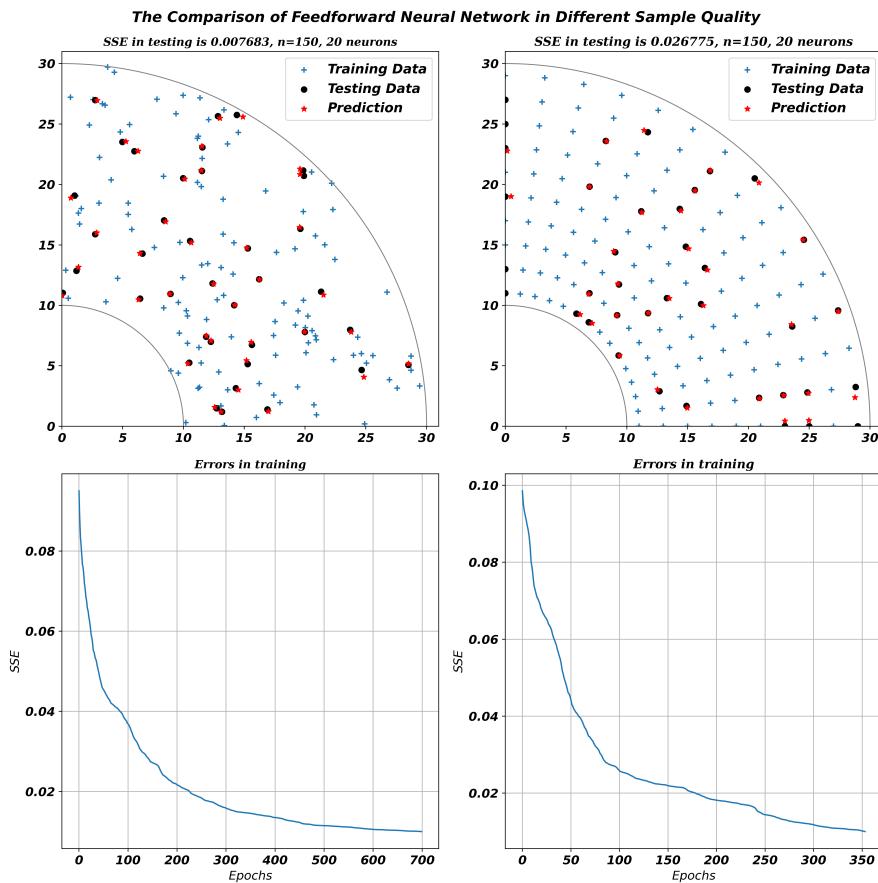


圖 6.8: 均勻樣本與不均勻樣本使用於 ANN 模型的學習結果

綜合以上觀察，使用均勻樣本於 ANN 模型可以獲得比不均勻樣本還高的預測準確率。均勻樣本彼此之間的「獨立」程度較高，而不均勻樣本在內圈時樣本密集，外圈樣本較為發散，使得不均勻樣本彼此之間有相關。這些樣本「品質」上的改變在此實作確實表現出了差異性。

6.2.2 圖形辨識

本節將進行手寫數字辨識實作。因為手寫辨識屬於群組判別，其輸出為類別型資料。所以使用 sklearn 中的神經網路分類器 MLPClassifier 此中多層次感知器做預測。

至於用於訓練與測試的資料採用了 MNIST 數據集，資料集包含 10 個類別（0 到 9）下的 60000 個訓練圖片和 10000 個測試圖片，每個圖片大小是 28 * 28 像素。為了用於訓練與測試，我們只從其中抽取前 600 張圖片，製成如圖 6.9 的 20×30 蒙太奇圖陣（Montage）。

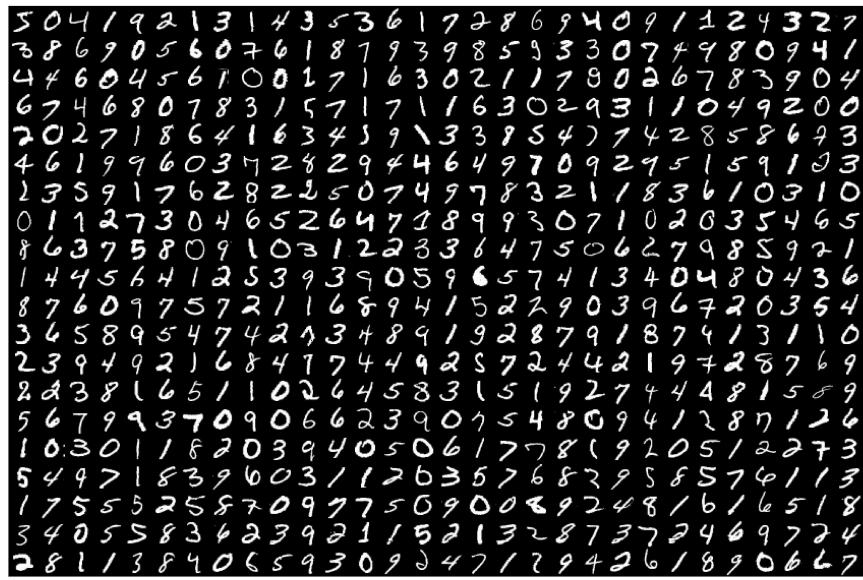


圖 6.9: 手寫數字

將 450 筆訓練資料放入神經網路建模後，圖 6.10 展示剩餘 250 筆測試料的總預測正確率以及各群組的預測正確率。其中數字「2」與「8」的判別表現最差，只有 84% 的「2」與 83% 的「8」。而分析者若想了解數字被認錯的原因可以回去觀察圖 6.9 蒙太奇圖陣。

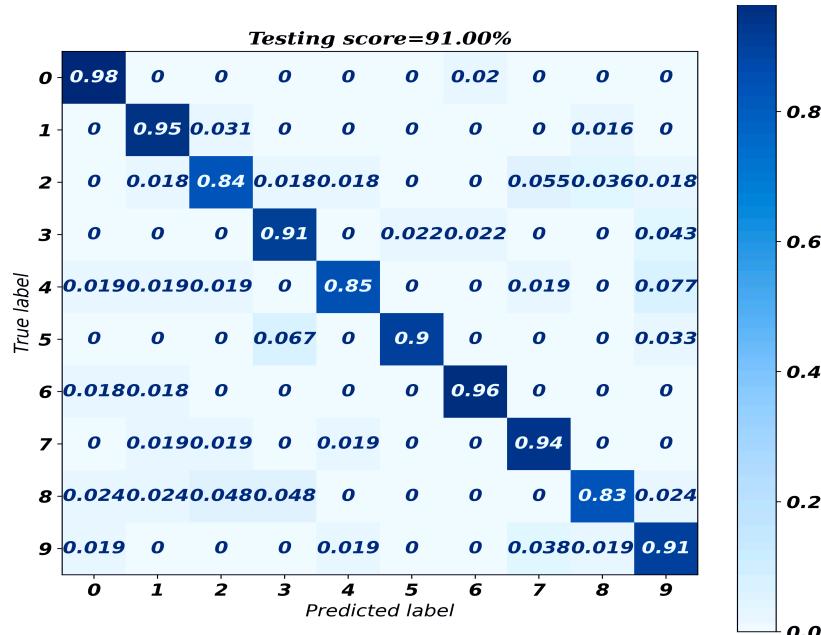


圖 6.10: MLPClassifier 於一個隱藏層（30 個神經）的測試資料的預測能力以混淆矩陣表示

表 6.3 以 30 個神經元為基礎，再加入 (50, 80, 100) 個神經元數量進行比較。可以看到神經元數提升後，訓練正確率都有提升。但是測試正確率在 50 個神經元時下降，而到達 80 與 100 時從 91% 提高到 92% 左右。我認為差異並不大，但模型在訓練時所消耗的資源卻增加很多。因為當神經元數提升時訓練時間會跟著增加。

表 6.3: 四種神經元數量在 MLPClassifier 於一個隱藏層的訓練資料及測試資料正確率

資料集	神經元數	30		50		80		100	
		訓練	測試	訓練	測試	訓練	測試	訓練	測試
手寫數字		92.74%	91.00%	93.11%	90.06%	94.42%	92.31%	94.44%	92.35%

6.3 結語

此篇文章前段主要在介紹類神經網路的原理，後段以機械手臂以及數字辨識進行實作。而類神經網路整體架構有許多參數需要人工調整，例如激活函數、神經元與隱藏層數量等。對於輸入變數型態的不同，需要在有背景知識下對參數進行修改以達到理想的模型學習表現，否則只是枯燥無味的在調整。因此我在實作中加入總樣本數改變、樣本品質改變等，期望能透過測試結果了解參數改變對於模型的影響為何。最後，若能在本文新增神經網路的運算時間以及不同演算法的評比將會更加完整。

第 7 章

蒙地卡羅模擬實驗：學習器的評比

此文將通過蒙地卡羅模擬的方式，對前面提到的學習器 LDA、QDA、羅吉斯回歸、KNN 以及 ANN 做出總體評比。這些學習器各具特點。通過實驗數據對其進行比較和評估，並根據實驗結果提出總結與建議。

7.1 學習器的回顧

首先，讓我們回顧前面提到的每個學習器 LDA、QDA、羅吉斯回歸、KNN 和 ANN 的原理和特點。

- LDA (Linear Discriminant Analysis) 是一種基於線性鑑別函數的分類器。它假設資料服從常態分佈並且各類資料具有相同的共變異矩陣。LDA 是一種經典的非參數分類器，在多元分類問題中表現較好。
- QDA (Quadratic Discriminant Analysis) 是 LDA 的擴展，它允許各類資料具有不同的共變異矩陣。比起 LDA，QDA 更為精確，但計算複雜度較高。
- 羅吉斯回歸是一種基於羅吉斯回歸模型的二元分類器，它可以解決類似羅吉斯回歸的問題，但是有更強的推廣能力。它也可以應用於多元分類，通常稱為多元羅吉斯回歸 (Multinomial Logistic Regression)。
- KNN 是一種基於距離度量的分類器，它將新的測試樣本與訓練集中的樣本進行比較，並決定該樣本的類別標籤。

- ANN 是一種類似人類大腦的網絡，由許多神經元構成，可以自動學習特徵和分類規則。

以上將會是我們總評比的基礎。

7.2 分析學習器在不同情境下的性能

此節將比較具有相同與相異共變異矩陣的兩群和三群資料的表現，並對環型資料集進行獨立分析。

7.2.1 具有相同共變異矩陣的資料

圖 7.1 展示了七種學習器在相同共變異矩陣的兩組資料的學習狀況。另外，根據表 7.1 的結果顯示，5-NN 在訓練時誤判率最低，而在測試時，ANN-80 的誤判率最低。因此，我們可以得出 5-NN 在訓練上效果最佳，ANN-80 在測試上效果最佳。

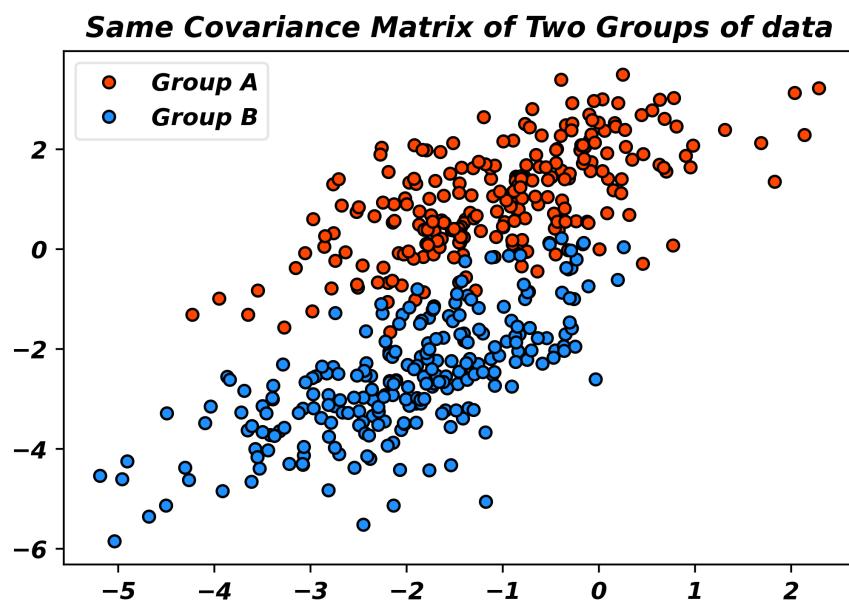


圖 7.1: 相同共變異矩陣的兩群資料

表 7.1: 七種學習器在兩共變異矩陣相同資料的學習誤判率

LDA		QDA		5-NN		15-NN		Logistic		ANN-30		ANN-80	
訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試
2.51%	2.53%	2.66%	2.76%	1.95%	3.11%	2.60%	2.71%	2.57%	2.64%	2.11%	2.14%	1.97%	2.13%

接下來，圖 7.2 展示了七種學習器在相同共變異矩陣的兩組資料的學習狀況。另外，根據表 7.2 的結果顯示，5-NN 在訓練時誤判率最低，而在測試時，QDA 的誤判率最低。因此，我們可以得出 5-NN 在訓練上效果最佳，QDA 在測試上效果最佳。

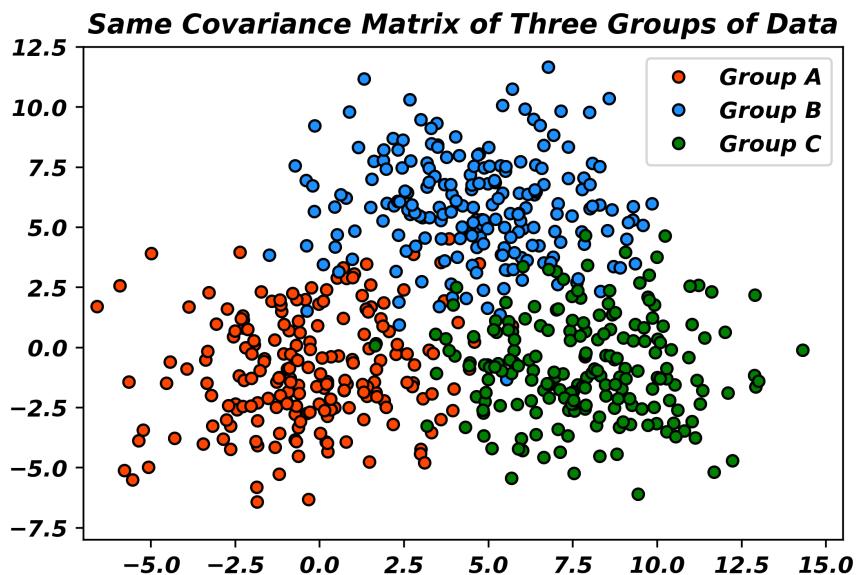


圖 7.2: 相同共變異矩陣的三群資料

表 7.2: 七種學習器在三共變異矩陣相同資料的學習誤判率

LDA		QDA		5-NN		15-NN		Logistic		ANN-30		ANN-80	
訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試
5.80%	6.04%	5.85%	5.89%	5.03%	7.47%	5.35%	6.07%	5.82%	5.94%	5.88%	5.92%	5.86%	5.94%

7.2.2 具有相異共變異矩陣的資料

圖 7.1 展示了七種學習器在相異共變異矩陣的兩組資料的學習狀況。另外，根據表 7.1 的結果顯示，5-NN 在訓練時誤判率最低，而在測試時，ANN-80 的誤判率最低。因此，我們可以得出 5-NN 在訓練上效果最佳，ANN-80 在測試上效果最佳。

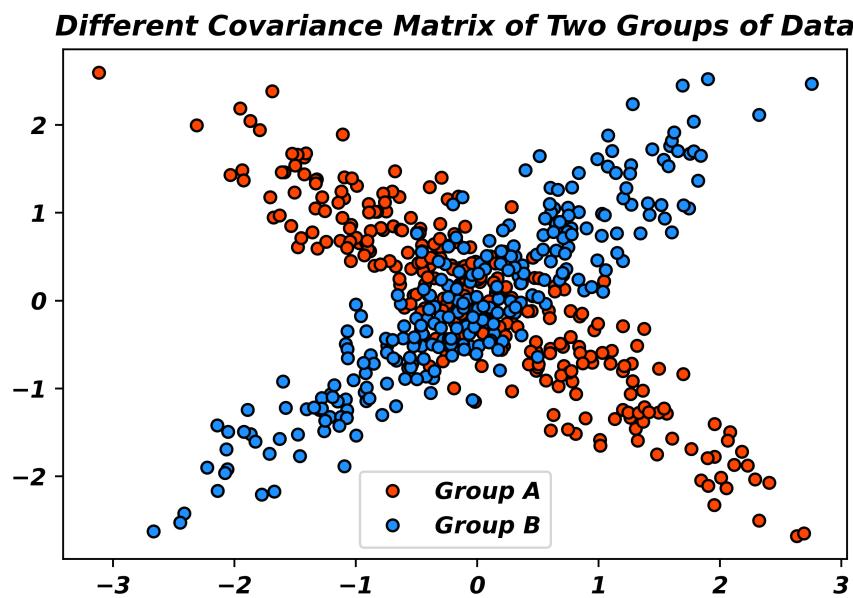


圖 7.3: 相異共變異矩陣的兩群資料

表 7.3: 七種學習器在兩共變異矩陣相異資料的學習誤判率

LDA		QDA		5-NN		15-NN		Logistic		ANN-30		ANN-80	
訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試
48.49%	54.34%	16.04%	15.71%	12.21%	16.94%	15.22%	17.46%	35.66%	40.23%	15.27%	16.03%	15.09%	15.69%

圖 7.4 展示了七種學習器在相異共變異矩陣的三組資料的學習狀況。另外，根據表 7.4 的結果顯示，5-NN 在訓練時誤判率最低，而在測試時，15-NN 的誤判率最低。因此，我們可以得出 5-NN 在訓練上效果最佳，15-NN 在測試上效果最佳。

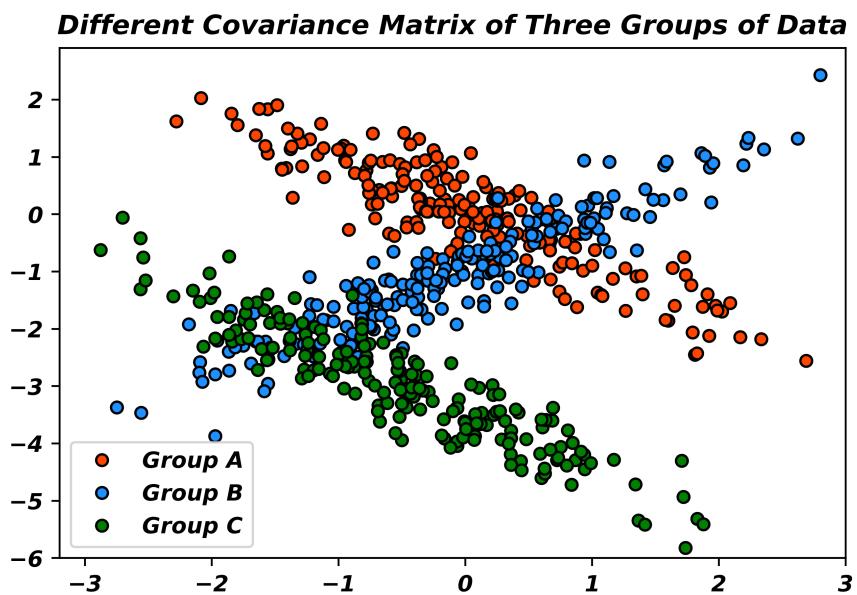


圖 7.4: 相異共變異矩陣的三群資料

表 7.4: 七種學習器在三共變異矩陣相異資料的學習誤判率

LDA		QDA		5-NN		15-NN		Logistic		ANN-30		ANN-80	
訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試
25.86%	26.31%	25.92%	26.18%	9.5%	13.32%	11.09%	12.79%	26.33%	26.81%	15.36%	15.50%	15.33%	15.40%

圖 7.5 展示了七種學習器在雙環資料集的學習狀況。另外，根據表 7.5 的結果顯示，5-NN 在訓練時誤判率最低，而在測試時，ANN-30 的誤判率最低。因此，我們可以得出 5-NN 在訓練上效果最佳，ANN-30 在測試上效果最佳。

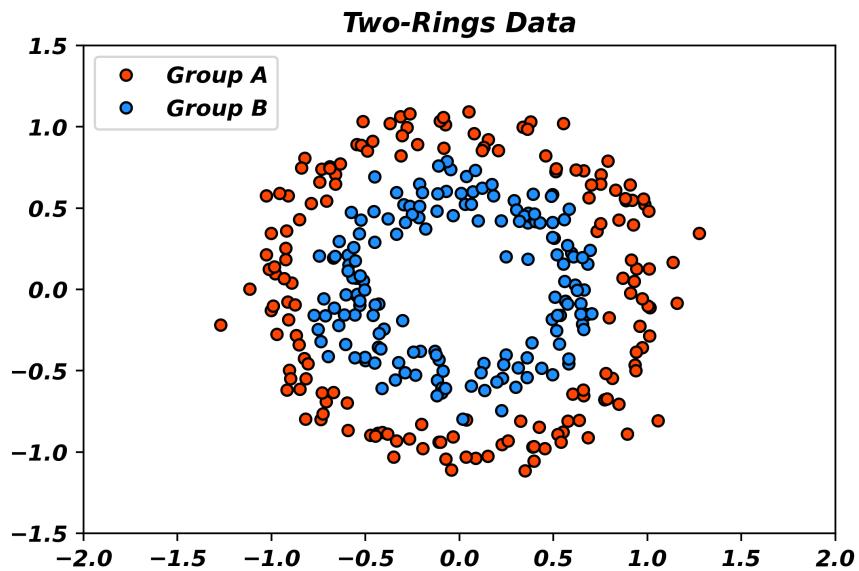


圖 7.5: 雙環資料集

表 7.5: 七種學習器在雙環資料集的學習誤判率

LDA		QDA		5-NN		15-NN		Logistic		ANN-30		ANN-80	
訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試	訓練	測試
47.74%	56.08%	3.13%	4.39%	1.01%	1.82%	1.41%	1.72%	3.53%	4.12%	1.05%	1.23%	1.02%	1.33%

7.3 結論

在本實驗中，我們比較了 KNN、LDA、QDA、ANN、Logistic 學習器之間的整體學習狀況。結果顯示，對於分類問題，KNN 學習器的整體學習表現優於其他學習器，特別是與 ANN 學習器相比，KNN 學習器表現更為優異，但是 ANN 學習器也表現良好。因此，我們可以得出結論，KNN 學習器是一種高效且有效的學習器，而 ANN 也是值得採用的學習器。而 LDA、QDA、Logistic 回歸學習器相對於其他學習器表現較為普通，但仍有討論的價值。