H00

機器學習(ML)
深度學習(DL)
人工智慧(AI)
的
數學基礎

吳漢銘
國立政治大學 統計學系

https://hmwu.idv.tw

"Artificial Intelligence is All About Math"

科學月刊 SCIENCE MONTHLY
關於科月 | 購買科月 | 訂閱科月 | 投稿須知 | 各期
封面故事 News Focus 專訪 專欄 科技報導 評論 精選文章與其他 活動訊息

2018年3月27日

人工智慧浪潮下的數學教育

魏澤人／任教於國立東華大學，創立花蓮-py社群及實做數學粉專。

我們曾學過的數學，究竟對人生有什麼幫助？

第一個尷尬點是，數學的實用性變得十分明顯，明顯到令人尷尬。學數學的人，常會聽到人問：「學數學有什麼用？」、「我高中數學都忘光了，還不是活得好好的？」、「工作上好像完全沒用到。」、「我寫程式這麼久了，也沒用到什麼數學。」

當然，內行人都知道數學在科學、技術、工程中，應用十分廣泛。特別在網路時代，網路加密、電腦運算、影像壓縮，甚至網頁和應用程式的自動排版都得用到數學。即使連照片編修這種屬於藝術文化的活動，其中的圖層操作就包含了向量概念，調色盤會看到16進位及顏色轉換的

拉長、內容感知

了傅立葉轉換和

電腦工程師們重新拿起統計、微積分及線性代數課本，想要了解現代的人工智慧在玩什麼把戲。現代人工智慧的領軍人物之一勒丘恩（Yann LeCun）說「人工智慧就是數學（artificial intelligence is all about math）」，他給想從深入人工智慧領域大學生的建議是：「如果在『iOS程式設計』及『量子力學』中要選一門課來修的話，選量子力學，且一定要選修微積分一、微積分二、微積分三……、線性代數、機率與統計，和盡可能的多選物理課程。即便如此，最重要的還是要會寫程式。」

楊立昆

電腦科學家

楊立昆，法國籍計算機科學家，他在機器學習、計算機視覺、移動機器人和計算神經科學等領域都有很多貢獻。他最著名的工作是在光學字符識別和計算機視覺上使用卷積神經網絡，他也被稱為卷積網絡之父。他同Léon Bottou和Patrick Haffner等人創建了DjVu圖像壓縮技術。 維基百科

出生資訊： 1960 年 7 月 8 日（60歲），法國蘇瓦西蘇蒙特莫朗西

獲頒獎項： 圖靈獎 (2018); AAAI Fellow (2019); 法國榮譽軍團勳章 (2020)

論文： Modèles connexionnistes de l'apprentissage (1987)

模範生： 沃伊切赫·薩倫巴

獲獎記錄： 圖靈獎

學歷： 巴黎第六大學 (1983 年–1987 年)， 巴黎高等電子工程師學校， 索邦大學

勒丘恩：「人工智慧完全是數學。」（Wikipedia）

人工智慧只是統計學的延伸

每日頭條

首頁 | 健康 | 娛樂 | 時尚 | 遊戲 | 3C | 親子 | 文化 | 歷史 | 動

AI（人工智慧）就是統計學？

2018-10-26 由 素思生涯規劃 發表于科技

諾貝爾經濟學獎托馬斯·薩金特在《財經》世界科技創新論壇上的演講中說過一句話：人工智慧首先是一些很華麗的辭藻。人工智慧其實就是統計學，只不過用了一個很華麗的辭藻，其實就是統計學。

https://kknews.cc/zh-tw/tech/5z36z28.html

AI 不過是統計學

Thomas J. Sargent ：人工智慧只是統計學的延伸

2011年諾貝爾經濟學獎取得者Thomas J. Sargent在題為"共享全球智慧 引領將來科技"的世界科技創新論壇上表示：

計算機是非常擅長計算，它們可以非常快速地完成計算人算不了的東西，但最終必須由人來組織和分析這些計算。你可以看一些非常成功的人工智慧應用，它不僅是機器在「思考」，也是科學家在思考。像 AlphaGo 的演算法看上去是第一次出現，但其實有很多非常聰明的數學，並且是由人設定教學內容。人工智慧是由機器和人分飾兩角的，非常有趣。

AI時代的中層支柱：統計學: https://www.mdeditor.tw/pl/2nBY/zh-tw

但是為什麼不說搞統計學呢？很簡單，因為不如人工智慧說法高大上，為什麼要高大上？因為高大上有人投錢。人工智慧代表了最新科技、最熱行業，能吸引投資，你要說是做統計學的誰投錢？誰買單？實際上這些工作很早就有人研究，只是那時候都歸類於統計學領域。

就像顯示系統縱橫位置指示器就是滑鼠；人體表皮污垢學就是搓灰；人體表皮死細胞分離器就是搓澡巾；智能高端數字通訊設備表面高分子化合物線性處理就是手機貼膜……

所以現在大家都學聰明了，那怕是老生常談也得包裝一個好聽的名稱，你說搞機器學習、深度學習就有人投錢，有人出大價錢挖你，你要說搞統計學大家立馬就不感興趣了，其實做的還是一回事。當然資本也是知道這些道理的，那為什麼還要投錢，因為資本是逐利的，投錢是為了掙更多錢，高大上的外衣就是掙錢的保障之一。

人工智慧和統計學不能完全劃等號

人工智慧和統計學存在莫大的關係，或者說統計學是人工智慧的最重要的理論基礎，但統計學和人工智慧依然有著很大不同，更不是一回事。

科技領域都需要數學

聯合新聞網

願景 時尚 汽車 NBA台灣 運動筆記 遊戲 國際 鳴人堂 新鮮事 部落格 全部

即時 要聞 娛樂 運動 全球 社會 專題 產經 股市 房市 健康 生活 文教 評論 地方 兩岸 數位 旅遊 閱讀 雜誌 購物

udn / 生活 / 職場觀測 相關新聞

郭董指數學很重要 台師大教授：數學人才出路好的時代來了

分享 分享 留言 列印 存新聞 A- A+

2019-03-01 21:28 聯合報 記者陳智華／即時報導 讚 7,878 分享

在美國，數學家的出路非常好，台灣這樣的時代也要來了嗎？

台師大電機工師系助理教授、數學專欄作家賴以威今在臉書貼文指出，鴻海集團董事長郭台銘說：「數學很重要！」他因之前受邀鴻海跟高階主管演講時，親耳聽郭董對總部3、4 百位高階主管、十幾個遠端連線的各地分公司主管員工這麼說。

賴以威聽到郭董跟員這樣說時，才知道郭董找他去演講是為了推廣數學，推廣成長型數學思維。

郭董指出，工業物聯網、人工智慧、資料分析，這些鴻海現今著重的科技領域，背後都需要數學。因此，讓集團意識到數學的重要性，知道如何學好數學的心態是非常重要的。

他表示，演講後的隔天，郭董跟他聊到，鴻海很歡迎電機系和數學系的人才加入，與相關的產學合作。他這兩天認識許多裡面的同事、主管也是相關科系。

賴以威說，美國就業網站 CareerCast 曾經統計過， 在美國數學人才有非常好的工作機會。在台灣，看來這樣的時代也要來臨了。

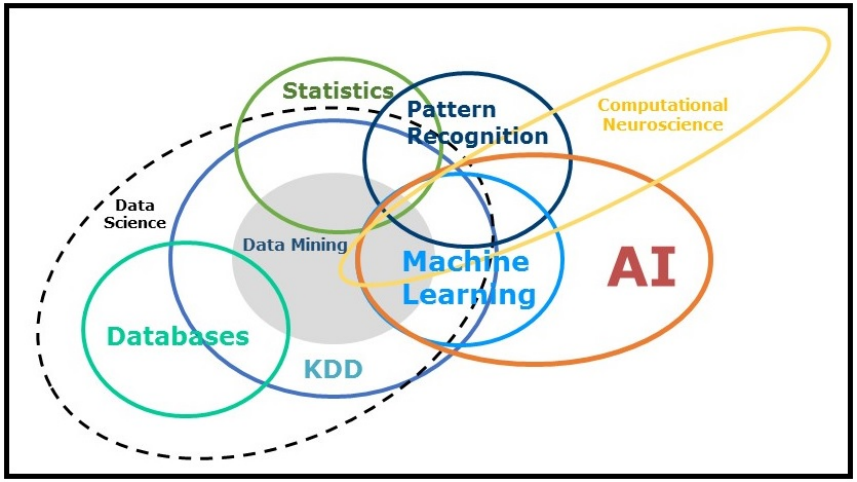
但賴以威強調，數學很重要，這指的不是程序性的計算，而是懂得活用的數感。

賴以威說：「郭台銘董事長都這樣說了，你不覺得嗎?」

https://udn.com/news/story/7266/3672385

Statistics, Data Mining, Machine Learning, Deep Learning, and AI

- **Statistics (STAT)**: Statistics is the discipline that concerns the collection, organization, analysis, interpretation and presentation of data.
- **Data Mining (DM)**: Data mining is a process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.
- **Machine Learning (ML)**: Machine learning (ML) is the study of computer algorithms that improve automatically through experience. Machine learning algorithms build a mathematical model based on the training data, in order to make predictions or decisions.
- **Deep learning (DL)**: Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning.
- **Artificial intelligence (AL)**: the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

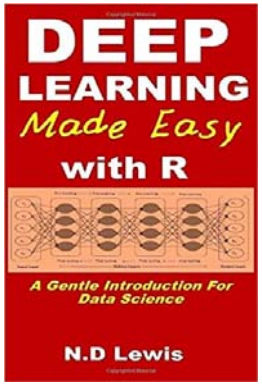


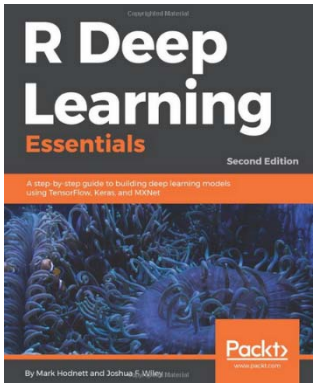
- **人工智慧**是讓機器具有如同人類甚至更多的思辨能力。
- **機器學習**則是能夠達成人工智慧的方法，透過與人類相似的學習方法，訓練機器進行資料分類、處理與預測。
- **深度學習**代表實現機器學習的一種技術。

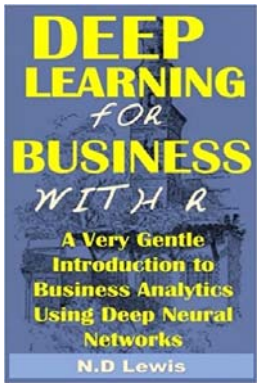
https://jamesmccaffrey.wordpress.com/2016/09/29/machine-learning-data-science-and-statistics/

Deep Learning with R

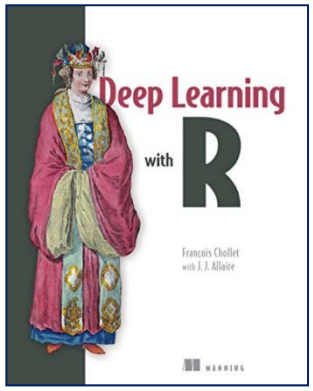
2016

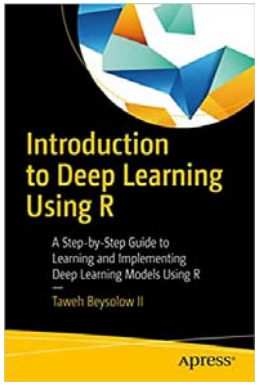


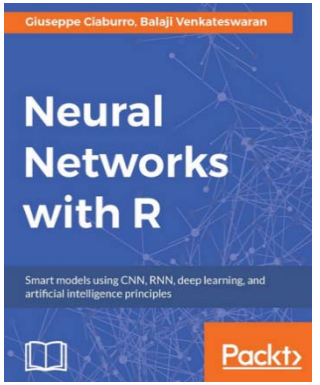


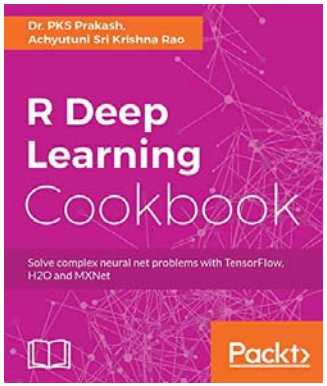


2017

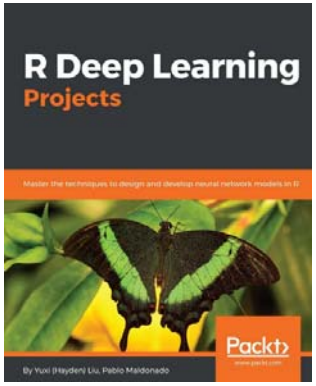




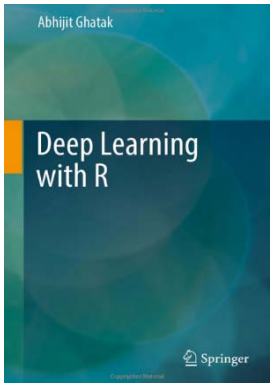


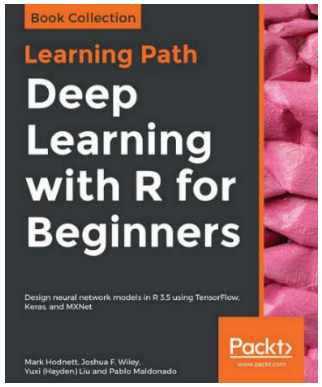


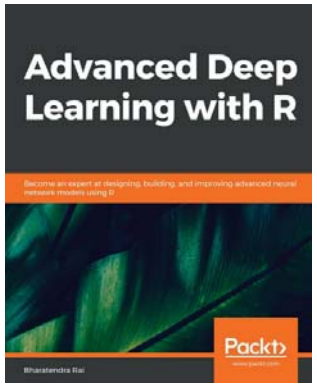
2018

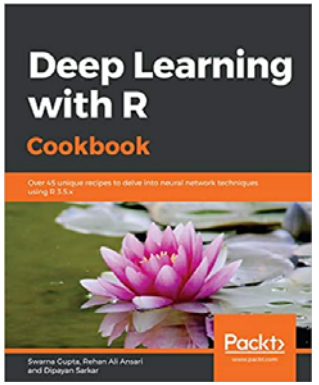


2019









2020

書籍: 深度學習的數學

[日]涌井良幸、涌井貞美

深度學習的數學

出版商: 人民郵電

出版日期: 2019-05-01

語言：簡體中文

深度學習的數學

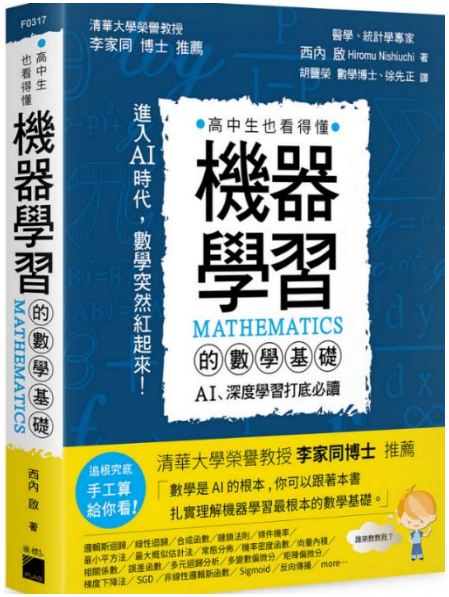
作者： 涌井良幸, 涌井貞美

譯者： 楊瑞龍

出版社：博碩

出版日期：2020/05/04

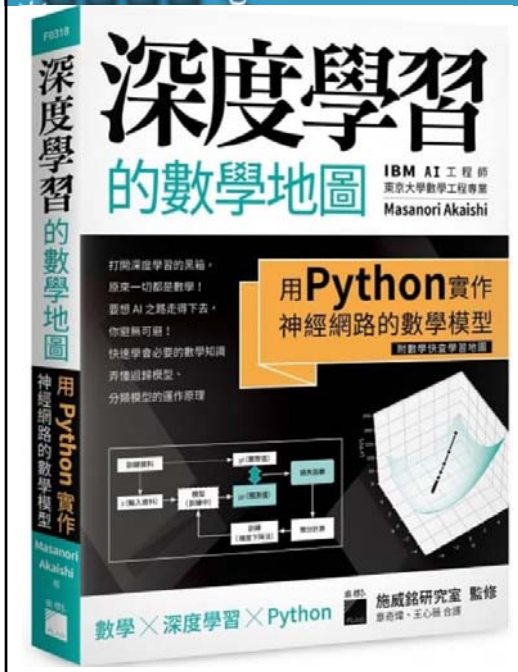
書籍: 機器學習的數學基礎



機器學習的數學基礎：
AI、深度學習打底必讀
醫學統計學專家 西內啟 著
胡豐榮博士, 徐先正 合譯，
出版商: 旗標科技出版
(2020-01-31)

目錄：

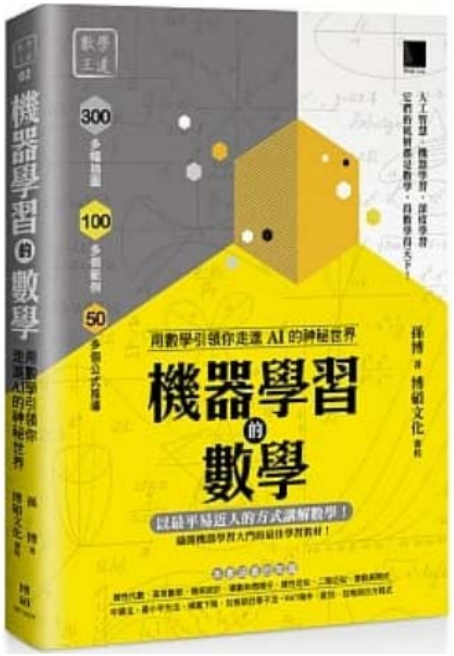
書籍: 深度學習的數學地圖



深度學習的數學地圖：
用 Python 實作神經網路的數學模型
作者： Masanori Akaishi
譯者： 章奇煒, 王心薇
出版社：旗標
出版日期：2020/05/28

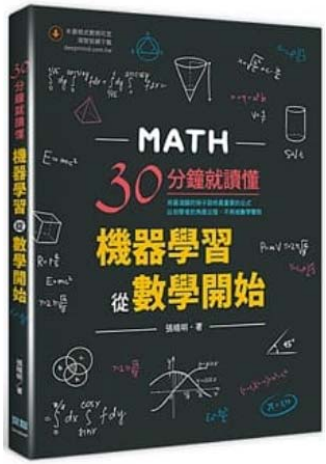
目錄

書籍: 機器學習的數學



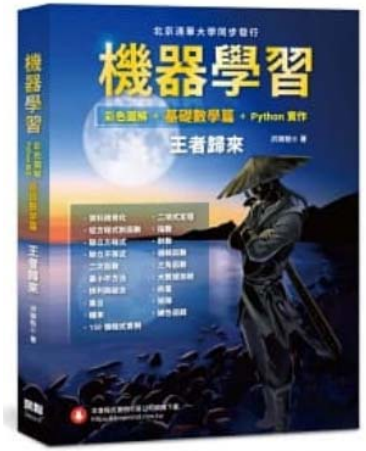
機器學習的數學：
用數學引領你走進AI的神秘世界
作者： 孫博
譯者： 博碩文化
出版社：博碩
出版日期：2020/09/09

目錄



30分鐘就讀懂：
機器學習從數學開始
作者： 張曉明
出版社：深智數位
出版日期：2020/11/21

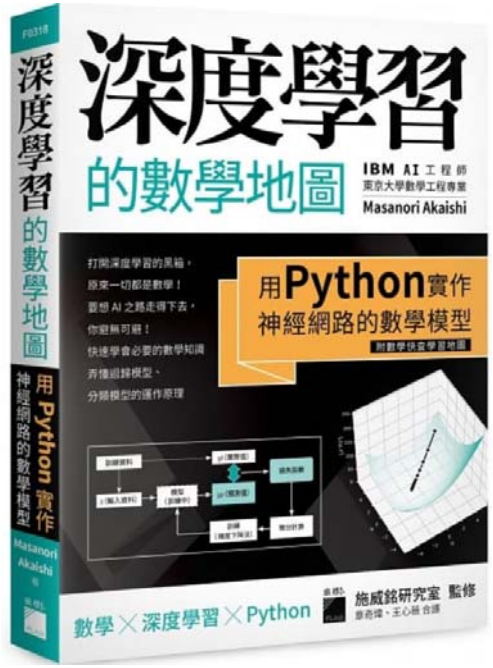
- 第1篇 線性代數: 向量、矩陣、多項式回歸、嶺回歸、Lasso回歸、矩陣分解。
- 第2篇 機率: 最大似然估計、最大後驗機率、貝氏定理。
- 第3篇 最佳化: 凸最佳化、梯度下降演算法、邏輯回歸演算法。



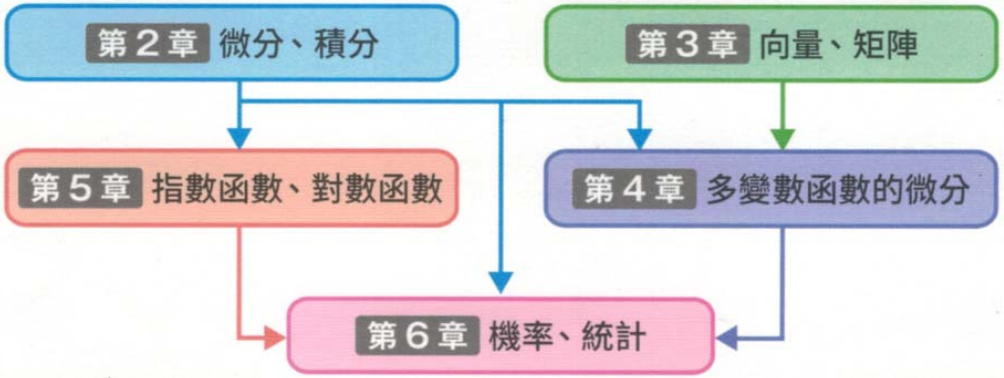
機器學習：彩色圖解 +
基礎數學篇 + Python實作
作者： 洪錦魁
出版社：深智數位
出版日期：2020/08/17

資料視覺化、、認識方程式/函數/座標圖形、畢氏定理、聯立不等式、二次函數、最小平方法、集合、排列與組合、機率、二項式定理、指數觀念與指數函數、歐拉數與邏輯函數、三角函數、向量、矩陣、多元線性回歸、

理論篇: 數學速學課程



深度學習的數學地圖：
用 Python 實作神經網路的數學模型
作者： Masanori Akaishi
譯者： 章奇煒, 王心薇
出版社：旗標
出版日期：2020/05/28



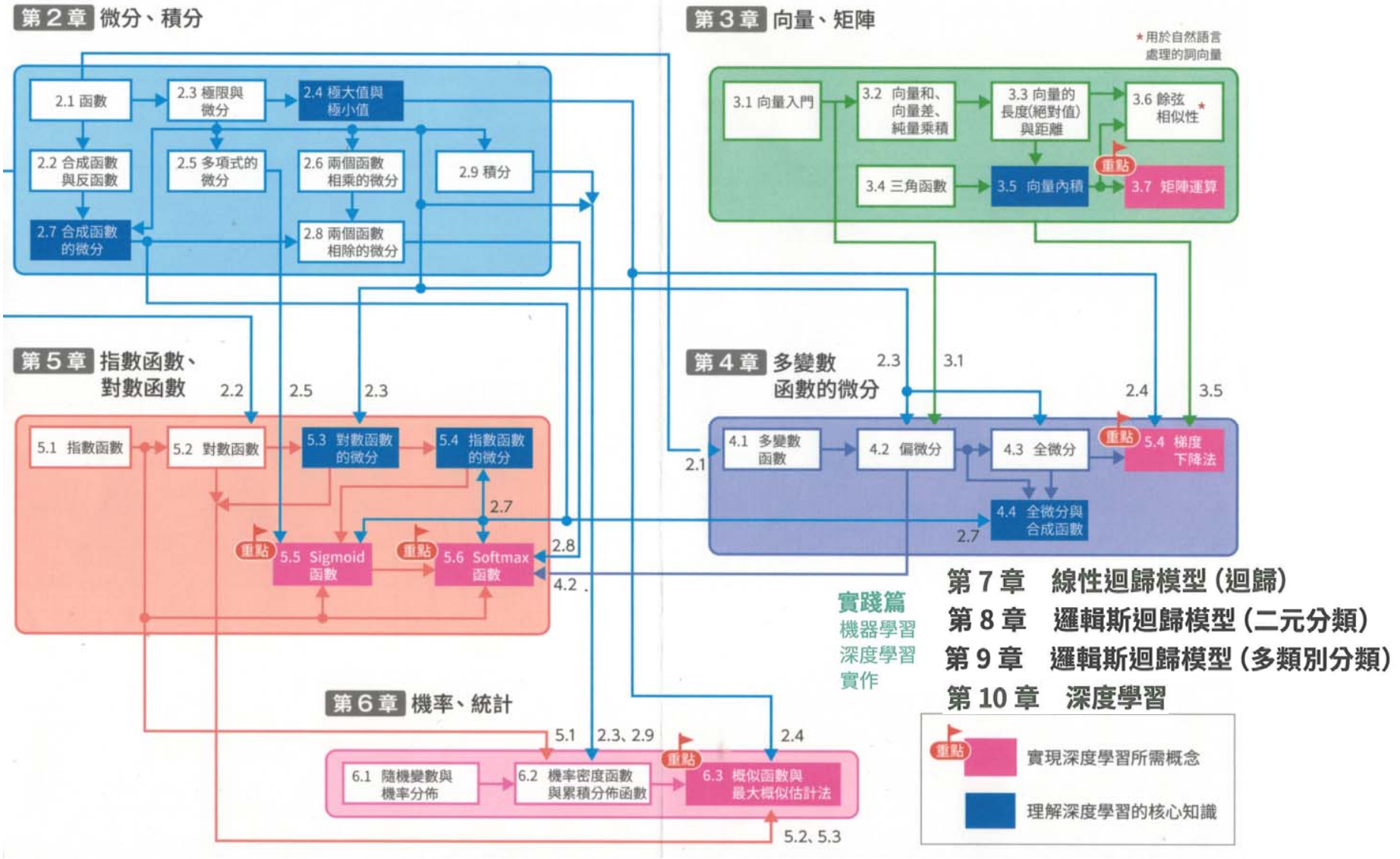
實踐篇 機器學習、深度學習實作

重點

實現深度學習所需概念	第 1 章 迴歸 1	第 7 章 迴歸 2	第 8 章 二元分類	第 9 章 多類別分類	第 10 章 深度學習
1 損失函數	○	○	○	○	○
3.7 矩陣運算				○	○
4.5 梯度下降法		○	○	○	○
5.5 Sigmoid 函數			○		○
5.6 Softmax 函數				○	○
6.3 概似函數與最大概似估計法			○	○	○
10 反向傳播					○

＊ 迴歸 1 是指簡單線性迴歸，迴歸 2 是指多元線性迴歸

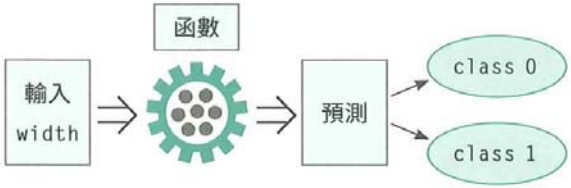
數學理論學習地圖


第 2 章 微分、積分
2.1 函數
2.3 極限與微分
2.4 極大值與極小值
2.2 合成函數與反函數
2.5 多項式的微分
2.6 兩個函數相乘的微分
2.9 積分
2.7 合成函數的微分
2.8 兩個函數相除的微分
第 3 章 向量、矩陣
*用於自然語言處理的詞向量
3.1 向量入門
3.2 向量和、向量差、純量乘積
3.3 向量的長度(絕對值)與距離
3.6 餘弦相似性 *
3.4 三角函數
3.5 向量內積
重點
3.7 矩陣運算
第 5 章 指數函數、對數函數
2.2
2.5
2.3
5.1 指數函數
5.2 對數函數
5.3 對數函數的微分
5.4 指數函數的微分
2.7
重點
5.5 Sigmoid 函數
重點
5.6 Softmax 函數
2.8
4.2
第 4 章 多變數函數的微分
2.3
3.1
2.4
3.5
2.1
4.1 多變數函數
4.2 偏微分
4.3 全微分
重點
5.4 梯度下降法
4.4 全微分與合成函數
2.7
第 6 章 機率、統計
5.1
2.3、2.9
2.4
6.1 隨機變數與機率分佈
6.2 機率密度函數與累積分佈函數
重點
6.3 概似函數與最大概似估計法
5.2、5.3
實踐篇
機器學習
深度學習
實作
第 7 章 線性迴歸模型 (迴歸)
第 8 章 邏輯斯迴歸模型 (二元分類)
第 9 章 邏輯斯迴歸模型 (多類別分類)
第 10 章 深度學習
重點
實現深度學習所需概念
理解深度學習的核心知識
發展篇 實務上的解決方法 第 11 章 以實用的深度學習為目標

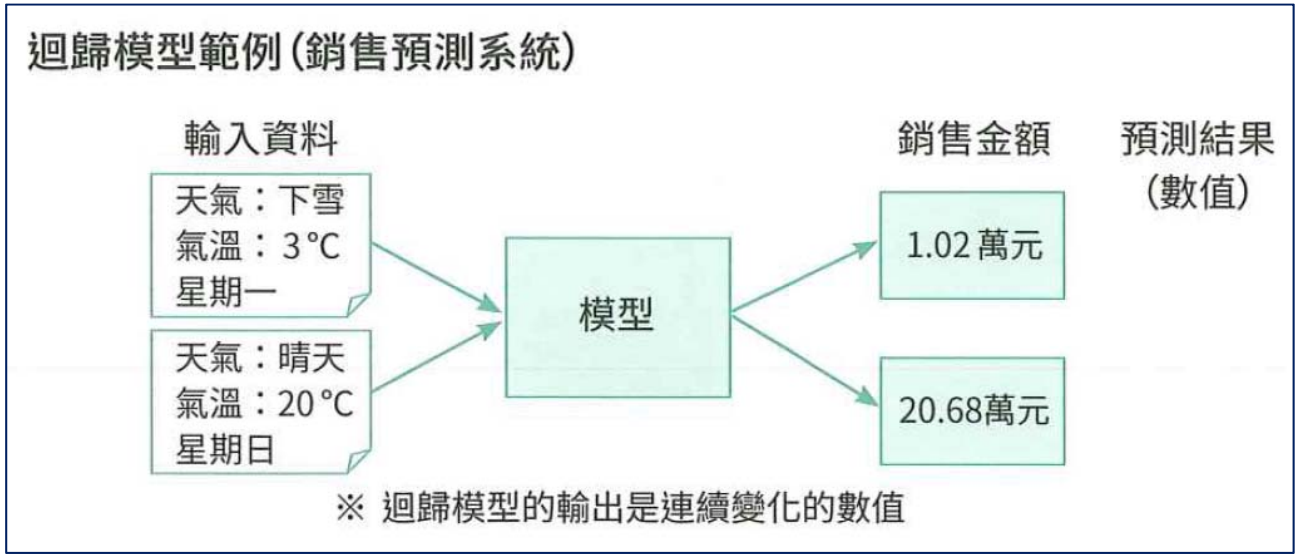
什麼是機器學習模型

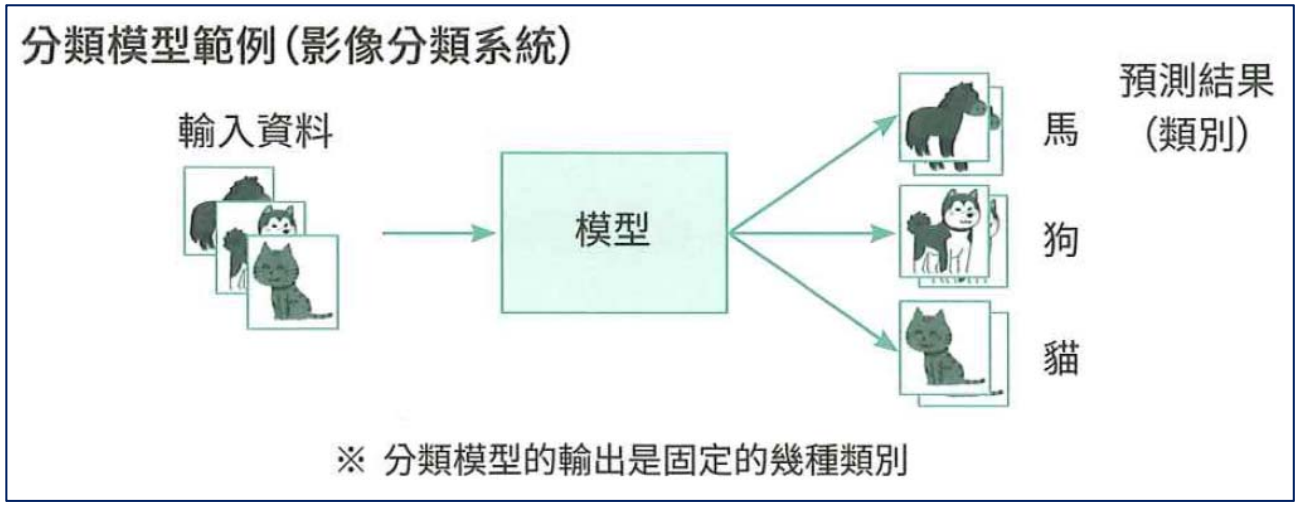
class（類別）	length (cm)	width (cm)
0	1.4	0.2
1	4.7	1.4
0	1.3	0.2
1	4.9	1.5
0	1.4	0.2
1	4.9	1.5

表 1-1　鳶尾花 2 個品種的花瓣大小



- 原則 1:機器學習模型是一個函數，它將輸入資料映射成預測資料。
- 原則 2 :機器學習模型的函數行為，是由訓練決定的。





訓練是機器學習的核心，其具體做法有三種

監督式學習 (supervised learning)

- 用大量的資料來訓練模型，每筆資料都搭配一個標準答案(又稱 teaching data)，模型的預測結果會和標準答案做比對，以修正模型參數，進而達成學習效果。
  - **迴歸(regression)模型**: 輸出值為連續變化的數值，例如預測商店的每日銷售額;
  - **分類(classification)模型**: 輸出值為離散值，稱為「類別(class)」, 例如辨別照片中的動物是甚麼種類。

非監督式學習 (unsupervised learning)

- 只提供訓練資料，不提供標準答案，讓模型自行摸索出資料的規則並產生預測(答案)的訓練法。例如 :僅根據資料的特性，將資料自動分組的「分群(clustering)」便是一種典型的非監督式學習。

強化式學習 (reinforcement learning)

- 沒有訓練資料與標準答案，讓機器從與環境立動中去學習，進而擬定行動對策。由於沒有已知的大量資料，適合用於探索未知的領域。在學習中，藉由犯錯給予處罰，未犯錯給予獎勵的方式，讓機器去探索出正確的答案。例如：打敗棋王的AlphaGo。

監督式學習由訓練階段 (learning phase)與預測階段 (prediction phase)組成

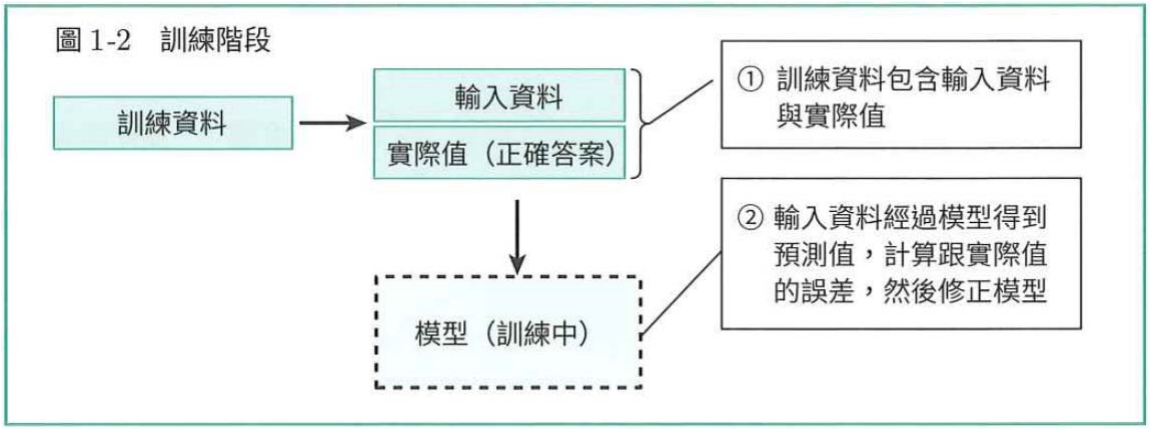


圖 1-2　訓練階段

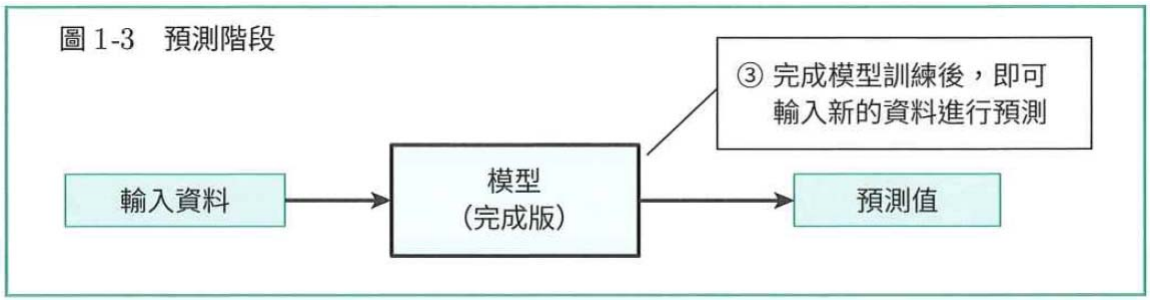


圖 1-3　預測階段

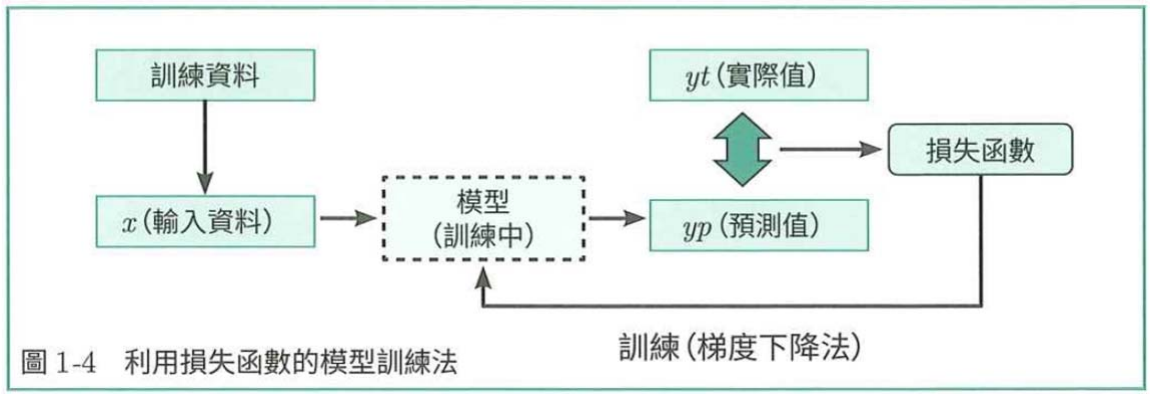


圖 1-4　利用損失函數的模型訓練法

第一個機器學習模型: 簡單線性迴歸模型

身高 x (cm)	體重 y (kg)
166	58.7
176	75.7
171	62.1
173	70.4
169	60.1

表 1-2　訓練資料 1

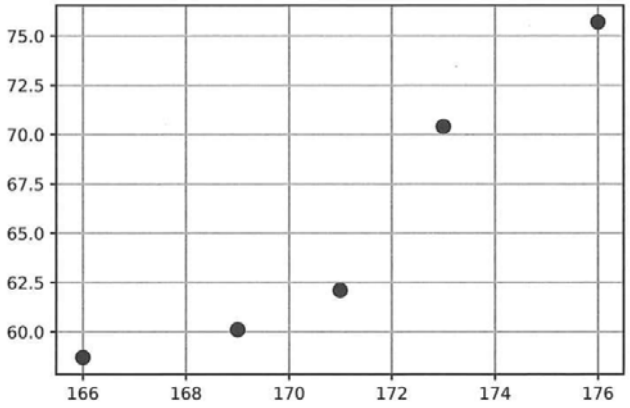


圖 1-5　訓練資料的散佈圖

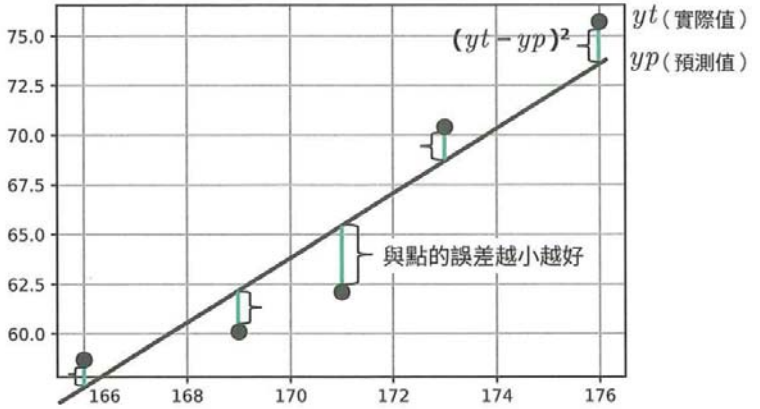


圖 1-6　顯示實際值與預測值之間的誤差

$$(y_1, x_1), \cdots, (y_n, x_n) \qquad y = w_0 + w_1 x$$

$$e_i = y_i - \hat{y}_i \qquad \hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

最小平方法: 計算各樣本點實際值與預測值的差值，個別取平方之後再加總，亦即「誤差平方和J'並令其為損失函數，再利用「最小平方法」求出讓損失函數最小的參數值。

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\frac{\partial S(\beta_0, \beta_1)}{\partial \beta_0} = 0$$

$$\frac{\partial S(\beta_0, \beta_1)}{\partial \beta_1} = 0$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

沒有隱藏層的分類模型結構

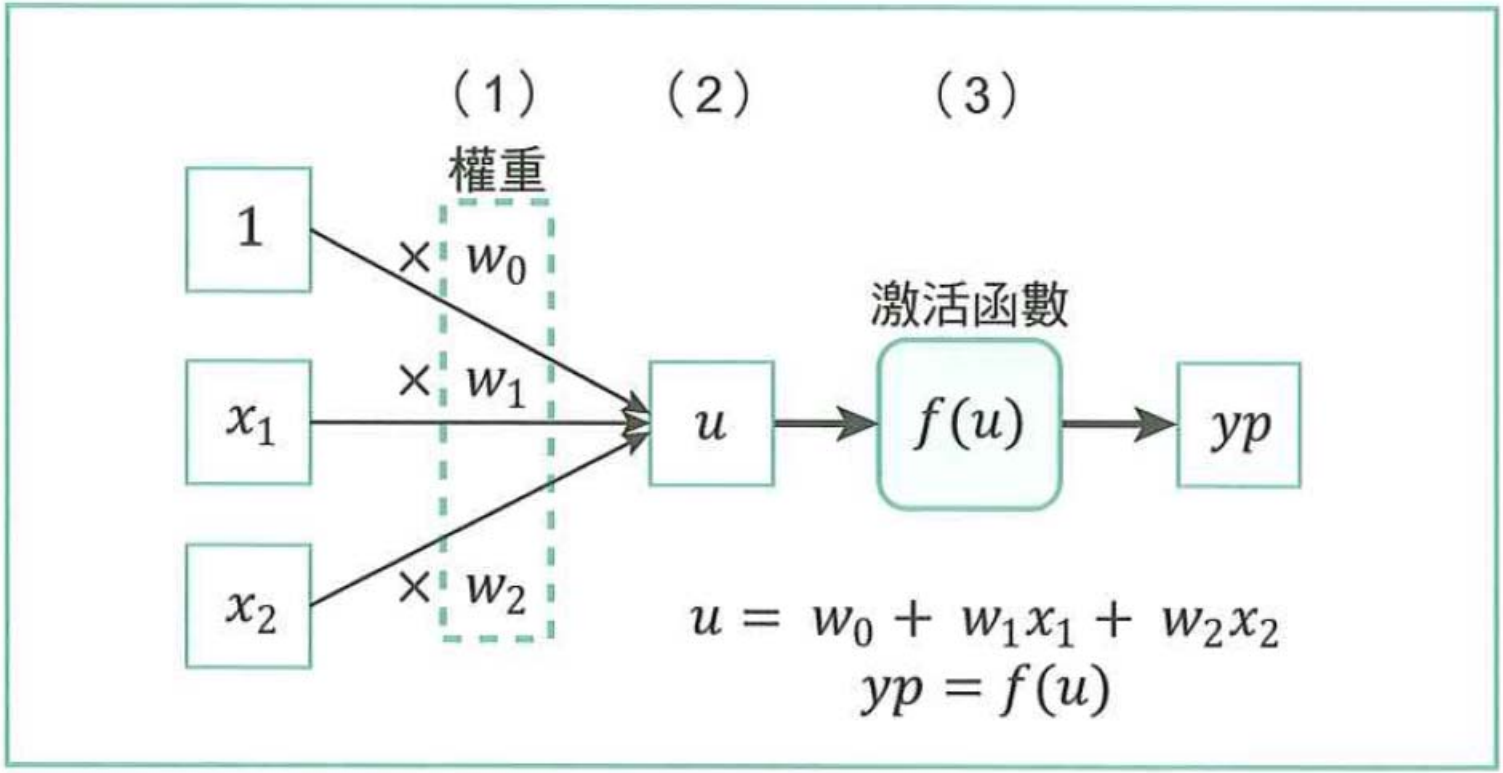


圖 1-12　沒有隱藏層的分類模型結構

有隱藏層的分類模型結構

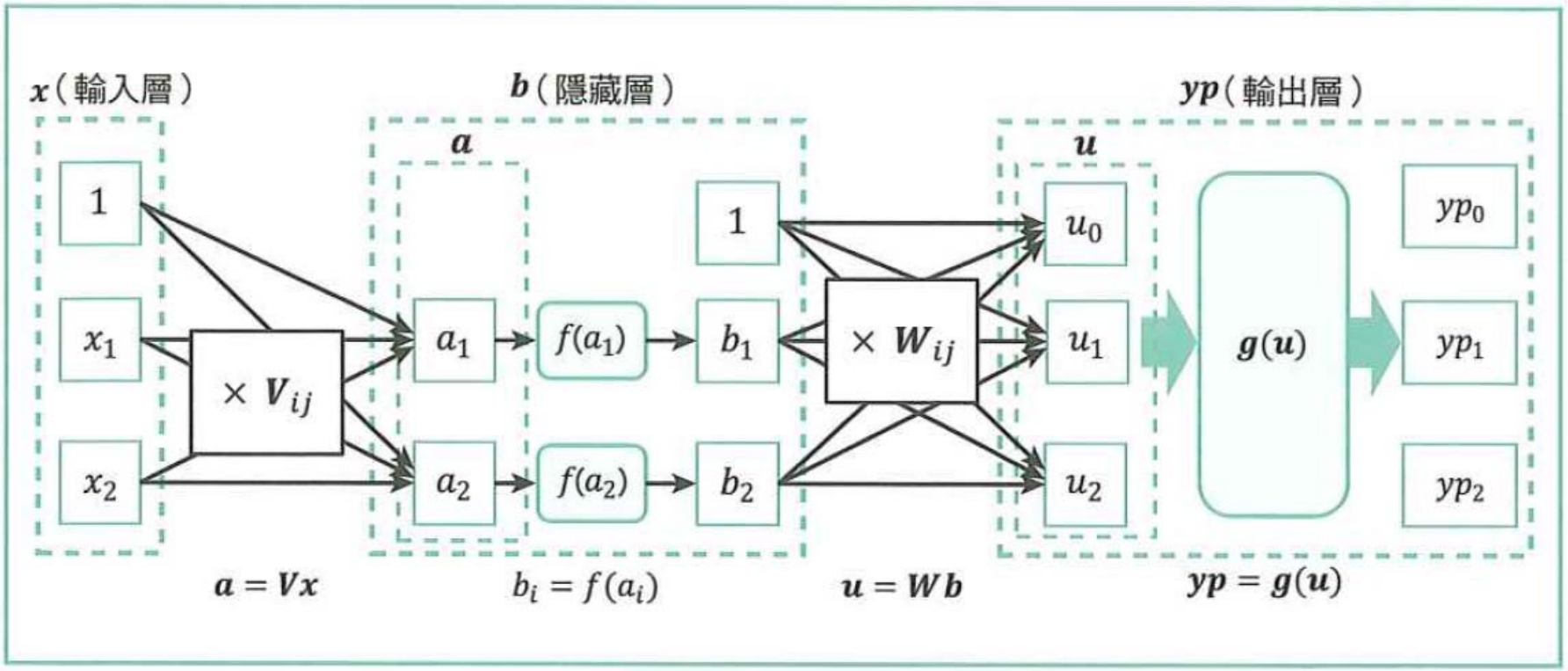
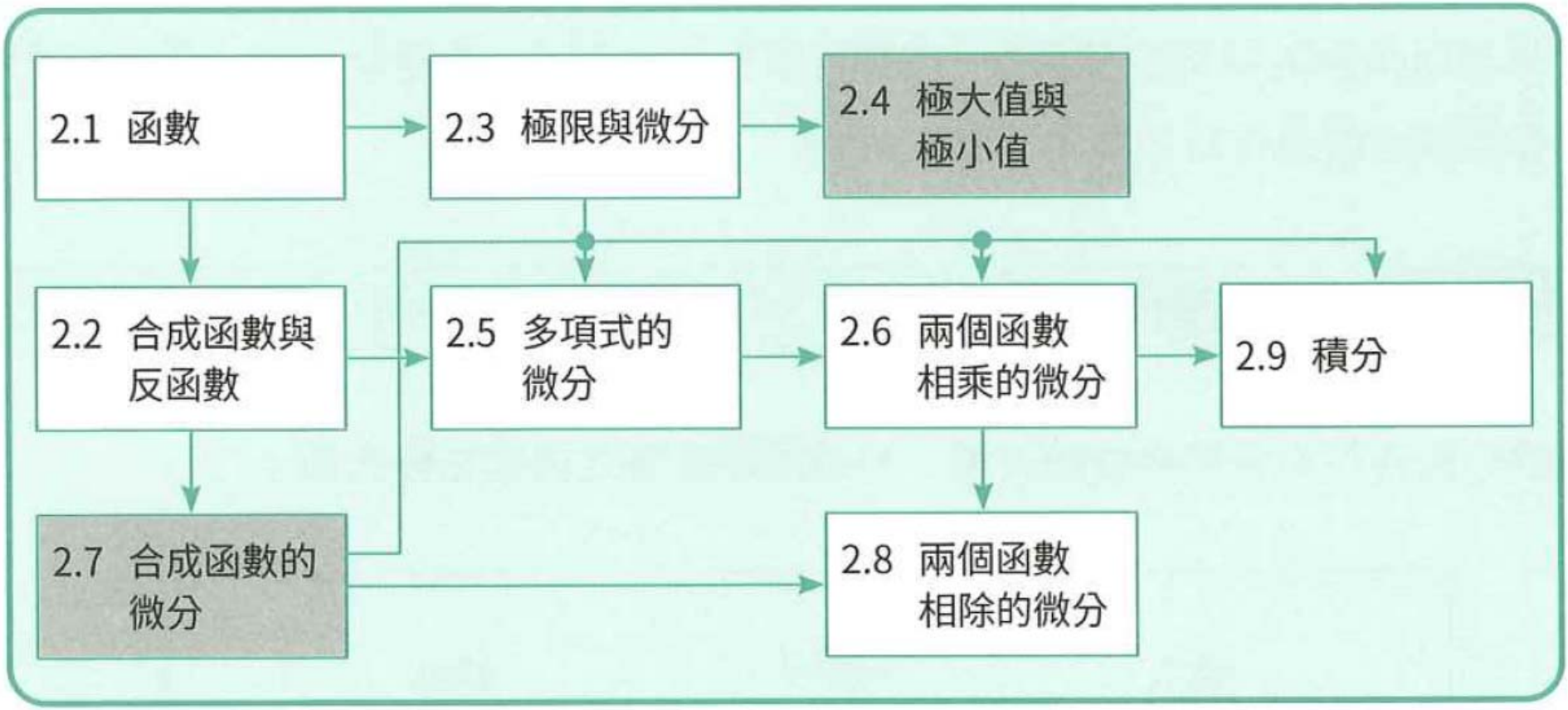


圖 1-13　有隱藏層的分類模型結構

Chapter 2: 積分、微分



瞭解函數(function)運作行為

- 深度學習的基本原理，就是找出讓**損失函數**最小化的**參數值**。
- 要找出這些參數值，必須使用到梯度下降演算法，而這個演算法就是從**函數微分**推導出來的。
- 要深入了解機器學習與深度學習，就必須了解微分。
- 機器學習會利用機率與統計的方法，也需要用到微分和積分。

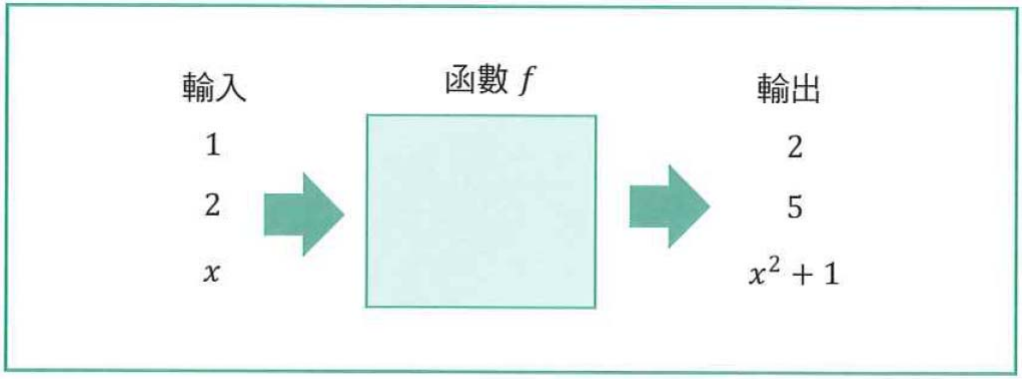


圖 2-1　函數的概念

函數的圖形

函數 $y = f(x) = x^2 + 1$ 的圖形

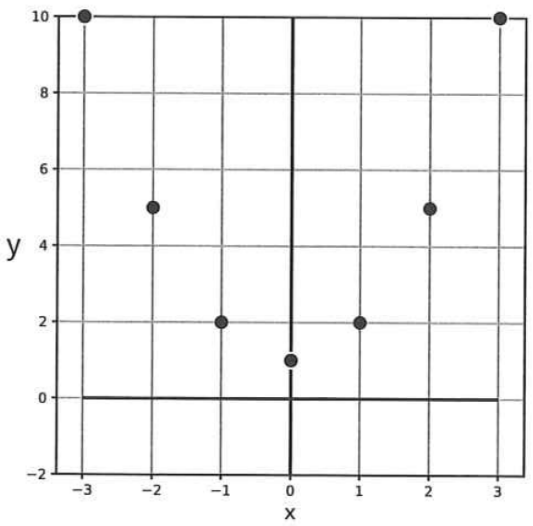


圖 2.2a 在平面座標上畫出 7 個點

(3， 10)、(2， 5)、(1， 2)、(0， 1)、(-1， 2)、(-2， 5)、(-3， 10)

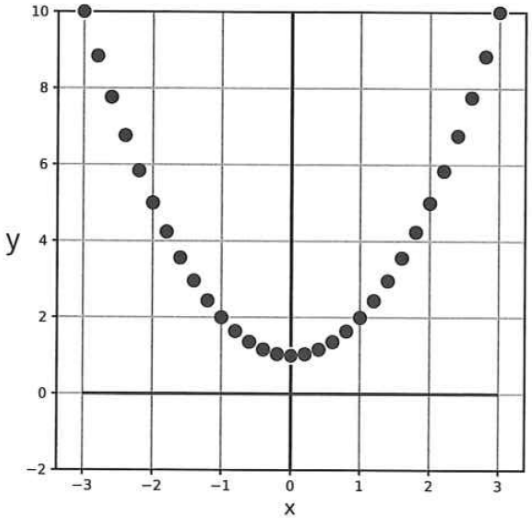


圖 2.2b 畫出更多的點

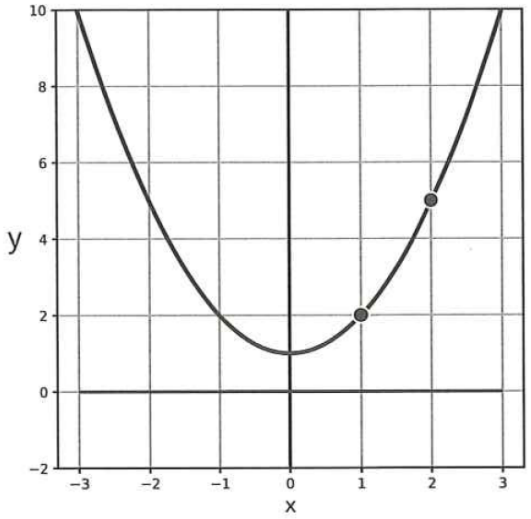


圖 2.2c $f(x)$ 在平面座標上的圖形

連續函數

合成函數 (Composite function)

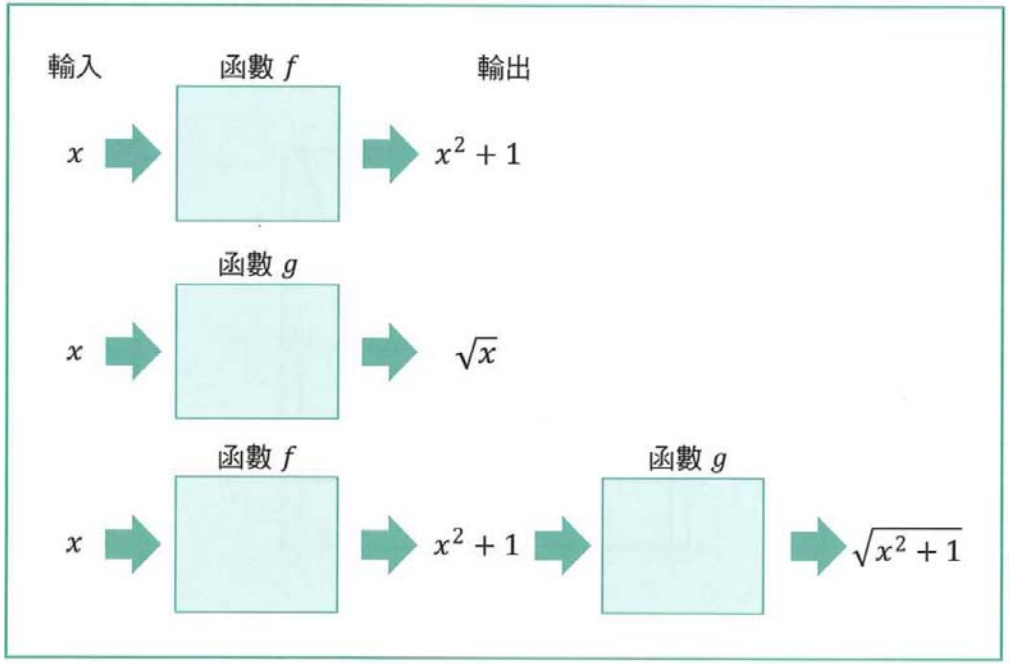


圖 2-3　合成函數的概念

$$f(x) = x^2 + 1$$
$$g(x) = \sqrt{x}$$

$f(x) = \underbrace{x^2+1}$ → 把 $f(x)$ 的輸出值，輸入到 $g(x)$ 中

$$g(x) = \sqrt{x^2+1}$$

$$\Downarrow$$

$$x \to f \to g \to \sqrt{x^2+1}$$

$$h(x) = \sqrt{x^2+1}$$

$$h(x) = \underbrace{g \circ f(x)} = \underbrace{g\,(f(x))}$$

這兩個都是合成函數的符號，
要用哪一個都可以

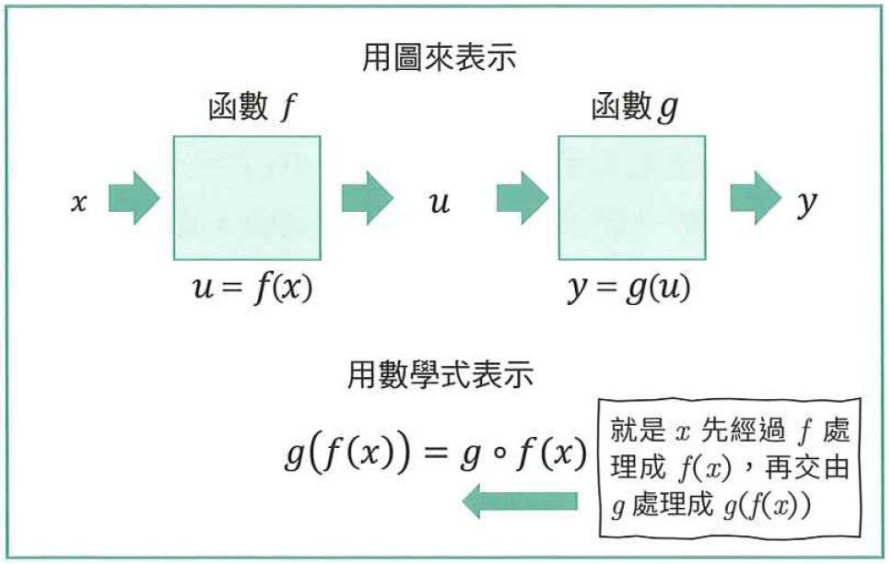


圖 2-4　合成函數的表示法

反函數 (Inverse function)

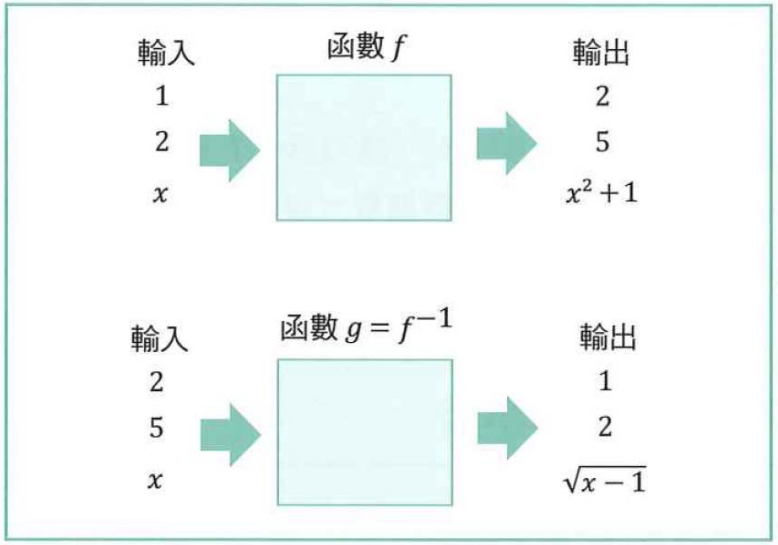


圖 2-5　函數與反函數

- 並非所有的函數都存在反函數，只有一對一的函數(每個x都必須對應到不同的y值)才會存在反函數。

$$g = f^{-1}(x)$$

反函數的具體求法如下：

(1) 將原本函數中的 x 都換成 y，且 y 都換成 x ⟵ 其實只有這一步就夠了

(2) 再將對調後的式子，整理成 $y = \cdots$ 的形式 ⟵ 這一步只是整理而已

例如原函數為 $y = x^2 + 1$，其反函數為：

x、y 對調

第一步　$x = y^2 + 1$

第二步　$y^2 = x - 1 \Rightarrow \quad y = \sqrt{x-1}$，並限定 $x \geq 1$

或 $= -\sqrt{x-1}$，並限定 $x \geq 1$

最後求得反函數為 $y = \sqrt{x-1}$ 或 $-\sqrt{x-1}$ 。

反函數的圖形

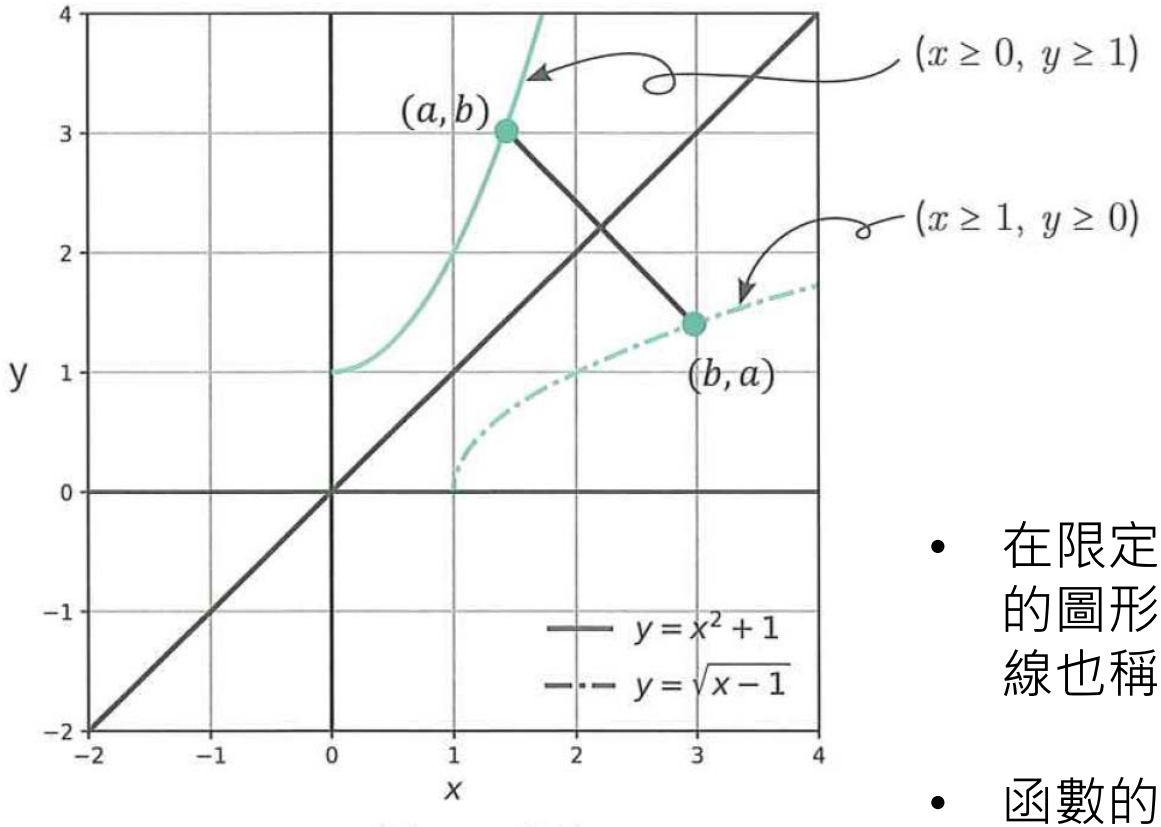


圖 2-6　函數與反函數的圖形

- 在限定的範圍內，函數與反函數的圖形會對稱於直線 $y = x$，此條線也稱為 $y = x$對稱軸。
- 函數的**定義域**就是反函數的**值域**。
- 函數的**值域**就是反函數的**定義域**。

函數在該點上的微分(derivative)

函數 $y = x^3 - x$ 的圖形　以點 $(\frac{1}{2}, -\frac{3}{8})$ 為中心

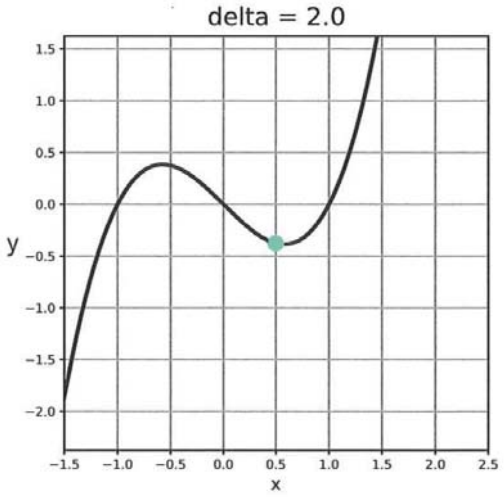


圖 2-7a　點$(\frac{1}{2}, -\frac{3}{8})$在函數的位置

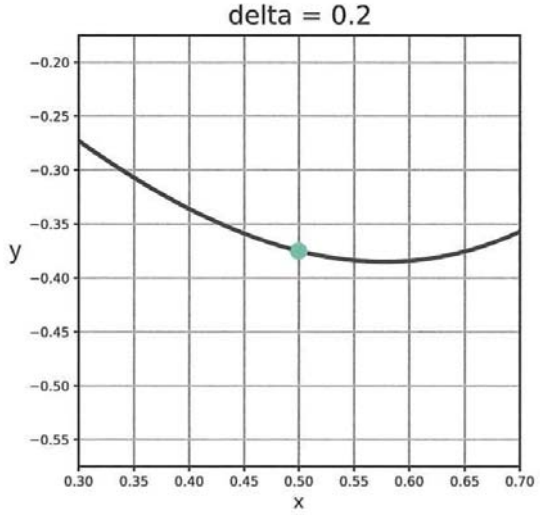


圖 2-7b　放大 10 倍，局部曲線弧度變得較平緩

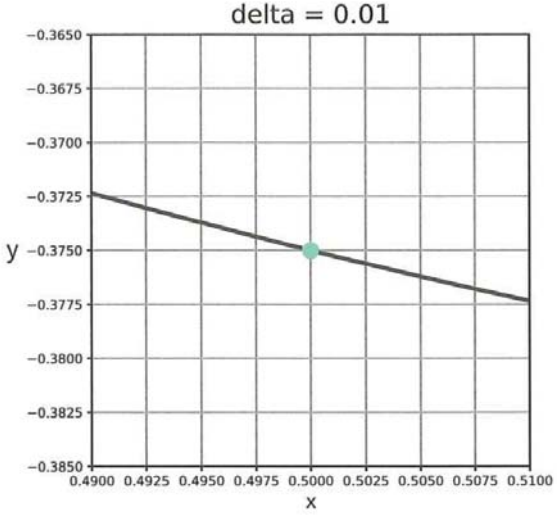
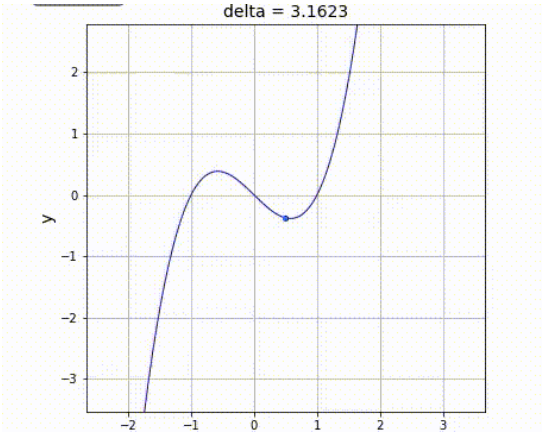


圖 2-7c　放大 200 倍，局部曲線接近直線

以直觀的角度定義微分(Differentiation, Derivative):
以函數圖形上的某點為中心，將函數圖形無限放大，當放到無限大時，圖形會趨近於一直線。此時，這條**直線的斜率**就稱為**函數在該點的微分**。而且這條直線會等於該點的切線。



https://raw.githubusercontent.com/makaishi2/math-sample/master/movie/diff.gif

過曲線上一點的切線斜率就是此函數$f(x)$在這一點的微分

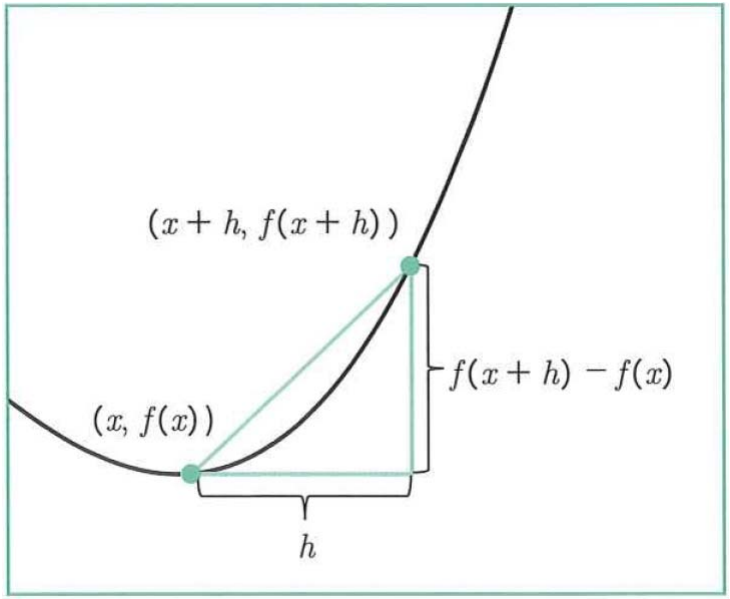


圖 2-8　函數圖形上，兩點相連的直線斜率

$$\frac{f(x+h)-f(x)}{h} \qquad f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

$f(x)=x^2+1$ 為例來進行微分

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}=\lim_{h\to 0}\frac{((x+h)^2+1)-(x^2+1)}{h}$$

$$=\lim_{h\to 0}\frac{2xh+h^2}{h}=\lim_{h\to 0}(2x+h)=2x$$

y'

$\frac{dy}{dx}$

$\frac{d}{dx}f(x)$

} 這些都是 $f'(x)$ 的另一種寫法，意思是一樣的

$y'=2x$

$\frac{dy}{dx}=2x$

$\frac{d}{dx}f(x)=2x$

函數值增量與微分的關係

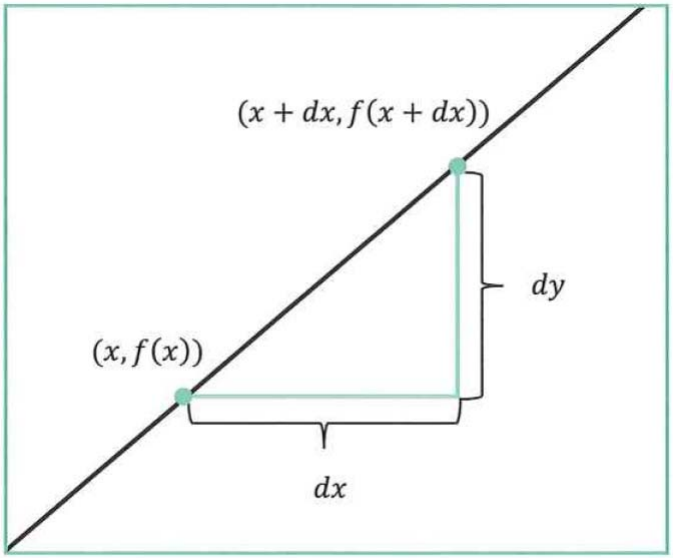


圖 2-9　微分與函數值的近似關係

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x+h) - f(x) \fallingdotseq hf'(x)$$

$$dy = f(x+dx) - f(x) \fallingdotseq f'(x)dx$$

$f(x + dx) - f(x)$ 表示的變化量，

會等於f在x的微分乘上x的變化量 dx。

切線(Tangent line)方程式

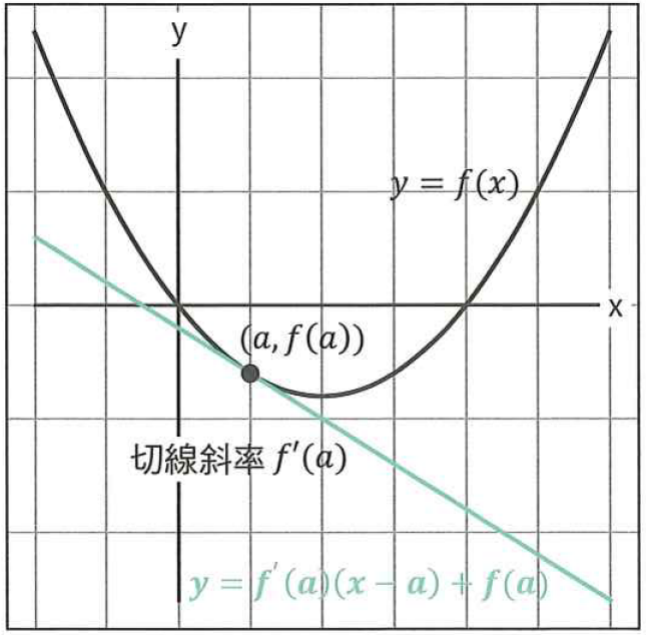


圖 2-10　切線方程式

假設一條直線，其斜率為 m，
且通過 (p, g) 點，則其直線方程式為

$$y = m(x - p) + q$$

從微分定義可知，
函數 $y = f(x)$ 圖形上的 $(a, f(a))$ 這一點的
切線斜率為 $f'(a)$

$$y = f'(a)(x - a) + f(a)$$

函數的極大值與極小值

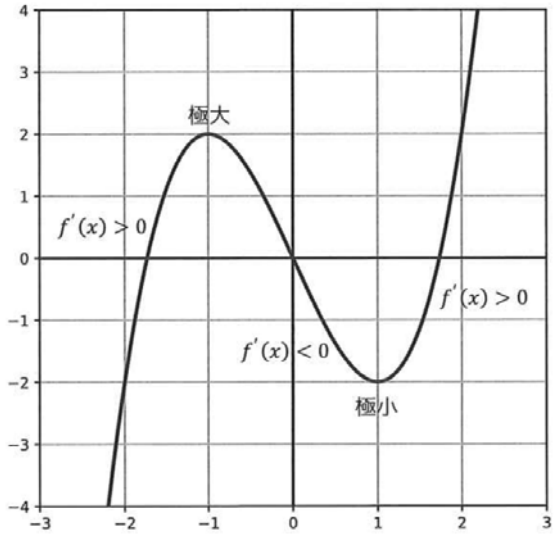


圖 2-11　函數 $y=x^3-3x$ 的極大、極小值

- 極大值與極小值所在的點，其切線斜率會等於0。
- 可透過函數的一次微分$f'(x)=0$來找到該函數可能有極大值或極小值的點。
- (註:在微分等於0的點，有可能是相對極值或絕對極值，也可能根本就不是極值)。

設函數 $f(x)$ 在 $x=a$ 的點，且 $f'(a)=0$，則：

(1) 若在 a 點附近，當 $x<a$ 時，$f'(x)>0$；當 $x>a$ 時，$f'(x)<0$，則 $f(x)$ 在 $x=a$ 處有「相對」極大值。因為當 $x<a$ 時，$f'(x)>0$，表示函數在 a 的左邊遞增。而當 $x>a$ 時，$f'(x)<0$，表示函數在 a 的右邊遞減，所以 $f(a)$ 是相對極大值。

(2) 若在 a 點附近，當 $x<a$ 時，$f'(x)<0$；當 $x>a$ 時，$f'(x)>0$，則 $f(x)$ 在 $x=a$ 處有「相對」極小值。理由同 (1) 的說明。

舉例: 函數微分等於0的點(0, 0)，沒有極大、極小值

有時候即使 $f'(x) = 0$，也不一定會有極值，像下圖的函數就是極值不存在的例子：

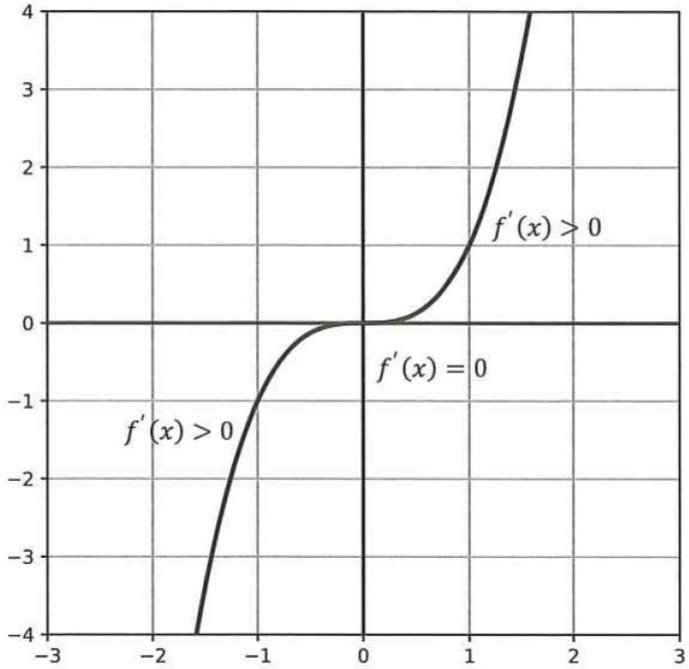


圖 2-12　函數微分等於 0 的點 (0, 0) 沒有極大、極小值 ($y = x^3$ 的圖形)

梯度下降法的原理就是來自這個極值原理：「**藉由計算損失函數的斜率，去找出能讓損失函數產生極小值的參數**」。

一個函數的二次微分叫做 *Hessian*，在機器學習中扮演了二次最佳化的角色

多項式函數的微分

$f(x) = x^n$ 的微分

$$\frac{d}{dx}(x^n) = nx^{n-1}$$

x^r 的微分公式為：

$$(x^r)' = rx^{r-1}$$

二項式定理：

$$(x+h)^n = x^n + {}_nC_1 x^{n-1}h + {}_nC_2 x^{n-2}h^2 + \cdots$$

$$(x+h)^n - x^n = (x^n + {}_nC_1 x^{n-1}h + {}_nC_2 x^{n-2}h^2 + \cdots) - x^n$$

$$= nhx^{n-1} + \frac{n(n-1)}{2}h^2x^{n-2} + \cdots$$

$f(x+h)$ $f(x)$

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h} = \lim_{h\to 0}\frac{nhx^{n-1} + \frac{n(n-1)}{2}h^2x^{n-2} + \cdots}{h}$$

$$= \lim_{h\to 0}(nx^{n-1} + \frac{n(n-1)}{2}hx^{n-2} + \cdots) = nx^{n-1}$$

合成函數的微分

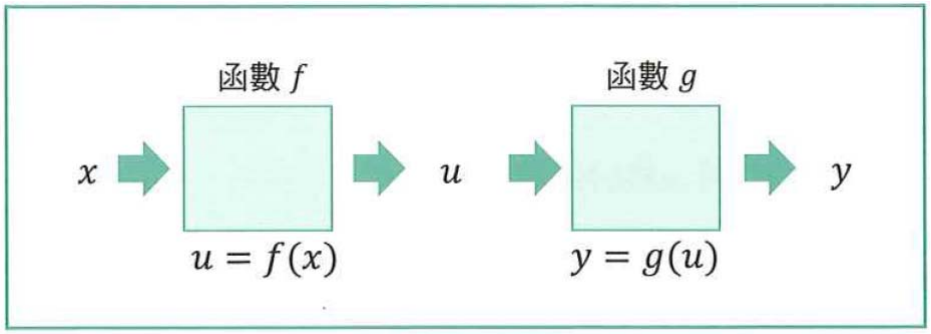


圖 2-13　合成函數

有一個合成函數，其輸入為 x，輸出為 y，且：

$$u = f(x)$$
$$y = g(u)$$

鏈鎖法則 (*chain rule*)

此合成函數的微分公式如下：

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

假設有兩個函數 $f(x)$、$g(x)$，如果將函數 $f(x)$ 的輸出值，當做函數 $g(x)$ 的輸入值，則這兩個函數組合成一個新函數，就稱為合成函數。

$$u = f(x) = x^2 + 1$$
$$y = g(u) = \sqrt{u}$$

$$y = \sqrt{x^2 + 1}$$

$$\frac{dy}{du} = g'(u) = \left(u^{\frac{1}{2}}\right)' = \frac{1}{2}u^{-\frac{1}{2}} = \frac{1}{2\sqrt{u}} = \frac{1}{2\sqrt{x^2+1}}$$

$$\frac{du}{dx} = f'(x) = 2x$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} = \frac{1}{2\sqrt{x^2+1}} \cdot 2x = \frac{x}{\sqrt{x^2+1}}$$

反函數的微分

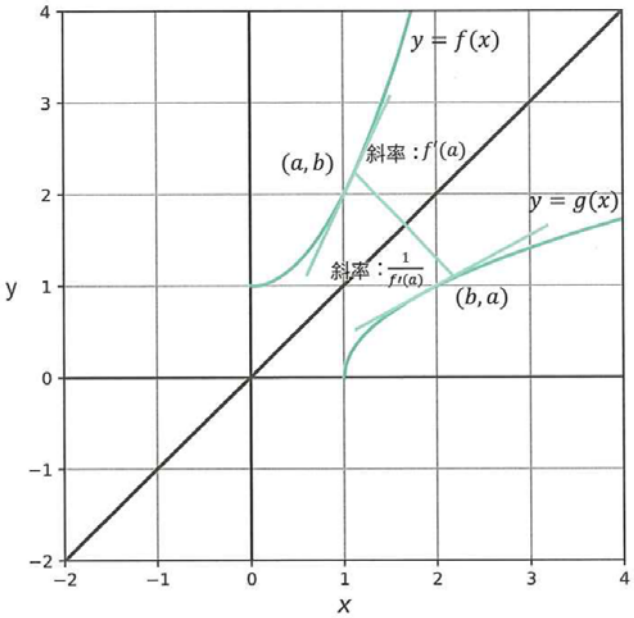


圖 2-14　反函數的微分

$$\frac{dx}{dy} = \frac{1}{\frac{dy}{dx}}$$

若 $y = f(x)$，則 $f'(x) = \frac{dy}{dx}$。

若 $x = g(y)$，則 $g'(y) = \frac{dx}{dy}$。

因此若 $b = f(a)$，則反函數 g 的微分公式：

$$g'(b) = \frac{1}{f'(a)}$$

假設 $y = f(x)$ 的反函數為 $g(x)$，則根據反函數的定義，可知 $x = g(y)$。

(a, b) 在 $y = f(x)$ 上的切線斜率為 $f'(a)$。根據圖形的對稱性，(b, a) 在 $y = g(x)$ 上的切線斜率為 $\frac{1}{f'(a)}$。

$g(f(x)) = x$ ⇒ $\frac{dg}{df} \cdot \frac{df}{dx} = 1 \Rightarrow g'(y) = \frac{1}{f'(x)}$ 或 $\frac{dx}{dy} = \frac{1}{\frac{dy}{dx}}$

微分常用公式整理

$$(p \cdot f(x) + q \cdot g(x))' = p \cdot f'(x) + q \cdot g'(x)$$

$$(x^r)' = rx^{r-1}$$

$$\frac{dx}{dy} = \frac{1}{\frac{dy}{dx}}$$

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$$

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

微分是「將函數圖形在某一點無限放大後，找出切於該點的直線斜率」。
積分則是計算「函數圖形與直線 $y = 0$（即 x 軸）圍成的面積」。

積分則計算「函數圖形與直線 y = 0 (即x軸) 」圍成的面積

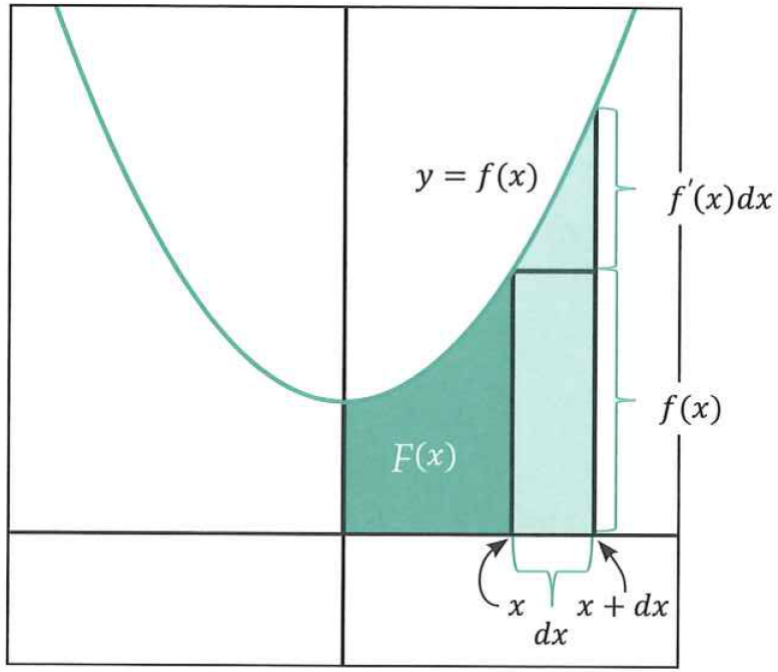


圖 2-15　面積函數 $F(x)$ 與 $f(x)$ 的關係

假設函數 $y = f(x)$ 與 x 值皆為正數。

令下圖中圍出 面積的函數為 $F(x)$ 。

探討面積函數 $F(x)$ 的微分 $F'(x)$ 。

把梯形的面積，看成是一個長方形加上一個三角形

$$F(x+dx) - F(x) \fallingdotseq f(x)dx + \frac{1}{2}f'(x)(dx)^2$$

$$f(x)dx + \frac{1}{2}dx \cdot f'(x)dx$$

$$F'(x) = \lim_{dx \to 0} \frac{1}{dx}(F(x+dx) - F(x)) = \lim_{dx \to 0}\left(f(x) + \frac{1}{2}f'(x) \cdot dx\right) = f(x)$$

面積函數 $F(x)$ 微分就是函數 $f(x)$ ，此即為**微積分的基本定理**。對函數 $f(x)$ 而言， $F(x)$ 稱為 $f(x)$ 的**原始函數**，通常用大寫的 F 表示。

不定積分(Indefinite integral)

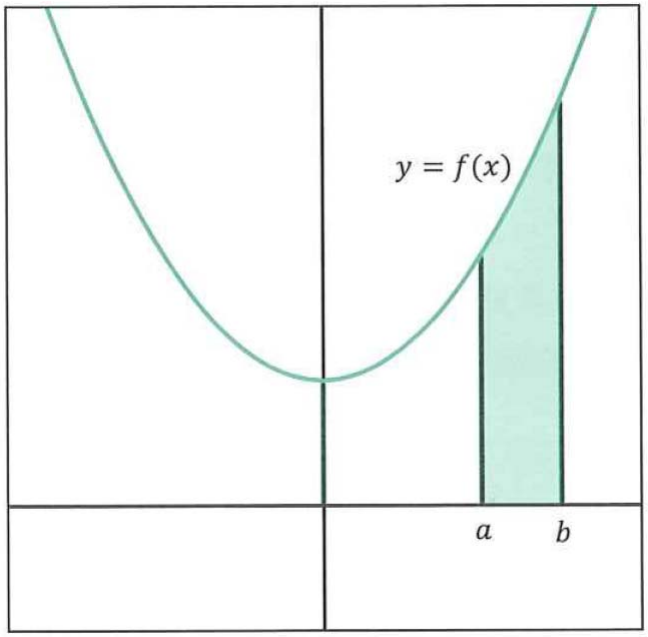


圖 2-16　圖形的面積與定積分

給定一個函數 $f(x)$，求滿足 $F'(x) = f(x)$ 的函數 $F(x)$，而且不限制 x 的範圍，其計算過程稱為求「**不定積分**」。

$$\int f(x)dx = F(x) + C$$

例如求 $f(x) = x^2$ 不定積分的式子就會寫成下面這樣：

$$\int x^2 dx = \frac{x^3}{3} + C$$

如果積分時有限制要積分的範圍，例如下圖是要計算 x 從 a 到 b 圍出的面積，則稱為「**定積分**」，用 $F(x)$ 來表示可寫成：

$$\int_a^b f(x)dx = F(b) - F(a)$$

圖形的面積與定積分

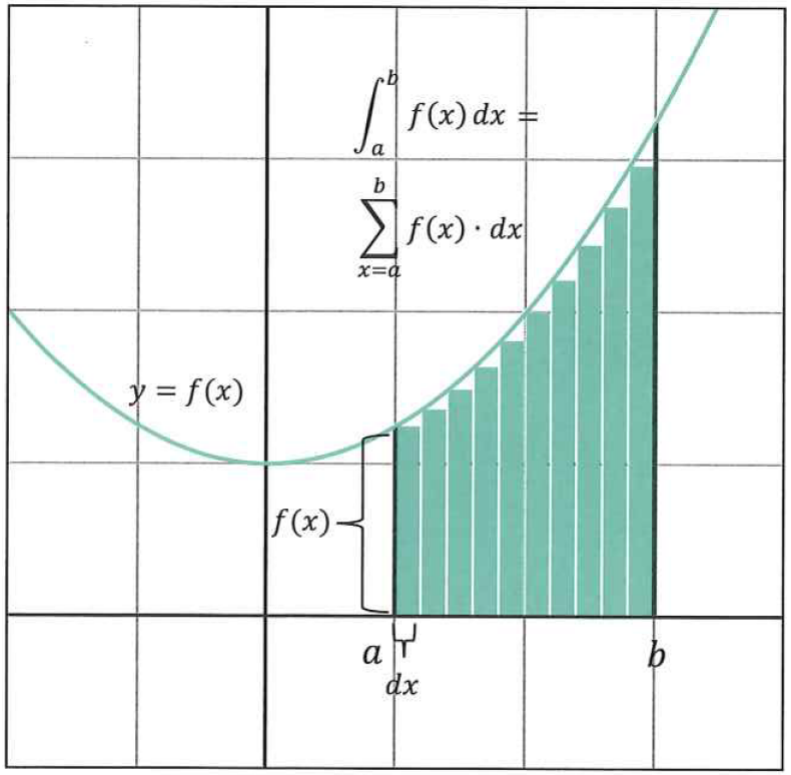


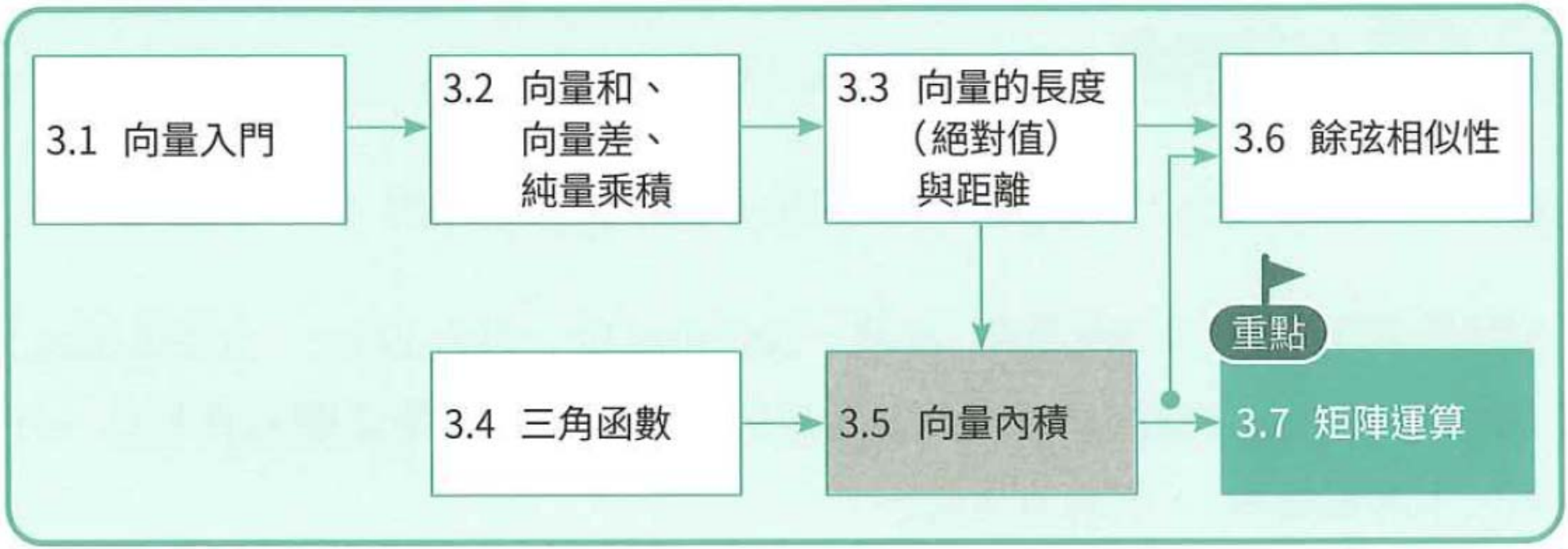
圖 2-17　積分與面積的關係

定積分在圖形上可看成是由 a 到 b 之間用高度 $f(x)$、寬度 dx 組成的許多細長方形填滿的面積。

$$\lim_{\|P\|\to 0}\sum_{k=1}^{n} f(c_k)\Delta x_k = I = \int_a^b f(x)\ dx$$

積分符號 $\int$，原本是英文字母 $S(Sum)$ 的意思，表示將上面定積分式子中「從 $x = a$ 到 $x = b$ 之間所有細長方形 $f(x)\,dx$ 都加起來」的意思。

Chapter 3: 向量、矩陣



向量入門

- 機器學習與深度學習中用到的向量(vector)運算，以「**內積**」最為重要。
- 瞭解並熟悉向量、矩陣(matrix)的符號及運算規則，才能看懂深度學習用到的算式。

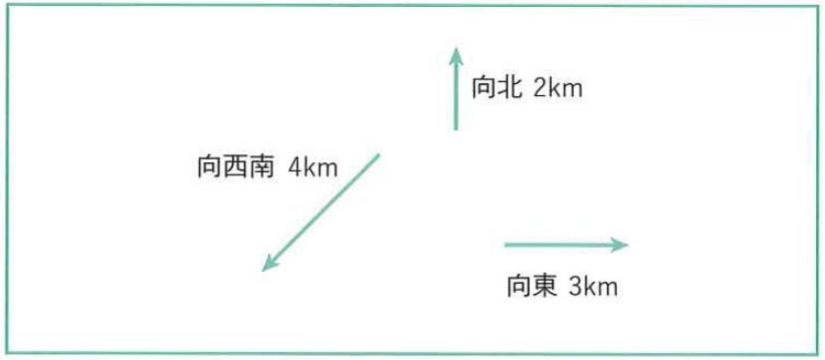


圖 3-1　向量用方向及大小來表現

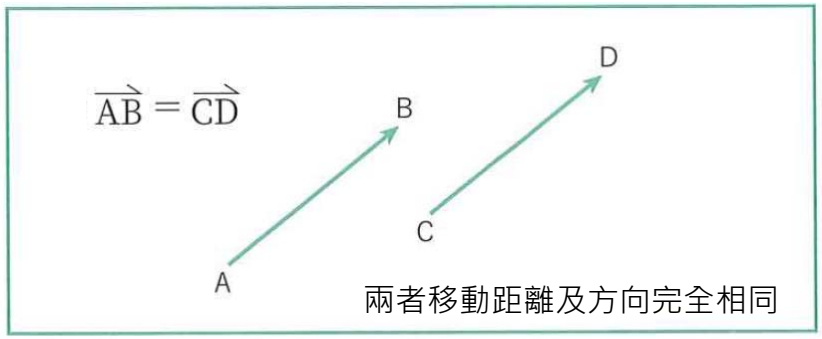


圖 3-2　表示起點和終點的向量

- 常見的向量表示法: (1) 用粗體小寫字母。(2) 小寫字母上方加上箭頭。
- (而一般小寫字母的則用來表示一般的純量。)

使用分量前，必須定出x軸、y軸的方向，以及單位向量的長度。

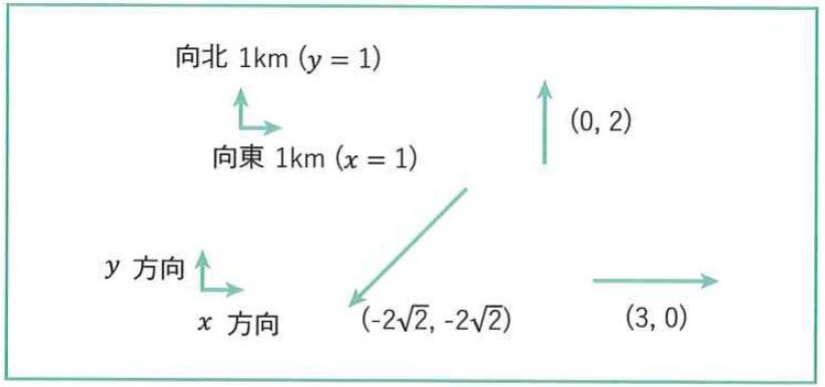


圖 3-3　各向量以分量表示

編註： 我們也可以將一個向量用「單位向量（*unit vector*）」的形式來表示，例如：以 i、j 表示 x、y 軸上的單位向量，然後 $(-2\sqrt{2}, -2\sqrt{2})$ 這個向量就可以寫成 $-2\sqrt{2}\,i - 2\sqrt{2}\,j$。這在數學、物理或工程上較常使用。

三維空間的向量

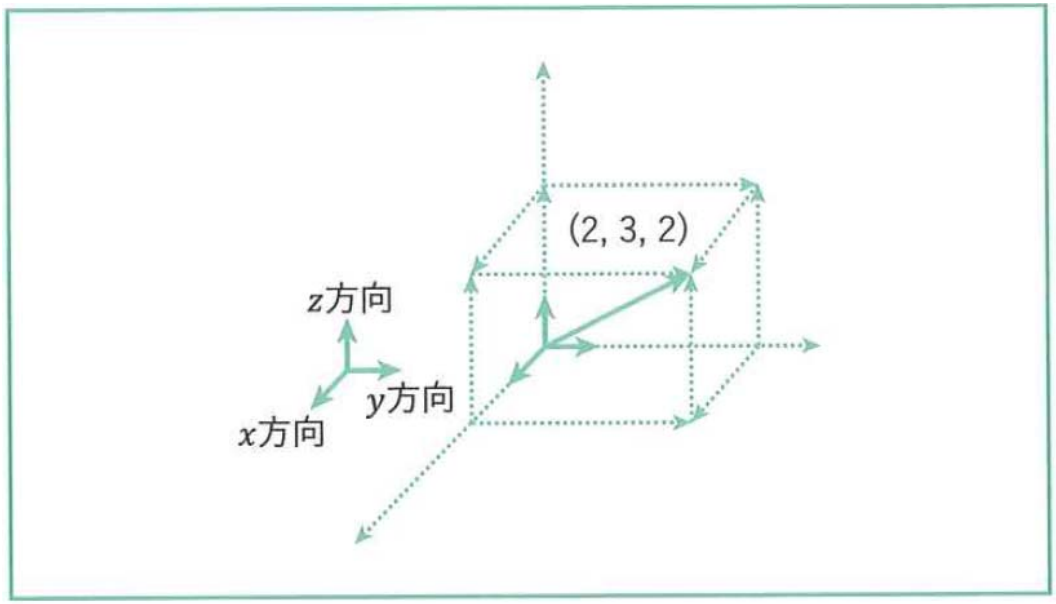


圖 3-4　三維向量的分量表示法

多維向量

列向量 (*row vector*)

$$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$$

行向量 (*column vector*)

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

向量的運算:和、差

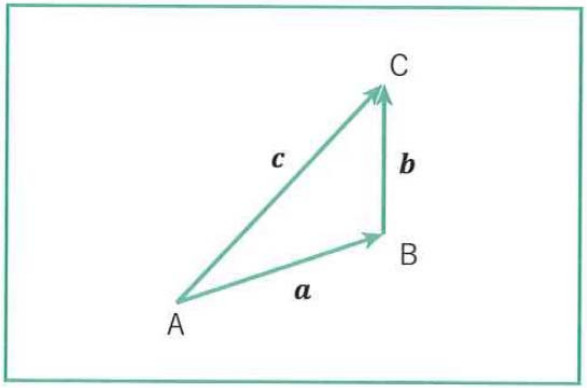


圖 3-5　向量和

$\boldsymbol{a} = \overrightarrow{AB}$

$\boldsymbol{b} = \overrightarrow{BC}$

$\boldsymbol{c} = \overrightarrow{AC}$

$\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b} = (a_1 + b_1, a_2 + b_2)$

$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{c}$　或　$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$

若有兩個 n 維向量 $\boldsymbol{a}$、$\boldsymbol{b}$：

$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$

$\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$

$\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b} = (a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n)$

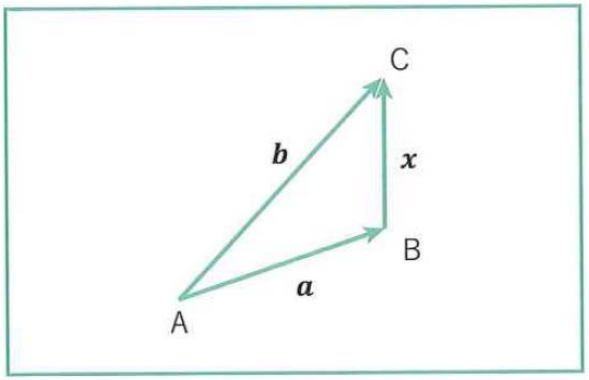


圖 3-6　向量差

$\boldsymbol{a} = (a_1, a_2)$

$\boldsymbol{b} = (b_1, b_2)$

$\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{a} = (b_1 - a_1, b_2 - a_2)$

若有兩個 n 維向量 $\boldsymbol{a}$、$\boldsymbol{b}$：

$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$

$\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$

$\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{a} = (b_1 - a_1,\ b_2 - a_2\ \cdots,\ b_n - a_n)$

編註： 兩個向量差也可以反過來相減，會差一個負號，表示向量大小相同，但方向相反。

向量與純量的乘積，向量的長度

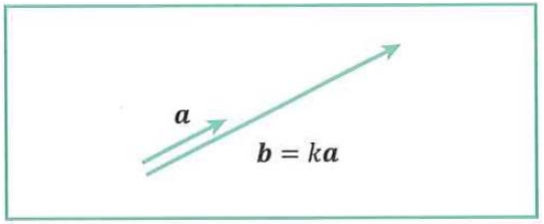


圖 3-7　向量的純量乘積

$\boldsymbol{b} = k\boldsymbol{a}$

$\boldsymbol{a} = (a_1,\ a_2)$

$\boldsymbol{b} = (ka_1,\ ka_2)$ ← $\boldsymbol{b}$ 的分量皆為 $\boldsymbol{a}$ 的 k 倍

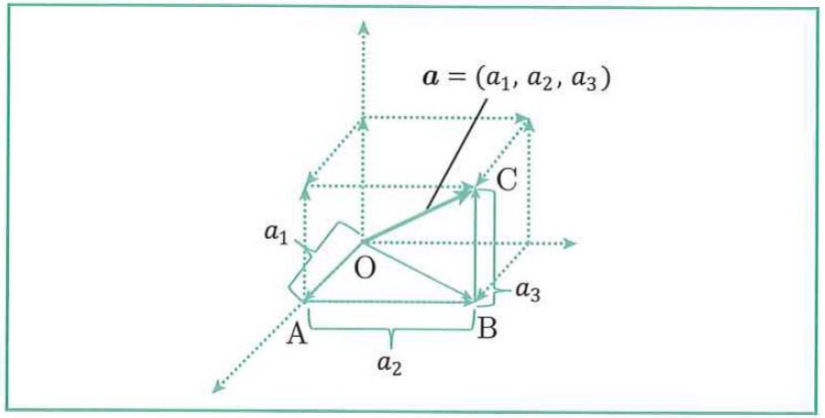


圖 3-9　三維向量的長度

$\boldsymbol{a} = \overrightarrow{OC} = (a_1, a_2, a_3)$

$OA^2 + AB^2 = OB^2$

$OB^2 + BC^2 = OC^2$

$OA^2 + AB^2 + BC^2 = OC^2$

$OC^2 = {a_1}^2 + {a_2}^2 + {a_3}^2$

$OC = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2} = |\overrightarrow{OC}| = |\boldsymbol{a}|$

$|\boldsymbol{a}| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2}$

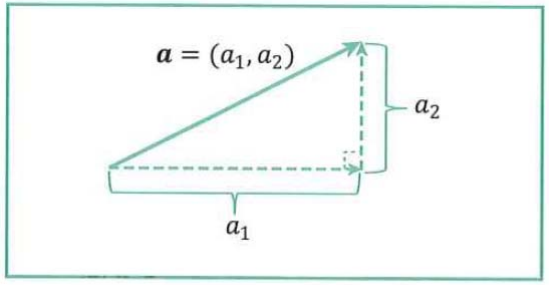


圖 3-8　用分量來表示向量長度

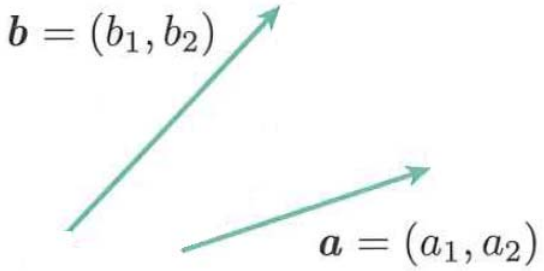
假設向量 $\boldsymbol{a}$ 的長度為 $|\boldsymbol{a}|$(也可以寫為 $\|\boldsymbol{a}\|$)，根據畢氏定理：

$|\boldsymbol{a}|^2 = {a_1}^2 + {a_2}^2$　　$|\boldsymbol{a}| = \sqrt{{a_1}^2 + {a_2}^2}$

$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$　　$|\boldsymbol{a}| = \sqrt{\sum_{k=1}^{n} {a_k}^2}$

$|\boldsymbol{a}| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2 + \cdots + {a_n}^2}$

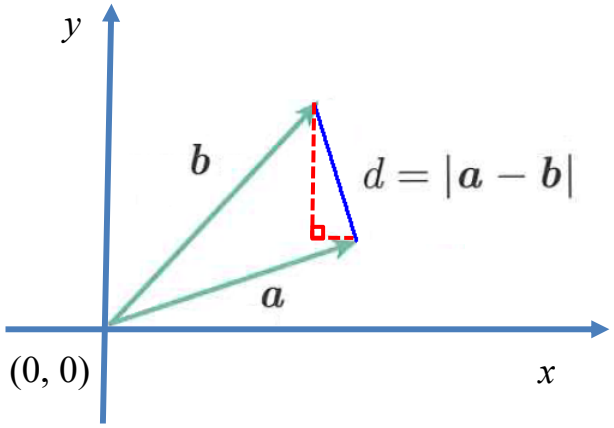
向量間的距離
是兩個向量相減的絕對值



向量 $\boldsymbol{a}$、$\boldsymbol{b}$ 的距離 d

$$\boldsymbol{a} = (a_1, a_2)$$
$$\boldsymbol{b} = (b_1, b_2)$$
$$d = |\boldsymbol{a} - \boldsymbol{b}| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

計算向量間的距離前，必須將兩向量的起點平移到原點(0, 0)，重新算出向量終點的位置後，再計算兩向量終點的距離。



$\boldsymbol{a}$、$\boldsymbol{b}$ 是 n 維向量

$$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$$
$$\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$$
$$d = |\boldsymbol{a} - \boldsymbol{b}| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} = \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2}$$

三角函數

角度範圍: 0 ~ 90度

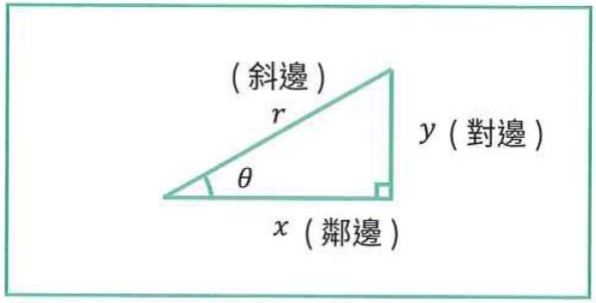


圖 3-10　三角比的定義

$$\sin\theta = \frac{y}{r} \longleftarrow \theta \text{ 的對邊除以斜邊}$$

$$\cos\theta = \frac{x}{r} \longleftarrow \theta \text{ 的鄰邊除以斜邊}$$

$$\tan\theta = \frac{y}{x} \longleftarrow \theta \text{ 的對邊除以鄰邊}$$

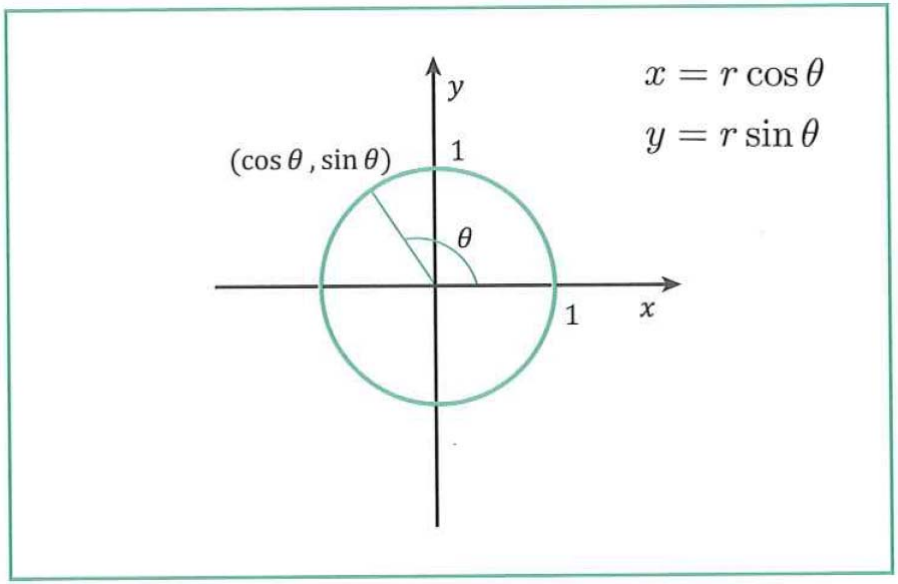


圖 3-11　單位圓上的三角函數座標定義

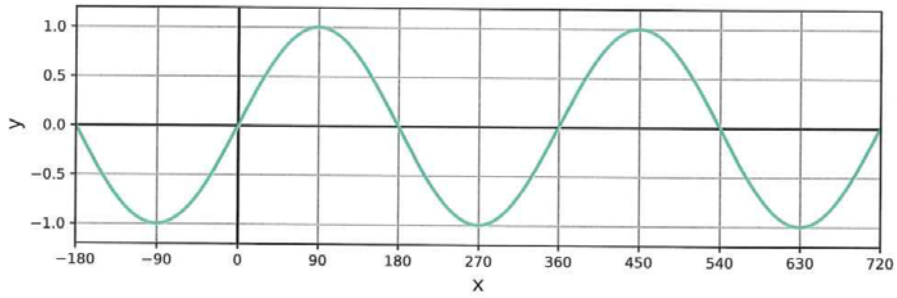


圖 3-12　$y = \sin\theta$ 的圖形　正弦曲線

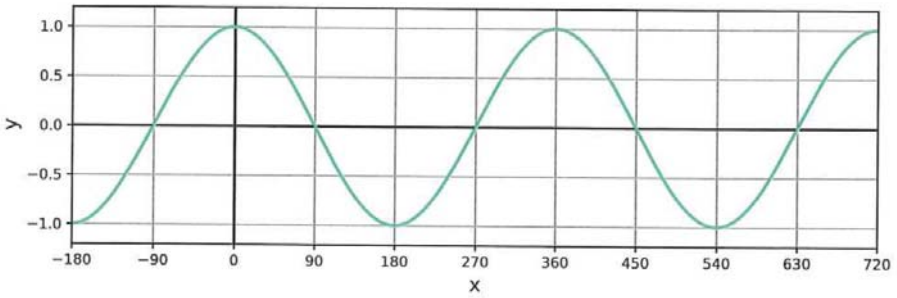


圖 3-13　$y = \cos\theta$ 的圖形　餘弦曲線

向量內積

兩個向量做內積運算有兩種解釋方法:

- 兩個相同維度的向量，將對應的分量兩兩相乘後再相加，亦稱為**點積(dot product)**。

向量 $\boldsymbol{a}$、$\boldsymbol{b}$ 的內積

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2$$

- 指幾何座標上的兩個向量，其一的長度乘上夾角的cos值(即其中一個向量在另一向量上的投影長度，再乘上被投影向量的長度)，稱為**內積(inner product)**。

當二維向量 $\boldsymbol{a}$、$\boldsymbol{b}$ 的夾角為 θ 時

$$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \cos\theta$$

- 在機器學習中，內積與點積的意思相同。

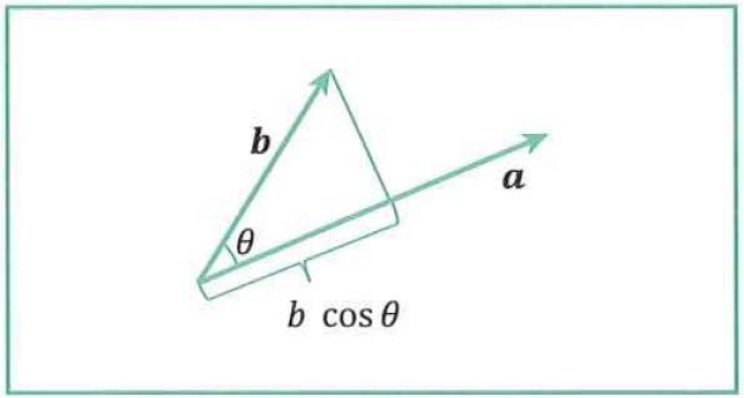


圖 3-14　內積的圖形意義

θ 值	向量 a、b 的關係	內積的數值
0°	方向完全相同	最大值
90°	方向垂直	0
180°	方向完全相反	最小值

表 3-1　角度 θ 與內積的關係

用兩向量的分量兩兩相乘來表示內積

$$\boldsymbol{a} = (a_1, a_2) = a_1\,\boldsymbol{i} + a_2\,\boldsymbol{j}$$
$$\boldsymbol{b} = (b_1, b_2) = b_1\,\boldsymbol{i} + b_2\,\boldsymbol{j}$$

→ 用單位向量表示

$$\boldsymbol{i}\cdot\boldsymbol{i} = \boldsymbol{j}\cdot\boldsymbol{j} = 1\,,\ \boldsymbol{i}\cdot\boldsymbol{j} = \boldsymbol{j}\cdot\boldsymbol{i} = 0$$

$\boldsymbol{a}$ 和 $\boldsymbol{b}$ 的內積

$$\boldsymbol{a}\cdot\boldsymbol{b} = (a_1\,\boldsymbol{i} + a_2\,\boldsymbol{j})\cdot(b_1\,\boldsymbol{i} + b_2\,\boldsymbol{j})$$
$$= a_1b_1\,\boldsymbol{i}\cdot\boldsymbol{i} + a_1b_2\,\boldsymbol{i}\cdot\boldsymbol{j} + a_2b_1\,\boldsymbol{j}\cdot\boldsymbol{i} + a_2b_2\,\boldsymbol{j}\cdot\boldsymbol{j}$$

$$\boldsymbol{a}\cdot\boldsymbol{b} = a_1b_1 + a_2b_2$$

向量內積符合分配律

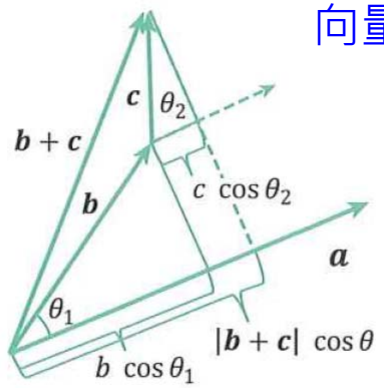


圖 3-15　內積的線性關係

$$\boldsymbol{a}\cdot(\boldsymbol{b}+\boldsymbol{c}) = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\cdot\boldsymbol{c}$$

($\boldsymbol{b}+\boldsymbol{c}$ 與 $\boldsymbol{a}$ 同方向的分量)
= ($\boldsymbol{b}$ 與 $\boldsymbol{a}$ 同方向的分量) +
($\boldsymbol{c}$ 與 $\boldsymbol{a}$ 同方向的分量)

n 維向量內積

$$\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$$
$$\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$$

$$\boldsymbol{a}\cdot\boldsymbol{b} = a_1b_1 + a_2b_2 + \cdots + a_nb_n = \sum_{k=1}^{n} a_kb_k$$

餘弦相似性

$$\boldsymbol{a} = (a_1, a_2)$$

$$\boldsymbol{b} = (b_1, b_2)$$

$$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \cos\theta = a_1 b_1 + a_2 b_2 \quad \Rightarrow \quad \cos\theta = \frac{a_1 b_1 + a_2 b_2}{|\boldsymbol{a}||\boldsymbol{b}|} = \frac{a_1 b_1 + a_2 b_2}{\sqrt{{a_1}^2 + {a_2}^2}\sqrt{{b_1}^2 + {b_2}^2}}$$

- 當 cosθ接近1時(夾角約0度)，我們可以說這兩個向量的方向幾乎相同。
- 當 cosθ接近-1時(夾角約 180度)，代表這兩個向量的方向幾乎相反。
- 當 cos θ接近0時，則代表這兩個向量的方向接近垂直。
- 以上這個公式的特性稱為**餘弦相似性**，也就是用二維的概念來描述n維的行為，常做為判斷兩向量方向是否接近的指標。

n 維向量

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{\sqrt{{a_1}^2 + {a_2}^2 + \cdots + {a_n}^2}\sqrt{{b_1}^2 + {b_2}^2 + \cdots + {b_n}^2}} = \frac{\sum_{k=1}^{n} a_k b_k}{\sqrt{\sum_{k=1}^{n} {a_k}^2}\sqrt{\sum_{k=1}^{n} {b_k}^2}}$$

編註： 假設上式算出 cosθ 的值是 x，則在 *Google* 搜尋欄內輸入 *arccos*（x）即可得到 θ 的徑度，再除以 π，乘以 180 度，即可得到度數。假設上式算出的 cos$\theta = -0.5$，則輸入 *arccos*（−0.5）會得到 2.0943951 *rad*（徑度），則 *arccos*（−0.5）/ *pi* * 180 可得到 120 度。

矩陣運算: 輸出節點的內積表示法

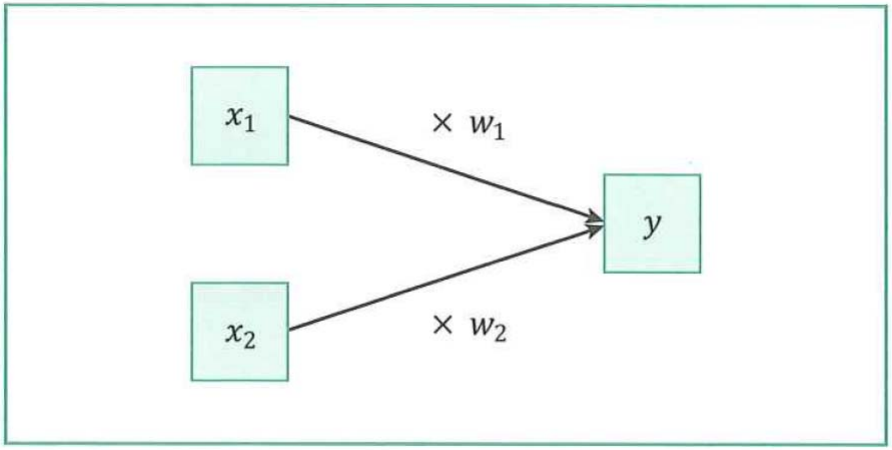


圖 3-16　2 個輸入節點，1 個輸出節點的架構

$$y = w_1 x_1 + w_2 x_2$$

$$\boldsymbol{w} = (w_1, w_2) \text{、} \boldsymbol{x} = (x_1, x_2)$$

$$y = \boldsymbol{w} \cdot \boldsymbol{x}$$

$\boldsymbol{w}$在機器學習中稱為權重參數

3個輸出節點的矩陣相乘

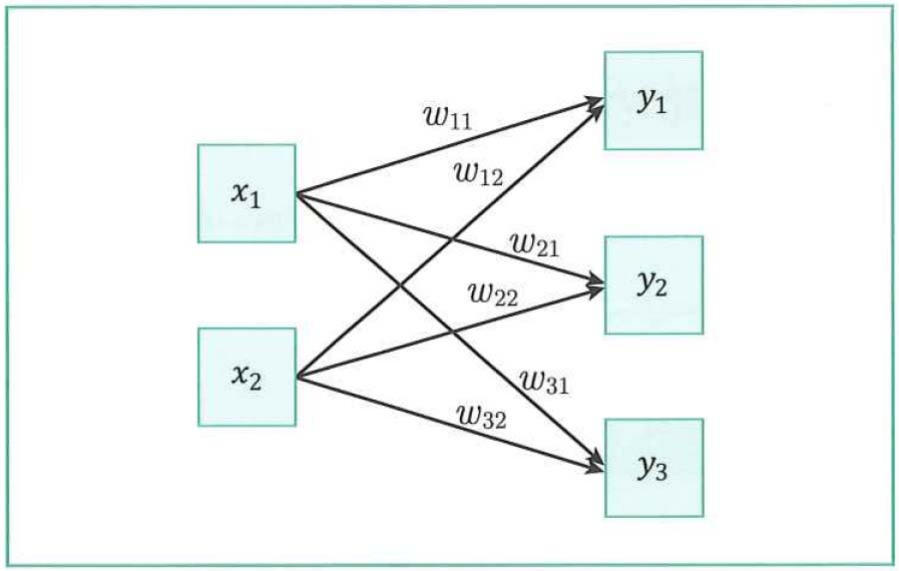


圖 3-17　2 個輸入節點與 3 個輸出節點的架構

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$y_1 = w_{11}x_1 + w_{12}x_2$$
$$y_2 = w_{21}x_1 + w_{22}x_2$$
$$y_3 = w_{31}x_1 + w_{32}x_2$$

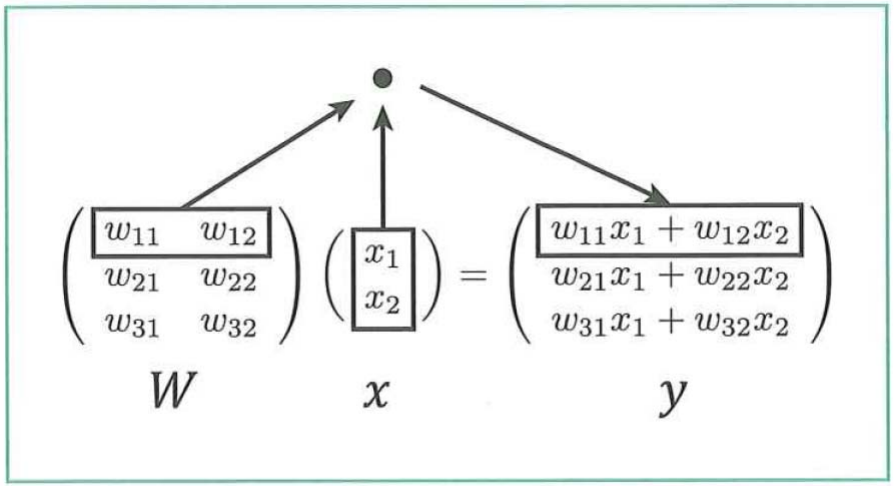


圖 3-18　矩陣與向量的乘積

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}$$

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

純量、向量、張量(Tensor)

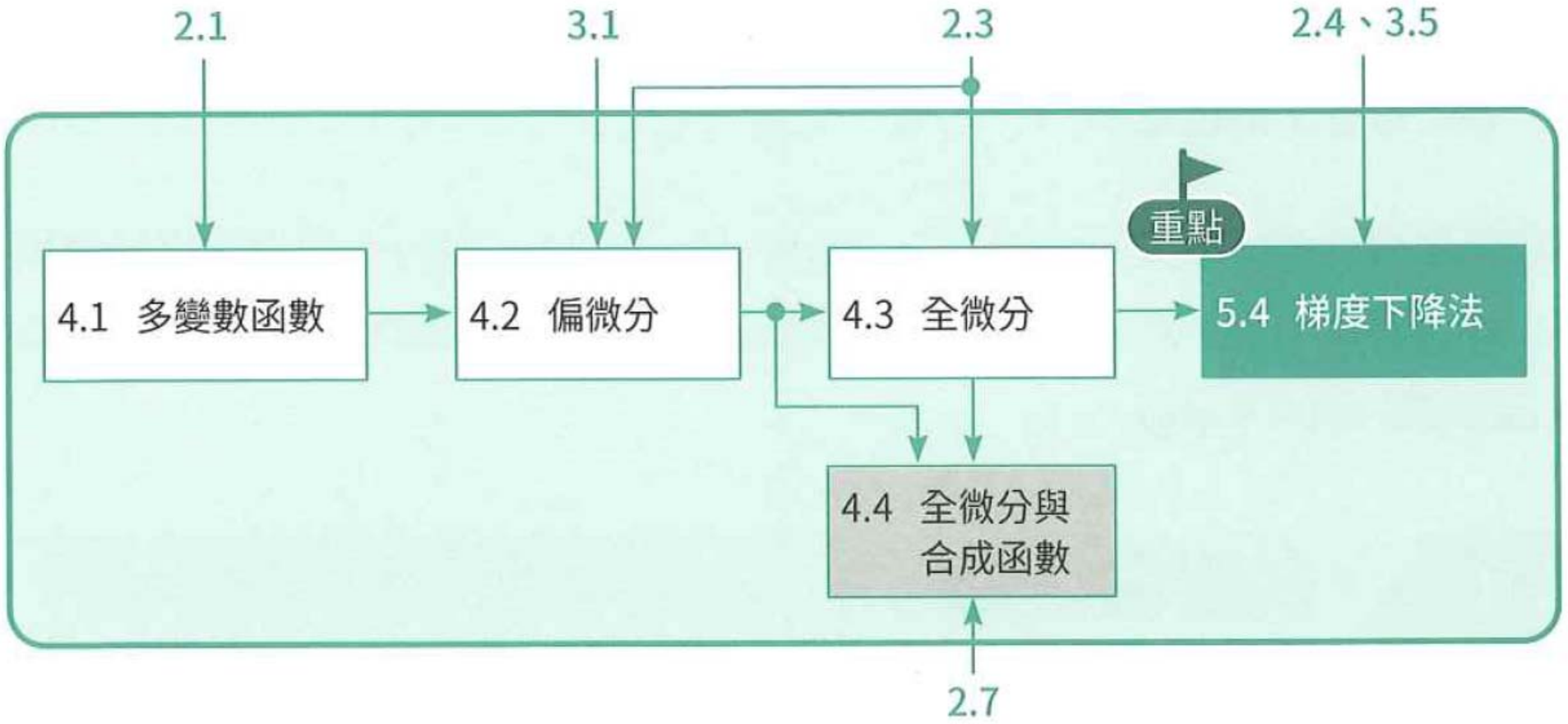
張量也是機器學習的核心，它和向量、短陣的關係如下:

本章介紹的項目	張量	Python 的實作
純量	0 階張量	純數值
向量	一階張量	1D 陣列
矩陣	二階張量	2D 陣列
	三階張量	3D 陣列
	……	……
	n 階張量	nD 陣列

表 3-2　純量、向量、矩陣、張量對照表

- 矩陣與張量都可視為**存放數字的容器**，外觀看起來相同，但意義不同。
- 張量在幾何上代表**座標轉換**的意思，也就是說張量作用在某個向量或矩陣上，就如間對該向量或矩陣做座標轉換運算;若沒有被作用的對象，則張量就形同矩陣。
- 例如 $\boldsymbol{W}\cdot\boldsymbol{x}$就可視為權重$\boldsymbol{W}$「張量」對 向量$\boldsymbol{x}$做轉換運算。
- 為了讓張量與矩陣的維數做區別，我們將前者稱為 **n階 (rank)張量**，後者稱為n維矩障。不過有些文件中仍將張量稱為n維張量，不要弄混就好。

Chapter 4: 多變數函數的微分



多變數函數: $f(x, y)$, $L(u, v)$

輸入 x	單變數函數	輸出 y
1	$(x-2)^2+1$	2
-1		10

圖 4-1　單變數函數

輸入(u, v)	雙變數函數	輸出 $L(u, v)$
$(-1, 1)$	$3u^2+3v^2-uv$	3
$(0, 2)$	$+7u-7v+10$	8

圖 4-2　雙變數的函數

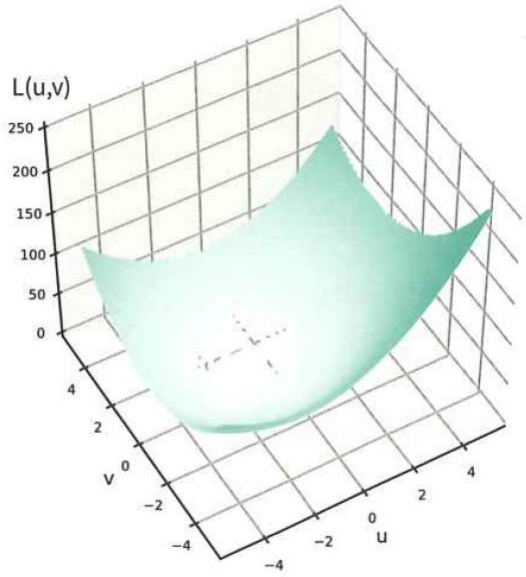


圖 4-3a　雙變數函數在三維座標的曲面

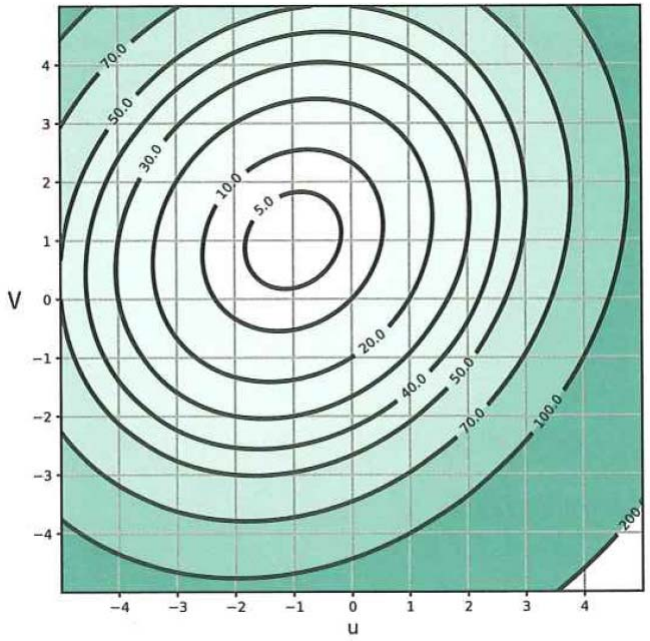


圖 4-3b　用等高線來呈現函數的圖形

雙變數函數的偏微分

- 如果一個函數的變數不只一個時，為了能夠看出每一個變數在變化時的影響，就要對各該變數分別做微分，稱為「**偏微分**」。
- 在對某一個變數做微分時，其餘的變數因與該變數無關，因此會被當成常數看待。

$\frac{\partial L}{\partial u}$ ← 將 L 寫在上面

$\frac{\partial}{\partial u}L(u,v)$ ← 將 L 寫在右邊

多變數函數在偏微分時，
把偏微分目標以外的變數都當成常數就可以了。

$L_u(u,\ v)$ 代表 $\frac{\partial}{\partial u}L(u,v)$

$$L(u,\ v,\ w) = 3u^2 + 3v^2 + 3w^2 - uv + uw + 7u - 7v - 7w + 10$$

$$L(u,\ v) = 3u^2 + 3v^2 - uv + 7u - 7v + 10$$

$L_u(u,\ v) = 6u - v + 7$ ← 對 u 偏微分

$L_v(u,\ v) = 6v - u - 7$ ← 對 v 偏微分

分別對 u、v、w 偏微分後會得到：

$$L_u(u,\ v,\ w) = 6u - v + w + 7$$
$$L_v(u,\ v,\ w) = 6v - u - 7$$
$$L_w(u,\ v,\ w) = 6w + u - 7$$

雙變數函數的偏微分

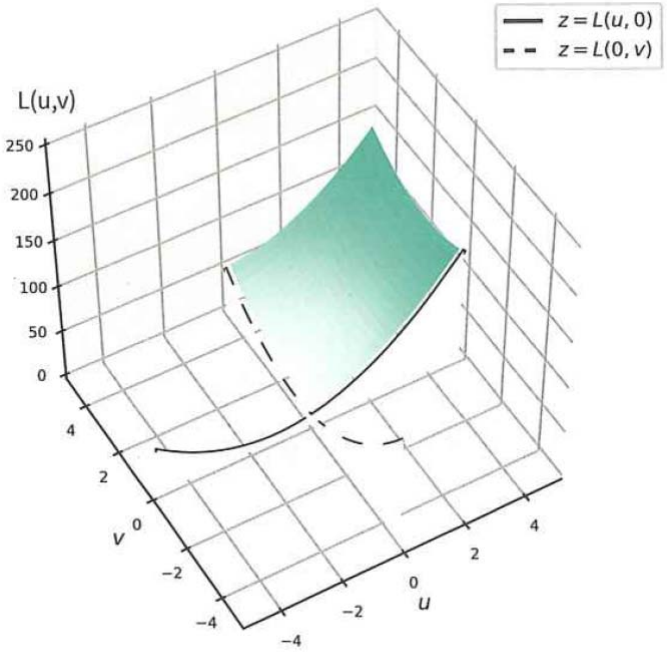


圖 4-4　三維圖形與切割面

$$L(u,\ v) = 3u^2 + 3v^2 - uv + 7u - 7v + 10$$

在點 $(u, v) = (0, 0)$ 對 u 偏微分的 $L_u(0, 0) = 7$ 的意義是什麼？

計算 $L_u(0, 0)$ 時，我們會將 v 值固定為 $v = 0$。
對 u 偏微分的圖形，可以想像成 是在函數曲面上
$v = 0$ 處切一刀所形成的一個切割面
(注意右圖中的實線曲線)

函數 $L(u, v)$ 當 $v = 0$ 時，
$L(u, 0) = 3u^2 + 7u + 10$ 就是此切割面曲線的函數。
偏微分 $L_u(0, 0)$ 代表的是這個**切割面函數**
$\mathbf{3u^2 + 7u + 10}$ **在** $\boldsymbol{u = 0}$ **的斜率**。

偏微分 $L_v(0, 0)$，則是 $\boldsymbol{u = 0}$ **切割面函數在** $\boldsymbol{v = 0}$ **的斜率**
(請看圖中虛線曲線)

雙變數函數的全微分

全微分就是單變數函數時的微分，當推廣到多變數函數的微分峙，就稱為「全微分」。

微分是**函數圖形上一小段區間，經過無限放大後會接近直線的特性**，來得到函數變化的狀態。

當 u、v 值小幅變動，亦即 $(u,\ v) \to (u+du,\ v+dv)$ 時，
函數 $L(u,\ v)$ 的值會如何變動？

三維空間的圖形中，
將函數**曲面上的某一小塊區間，無限放大後會接近平面**。

如此將曲面局部放大變成接近平面後，
看看 $L(u+\ du,\ v+\ du)$ 與 $L(u,\ v)$ 的差異：

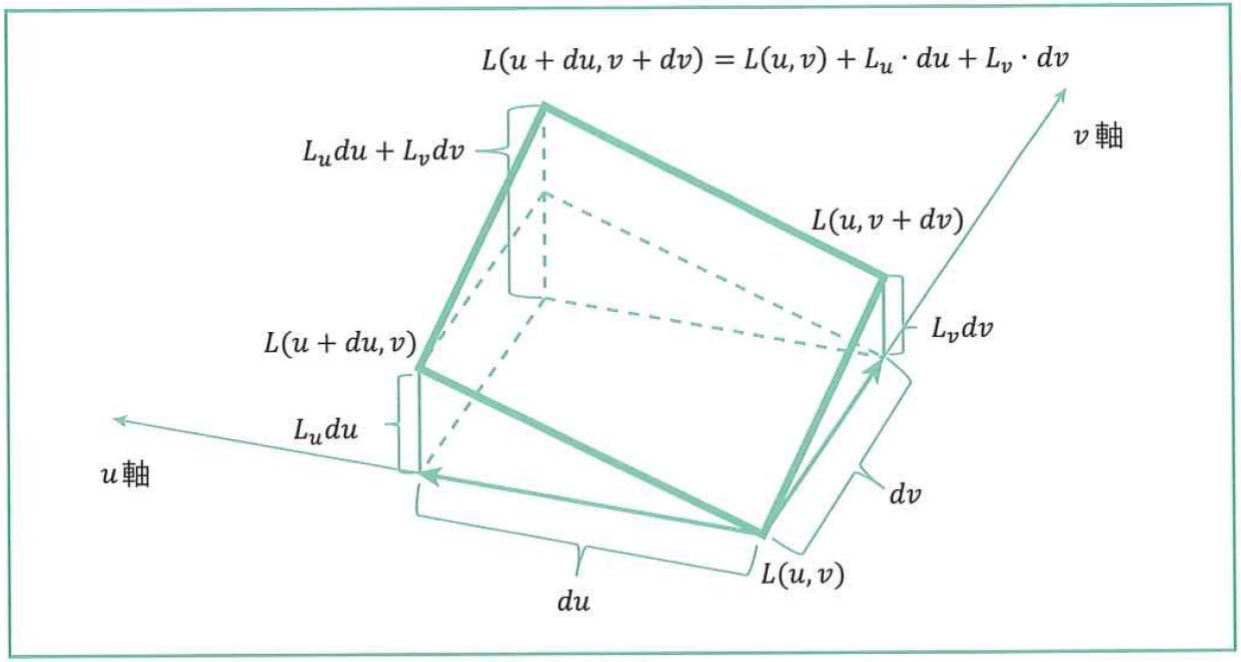


圖 4-5　雙變數函數小幅增量之後的變化

雙變數函數的全微分

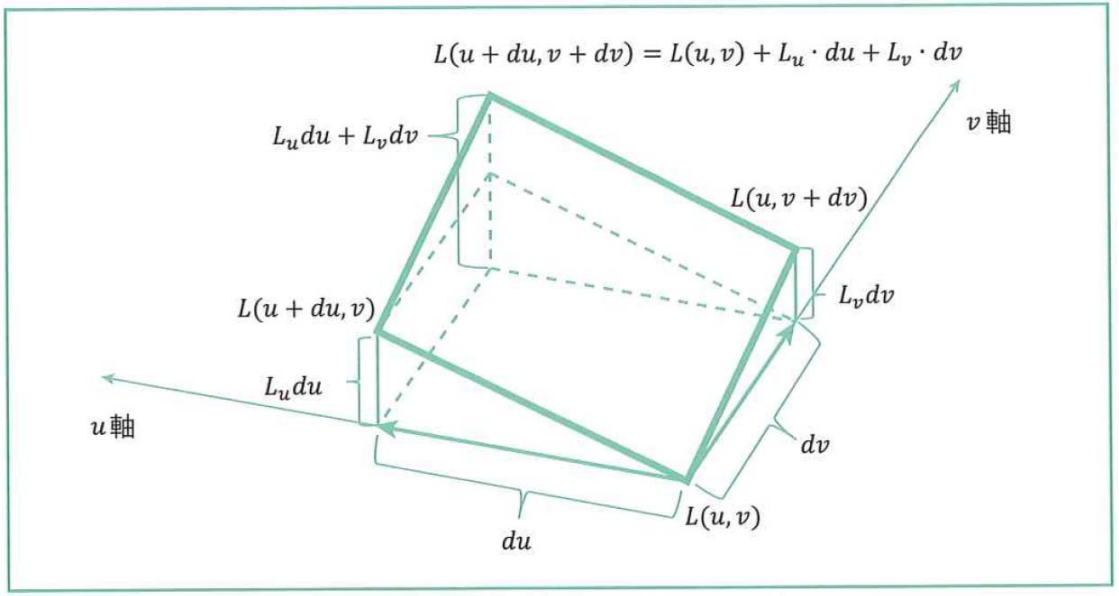


圖 4-5　雙變數函數小幅增量之後的變化

$$f(x+dx) \fallingdotseq f(x) + f'(x)dx$$

$$L(u+du, v) \fallingdotseq L(u,v) + L_u(u,v)du$$

$$L(u, v+dv) \fallingdotseq L(u,v) + L_v(u,v)dv$$

$$L(u+du, v+dv) \fallingdotseq L(u,v) + L_u(u,v)du + L_v(u,v)dv$$

$$L(u+du, v+dv) - L(u,v) \fallingdotseq L_u(u,v)du + L_v(u,v)dv$$

L 函數值在 $(u,\ v)$ 微量增加 $(du,\ dv)$ 時的變化

$$dL = L_u du + L_v dv$$

原函數：$L(w_1, w_2, \cdots, w_N)$

全微分公式：

$$dL = \frac{\partial L}{\partial w_1}dw_1 + \frac{\partial L}{\partial w_2}dw_2 + \cdots + \frac{\partial L}{\partial w_N}dw_N = \sum_{i=1}^{N}\frac{\partial L}{\partial w_i}dw_i$$

全微分的公式

$$dL = \frac{\partial L}{\partial u}du + \frac{\partial L}{\partial v}dv$$

全微分與合成函數

合成函數的微分公式 (鏈鎖法則)

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

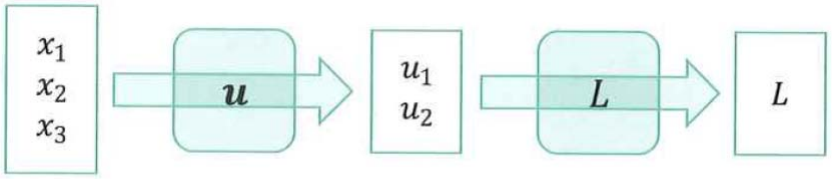


圖 4-6　例題的設定

$$dL = \frac{\partial L}{\partial u_1} du_1 + \frac{\partial L}{\partial u_2} du_2$$

$$u_1 = u_1(x_1,\ x_2,\ x_3)$$
$$u_2 = u_2(x_1,\ x_2,\ x_3)$$
$$L = L(u_1,\ u_2)$$

換成偏微分符號

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial x_1} + \frac{\partial L}{\partial u_2}\frac{\partial u_2}{\partial x_1}$$

將變數 x_1、x_2、x_3 組合成向量 $\boldsymbol{x} = (x_1,\ x_2,\ x_3)$

將 u_1、u_2 組合成 $\boldsymbol{u} = (u_1, u_2)$

$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$ ⟵ $\boldsymbol{u}$ 是 $\boldsymbol{x}$ 的函數，即向量函數

$L = L(\boldsymbol{u})$ ⟵ L 是 $\boldsymbol{u}$ 的函數，即合成函數

全微分與合成函數

$$dL = \frac{\partial L}{\partial u_1}du_1 + \frac{\partial L}{\partial u_2}du_2$$

換成偏微分符號

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial x_1} + \frac{\partial L}{\partial u_2}\frac{\partial u_2}{\partial x_1}$$

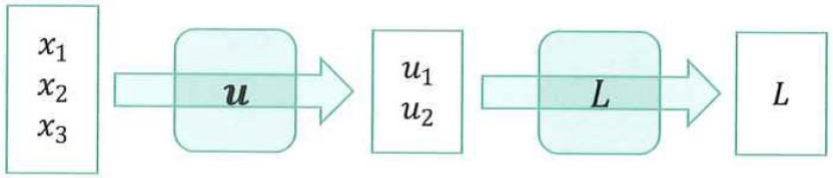


圖 4-6　例題的設定

$$u_1(x_1,\ x_2,\ x_3) = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

$$u_2(x_1,\ x_2,\ x_3) = w_{21}x_1 + w_{22}x_2 + w_{23}x_3$$

$$L(u_1,\ u_2) = u_1{}^2 + u_2{}^2$$

$\frac{\partial L}{\partial u_1} = 2u_1$ ← L 中的 $u_2{}^2$ 與 u_1 無關，因此只留下 $2u_1$

$\frac{\partial L}{\partial u_2} = 2u_2$ ← L 中的 $u_1{}^2$ 與 u_2 無關，因此只留下 $2u_2$

$\frac{\partial u_1}{\partial x_1} = w_{11}$ ← u_1 中的 x_2、x_3 兩項與 x_1 無關，只留下 w_{11}

$\frac{\partial u_2}{\partial x_1} = w_{21}$ ← u_2 中的 x_2、x_3 兩項與 x_1 無關，只留下 w_{21}

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial x_i} + \frac{\partial L}{\partial u_2}\frac{\partial u_2}{\partial x_i}$$

$$i = 1,\ 2,\ 3$$

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial x_1} + \frac{\partial L}{\partial u_2}\frac{\partial u_2}{\partial x_1} = 2u_1 \cdot w_{11} + 2u_2 \cdot w_{21} = 2(u_1 \cdot w_{11} + u_2 \cdot w_{21})$$

全微分與合成函數:中間變數u只有一個

如果中間的 $\boldsymbol{u}$ 包括 N 個函數 u_1、u_2、$\cdots$、u_N，則

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial x_i} + \frac{\partial L}{\partial u_2}\frac{\partial u_2}{\partial x_i} + \cdots + \frac{\partial L}{\partial u_N}\frac{\partial u_N}{\partial x_i}$$

$$= \sum_{j=1}^{N} \frac{\partial L}{\partial u_j}\frac{\partial u_j}{\partial x_i}$$

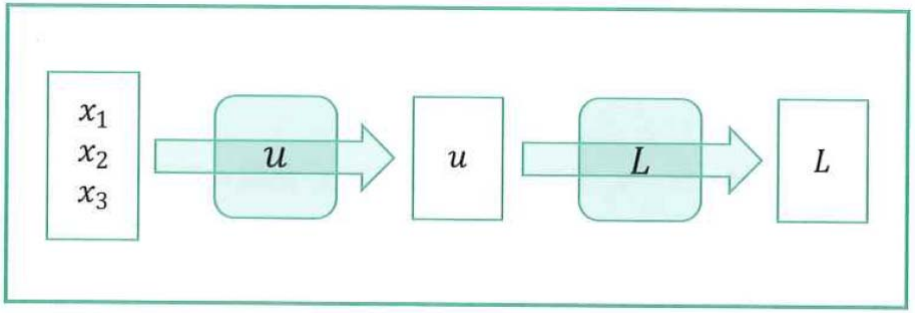


圖 4-7　中間只有一個 u

$$dL = \frac{dL}{du} \cdot du$$

$$\frac{\partial L}{\partial x_1} = \frac{dL}{du} \cdot \frac{\partial u}{\partial x_1}$$

u 中包括 x_1、x_2、x_3，只對 x_1 微分則換成偏微分符號

只有一個 u，仍是常微分

L 對 x_1 偏微分，換成偏微分符號

$$\frac{\partial L}{\partial x_i} = \frac{dL}{du} \cdot \frac{\partial u}{\partial x_i}$$

梯度下降法(Gradient Descent)

給定某雙變數函數 $L(u,\ v)$，求出能讓 $L(u,\ v)$ 函數值最小化的 $(u_{min},\ v_{min})$ 值。

運算步驟：

1) 先設定$(u,\ v)$的初始值為(u_0, v_0),即曲面上的某個點。
2) 從(u_0, v_0)值開始，找出能讓$L(u, v)$函數值減少最多的方向。 可視為向量移動的方向
3) 配合(2)的方向，調整 $(u,\ v)$ 的變化量，然後將新值設為 (u_1, v_1) 。 可視為向量移動的大小
4) 再以新值(u_1, v_1)為基準，重複迭代 (2)和 (3)的步驟。

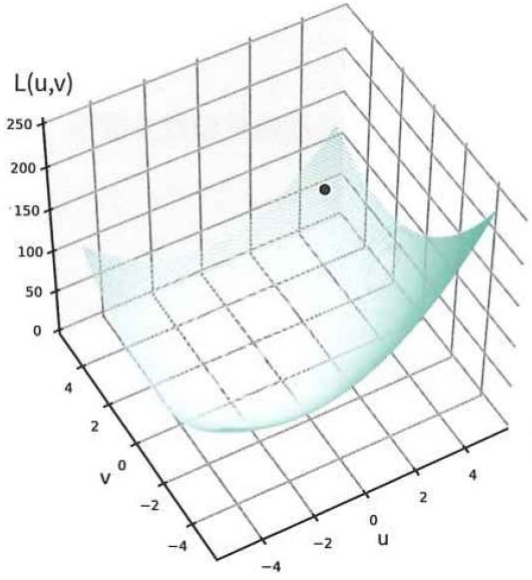
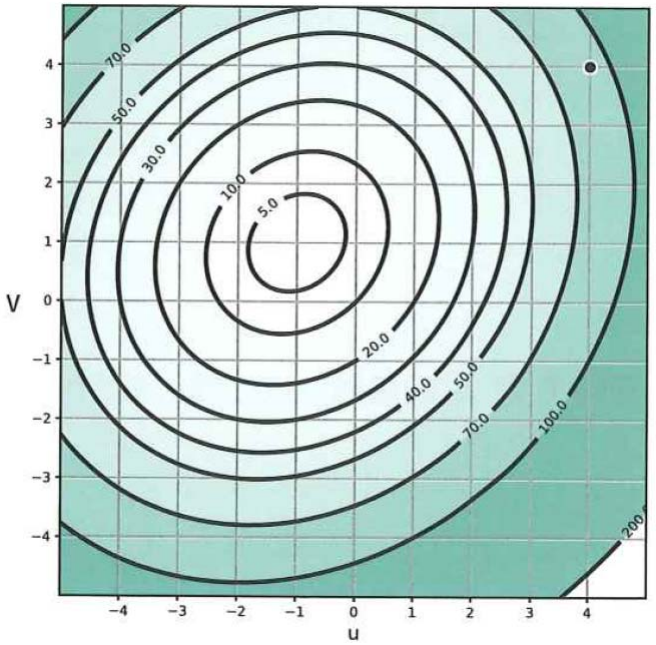


圖 4-8*a*
圖中黑點為曲面上
$(u,\ v)$ 初值的位置

圖 4-8*b*
這是左圖的等高線圖，
$(u,\ v)$ 初值位於右上角高處



梯度下降法的重點

(A)決定下一步移動的方向

(B)決定下一步移動的大小

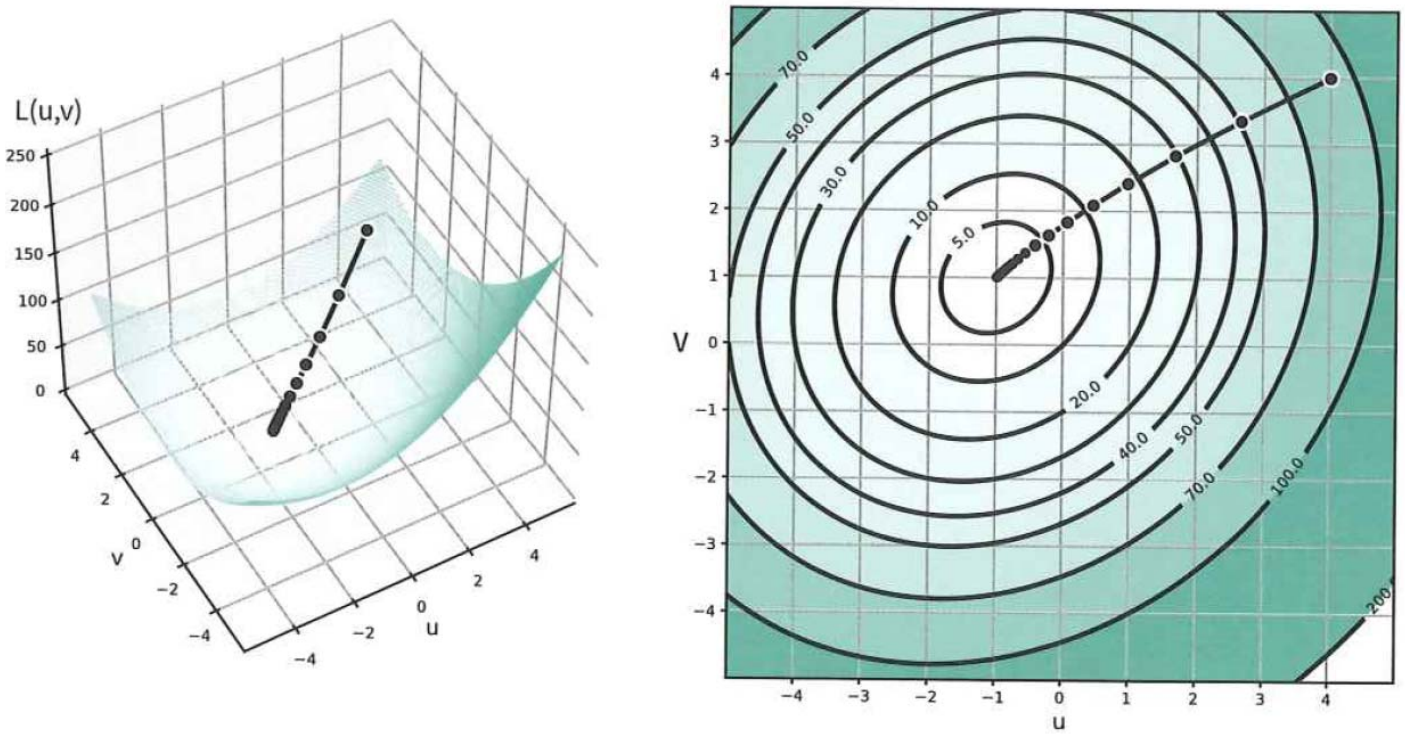


圖 4-11　迭代 20 次的圖形(左：曲面圖　右：等高線圖)

梯度下降法: 決定下一步移動的方向

- 假設現在尋找最低點已經送代k次，所以$(u,\ v)$已經移動到$(u_k\ v_k)$點的位置。那麼，從$(u_k\ v_k)$點要移動到下一個點$(u_{k+1}\ v_{k+1})$，該往那個方向移動?
- 要尋找向量的方向有兩個前提:

  (1) 移動量(du, dv)很微小。

  (2) 移動量$\sqrt{(du)^2+(dv)^2}$ 是一個定值。

哪個移動方向能最有效讓$L(u,v)$的值下降?

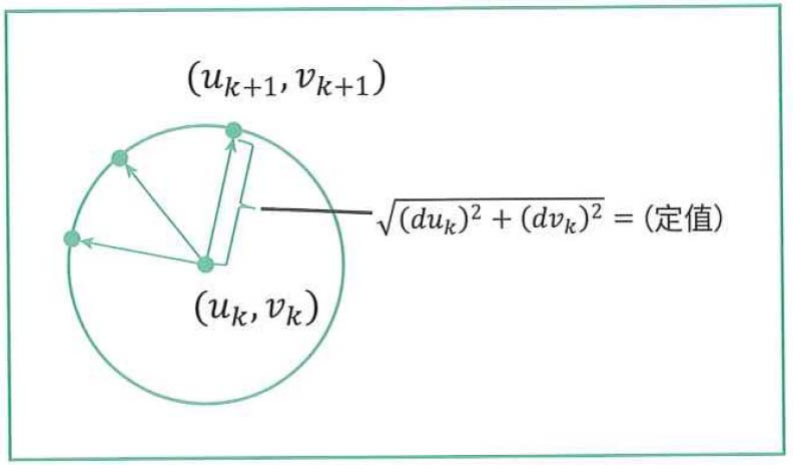


圖 4-12　移動到下一步的示意圖

假設函數 $L(u,v)$的變化量為$dL(u,v)$

$$dL(u_k, v_k) = L_u(u_k, v_k)du + L_v(u_k, v_k)dv$$

$$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \cos\theta$$

假設(L_u, L_v)與(du, dv)兩向量的夾角為θ

$$dL(u_k, v_k) = (L_u(u_k, v_k), L_v(u_k, v_k)) \cdot (du, dv)$$

$$dL(u_k, v_k) = (L_u(u_k, v_k), L_v(u_k, v_k)) \cdot (du, dv)$$
$$= |(L_u, L_v)|\ |(du, dv)| \cos\theta$$

梯度下降法: 決定下一步移動的方向

- 假設L是一個圓形為曲面的多變數函數，假設在$(u_k\ v_k)$點可微分，那麼 L在 $(u_k\ v_k)$上的**梯度**，就是 L在$(u_k\ v_k)$的**偏微分**。
- 要從$(u_k\ v_k)$點找一個能讓 $L(u,v)$值最小的方向，會由 $cos\theta$決定$dL(u_k\ v_k)$ 的大小。

若 (L_u, L_v) 和 (du, dv) 兩向量：

- 方向相同：$\cos 0^\circ = 1$，表示 $dL(u_k, v_k)$ 為最大值。
- 互相垂直：$\cos 90^\circ = 0$，表示 $dL(u_k, v_k)$ 為 0。
- 方向相反：$\cos 180^\circ = -1$，表示 $dL(u_k, v_k)$ 為最小值。

梯度dL是$(L_u\ L_v)$和(du, dv)兩向量在$(u_k\ v_k)$點的最大移動量，所以將梯度大小乘以-1就是會讓$L(u,v)$值下降的方向。

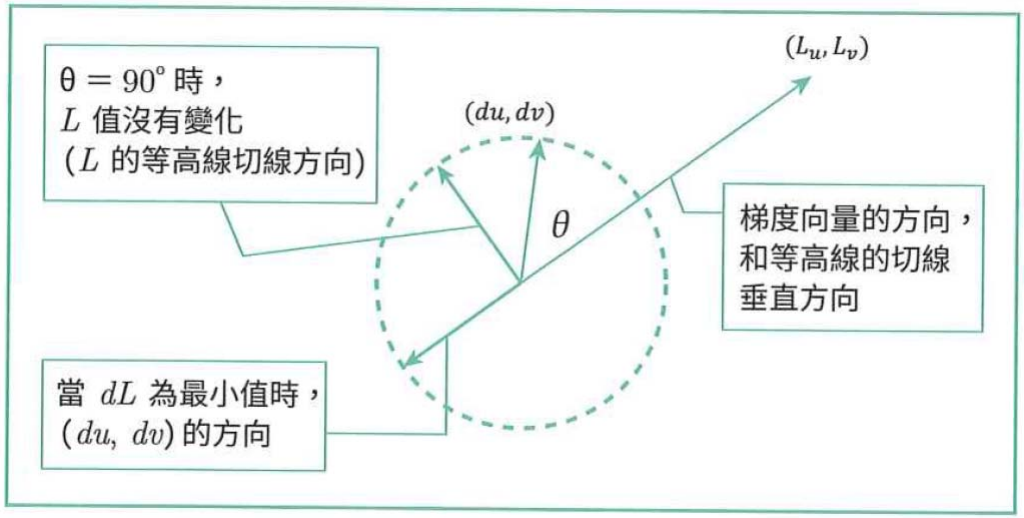


圖 4-13　(du, dv) 的方向與 $L(u, v)$ 值的關係

- **結論**: 只要找到 $L(u,v)$在$(u_k\ v_k)$偏微分向量 $(L_u(u_k\ v_k), L_v(u_k\ v_k))$的方向，其反方向就能找到讓$L(u,v)$值最小的$(u,v)$。
- **(L_u, L_v)即為梯度向量**。梯度向量會指向能讓函數值增加最多的方向，因此必須反方向才能讓函數值減少最多。

梯度下降法: 決定下一步移動的大小

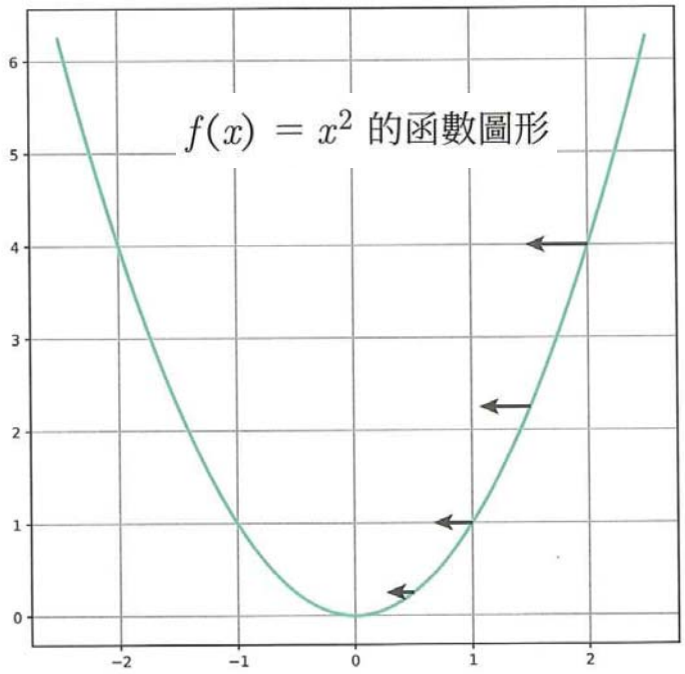


圖 4-14　微分值與移動量的關係

箭頭大小代表函數上 4 個點的微分值分別乘以固定值 (-0.1) 後產生的 x 方向移動量。

當 $x = 0$ 時，$f(x)$ 會有最小值，我們可發現：

- 離 $x = 0$ 較遠 $\longrightarrow$ $f(x)$ 的微分變大 (即斜率變大)，移動量也變大
- 離 $x = 0$ 較近 $\longrightarrow$ $f(x)$ 的微分變小 (即斜率變小)，移動量也變小

結論: 各點的微分乘上一個固定的負值，就可變成適當的移動量。

下一步 (u_{k+1}, v_{k+1}) 的位置

梯度下降法公式

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix} - \underbrace{\alpha \begin{pmatrix} L_u(u_k, v_k) \\ L_v(u_k, v_k) \end{pmatrix}}_{\text{移動量}}$$

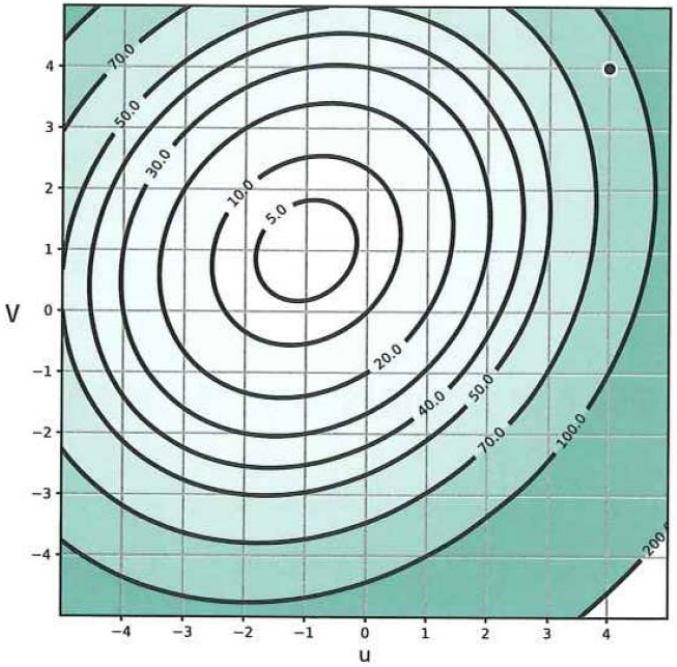
學習率

學習率的影響：

- 若學習率太大 :造成移動量偏大而越過最低點，無法收斂到極小值。
- 若學習率太小 :造成移動量過小，計算時間偏長，學習效率會變差。
- 實際上，在機器學習和深度學習中，都必須針對不同的問題來設定適當的學習率，再反覆尋找收斂的最佳解(**多半由經驗而得**) 。

等高線與梯度向量

- 等高線是由相同L值的各個點連接起來的曲線。
- **等高線上每一點的梯度向量，都會與等高線的切線垂直。**
- 梯度向量的方向代表函數值增加最快的方向，將梯度向量乘上-1使其反過來朝函數值最小的方向。
- 梯度向量的長度越長，表示梯度下降得越多;接近最低點時，梯度向量越短，下降也就減少。



等高線上某一點$(u,\ v)$的梯度向量是$\left(\frac{\partial L}{\partial u},\ \frac{\partial L}{\partial v}\right)$，也就是$(L_u,\ L_v)$。

假設：$u = u(t)$、$v = v(t)$

表示$(u(t),\ v(t))$在等高線上的位置會隨著 t 變動。

因此計算 u、v 微分，就 是計算$\left(\frac{du}{dt},\ \frac{dv}{dt}\right)$：

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{u(t+\Delta t) - u(t)}{\Delta t}$$

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{v(t+\Delta t) - v(t)}{\Delta t}$$

因此，$L(u,\ v)$可以寫成 $L(u(t),\ v(t))$。

假設在該點的等高線值等於 c，可知： $L(u(t),\ v(t)) = c$

$$\frac{\partial L}{\partial u}\frac{du}{dt} + \frac{\partial L}{\partial v}\frac{dv}{dt} = \frac{dc}{dt} = 0 \quad \Rightarrow \quad \left(\frac{\partial L}{\partial u},\ \frac{\partial L}{\partial v}\right) \cdot \left(\frac{du}{dt},\ \frac{dv}{dt}\right) = 0$$

梯度向量　　等高線切線向量

兩者內積為 0，即表示互相垂直。

等高線與梯度向量

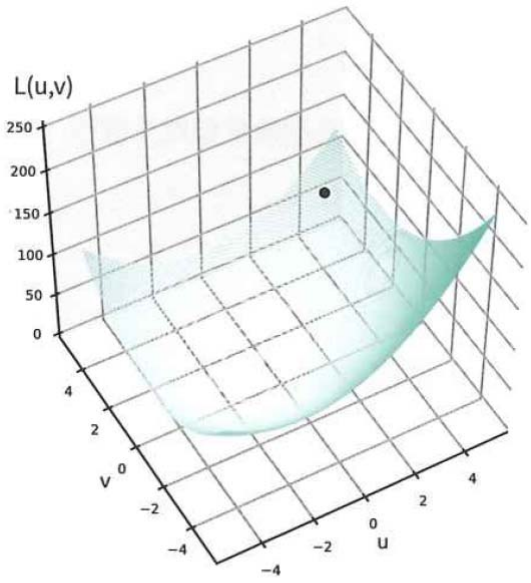


圖 4-8a 圖中黑點為曲面上 $(u,\ v)$ 初值的位置

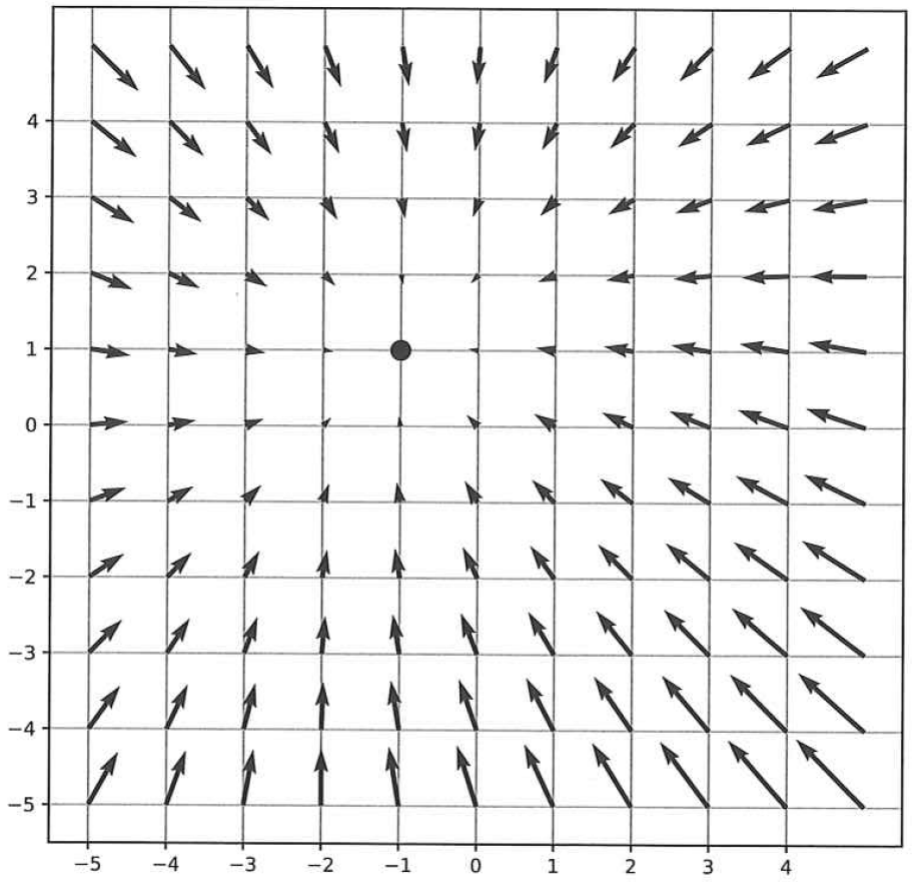
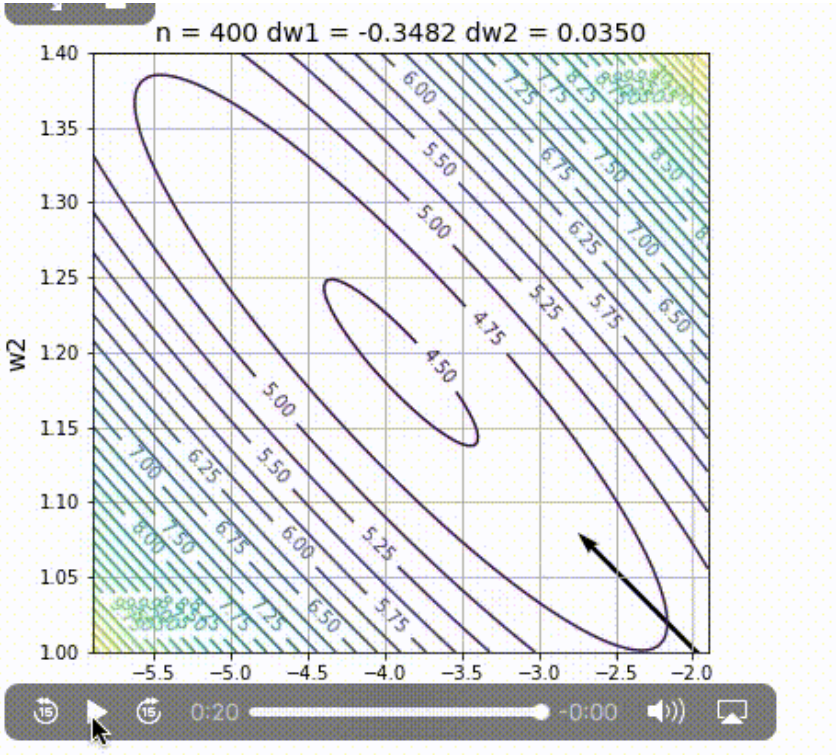


圖 4-15 $(u,\ v)$ 平面上各點的梯度向量示意圖

圖4-15，$(u,v)=(4,-4)$ 附近的梯度向量比較長。同理，$(u,v)=(-2,0)$附近各點向下的梯度比較小，梯度向量比較短。

梯度下降法的動畫範例

https://github.com/makaishi2/math-sample/blob/master/movie/gradient-descent.gif



如果函数 L 為三變數函數 $L(u, v, w)$，則梯度下降法的公式如下：

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \\ w_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \\ w_k \end{pmatrix} - \alpha \begin{pmatrix} L_u(u_k, v_k, w_k) \\ L_v(u_k, v_k, w_k) \\ L_w(u_k, v_k, w_k) \end{pmatrix}$$

學習率　　梯度向量

梯度下降法與局部最佳解

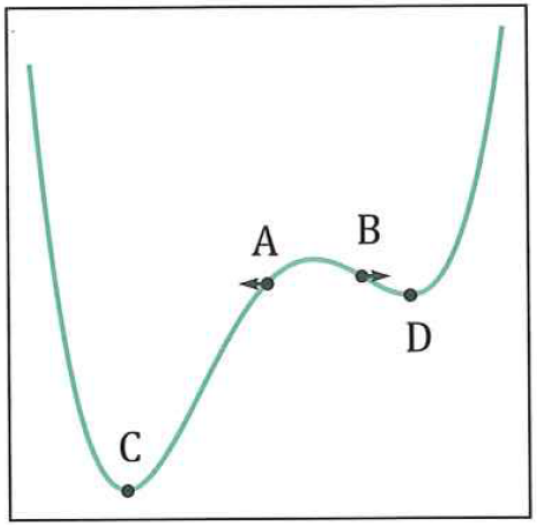


圖 4-16　因為有兩個局部最佳解，而無法決定最小值的例子

一開始選的初始值不同，梯度下降法未必都能找到函數真正的最小值。

- 因為梯度下降法是一次將所有的資料樣本都送進去運算，每次迭代時又全部再送去運算，當資料量龐大時，可能造成電腦負荷太大而無法處理，因此有兩種衍生的方法:
- 「**隨機梯度下降法**(Stochαstic Gradient Descent, SGD) 」: 每次送代只隨機挑選一筆資料做運算，雖然能夠減少只找到局部最佳解的機會，但其缺點是隨機選到的位置(資料樣本)可能會跳得很遠，而造成不易收斂。
- 「**小批量梯度下降法**(Mini Batch Gradient Descent, MBGD) J :每次迭代只挑選小批量資料分批運算。因為是一次取多筆資料運算，比較能做到穩定收斂。而且只要每個小批量的資料筆數夠多，就真有代表整體資料分佈的特性。這也是目前最普遍採用的方法。

H00-2

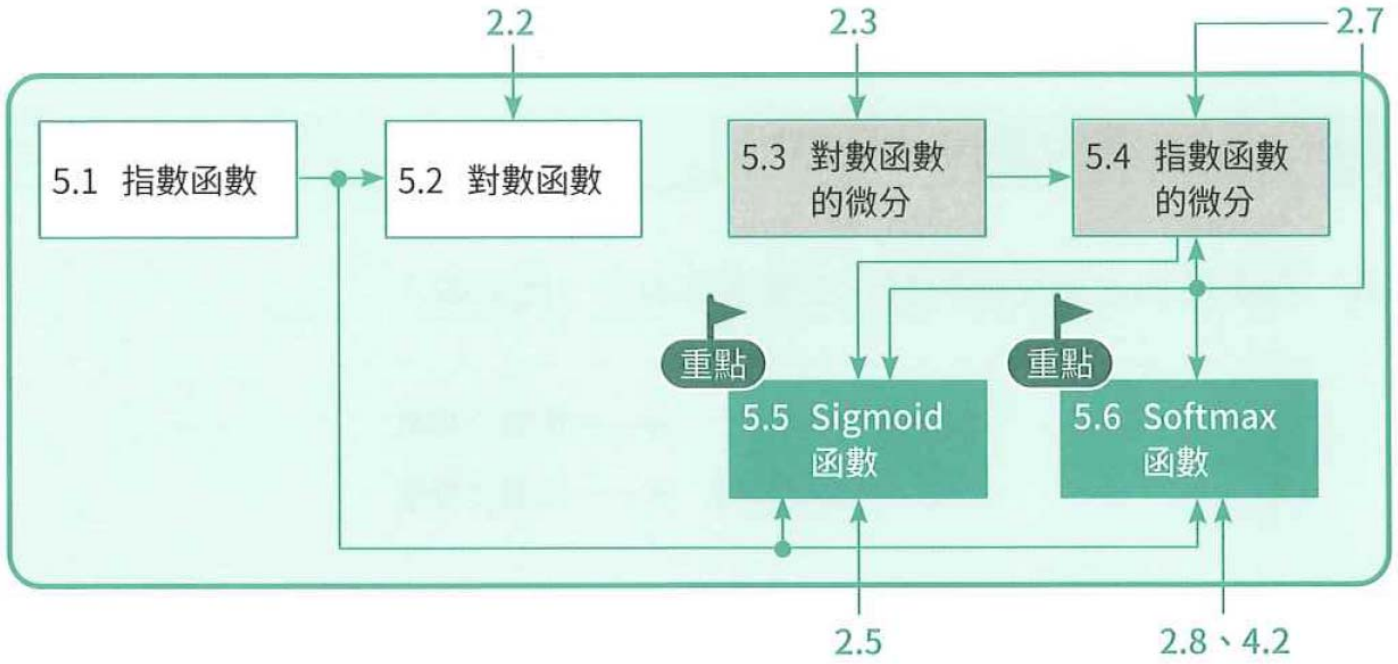
機器學習(ML)
深度學習(DL)
人工智慧(AI)
的
數學基礎 (II)

微積分、線性代數
機率、深度學習

吳漢銘
國立政治大學 統計學系

https://hmwu.idv.tw

Chapter 5: 指數函數、對數函數



指數函數$f(x) = a^x$

指數函數: $f(x) = a^x$

$$8 = 2^3$$

3 稱為指數

2 稱為底數

x	-2	$-\frac{3}{2}$	-1	$-\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{3}{2}$	2
$f(x)$	$\frac{1}{4}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	1	$\sqrt{2}$	2	$2\sqrt{2}$	4

表 5-1 $f(x)=2^x$ 的 9 個座標點

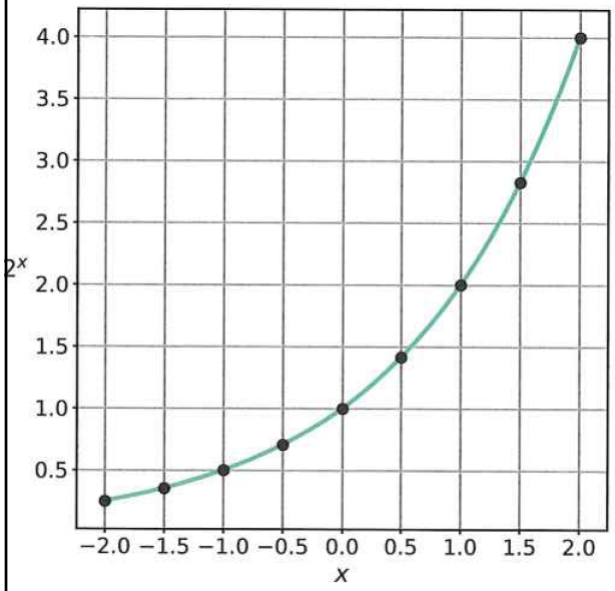


圖 5-1 $f(x)=2^x$ 的函數圖形

x	-2	$-\frac{3}{2}$	-1	$-\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{3}{2}$	2
$f(x)$	4	$2\sqrt{2}$	2	$\sqrt{2}$	1	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{4}$

表 5-2 $f(x)=\left(\frac{1}{2}\right)^x$ 的 9 個座標點

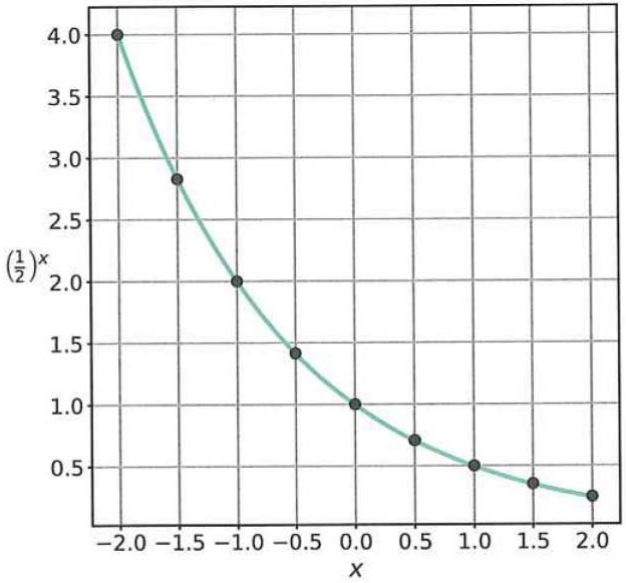


圖 5-2 $f(x)=\left(\frac{1}{2}\right)^x$ 的函數圖形

$$a^m \times a^n = a^{m+n}$$

$$(a^m)^n = a^{m \times n}$$

$$a^{-m} = \frac{1}{a^m}$$

$$a^{\frac{q}{p}} = (\sqrt[p]{a})^q$$

$$a^{\frac{1}{n}} = \sqrt[n]{a}$$

對數函數$f(x) = \log_b(x)$

真數，必須大於 0

$\log_b a = x \iff b^x = a$

底數，必須是不為 1 的正數

「以 b 為底，a 的對數為 x」。

其中，$b > 0$ 且 $b \neq 1$，$a > 0$，x 則為任意實數。

對數函數: $f(x) = \log_b(x)$
自然對數函數: $f(x) = \ln(x)$

指數算式	對數算式
$2^3=8$	$\log_2 8=3$
$3^4=81$	$\log_3 81=4$
$10^0=1$	$\log_{10} 1=0$
$10^{-2}=0.01$	$\log_{10} 0.01=-2$
$e^1=e$	$\log_e e=1$

表 5-3 指數及對數的對應表

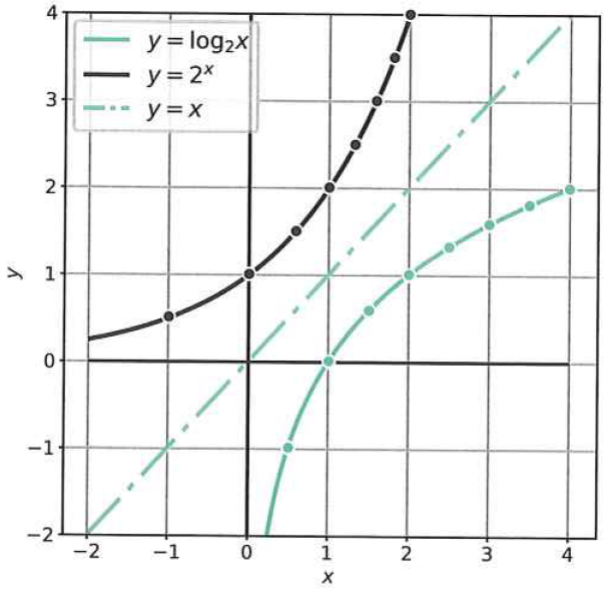


圖 5-4 指數與對數圖形對稱於 $y = x$ 直線

- 對數函數是指數函數的反函數。
- 對數函數是指數函數的反運算，從底數及乘積來找出對數。
- $log_a(a^x) = x, a^{log_a x} = x$
- 對數函數的變數x永遠是正值。而其反函數，指數函數的y值也永遠是正值。

$e^{lnx} = \underline{\quad x \quad}, \quad \forall x > 0$

$\ln(e^x) = \underline{\quad x \quad}, \quad \forall x$

對數函數$f(x) = \log_b(x)$的特徵

$$x = \log_a X \quad \Leftrightarrow \quad a^x = X$$
$$y = \log_a Y \quad \Leftrightarrow \quad a^y = Y$$

$$\log_a (X \times Y) = \log_a X + \log_a Y$$

$$\log_a \left(\frac{Y}{X}\right) = \log_a Y - \log_a X$$

$$\log_a \left(\frac{1}{X}\right) = -\log_a X$$

$$\log_a (X^y) = y \log_a X$$

$$\begin{aligned} & a^x \times a^y = a^{x+y} \\ \Leftrightarrow\ & X \times Y = a^{x+y} \\ \Leftrightarrow\ & \log_a (X \times Y) = x + y \\ \Leftrightarrow\ & \log_a (X \times Y) = \log_a X + \log_a Y \end{aligned}$$

$$\begin{aligned} & \frac{a^y}{a^x} = a^{y-x} \\ \Leftrightarrow\ & \frac{Y}{X} = a^{y-x} \\ \Leftrightarrow\ & \log_a \left(\frac{Y}{X}\right) = y - x \\ \Leftrightarrow\ & \log_a \left(\frac{Y}{X}\right) = \log_a Y - \log_a X \end{aligned}$$

$$\begin{aligned} & (a^x)^y = a^{xy} \\ \Leftrightarrow\ & X^y = a^{xy} \\ \Leftrightarrow\ & \log_a (X^y) = xy \\ \Leftrightarrow\ & \log_a (X^y) = y \log_a X \end{aligned}$$

換底公式

$$\log_a X = \frac{\log_b X}{\log_b a}$$

$X = a^x$

$x = \log_a X$

$\log_b X = \log_b (a^x)$

$\log_b (a^x) = x \log_b a$

$\log_b X = \log_a X \log_b a$

對數函數$f(x) = \log_b(x)$的意義

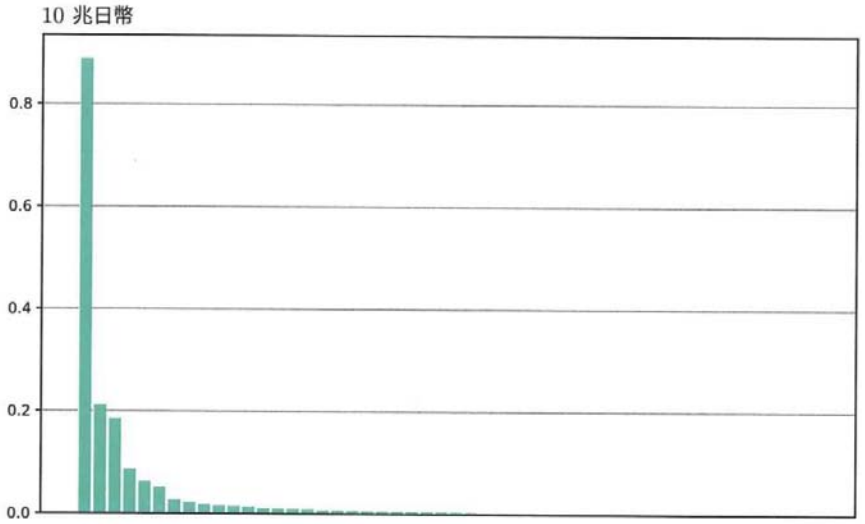


圖 5-5　營業額前 50 名的公司 (以營業額為比例尺)

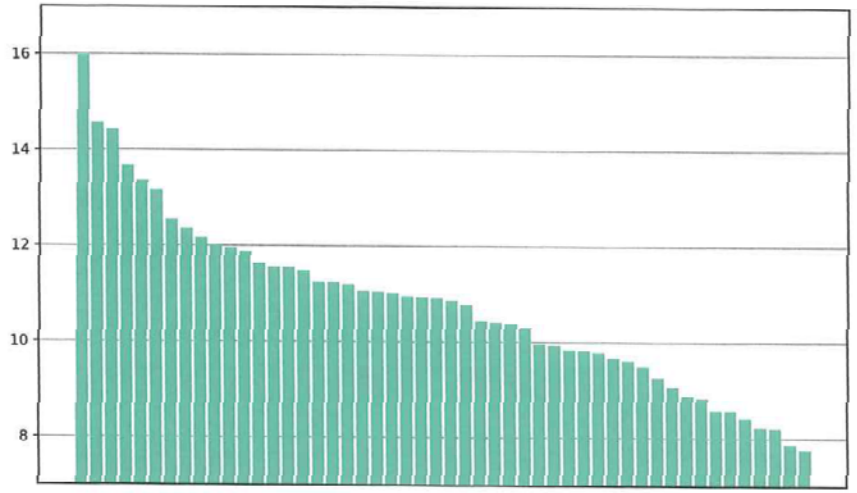
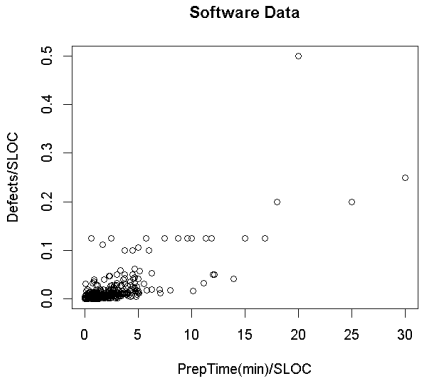
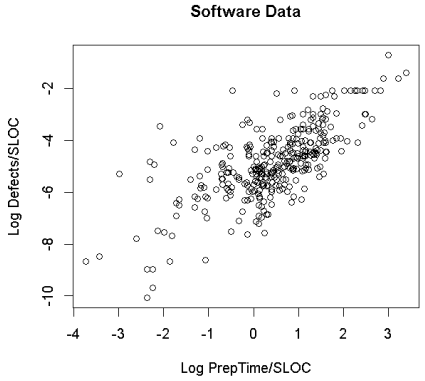
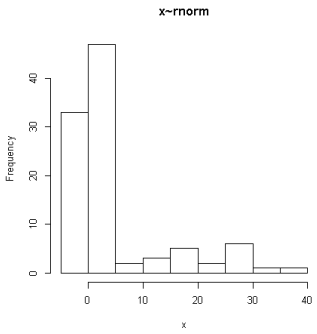
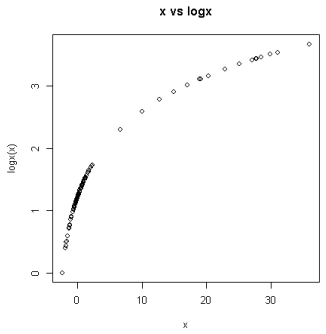


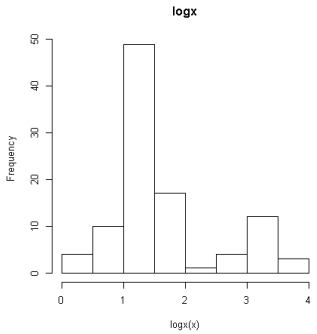
圖 5-6　營業額前 50 名的公司 (以營業額取對數為比例尺)











對數函數$f(x) = \log_b(x)$的微分

$$f'(x) = \frac{d}{dx}\log_a x = \frac{1}{x}\,\frac{1}{\log_e a}$$

$$f'(x) = \frac{d}{dx}\log_e x = \frac{1}{x}$$

$$f'(x) = \lim_{h\to 0}\frac{\log_a(x+h) - \log_a x}{h}$$

$$\log_a(x+h) - \log_a x = \log_a\left(\frac{x+h}{x}\right) = \log_a\left(1+\frac{h}{x}\right)$$

$$h' = \frac{h}{x}$$

$$f'(x) = \lim_{h'\to 0}\frac{\log_a(1+h')}{xh'} = \frac{1}{x}\lim_{h'\to 0}\frac{\log_a(1+h')}{h'} = \frac{1}{x}\lim_{h'\to 0}\log_a\left((1+h')^{\frac{1}{h'}}\right)$$

Theorem 4: The Number e as a Limit

The number e can be calculated as the limit: $e = \lim_{x\to 0}(1+x)^{1/x}$

$$\lim_{h\to 0}\log_a(1+h)^{\frac{1}{h}} = \log_a \underbrace{\lim_{h\to 0}(1+h)^{\frac{1}{h}}}_{\text{此極限值會收斂到 } 2.71828\cdots\text{，也就是尤拉數 } e} = \log_a e = \frac{1}{\log_e a}$$

尤拉數 e

指數函數$f(x) = a^x$的微分

$$\frac{d}{dx}(a^x) = (\log a)a^x \qquad (e^x)' = e^x$$

$y = a^x$ ⇒ $\log y = \log a^x = x \log a$

$$\frac{d(\log y)}{dx} = \frac{d(x \log a)}{dx} = \log a$$

$$\frac{d(\log y)}{dx} = \frac{d(\log y)}{dy}\frac{dy}{dx} = \frac{1}{y}\frac{dy}{dx}$$

$$\log a = \frac{1}{y}\frac{dy}{dx}$$

⇒ $y' = \frac{dy}{dx} = (\log a)y = (\log a)a^x$

$f(x)$	e^x	$\ln x$	a^x	$\log_a x$
$\frac{d}{dx}f(x)$	e^x	$\frac{1}{x}$	$a^x \ln a$	$\frac{1}{x}\frac{1}{\ln a}$
$\int f(x)\,dx$	$e^x + c$	$x \ln x - x + c$	$\frac{a^x}{\ln a} + c$	$\frac{1}{\ln a}(x \ln x - x) + c$

$a^x = e^{x \ln a}$

$\log_a x = \frac{\ln x}{\ln a}$

Sigmoid函數 $f(x) = \frac{1}{1+e^{-x}}$

- Sigmoid函數(也稱為S函數)是用於將輸入值轉換為0~1 (0%~100%)的值，因此可當做機率值來使用。

性質:

- 函數值介於0~1之間的單調遞增函數 。
- x值趨近-∞時，函數值趨近0 。
- x值趨近∞時，函數值趨近1 。
- $x = 0$時，函數值為0.5 。
- 圖形在座標(0， 0.5)會呈對稱的轉折。

$$y = \frac{1}{1 + \exp(-x)}$$

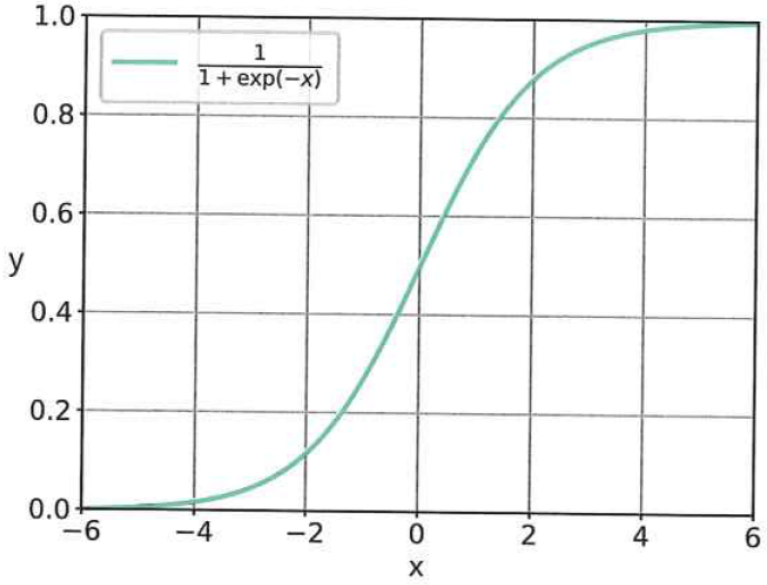


圖 5-9　*Sigmoid* 函數的圖形

Sigmoid 函數一般常見的形式是寫成 $y = \frac{1}{1+\exp(-(ax+b))}$，其中 a 為正數。當取 $a = 1$、$b = 0$ 時則為最簡單的形式。

Sigmoid函數 $f(x)=\frac{1}{1+e^{-x}}$ 的微分

很重要，背起來！

$$y=\frac{1}{1+\exp(-x)} \Rightarrow y'=y(1-y)$$

$$\boxed{\frac{dy}{dx}=f'(x)=y(1-y)}$$

訓練神經網路中若採用Sigmoid函數做為激活函數時，就會用到此微分式。

令 $u(x)=1+\exp(-x)$ $\Rightarrow$ $y(u)=\frac{1}{u}$ $\Rightarrow$ $\frac{dy}{dx}=\frac{dy}{du}\cdot\frac{du}{dx}$

$$\frac{dy}{du}=\left(\frac{1}{u}\right)'=(u^{-1})'=(-1)\cdot u^{-2}=-\frac{1}{u^2}$$

令 $v=-x$

$u=1+\exp(-x)=1+\exp(v)$ $\Rightarrow$ $\frac{du}{dx}=\frac{du}{dv}\cdot\frac{dv}{dx}=\exp(v)\cdot(-1)=-\exp(-x)$

$$\frac{dy}{dx}=-\frac{1}{u^2}\cdot-\exp(-x)=\frac{\exp(-x)}{(1+\exp(-x))^2}=\frac{1+\exp(-x)-1}{(1+\exp(-x))^2}$$

$$=\frac{1}{1+\exp(-x)}-\frac{1}{(1+\exp(-x))^2}=y-y^2=y(1-y)$$

Softmax函數
i.e., 常規化指數函數 (normalized exponential function)

The standard (unit) softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

- Softmax函數，輸入的是向量(vector)，輸出的也是個向量，且各分量皆介於0~1之間。
- 因為Softmax函數輸出與輸入值皆為向量，所以又稱為向量函數(或向量值函數)

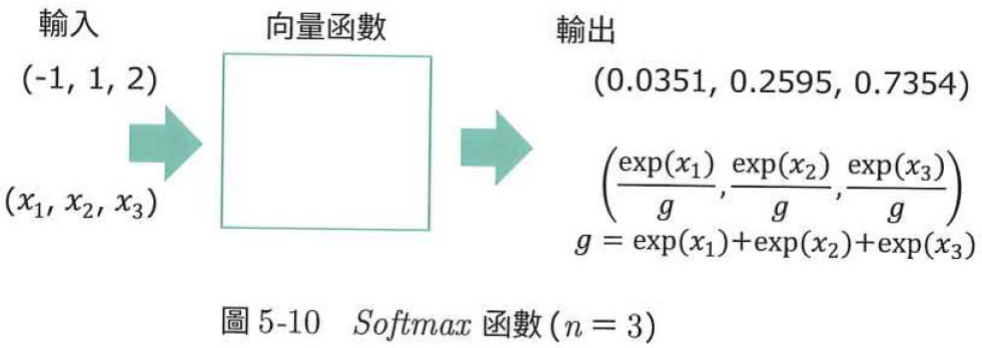


圖 5-10　*Softmax* 函數 $(n = 3)$

也因為輸出向量的每個分量都會介於0~1之間，可當做機率值來使用。例如:輸入一張動物照片要判斷是狗、貓、鼠，運算後的輸出向量為 $(y_1, y_2, y_3) = (0.85, 0.10, 0.05)$，則可判斷該照片應該是狗。

輸入向量：$(x_1,\ x_2,\ x_3)$

輸出向量：$(y_1,\ y_2,\ y_3)$

輸出向量的各分量值 (都介於 0~1)：

$$\begin{cases} y_1 = \dfrac{\exp(x_1)}{g(x_1, x_2, x_3)} \\ y_2 = \dfrac{\exp(x_2)}{g(x_1, x_2, x_3)} \\ y_3 = \dfrac{\exp(x_3)}{g(x_1, x_2, x_3)} \end{cases}$$

$$g(x_1, x_2, x_3) = \exp(x_1) + \exp(x_2) + \exp(x_3)$$

$$y_1 + y_2 + y_3 = 1$$

$$0 \le y_i \le 1 \quad (i = 1, 2, 3)$$

Softmax函數的微分

$$\begin{cases} y_1 = \dfrac{\exp(x_1)}{g(x_1,x_2,x_3)} \\ y_2 = \dfrac{\exp(x_2)}{g(x_1,x_2,x_3)} \\ y_3 = \dfrac{\exp(x_3)}{g(x_1,x_2,x_3)} \end{cases} \qquad g(x_1,x_2,x_3) = \exp(x_1)+\exp(x_2)+\exp(x_3)$$

✡也很重要！

$$\frac{\partial y_j}{\partial x_i} = \begin{cases} y_i(1-y_i) & (i=j) \\ -y_i y_j & (i \neq j) \end{cases}$$

先令 $\exp(x_1) = h(x_1)$ ➡ $y_1 = \dfrac{h(x_1)}{g(x_1,x_2,x_3)} = \dfrac{h}{g}$

$$h_{x_1} = \exp(x_1)' = \exp(x_1) = h$$

➡ $\dfrac{\partial y_1}{\partial x_1} = \dfrac{g \cdot h_{x_1} - h \cdot g_{x_1}}{g^2}$ ➡ $g_{x_1} = \dfrac{\partial g}{\partial x_1} = \exp(x_1)' = \exp(x_1) = h$

➡ $\dfrac{\partial y_1}{\partial x_1} = \dfrac{g \cdot h - h \cdot h}{g^2} = \dfrac{h}{g} \cdot \dfrac{g-h}{g} = \dfrac{h}{g} \cdot \left(1 - \dfrac{h}{g}\right) = y_1(1-y_1)$

$y_2 = \dfrac{\exp(x_2)}{g(x_1,x_2,x_3)} = \dfrac{h(x_2)}{g}$ ➡ $\dfrac{\partial y_2}{\partial x_1} = \dfrac{g \cdot h(x_2)_{x_1} - h(x_2) \cdot g_{x_1}}{g^2} = \dfrac{g \cdot 0 - h(x_2) \cdot g_{x_1}}{g^2} = -\dfrac{h(x_2) \cdot g_{x_1}}{g^2}$

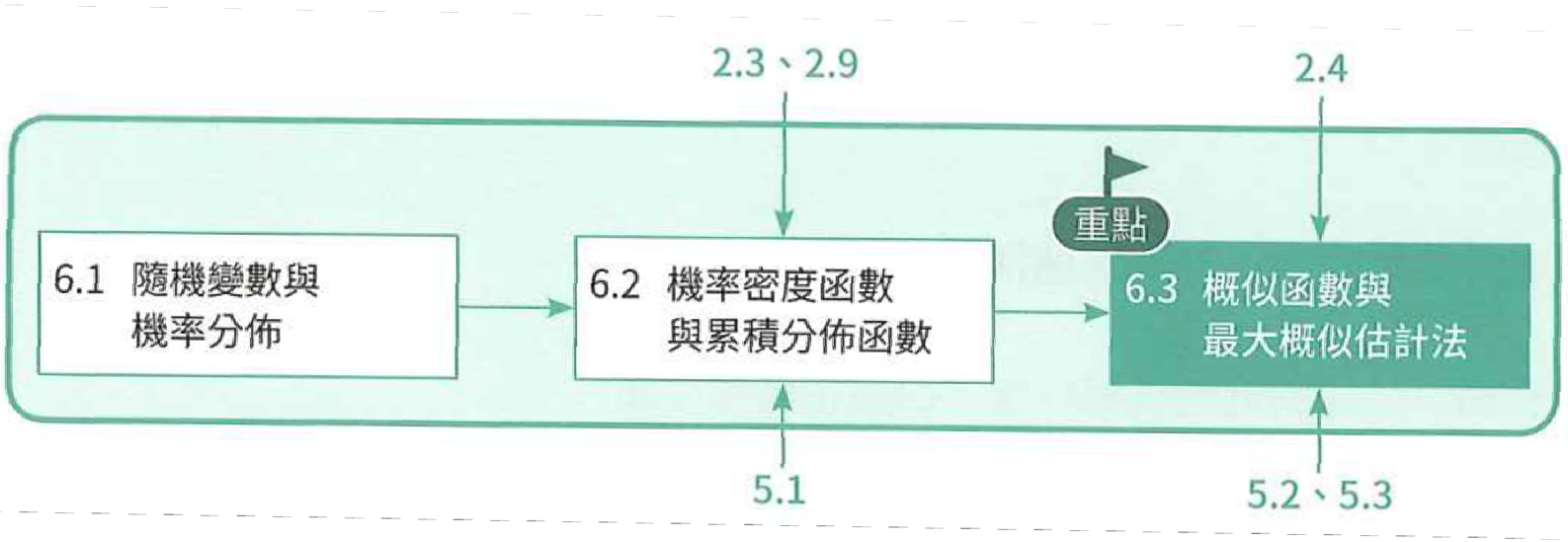
➡ $\dfrac{\partial y_2}{\partial x_1} = -\dfrac{h(x_2) \cdot h(x_1)}{g^2} = -\dfrac{h(x_2)}{g} \cdot \dfrac{h(x_1)}{g} = -y_2 \cdot y_1$

Sigmoid 和 Softmax函數的關係

$$y_1 = \frac{\exp(x_1)}{\exp(x_1) + \exp(x_2)} = \frac{1}{1 + \exp(-(x_1 - x_2))}$$

- 當輸入的是 2維向量 $(n = 2)$ 時，對 Softmax函數進行運算(將分子分母同除以 $\exp(x_1)$)。
- 將$x_1 - x_2$用x取代，就和Sigmoid函數的式子完全一樣。
- 當 $(n = 2)$ 時，Softmax函數實質上就等於Sigmoid函數。也就是說 Sigmoid函數是Softmax的函數的簡化版本。
- 這兩個函數都是屬於分類函數。

Chapter 6: 機率統計



隨機變數(random variable)與機率分佈(probability distribution)

機率在機器學習的分類模型中的重要性:

- 機率是非常重要的基礎，因為**分類模型**就是藉由**激活函數**將運算結果**用機率呈現**，才得以依據輸入資料預測最後應該分到哪一類。

機率是用百分比來表示某事件發生的可能性。

- 在數學上，~~將事件 X,Y 發生的機率分別用 $P(X),P(Y)$來表示~~。
- 在數學上，將事件 A,B 發生的機率分別用 $P(A),P(B)$來表示。
- 機率符號中的 X,Y稱為隨機變數(隨機變數是一個函數，將事件對應到實數)。

隨機變數 X	正面	反面
$P(X)$	$\frac{1}{2}$	$\frac{1}{2}$

表 6-1　X(硬幣)的機率分佈

(離散型)機率分佈例子:

- 「公正的硬幣投擲一次」的樣本空間= {正面，反面}共有 2種，出現機率各為1/2。
- X代表「公正的硬幣投擲一次，出現正面或反面」: $X(\text{正面})=1,\ X(\text{反面})=0$。
- $P(\text{正面})=P(X(\text{正面})=1)=P(X=1)=1/2;\ P(\text{反面})=P(X(\text{正面})=0)=P(X=0)=1/2$

- 「公正的骰子投擲一次」的樣本空間={1，2， 3， 4， 5， 6}共有 6種，出現機率各為1/6。
- Y代表「公正的骰子投擲一次，骰子的點數」: $Y(\text{點數}1)=1$，…，$Y(\text{點數}6)=6$。
- $P(\text{點數}k)=P(Y(\text{點數}k)=k)=P(Y=k)=\frac{1}{6},k=1,\ldots,6.$

隨機變數 Y	1	2	3	4	5	6
$P(Y)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

表 6-2　Y(骰子)的機率分佈

若將隨機變數每種情況的機率值都列出來，就是其機率分佈。

編註： 所謂機率分佈，就是每個可能出現的值 (例如：1、2、3、4、5、6)，各有多少機率 (例如：$\frac{1}{6}$、$\frac{1}{6}$、$\frac{1}{6}$、$\frac{1}{6}$、$\frac{1}{6}$、$\frac{1}{6}$)，把它們全部列出來，如上表所示。

範例: 硬幣投擲n次時，正面出現的次數

將隨機變數X_n定義為：X_n=「硬幣投擲n次時，正面出現的次數」擲硬幣這種只有二擇一(正面、反面)的**機率分佈**又稱為「**二項式分佈**」。

$n=1$ 時(正面出現 0 次、1 次的機率)

隨機變數 X_1	0	1
$P(X_1)$	$\frac{1}{2}$	$\frac{1}{2}$

$n=2$ 時(正面出現 0 次、1 次、2 次的機率)

隨機變數 X_2	0	1	2
$P(X_2)$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{1}{4}$

$n=3$ 時(正面出現 0 次、1 次、2 次、3 次的機率)

隨機變數 X_3	0	1	2	3
$P(X_3)$	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

$n=4$ 時(正面出現 0 次、1 次、2 次、3 次、4 次的機率)

隨機變數 X_4	0	1	2	3	4
$P(X_4)$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{6}{16}$	$\frac{4}{16}$	$\frac{1}{16}$

表 6-3　二項分佈的機率分佈

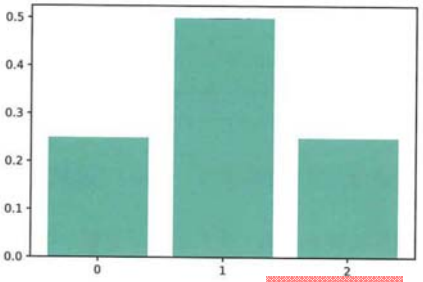


圖 6-1　擲 2 個硬幣很多次的機率分佈直方圖

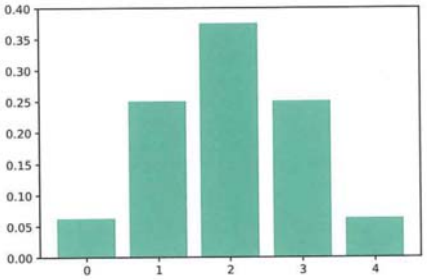


圖 6-3　擲 4 個硬幣很多次的機率分佈直方圖

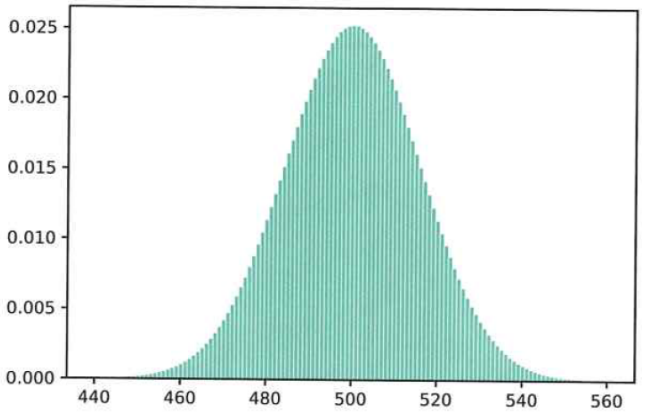


圖 6-6　擲 1000 個硬幣很多次的機率分佈直方圖

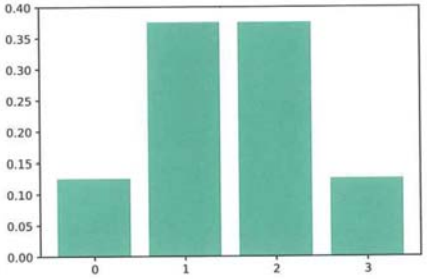


圖 6-2　擲 3 個硬幣很多次的機率分佈直方圖

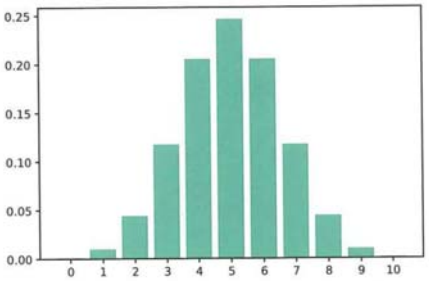
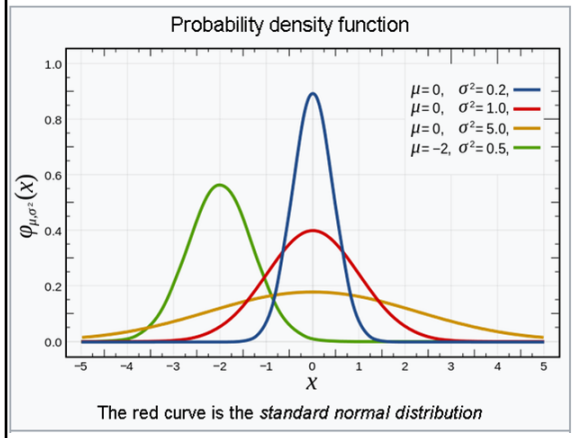


圖 6-4　擲 10 個硬幣很多次的機率分佈直方圖

編註： 請注意！這邊 $n=1$、$n=2$、$n=3$、… 並不是只擲 1 次、2 次、3 次！例如：圖 6-1，你擲 2 次硬幣不會有那樣的圖出現！圖 6-1 是機率分佈，意思是說，如果以「擲 2 個硬幣為一組」來觀察，則擲很多組之後，會出現 0 次正面、1 次正面、2 次正面的機率分佈會像圖 6-1 那樣！這很重要，請勿誤解！

機率密度函數(probability density function)與累積分佈函數(cumulative distribution function)

- 當二項分佈的n值很大時，其機率分佈圖的形狀會呈**常態分佈**(亦稱為**高斯分佈**)，也就是呈現中央高起且兩側快速下降的鐘型曲線(bell-shaped curve)，這稱為「**中央極限定理**」。
- 這條連續的**常態分佈曲線**的函數稱為「**機率密度函數**」，定義如下(其中 μ為平均數， σ為標準差):



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

當擲 1 個硬幣的試驗做 n 次，則出現正面的平均數是 $\mu=np$ 變異數 $\sigma^2=np(1-p)$

因此可知，當 1 個硬幣出現正面的機率是 $p=\frac{1}{2}$ 時，則平均數 $\mu=np=\frac{n}{2}$，$\sigma^2=np(1-p)=\frac{n}{4}$。令 $\frac{n}{2}=m$，則投擲硬幣的機率就會近似於：

$$P(X_n=x)\approx\frac{1}{\sqrt{m\pi}}\exp\left(-\frac{(x-m)^2}{m}\right)$$

The Normal Approximation to the Binomial Distribution

WIKIPEDIA
The Free Encyclopedia

Article Talk

Binomial distribution

From Wikipedia, the free encyclopedia

Normal approximation [edit]

If n is large enough, then the skew of the distribution is not too great. In this case a reasonable approximation to B(n, p) is given by the normal distribution

$$\mathcal{N}(np,\ np(1-p)),$$

and this basic approximation can be improved in a simple way by using a suitable continuity correction. The basic approximation generally improves as n increases (at least 20) and is better when p is not near to 0 or 1.[22] Various rules of thumb may be used to decide whether n is large enough, and p is far enough from the extremes of zero or one:

https://en.wikipedia.org/wiki/Binomial_distribution

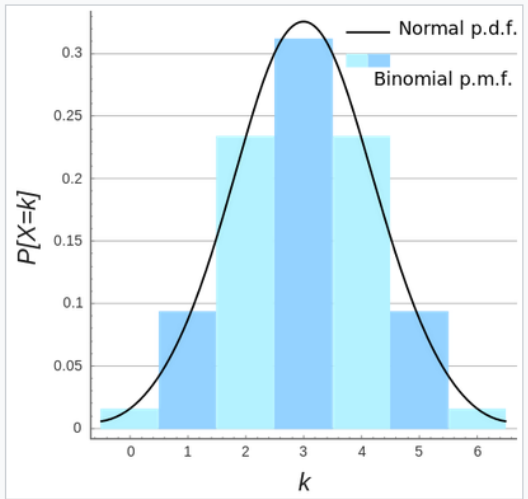
想知道擲 1000 次且出現正面不超過 480 次的機率

$$P(X_{1000} \leqq 480) \qquad m = \frac{1000}{2} = 500$$

$$P(X_{1000} \leq 480) \approx \int_0^{480} f(x)dx$$

$$f(x) = P(X_n = x) \approx \frac{1}{\sqrt{500\pi}} \exp\left(-\frac{(x-500)^2}{500}\right)$$

可知由 0~480 的積分結果約為 0.103，
表示擲 1000 次硬幣，出現正面不超過
480 次的機率約為 10.3%。



Binomial probability mass function and normal probability density function approximation for n = 6 and p = 0.5

Sigmoid函數的機率密度函數

$$f(x) = \frac{1}{1+\exp(-x)}$$

累積分佈函數：$f(x)$

機率密度函數：$f(x)\ (1-f(x))$

$$f'(x) = f(x)(1-f(x))$$

Derivative [edit]

The standard logistic function has an easily calculated derivative. The derivative is known as the logistic distribution:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x},$$

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{e^x \cdot (1+e^x) - e^x \cdot e^x}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} = f(x)\big(1-f(x)\big) = f(x)f(-x).$$

- Sigmoid函數就是某個機率密度函數的積分，因此只要將Sigmoid函數微分，就可以得到機率密度函數: f'(x) = f(x)(l - f(x))
- 當我們從機率的角度來看 Sigmoid函數，可整理成下面兩式:
  - 累積分佈函數: f(x)
  - 機率密度函數: f(x) (1 -f(x))
- **Sigmoid function is a cumulative distribution function of logistic distribution.**

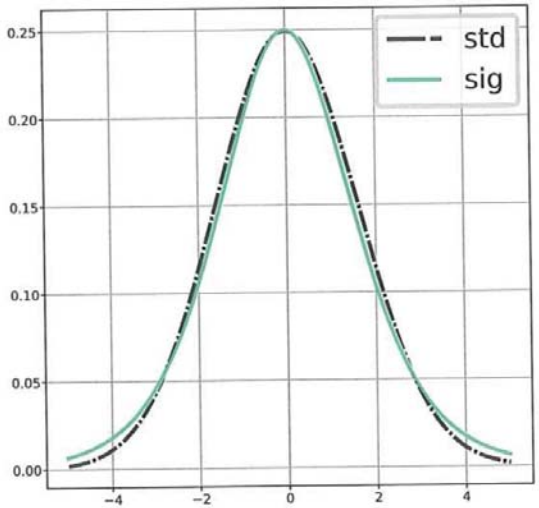


圖 6-12　兩個函數圖形相當接近

Logistic function

Logistic function

From Wikipedia, the free encyclopedia

For the recurrence relation, see Logistic map.

A **logistic function** or **logistic curve** is a common S-shaped curve (sigmoid curve) with equation

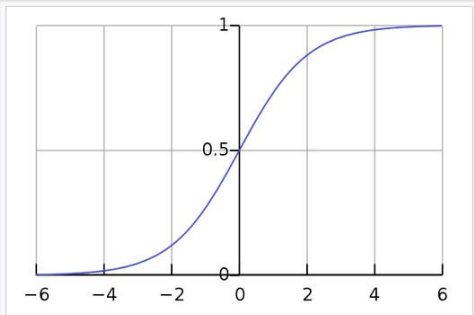
$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$

where

x_0, the x value of the sigmoid's midpoint;

L, the curve's maximum value;

k, the logistic growth rate or steepness of the curve.[1]



Standard logistic sigmoid function i.e.
$L = 1, k = 1, x_0 = 0$

WIKIPEDIA
The Free Encyclopedia

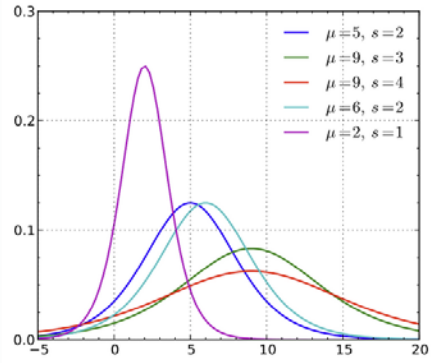
Article Talk

Logistic distribution

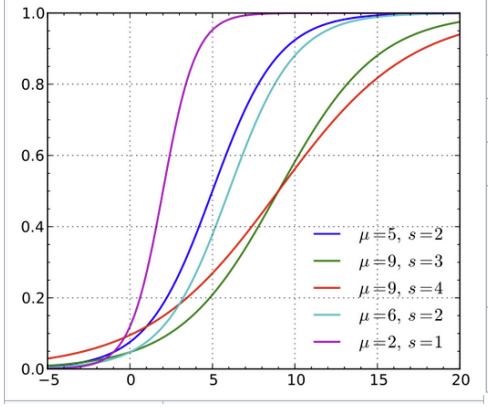
From Wikipedia, the free encyclopedia

Logistic

Probability density function



Cumulative distribution function



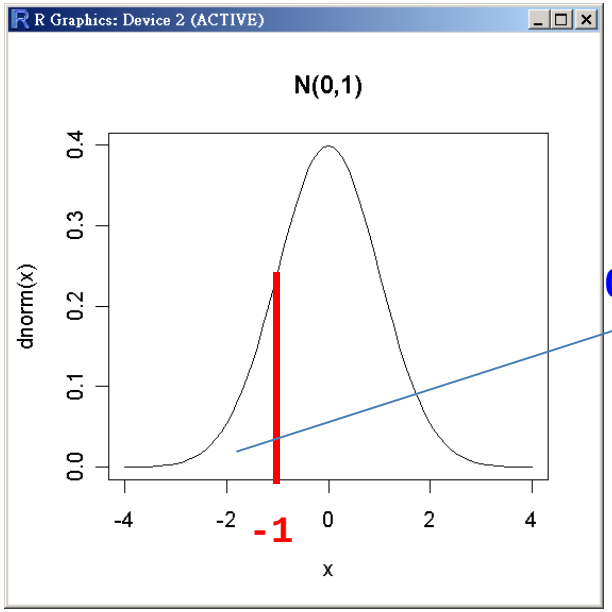
Parameters	μ, location (real) $s > 0$, scale (real)
Support	$x \in (-\infty, \infty)$
PDF	$\frac{e^{-(x-\mu)/s}}{s\left(1 + e^{-(x-\mu)/s}\right)^2}$
CDF	$\frac{1}{1 + e^{-(x-\mu)/s}}$
Mean	μ
Median	μ
Mode	μ
Variance	$\frac{s^2\pi^2}{3}$
Skewness	0
Ex. kurtosis	$6/5$
Entropy	$\ln s + 2$
MGF	$e^{\mu t}\mathrm{B}(1 - st, 1 + st)$ for $t \in (-1/s, 1/s)$ and B is the Beta function
CF	$e^{it\mu}\frac{\pi st}{\sinh(\pi st)}$

累積機率分配函數 CDF (p)

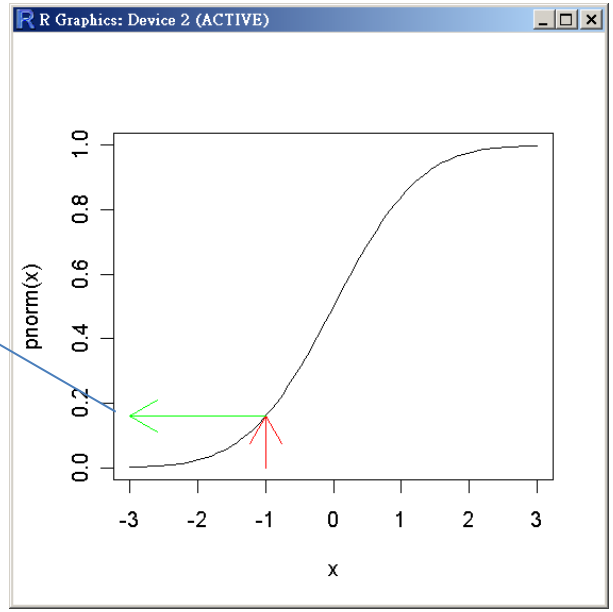
$$F_X(x) = P(X \leq x)$$

- The probability of obtaining a sample value that is less than or equal to x.

PDF



0.1586553



CDF

中央極限定理 (Central Limit Theorem)

- 由一具有平均數μ，標準差σ的母體中抽取樣本大小為n的簡單隨機樣本，當樣本大小n夠大時，**樣本平均數**的**抽樣分配**會近似於常態分配。

$X_1, X_2, X_3, \cdots$ be a set of n independent and identically distributed random variables having finite values of mean μ and variance $\sigma^2 > 0$.

$$S_n = X_1 + \cdots + X_n$$

$$Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}} \to N(0,1) \quad \text{as} \quad n \to \infty$$

$$Z_n = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}}$$

$$E(\bar{X}) = \mu_{\bar{X}} = \mu$$

$$Var(\bar{X}) = \sigma^2_{\bar{X}} = \frac{\sigma^2}{n}$$

- 在一般的統計實務上，大部分的應用中均假設當樣本大小為30(含)以上時 $\bar{X}_n$的抽樣分配即近似於常態分配。
- 當母體為常態分配時，不論樣本大小，樣本平均數的抽樣分配仍為常態分配。

概似函數(likelihood function)與最大概似估計法(maximum likelihood estimation, MLE)

假設有一台抽獎機，每次抽的中獎機率都不會改變，也就是說每次抽中與否，都與前一次是否抽中無關，表示每次抽都是獨立事件。

假設此抽獎機連抽 5 次，只有第 1 次和第 4 次中獎，其他 3 次沒有中獎。若每次中獎機率為 p，請推測最有可能的 p 值為多少？

抽獎機的機率模型

將隨機變數 X_i 定義為：$X_i = \begin{cases} 1\ (\text{中獎}) \\ 0\ (\text{沒中獎}) \end{cases}$

每次中獎的機率為 p

沒中獎的機率為 $(1-p)$

想要推估的參數就是 p 的值

則抽 5 次的中獎機率可分別寫為：

$$P(X=X_1)\cdot P(X=X_2)\cdot P(X=X_3)\cdot P(X=X_4)\cdot P(X=X_5)$$
$$=p\cdot(1-p)\cdot(1-p)\cdot p\cdot(1-p)$$
$$=p^2\cdot(1-p)^3 \qquad (6.3.2)$$

- 式子(6.3.2)稱為**概似函數 (L ikelihood function)**。
- 只要找出能讓概似函數出現極大值的p就是最能符合此抽獎機機率模型的答案。
- 要找出極大值，就是找出概似函數微分後等於 0的 p, 且此p可以讓概似函數出現極大值。
- 概似函數習慣上會用 L (Likelihood)做為函數名稱，但許多機器學習的書中習慣用L表示損失函數 (Loss function)，應避免還淆。

對數概似函數

$$P(X = X_1) \cdot P(X = X_2) \cdot P(X = X_3) \cdot P(X = X_4) \cdot P(X = X_5)$$
$$= p \cdot (1-p) \cdot (1-p) \cdot p \cdot (1-p)$$
$$= p^2 \cdot (1-p)^3$$

$$\log(p^2(1-p)^3) = 2\log p + 3\log(1-p)$$

$$\frac{2}{p} + \frac{3 \cdot (-1)}{1-p} = 0$$
$$\Leftrightarrow \quad 2(1-p) - 3p = 0$$
$$\Leftrightarrow \quad 5p = 2$$
$$\Leftrightarrow \quad p = \frac{2}{5} = 0.4$$

最大概似估計量
(maximum likelihood estimator, MLE)

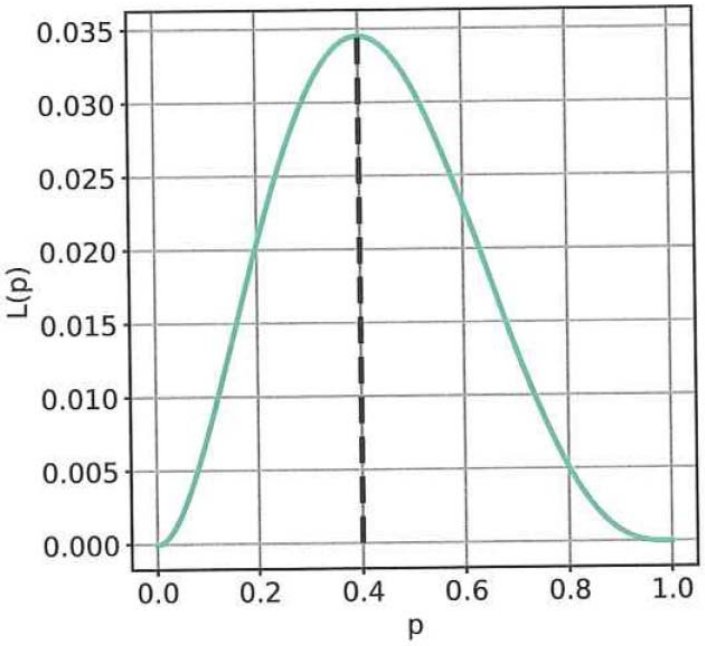


圖 6-13　橫軸為 p，縱軸為概似函數的值

為何概似函數的極值是求最大值，而不是最小值?

- 最大概似估計法是找出「概似函數微分等於 0」的參數值。照理講，找出的參數也有可能讓概似函數出現極小值或無極值。
- 概似函數是由各已知事件的機率(介於0~1)相乘而來，數值只會大於等於0，而等於 0就是極小值，也就是此機率模型最不可能發生的情況。
- 我們希望的是此機率模型最可能發生的情況，因此能產生極大值的參數才是我們要的。

Chapter 7: 線性迴歸模型 (迴歸)

- 本章以線性迴歸模型 (linear regression model)來認識深度學習中的損失函數與梯度下降法。
- 在線性迴歸模型中，訓練的基本原理是以誤差平方和做為損失函數 (loss function) ，並尋找使其函數值最小的參數。
- 用數學方程式求得線性迴歸模型中，損失函數(誤差平方和)的解稱為「**解析解**」。
- 藉由一組粗估初始值開始，經過反覆運算得到的解則稱為「**近似解**」。

損失函數的偏微分與梯度下降法

試求損失函數解析解

$$L(u, v)=3u^2 + 3v^2 - uv + 7u - 7v + 10$$

$$\begin{cases} L_u(u,v) = 6u - v + 7 = 0 \\ L_v(u,v) = -u + 6v - 7 = 0 \end{cases}$$

解聯立方程式

$$(u,\ v) = (-1,\ 1)$$

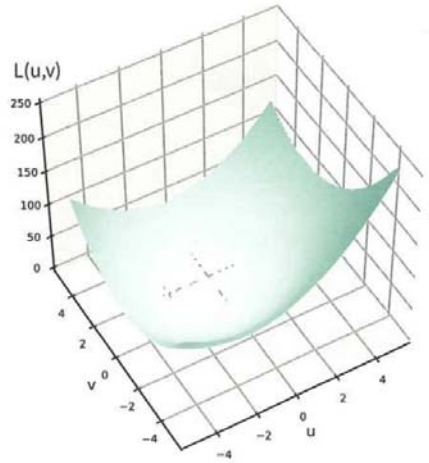


圖 4-3a　雙變數函數在三維座標的曲面

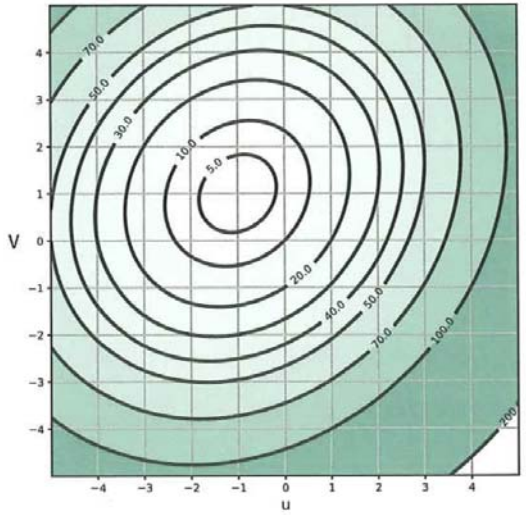
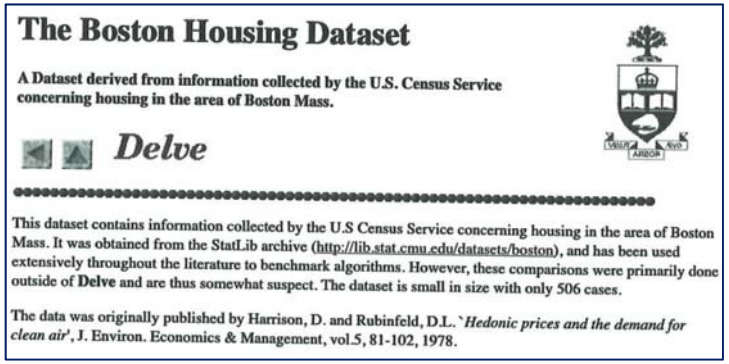


圖 4-3b　用等高線來呈現函數的圖形

美國波士頓房地產資料集

1970年代波士頓郊區的不動產物件相關統計資料:

- **不動產物件相關屬性(attribute):**
  - PRICE: 房產物件價格(平均值)
  - RM: 各物件的房間數(平均值)
  - AGE: 於1940年前建造的房屋比例等等
- **區域特性:**
  - LSTAT: 低所得者比例
  - CRIM: 犯罪率
  - CHAS: 是否位於查爾斯河沿岸(1 : Yes、 0: No) 等等



The Boston Housing Dataset

A Dataset derived from information collected by the U.S. Census Service concerning housing in the area of Boston Mass.

Delve

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston), and has been used extensively throughout the literature to benchmark algorithms. However, these comparisons were primarily done outside of **Delve** and are thus somewhat suspect. The dataset is small in size with only 506 cases.

The data was originally published by Harrison, D. and Rubinfeld, D.L. `*Hedonic prices and the demand for clean air*', J. Environ. Economics & Management, vol.5, 81-102, 1978.

- **目的:** 利用物件價格以外的屬性值，建立可預測物件價格之模型。因為要預測的是連續數值(價格)，即反應變數為「房產物件價格(PRICE)」，因此採用迴歸模型。
  - 簡單線性迴歸模型: 自變數為平均房間數(RM)。
  - 多元線性迴歸模型: 自變數為平均房間數(RM)，低所得者比例 (LSTAT)。

訓練資料與預測值的數學符號標示法

筆數	RM	PRICE
1	6.575	24
2	6.421	21.6
3	7.185	34.7
:	:	:
506	6.03	11.9

表 7-1　本次範例使用的資料

配合 *Python* 語言陣列
的索引起始值為 0

	RM (x)	PRICE (yt)
第 0 筆資料	$x^{(0)} = 6.575$	$yt^{(0)} = 24.0$
第 1 筆資料	$x^{(1)} = 6.421$	$yt^{(1)} = 21.6$
第 2 筆資料	$x^{(2)} = 7.185$	$yt^{(2)} = 34.7$
	:	:
	$x^{(505)} = 6.03$	$yt^{(505)} = 11.9$

表 7-2　將訓練資料改用標示法呈現

預測值標示為
yp (*predict*)

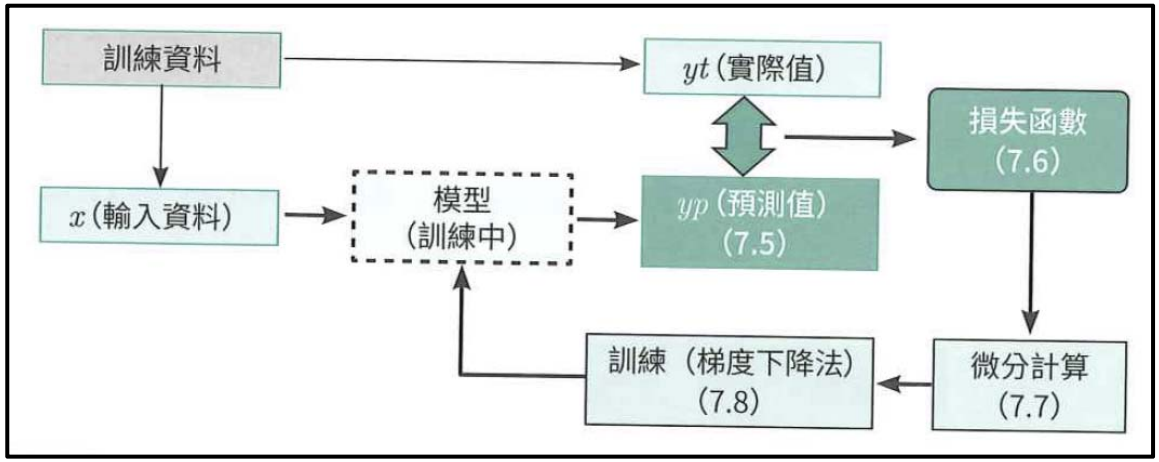


圖 7-2　梯度下降法運作流程架構圖

建立預測模型

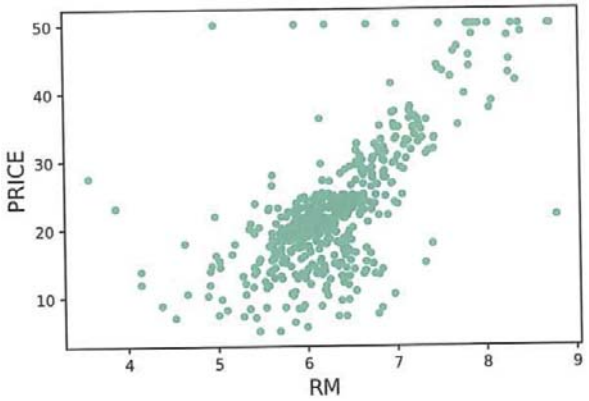
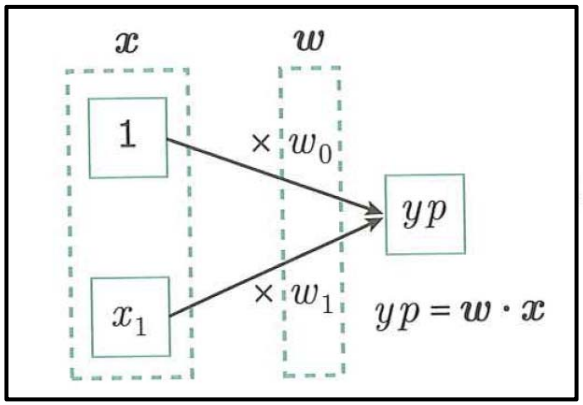


圖 7-3 平均房間數 *vs.* 物件價格的散佈圖



簡單線性迴歸模型的預測方程式關係圖

- **散佈圖**: 資料點雖然散佈很廣，但從大多數集中的點可看出由左下到右上的趨勢是趨近於直線分佈(表示房間數越多的房屋，其價格也趨向越高)。
- 要建立一個簡單線性迴歸模型，也就是要找出最符合散佈圓趨勢的一條直線數學模型。

簡單線性迴歸模型的預測值 yp

$$yp = w_0 + w_1 x \qquad x \text{ 加上下標改寫為 } x_1 \qquad \boldsymbol{x} = (x_0, x_1)$$

$$= w_0 \cdot 1 + w_1 \cdot x \qquad 1 \text{ 用虛擬變數 } x_0 \text{ 來表示} \qquad \boldsymbol{w} = (w_0, w_1)$$

$$yp = \boldsymbol{w} \cdot \boldsymbol{x} \qquad \text{可看作是兩個向量 } (w_0, w_1) \text{ 和 } (1, x) \text{ 的內積}$$

上式是一組資料 $\boldsymbol{x} = (x_0, x_1)$ 的運算式，在機器學習實際計算時，是針對訓練樣本中數量眾多的資料 $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ 各別進行運算並得出預測值。

$$yp^{(m)} = \boldsymbol{w} \cdot \boldsymbol{x}^{(m)}$$

建立損失函數

- 線性迴歸模型的**損失函數L**是使用y 值的預測值(yp)與實際值 (yt)之間差值的平方加總，也就是「誤差平方和」。

$$L = (yp^{(0)} - yt^{(0)})^2 + (yp^{(1)} - yt^{(1)})^2 + \cdots + (yp^{(M-1)} - yt^{(M-1)})^2$$
$$= \sum_{m=0}^{M-1} (yp^{(m)} - yt^{(m)})^2$$

M 代表資料總數共 506 筆

$$L(w_0, w_1) = \frac{1}{2M} \sum_{m=0}^{M-1} (yp^{(m)} - yt^{(m)})^2$$

將誤差平方和除以資料總筆數取平均值作為損失函數之值，如此就不受資料筆數多寡的影響了。

對損失函數做微分，以利後續梯度下降法的運算。因損失函數為二次方程式，微分後會產生一個係數2，故令損失函數再除以2，如此即可與微分後產生的2相抵銷。

損失函數的微分

$$L(w_0, w_1) = \frac{1}{2M} \sum_{m=0}^{M-1} (yp^{(m)} - yt^{(m)})^2$$

$$\Rightarrow \quad \frac{\partial L(w_0, w_1)}{\partial w_1} = \frac{1}{2M} \sum_{m=0}^{M-1} \frac{\partial ((yp^{(m)} - yt^{(m)})^2)}{\partial w_1}$$

$$yp^{(m)} = \boldsymbol{w} \cdot \boldsymbol{x}^{(m)}$$

$$\frac{\partial ((yp - yt)^2)}{\partial w_1} \quad \Rightarrow \quad yd(w_0, w_1) = yp - yt$$

$$= yp\,(w_0, w_1) - yt$$

$$yd^{(m)} = yp^{(m)} - yt^{(m)}$$

$$\Rightarrow \quad \frac{\partial (yd(w_0, w_1))}{\partial w_1} = \frac{\partial (yp(w_0, w_1))}{\partial w_1} = \frac{\partial (w_0 x_0 + w_1 x_1)}{\partial w_1} = x_1$$

$$\Rightarrow \quad \frac{\partial ((yd)^2)}{\partial w_1} = ((yd)^2)' \cdot \frac{\partial (yd)}{\partial w_1} = 2yd \cdot x_1 \quad \Rightarrow \quad \frac{\partial \left((yd^{(m)})^2\right)}{\partial w_1} = 2yd^{(m)} \cdot x_1^{(m)}$$

$$\Rightarrow \quad \frac{\partial L(w_0, w_1)}{\partial w_1} = \frac{1}{M} \sum_{m=0}^{M-1} yd^{(m)} \cdot x_1^{(m)}$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = \frac{1}{M} \sum_{m=0}^{M-1} yd^{(m)} \cdot x_0^{(m)}$$

$$\frac{\partial L(w_0, w_1)}{\partial w_i} = \frac{1}{M} \sum_{m=0}^{M-1} yd^{(m)} \cdot x_i^{(m)} \qquad (i = 0,\ 1)$$

梯度下降法的運用

- 將偏微分結果應用在梯度下降法，以求出能讓損失函數值最小化的(w_0, w_1)參數。

(本書中)變數的標示:

- i: 向量資料中的第 i個元素，放在下標位置。
- m: 全部資料樣本中的第m筆資料，放在上標位置。
- k: 迭代運算的第k次運算，放在上標位置。

- $w_i^{(k)}$: 權重向量$\boldsymbol{w}$的第 i個元素，經過k次迭代運算後的結果。
- $\boldsymbol{w}^{(k)}$: 權重向量$\boldsymbol{w}$經過k次迭代運算後的結果，用粗體表示。

- $x_i^{(m)}$: 資料樣本中第m筆輸入向量$\boldsymbol{x}$的第i個元素(也就是第i個分量)。
- $\boldsymbol{x}^{(m)}$: 資料樣本中第m筆輸入向量$\boldsymbol{x}$, 用粗體表示是一個向量，而不是向量中的元素。

- $yt^{(m)}$: 資料樣本中第 m筆的實際值。
- $yp^{(k)(m)}$: 資料樣本中第 m筆輸入向量，經過k次迭代得到的預測值。
- $yd^{(k)(m)}$: 第m筆資料經過k次迭代後得到的預測值再減掉第m筆的實際值$yt^{(m)}$。

梯度下降法的運用

「以簡單線性迴歸模型做線性預測，並以梯度下降法實踐的數學模型」

$$yp^{(k)(m)} = \boldsymbol{w}^{(k)} \cdot \boldsymbol{x}^{(m)}$$ 迭代第 k 次的權重向量 $\boldsymbol{w} = (w_0, w_1)$ 去算出新的預測值 yp

$$yd^{(k)(m)} = yp^{(k)(m)} - yt^{(m)}$$ 以新的預測值 yp 計算與實際值 yt 的誤差 yd

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix} - \alpha \begin{pmatrix} L_u(u_k, v_k) \\ L_v(u_k, v_k) \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} \Rightarrow \begin{pmatrix} {w_0}^{(k+1)} \\ {w_1}^{(k+1)} \end{pmatrix} \qquad \begin{pmatrix} u_k \\ v_k \end{pmatrix} \Rightarrow \begin{pmatrix} {w_0}^{(k)} \\ {w_1}^{(k)} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} {w_0}^{(k+1)} \\ {w_1}^{(k+1)} \end{pmatrix} = \begin{pmatrix} {w_0}^{(k)} \\ {w_1}^{(k)} \end{pmatrix} - \alpha \begin{pmatrix} \frac{\partial L({w_0}^{(k)}, {w_1}^{(k)})}{\partial w_0} \\ \frac{\partial L({w_0}^{(k)}, {w_1}^{(k)})}{\partial w_1} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} {w_0}^{(k+1)} \\ {w_1}^{(k+1)} \end{pmatrix} = \begin{pmatrix} {w_0}^{(k)} \\ {w_1}^{(k)} \end{pmatrix} - \alpha \begin{pmatrix} \frac{1}{M} \sum_{m=0}^{M-1} yd^{(k)(m)} \cdot {x_0}^{(m)} \\ \frac{1}{M} \sum_{m=0}^{M-1} yd^{(k)(m)} \cdot {x_1}^{(m)} \end{pmatrix}$$

$$\Rightarrow {w_i}^{(k+1)} = {w_i}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} yd^{(k)(m)} \cdot {x_i}^{(m)} \quad (i = 0, 1)$$

$$\Rightarrow \boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} yd^{(k)(m)} \cdot \boldsymbol{x}^{(m)}$$

學習率的大小會影響收斂

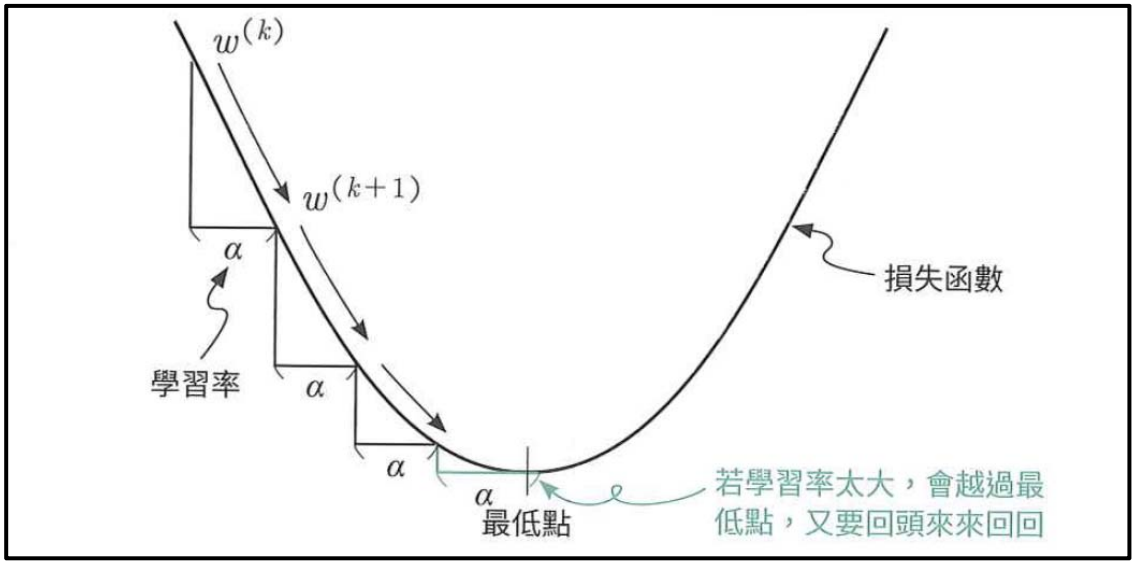


圖 7-5　學習率的大小會影響收斂

學習率與迭代次數的調整方法:

- 學習率的大小，對收斂與否的影響很大。如何設定最好的學習率並沒有明確的答案。
- 如果發現難以收斂，則試著將學習率調為原來的1/10，再視情況微調。
- 實務上要計算的資料量與參數量可能相當龐大，迭代次數越高就表示算得越久，此時權衡得失就很重要(學習率，迭代次數)。
- 在實務的機器學習訓練中，會先對資料做**正規化處理**，避免資料中不同特徵的數值範圍差異太大，而使得梯度下降過程不易收斂。
- 在正規化的情況下，將學習率設在 0.01到 0.001左右，通常都能順利達到收斂。

推廣到多元線性迴歸模型

模型描述

輸入項目：

RM：房間數 (x_1)

LSTAT：低所得者比例 (x_2)

輸出項目：

PRICE：物件價格 (y)

資料樣本內容

$x_1^{(0)} = 6.575$	$x_2^{(0)} = 4.98$	$y^{(0)} = 24.0$
$x_1^{(1)} = 6.421$	$x_2^{(1)} = 9.14$	$y^{(1)} = 21.6$
$x_1^{(2)} = 7.185$	$x_2^{(2)} = 4.03$	$y^{(2)} = 34.7$
$\vdots$	$\vdots$	$\vdots$
$x_1^{(505)} = 6.030$	$x_2^{(505)} = 7.88$	$y^{(505)} = 11.9$

預測式

$$yp = w_0x_0 + w_1x_1 + w_2x_2$$

損失函數

$$L(w_0, w_1, w_2) = \frac{1}{2M}\sum_{m=0}^{M-1}(yp^{(m)} - yt^{(m)})^2$$

$$yp^{(m)} = w_0x_0^{(m)} + w_1x_1^{(m)} + w_2x_2^{(m)}$$

偏微分的計算結果

$$\frac{\partial L(w_0, w_1, w_2)}{\partial w_i} = \frac{1}{M}\sum_{m=0}^{M-1} yd^{(m)} \cdot x_i^{(m)}$$

$(i = 0,\ 1,\ 2)$

$$yd^{(m)} = yp^{(m)} - yt^{(m)} = w_0x_0^{(m)} + w_1x_1^{(m)} + w_2x_2^{(m)} - yt^{(m)}$$

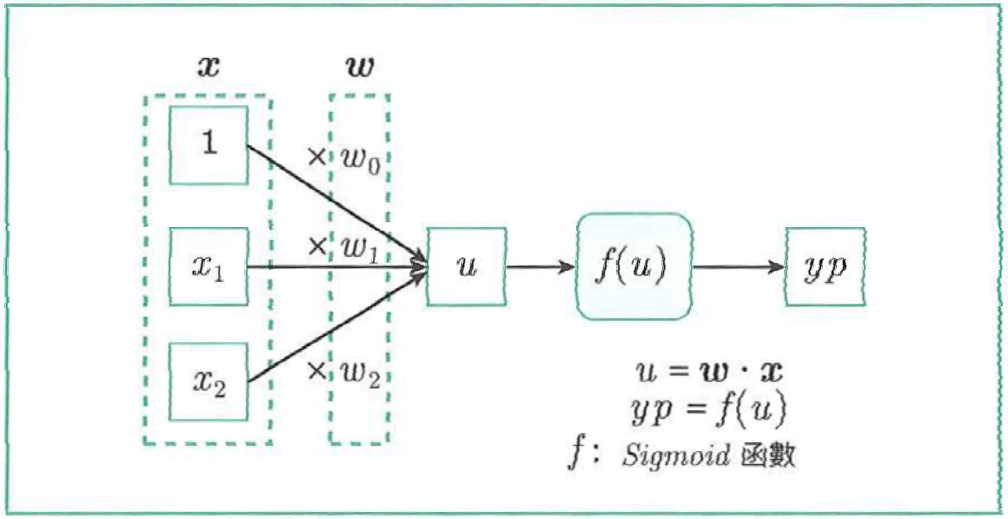
迭代運算的演算法

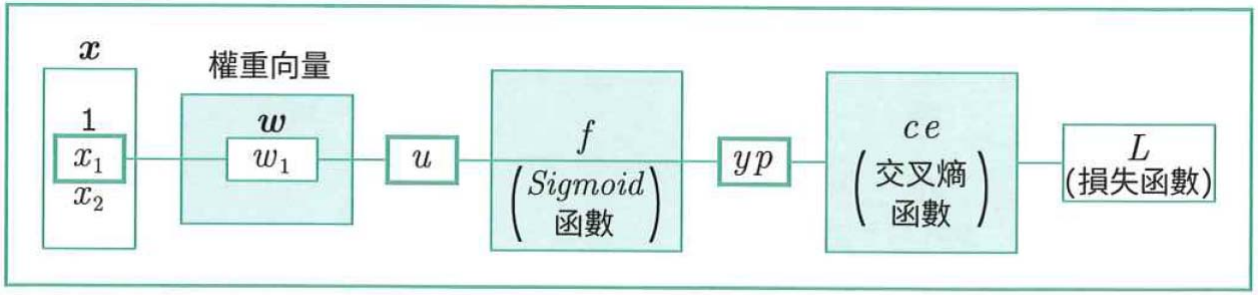
$$yp^{(k)(m)} = \boldsymbol{w}^{(k)} \cdot \boldsymbol{x}^{(m)}$$

$$yd^{(k)(m)} = yp^{(k)(m)} - yt^{(m)}$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \frac{\alpha}{M}\sum_{m=0}^{M-1} yd^{(k)(m)} \cdot \boldsymbol{x}^{(m)}$$

Chapter 8: 邏輯斯迴歸模型 (二元分類)





邏輯斯迴歸模型 (二元分類)範例資料

iris資料集:

- 觀察值(observations): 共150筆資料(150朵鳶尾花) 。 每朵花為3個品種其中之一，Setosa(山鳶尾)、Versicolour(變色鳶尾)、 Virginica(維吉尼亞鳶尾)。每個品種各有50筆資料，共150筆資料。
- 4種特徵(features)(變數)(variables): Sepa1 Length, Sepa1 Width (萼片長度、寬度)及Petal Length， Petal Width (花瓣長度、寬度) 。
- 僅挑選 Sepa1 Length, Sepa1 Width (萼片長度、寬度) (解說方便) 。
- **分類目標類別(2種類別)**: class 0 (Setosa)、 class 1 (Versicolour) 。
- **輸入項目名(2項)**: Sepal Length (cm) 萼片長度、Sepal Length (cm)萼片寬度。

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	3668419

https://archive.ics.uci.edu/ml/datasets/iris

yt（實際值）	x_1（萼片長度）	x_2（萼片寬度）
0	5	3.2
0	5	3.5
1	5	2.3
1	5.5	2.3
1	6.1	3

表 8-1 訓練資料

線性迴歸模型與分類模型的差異

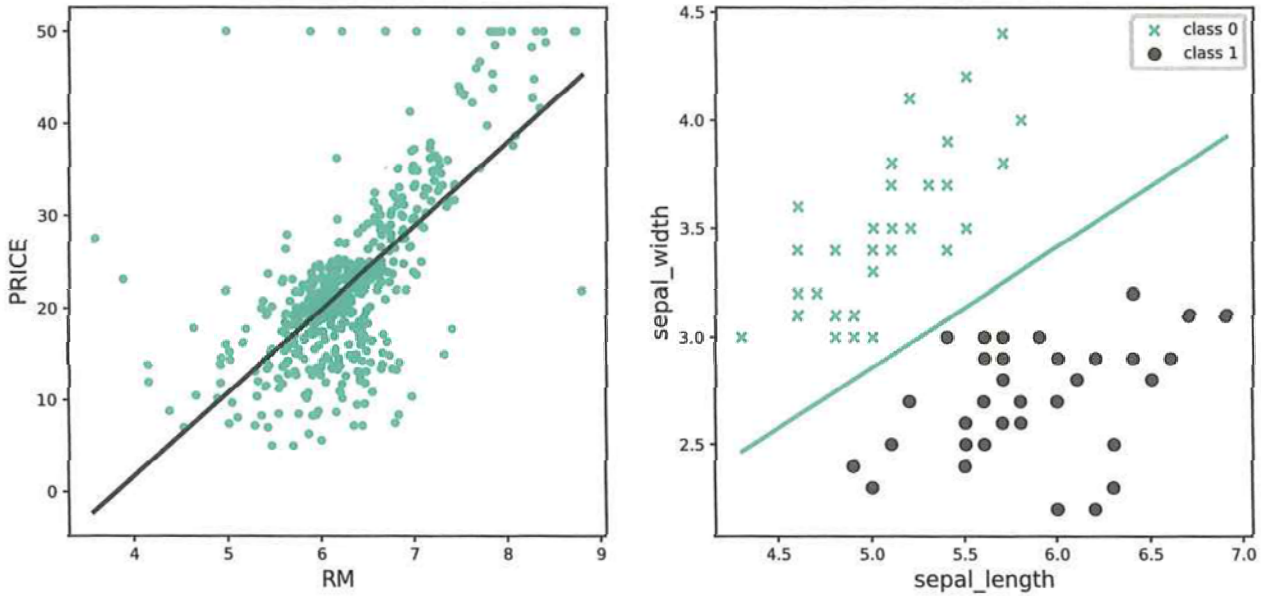


圖 8-3　線性迴歸問題 (左)、分類問題 (右) 的散佈圖

- 線性迴歸的目的是要找出一條距離各資料點誤差最小的直線。
- 分類則是將一群資料有效區分成不同的類別，例如二元分類就是用一條分界線(不一定是直線)將資料分成兩類。
- 分類問題中的分界線稱為**決策邊界(decision boundary)**。
- 迴歸問題與分類問題要考慮的完全不同，因此建立的**預測函數**與**損失函數**自然也不同。

針對預測模型之討論

- 先考慮以一條線性函數來將訓練資料進行分類。 $u = w_0 + w_1 x_1 + w_2 x_2$
- 接下來採用「梯度下降法(Gradient Descent)來做分類。
- 梯度下降法的重點在於建立可微分的損失函數(對參數w微分)，並逐步調整參數值，最後找出能讓損失函數最小化的參數，如此即可產生預測函數進行預測。

$Sigmoid$ 函數 $f(x) = \dfrac{1}{1 + \exp(-x)}$

- 使用Sigmoid函數，將線性函數式的計算結果「轉換」為0到1之間的機率值，並以此作為輸出的預測值。
- 在二元分類中，可設定只要預測值比較接近0就可預測是某個分類，比較接近 1則是另一個分類。

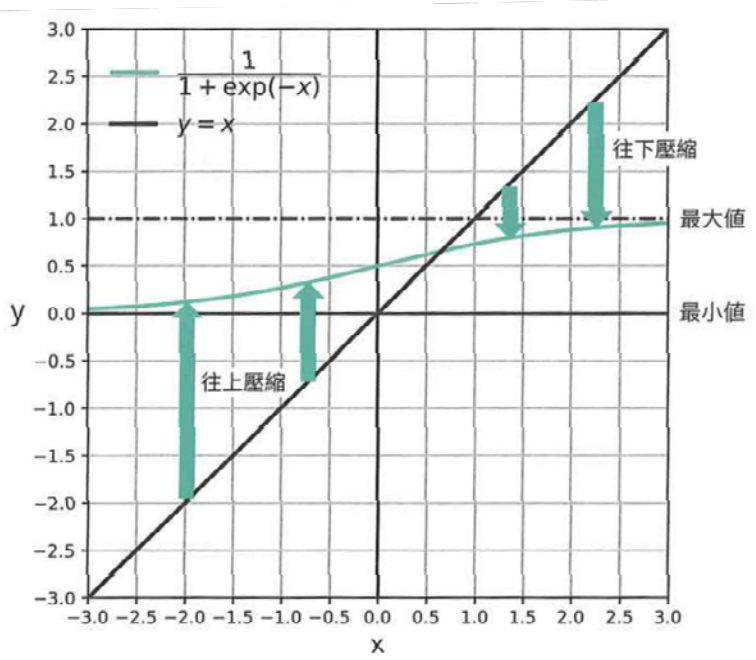


圖 8-4 藍色 S 形曲線就是 $Sigmoid$ 函數圖

決策邊界

決策邊界

$$w_0 + w_1 x_1 + w_2 x_2 = u$$

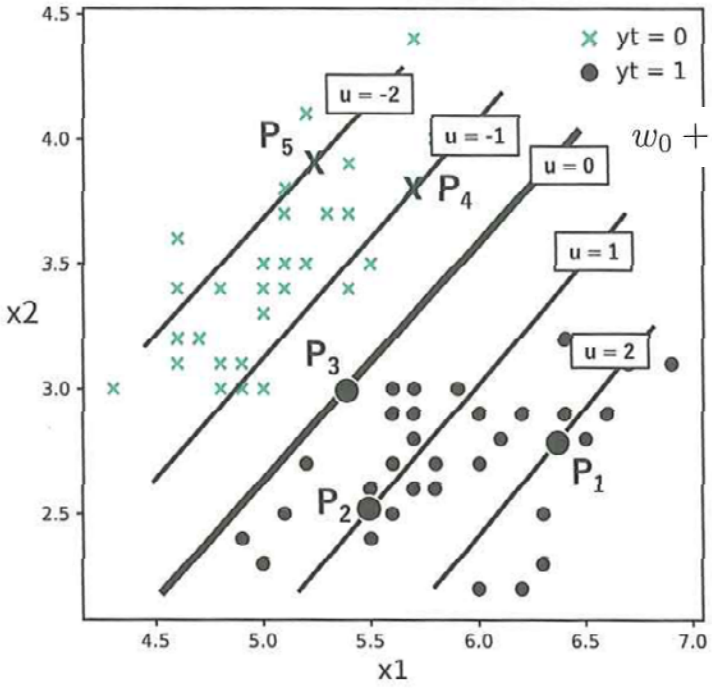


圖 8-5 資料散佈圖與決策邊界

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

P_m	yt（實際值）	u	f(u)
P_1	1	2	0.88
P_2	1	1	0.73
P_3	1	0	0.5
P_4	0	−1	0.27
P_5	0	−2	0.12

表 8-2　散佈圖中 5 個點的 u 值及 $f(u)$ 值

預測模型步驟

(1) 由輸入資料 (x_1, x_2) 計算 $u = w_0 + w_1x_1 + w_2x_2$ 之值。 ➡ $w_0 \cdot 1 + w_1x_1 + w_2x_2$ ➡ $\boldsymbol{x} = (x_0, x_1, x_2)$, $\boldsymbol{w} = (w_0, w_1, w_2)$

(2) 由 (1) 得到的 u 值計算 $f(u)$ 值。此處的 $f(u)$ 即 *Sigmoid* 函數：

$$f(u) = \frac{1}{1 + \exp(-u)}$$

➡

$$u = \boldsymbol{w} \cdot \boldsymbol{x}$$

$$yp = f(u)$$

(3) 經此計算得到的 **$f(u)$ 值，代表「該點屬於 $class = 1$ 的機率」**。

(4) **將此 $f(u)$ 值視為 y 的預測值 yp**。

$$f(u) = \frac{1}{1 + \exp(-u)}$$

(5) **使用預測值分類時，以預測值是否大於 0.5 進行判斷**。

(6) 將 yp 視為 $\boldsymbol{w}(w_0, w_1, w_2)$ 的函數時，
其值會隨著 $\boldsymbol{w}$ 的變化而產生連續變化。

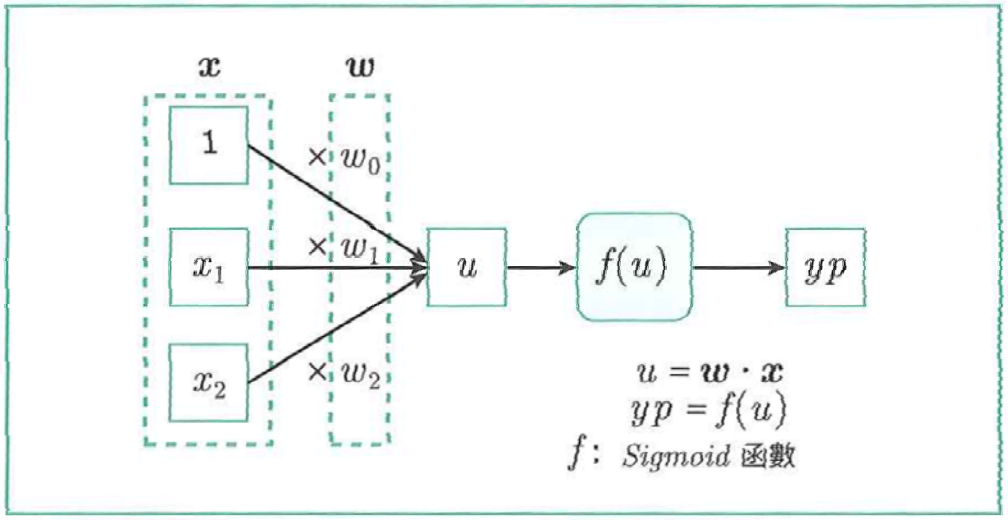


圖 8.6 二元邏輯斯迴歸之預測模型

損失函數(交叉熵 Cross entropy)

將模型在預測階段的行為表示如下： $u(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$

$$f(u) = \frac{1}{1 + \exp(-u)}$$

得到預測值 yp： $yp = f(u) = f(w_0 + w_1 x_1 + w_2 x_2)$

若 $yt = 1$ 的機率為 yp，則 $yt = 0$ 的機率就是 $1 - yp$。

因此機率可寫成下式：

$$P(yt, yp) = \begin{cases} yp & (yt = 1 \text{ 的時候}) \\ 1 - yp & (yt = 0 \text{ 的時候}) \end{cases}$$

要用 $P(yt,\ yp)$ 找出概似函數來定義損失函數

找出概似函數

限定輸入資料為 5 組 為方便說明

		輸入值	預測值	機率
m	$yt^{(m)}$ (實際值)	$u^{(m)}$	$yp^{(m)}$	$P^{(m)}$
1	1	$x^{(1)} \cdot w$	$f(u^{(1)})$	$yp^{(1)}$
2	0	$x^{(2)} \cdot w$	$f(u^{(2)})$	$1-yp^{(2)}$
3	0	$x^{(3)} \cdot w$	$f(u^{(3)})$	$1-yp^{(3)}$
4	1	$x^{(4)} \cdot w$	$f(u^{(4)})$	$yp^{(4)}$
5	0	$x^{(5)} \cdot w$	$f(u^{(5)})$	$1-yp^{(5)}$

表 8-3　5 組樣本資料與 P 值 (標示上標 m 則為 $P^{(m)}$)

$$u^{(m)} = x^{(m)} \cdot w$$

$$yp^{(m)} = f(u^{(m)})$$

概似函數　$Lk = P^{(1)} \cdot P^{(2)} \cdot P^{(3)} \cdot P^{(4)} \cdot P^{(5)}$

$$\log(Lk) = \log(P^{(1)} \cdot P^{(2)} \cdot P^{(3)} \cdot P^{(4)} \cdot P^{(5)})$$

$$= \log(P^{(1)}) + \log(P^{(2)}) + \log(P^{(3)}) + \log(P^{(4)}) + \log(P^{(5)})$$

$$\log(P^{(m)}) = yt^{(m)} \log(yp^{(m)}) + (1 - yt^{(m)}) \log(1 - yp^{(m)})$$

$yt^{(m)}$ 之值只會是 0 或 1

找出損失函數

$$\log(Lk) = \sum_{m=1}^{5} \log(P^{(m)})$$

$$= \sum_{m=1}^{5} (yt^{(m)} \log(yp^{(m)}) + (1 - yt^{(m)}) \log(1 - yp^{(m)}))$$

(1) 上式是為了講解方便才只用 5 筆資料，一般資料筆數會以 M 筆表示。

(2) 目前為訓練階段，權重參數 (w_0, w_1, w_2) 才是我們要求的值。

(3) 由於概似函數以求得最大值為目標，但梯度下降法的損失函數以求得最小值為目標，因此可將概似函數乘以 -1 做為損失函數。

(4) 上式為各樣本點代入 (8.4.2) 式計算後之和。但如上一章所述，損失函數值會隨資料的筆數成正比增加，使損失函數的準確率難以比較，因此要取平均值，使其不受資料件數影響。

(5) 由於 *Python* 的陣列索引是從 0 開始，因此需要令 m 的初始值為 0。

$$L(w_0, w_1, w_2) = -\frac{1}{M} \sum_{m=0}^{M-1} (yt^{(m)} \cdot \log(yp^{(m)}) + (1 - yt^{(m)}) \log(1 - yp^{(m)}))$$

交叉熵函數 (*cross entropy function*)

$$u^{(m)} = \boldsymbol{w} \cdot \boldsymbol{x}^{(m)} = w_0 + w_1 x_1^{(m)} + w_2 x_2^{(m)}$$

$$yp^{(m)} = f(u^{(m)})$$

$$f(u^{(m)}) = \frac{1}{1 + \exp(-u^{(m)})}$$

交叉熵函數(損失函數)的微分

$$L(w_0, w_1, w_2) = -\frac{1}{M}\sum_{m=0}^{M-1}(yt^{(m)} \cdot \log(yp^{(m)}) + (1 - yt^{(m)})\log(1 - yp^{(m)}))$$

為使式子易於了解，先將上標取下：$yt^{(m)} \Rightarrow yt$、$yp^{(m)} \Rightarrow yp$

再以 ce 表示 特定項目的交叉熵函數：　$ce = -(yt\log(yp) - (1 - yt)\log(1 - yp))$

yt 是常數

yp 為變數

$$\frac{d(ce)}{d(yp)} = -\frac{yt}{yp} - \frac{(1 - yt)(-1)}{1 - yp}$$

$$= \frac{-yt(1 - yp) + yp(1 - yt)}{yp(1 - yp)}$$

$$= \frac{yp - yt}{yp(1 - yp)}$$

利用最大概似估計法來找出能讓損失函數最小化的權重參數值。

訓練階段的 $\boldsymbol{x}$ 及 y 都是已知的實際值 (也就是常數)

只有權重向量 $\boldsymbol{w}$ 是需要做偏微分的變數即可。

損失函數的微分

$$u(w_0, w_1, w_2) = w_0 + w_1 x_1 + w_2 x_2$$

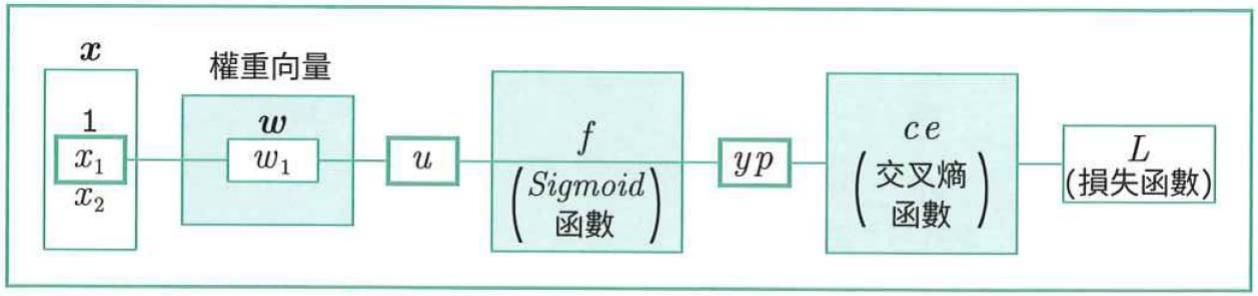


圖 8-7 輸入資料 $\boldsymbol{x}$ 與損失函數的關係

$$\frac{\partial L}{\partial w_1} = \frac{dL}{du} \cdot \frac{\partial u}{\partial w_1}$$

$$\frac{dL}{du} = \frac{dL}{d(yp)} \cdot \frac{d(yp)}{du}$$

$$\frac{\partial u}{\partial w_1} = x_1$$

很重要，背起來！

$$\frac{dy}{dx} = f'(x) = y(1-y)$$

$$\frac{dL}{d(yp)} = \frac{d(ce)}{d(yp)} = \frac{yp - yt}{yp(1-yp)}$$

$$\frac{d(yp)}{du} = yp(1-yp)$$

$$\frac{dL}{du} = \frac{yp - yt}{yp(1-yp)} \cdot yp(1-yp) = yp - yt$$

代表「誤差」

定義「誤差」為 yd

$$yd = yp - yt$$

$$\frac{dL}{du} = yd$$

$$\frac{\partial L}{\partial w_1} = x_1 \cdot yd$$

$$\frac{\partial L}{\partial w_1} = \frac{1}{M} \sum_{m=0}^{M-1} x_1^{(m)} \cdot yd^{(m)}$$

$$\frac{\partial L}{\partial w_i} = \frac{1}{M} \sum_{m=0}^{M-1} x_i^{(m)} \cdot yd^{(m)}$$

$$i = 0,\ 1,\ 2$$

梯度下降法的運用

【上下標】

k：迭代運算次數的 *index*

m：資料樣本的 *index*

i：向量分量的 *index*

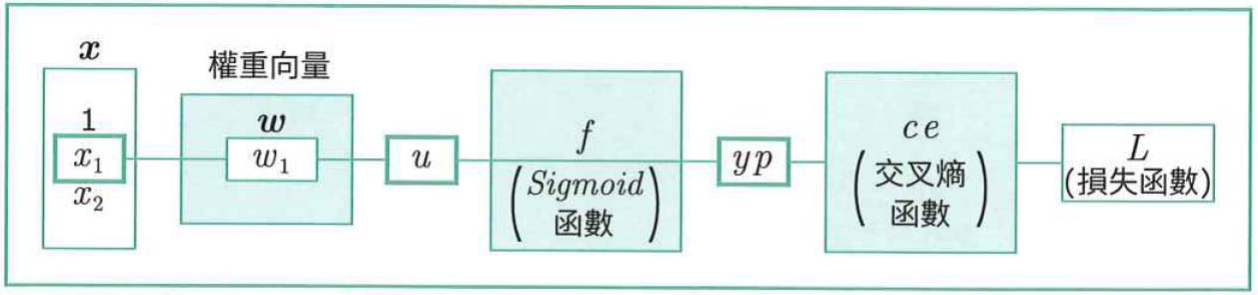


圖 8-7 輸入資料 $\boldsymbol{x}$ 與損失函數的關係

【變數】

M：資料樣本的總數

α：學習率

$$u^{(k)(m)} = \boldsymbol{w}^{(k)} \cdot \boldsymbol{x}^{(m)} \qquad yp^{(k)(m)} = f(u^{(k)(m)}) \qquad f(u) = \frac{1}{1 + \exp(-u)}$$

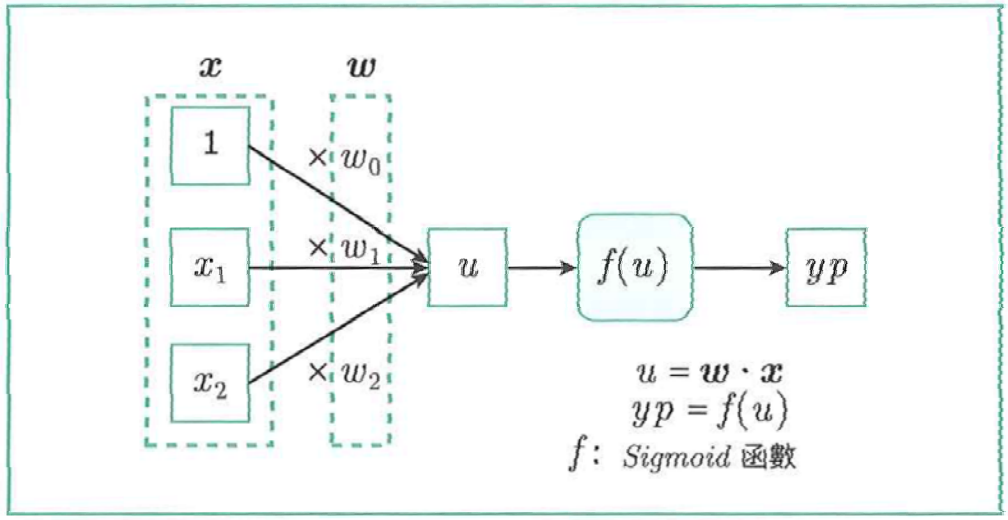


圖 8.6 二元邏輯斯迴歸之預測模型

$$yd^{(k)(m)} = yp^{(k)(m)} - yt^{(m)}$$

$$w_i^{(k+1)} = w_i^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} x_i^{(m)} \cdot yd^{(k)(m)}$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} \boldsymbol{x}^{(m)} \cdot yd^{(k)(m)}$$

$$i = 0,\ 1,\ 2$$

程式實作

Chapter 9: 邏輯斯迴歸模型 (多類別分類)

重點 實現深度學習所需概念	第 1 章 迴歸 1	第 7 章 迴歸 2	第 8 章 二元分類	第 9 章 多類別分類	第 10 章 深度學習
1 損失函數	○	○	○	○	○
3.7 矩陣運算				○	○
4.5 梯度下降法		○	○	○	○
5.5 Sigmoid 函數			○		○
5.6 Softmax 函數				○	○
6.3 概似函數與最大概似估計法			○	○	○
10 反向傳播					○

- 多類別分類的演算法基本流程:
  「建立預測函數」→「建立損失函數」→「以梯度下降法尋找最佳參數」。

- 多類別分類的做法:「建立多個可輸出0到1的分類器，並將數值最高的分類器所對應到的類別，視為整個模型的預測值」。

- 與二元分類不同處:

權重向量 → **權重矩陣** ← 由一維的向量變成矩陣

***Sigmoid* 函數** → ***Softmax* 函數** ← 換成可做多類別分類的 *Softmax* 函數

範例問題設定

iris資料集:

- 觀察值(observations): 共150筆資料(150朵鳶尾花) 。 每朵花為3個品種其中之一，Setosa(山鳶尾)、Versicolour(變色鳶尾)、 Virginica(維吉尼亞鳶尾)。每個品種各有50筆資料，共150筆資料。

- 4種特徵(features)(變數)(variables): Sepa1 Length, Sepa1 Width (萼片長度、寬度)及Petal Length， Petal Width (花瓣長度、寬度)。僅挑選 Sepa1 Length (萼片長度)及Petal Length (花瓣長度)(解說方便)

- **分類目標類別(3種類別)** : class 0 (Setosa)、 class 1 (Versicolour)、class 2 (Virginica)。

- **輸入項目名(2項)**: Sepal Length (cm) 萼片長度、Petal Length (cm) 花瓣長度。

yt(實際值)	x_1(萼片長度)	x_2(花瓣長度)
1	6.3	4.7
1	7	4.7
0	5	1.6
2	6.4	5.6
2	6.3	5
0	5	1.6
0	4.9	1.4
1	6.1	4
1	6.5	4.6

表 9-1　訓練資料內容

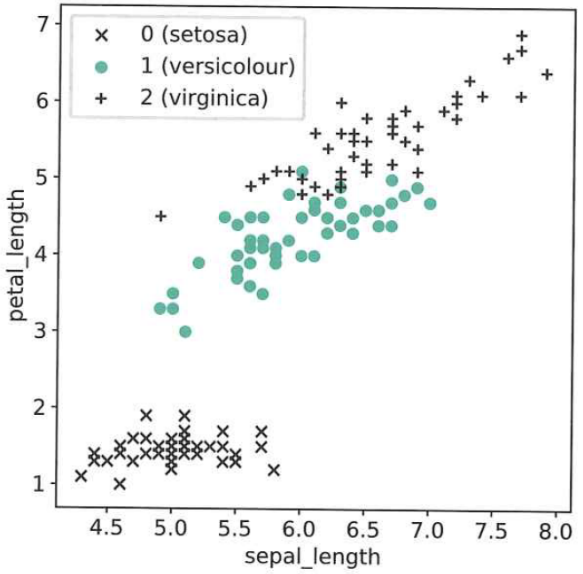


圖 9-2　3 種鳶尾花品種的資料散佈圖

建立模型的基本概念: 將實際值轉換成One-hot編碼

- 為了建立能夠同時輸出3種類別的分類器，我們將實際值yt由 0、 1、 2轉換成由 0與 1組合成的三維向量(1, 0, 0)、(0, 1, 0)、(0, 0, 1)表示法。這種表示法稱為「**one-hot向量(one-hot vector)**」。
- 在n維向量中只有1個分量是 1，其餘$n-1$個分量都是0，這種向量稱為**n維的one-hot向量**。
- 將類別資料轉換成one-hot向量的編碼方式，就叫做 **one-hot編碼(encoding)**。

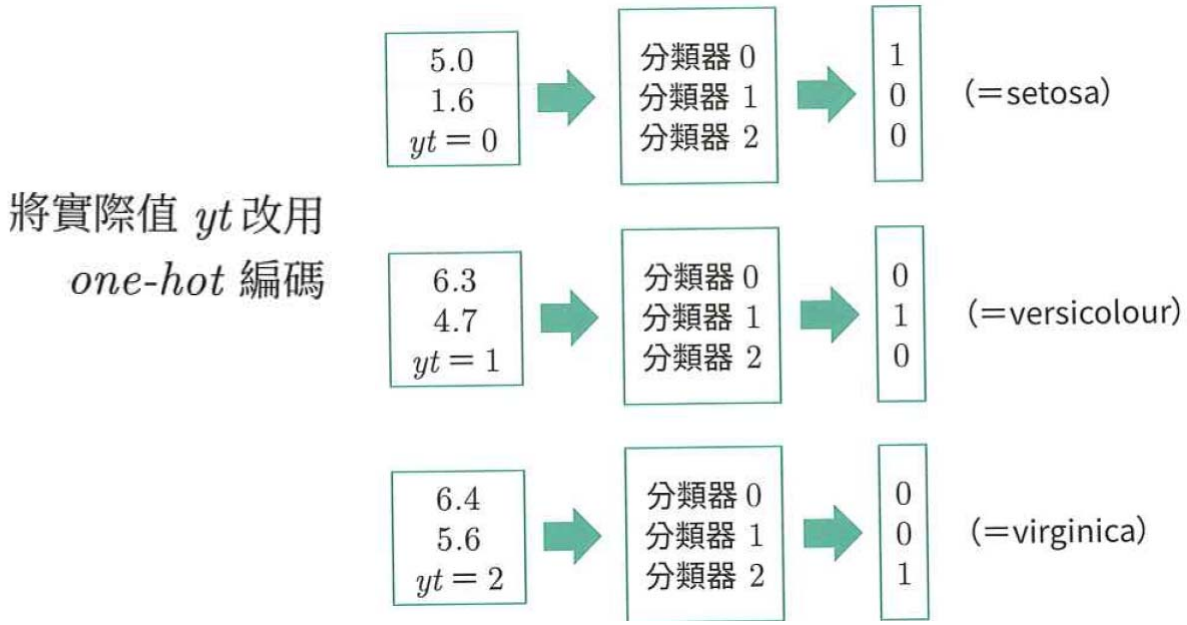


圖 9-3　*one-hot* 編碼做分類的概念圖

這種分類方式稱為「1對其餘分類器 (One vs Rest classifier)」。

權重矩陣

- 多類別分類模型的內部有N個(視需要數量而定)分類器同時運作，就會有N組權重向量，組合成「**權重矩陣**」來做運算。

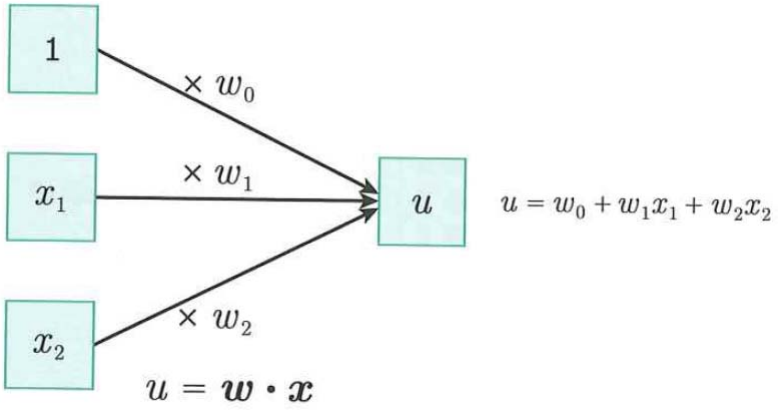
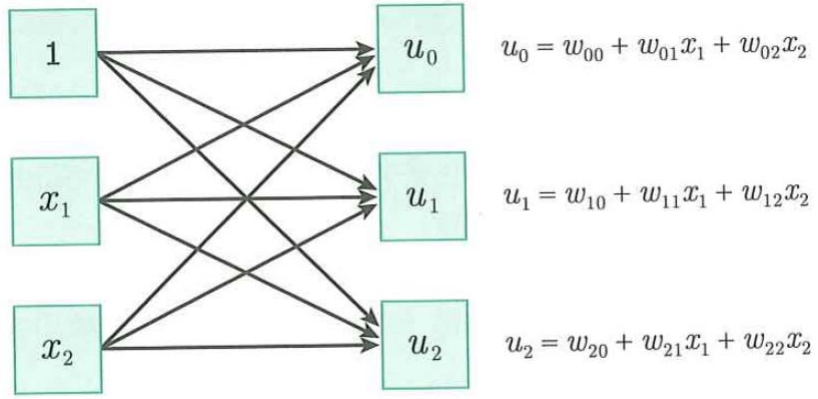


圖 9-4　二元分類與權重向量



$$\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x}$$

$$\begin{cases} u_0 = w_{00} + w_{01}x_1 + w_{02}x_2 \\ u_1 = w_{10} + w_{11}x_1 + w_{12}x_2 \\ u_2 = w_{20} + w_{21}x_1 + w_{22}x_2 \end{cases}$$

$$\boldsymbol{x} = ((x_0 = 1), x_1, x_2)$$

$$\boldsymbol{w} = (w_0, w_1, w_2)$$

向量內積

$$u = \boldsymbol{w} \cdot \boldsymbol{x}$$

$$\boldsymbol{W} = \begin{pmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix}$$

$$\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x}$$

Softmax 函數

- 二元分類: 將 $\boldsymbol{\omega}\cdot\boldsymbol{x}$的結果代入Sigmoid函數，以得出一個0~1的機率值做為預測之用。
- 多類別分類: 使用Softmax 函數。

- Softmax特性(適合做為「同時輸出多個機率值」的函數):
  - 輸入: N維向量。
  - 輸出: N維的向量值函數(vector-valued function)。
  - 各個輸出分量的值介於0~1。(皆為正值)
  - 所有輸出分量的值加總為1。

$$\begin{cases} y_0 = \dfrac{\exp(u_0)}{g(u_0,u_1,u_2)} \\ y_1 = \dfrac{\exp(u_1)}{g(u_0,u_1,u_2)} \\ y_2 = \dfrac{\exp(u_2)}{g(u_0,u_1,u_2)} \end{cases}$$

$$g(u_0,u_1,u_2) = \exp(u_0) + \exp(u_1) + \exp(u_2)$$

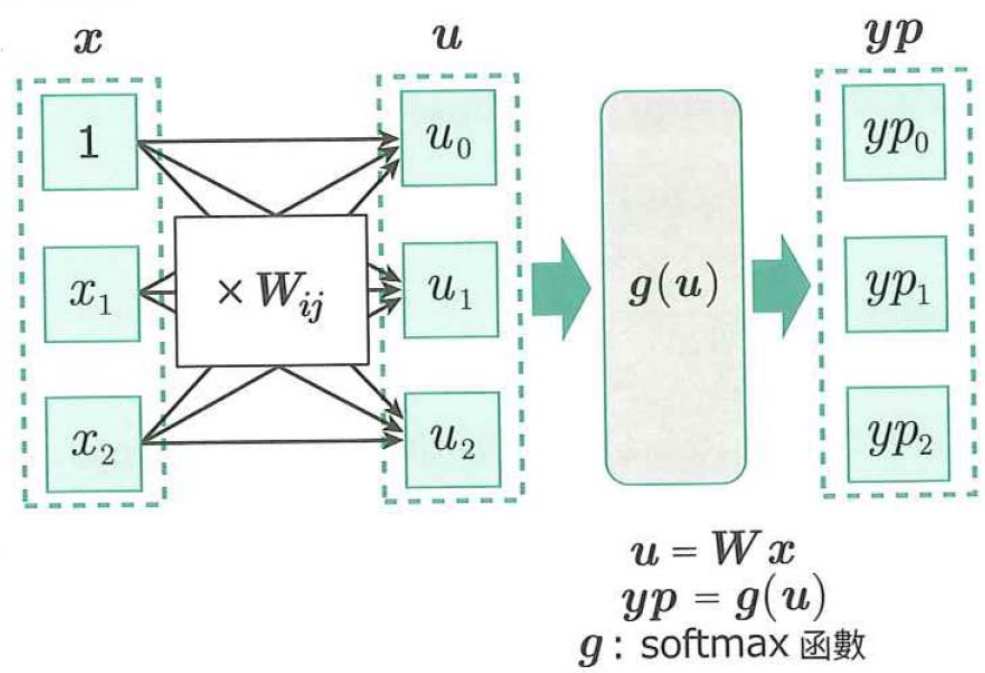


圖 9-6　多類別分類的模型架構圖

損失函數

$$\boldsymbol{yt} = (yt_0,\ yt_1,\ yt_2)$$
$$\boldsymbol{yp} = (yp_0,\ yp_1,\ yp_2)$$

實際值與對應預測值的對數概似函數: $\sum_{i=0}^{2}(yt_i \log(yp_i))$

損失函數即為對數概似函數乘以-1。因為有 M筆輸入資料，取平均值，最後得到損失函數如下:

$$L(\boldsymbol{W}) = -\frac{1}{M}\sum_{m=0}^{M-1}\sum_{i=0}^{2}(yt_i^{(m)} \log(yp_i^{(m)}))$$

推導多類別分類的對數概似函數

要得到損失函數，一般會先找出概似函數，再取對數概似函數，接著取平均值，再乘上−1之後得出損失函數。

m	yt (yt_0, yt_1, yt_2)	$P^{(m)}$
0	1 (0, 1, 0)	yp_1
1	1 (0, 1, 0)	yp_1
2	0 (1, 0, 0)	yp_0
3	2 (0, 0, 1)	yp_2
4	2 (0, 0, 1)	yp_2

表 9-2　多類別分類的 yt 與 P 的關係

$$P(\boldsymbol{yt},\ \boldsymbol{yp}) = \begin{cases} yp_0\text{，當 } yt = (1,0,0)\text{ 時} \\ yp_1\text{，當 } yt = (0,1,0)\text{ 時} \\ yp_2\text{，當 } yt = (0,0,1)\text{ 時} \end{cases}$$

$$P = {yp_0}^{yt_0} \cdot {yp_1}^{yt_1} \cdot {yp_2}^{yt_2}$$

當 $m = 0$ 時，將 $yt = (0, 1, 0)$ 代入上式，可得：

$$P^{(0)} = {yp_0}^{0} \cdot {yp_1}^{1} \cdot {yp_2}^{0} = yp_1$$

概似函數：　$Lk = P^{(0)} \cdot P^{(1)} \cdot \ldots \cdot P^{(M-1)}$

$$\log(P) = \sum_{i=0}^{2} \log({yp_i}^{yt_i}) = \sum_{i=0}^{2} yt_i \log(yp_i) \Rightarrow \log(P^{(m)}) = \sum_{i=0}^{2} yt_i \log({yp_i}^{(m)})$$

$$\log(Lk) = \sum_{m=0}^{M-1} \log(P^{(m)}) = \sum_{m=0}^{M-1} \sum_{i=0}^{2} yt_i \log({yp_i}^{(m)}) \Rightarrow L(\boldsymbol{W}) = -\frac{1}{M} \sum_{m=0}^{M-1} \sum_{i=0}^{2} (yt_i^{(m)} \log(yp_i^{(m)}))$$

損失函數的微分

- 損失函數決定之後，接下來要對損失函數偏微分以計算梯度。
- 先只考慮單1筆資料的yt與 yp:

$$\boldsymbol{yt}^{(m)} \longrightarrow \boldsymbol{yt} = (yt_0,\ yt_1,\ yt_2)$$
$$\boldsymbol{yp}^{(m)} \longrightarrow \boldsymbol{yp} = (yp_0,\ yp_1,\ yp_2)$$

用 ce 代表上述 1 筆訓練資料的交叉熵

$$ce(yp_0, yp_1, yp_2) = -\sum_{i=0}^{2}(yt_i \log(yp_i))$$
$$= -(yt_0 \log(yp_0) + yt_1 \log(yp_1) + yt_2 \log(yp_2))$$

$$\Rightarrow\ L(\boldsymbol{W}) = -\frac{1}{M}\sum_{m=0}^{M-1}\sum_{i=0}^{2}(yt_i^{(m)} \log(yp_i^{(m)}))$$

$$\Rightarrow\ \frac{\partial L}{\partial w_{ij}} = \frac{1}{M}\sum_{m=0}^{M-1} x_j^{(m)} \cdot yd_i^{(m)}$$

推導如後頁

損失函數L對權重矩陣的各參數w_{ij}偏微分

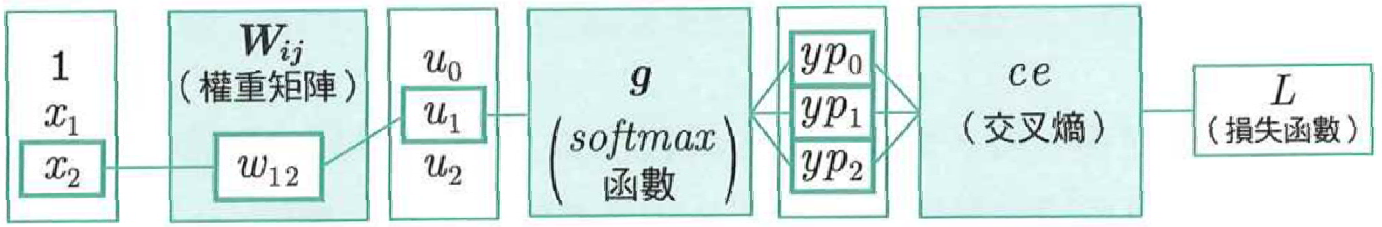


圖 9-7　權重矩陣、*Softmax* 函數與損失函數的關係

- w_{12} 的變化會對 u_1 產生影響 (但與 u_0 及 u_2 無關)
- u_1 的變化會對 yp_0, yp_1, yp_2 分別產生影響
- yp_0, yp_1, yp_2 的變化都會對 L 的值產生影響

計算 $\dfrac{\partial u_1}{\partial w_{12}}$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial w_{12}} \qquad u_1 = w_{10} + w_{11}x_1 + w_{12}x_2$$

$$\frac{\partial L}{\partial w_{12}} = x_2\frac{\partial L}{\partial u_1} \qquad \frac{\partial u_1}{\partial w_{12}} = x_2$$

損失函數L對權重矩陣的各參數w_{ij}偏微分

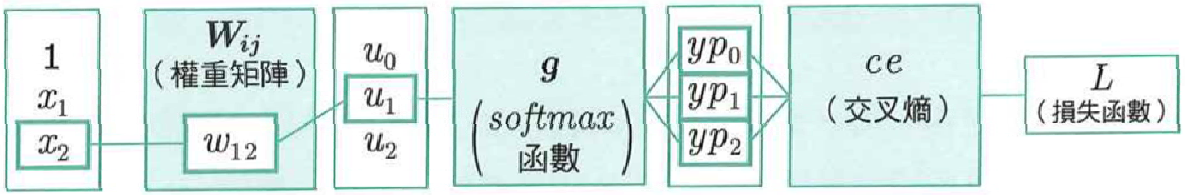


圖 9-7　權重矩陣、*Softmax* 函數與損失函數的關係

計算 $\frac{\partial L}{\partial u_1}$

$$L(yp_0, yp_1, yp_2)$$
$$= ce(yp_0, yp_1, yp_2)$$
$$= -(yt_0 \log(yp_0) + yt_1 \log(yp_1) + yt_2 \log(yp_2))$$

$$\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial yp_0}\frac{\partial yp_0}{\partial u_1} + \frac{\partial L}{\partial yp_1}\frac{\partial yp_1}{\partial u_1} + \frac{\partial L}{\partial yp_2}\frac{\partial yp_2}{\partial u_1}$$

計算 $\frac{\partial L}{\partial yp_i}$　$\frac{\partial L}{\partial yp_i}$ 為 **交叉熵函數偏微分**

$$\frac{\partial L}{\partial yp_0} = -\frac{\partial ce}{\partial yp_0} = -\frac{yt_0}{yp_0}$$
$$\frac{\partial L}{\partial yp_1} = -\frac{\partial ce}{\partial yp_1} = -\frac{yt_1}{yp_1}$$
$$\frac{\partial L}{\partial yp_2} = -\frac{\partial ce}{\partial yp_2} = -\frac{yt_2}{yp_2}$$

$\frac{\partial yp_i}{\partial u_1}$ 為 ***Softmax*** **函數的偏微分**

計算 $\frac{\partial yp_i}{\partial u_1}$

$$\frac{\partial y_j}{\partial x_i} = \begin{cases} y_i(1-y_i) & (i=j) \\ -y_i y_j & (i \neq j) \end{cases}$$

也很重要！

$$\frac{\partial yp_0}{\partial u_1} = -yp_1 \cdot yp_0$$
$$\frac{\partial yp_1}{\partial u_1} = yp_1(1-yp_1)$$
$$\frac{\partial yp_2}{\partial u_1} = -yp_1 \cdot yp_2$$

損失函數L對權重矩陣的各參數w_{ij}偏微分

$$\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial yp_0}\frac{\partial yp_0}{\partial u_1} + \frac{\partial L}{\partial yp_1}\frac{\partial yp_1}{\partial u_1} + \frac{\partial L}{\partial yp_2}\frac{\partial yp_2}{\partial u_1}$$

$$= -\frac{yt_0}{yp_0}\cdot(-yp_1\cdot yp_0) - \frac{yt_1}{yp_1}\cdot yp_1(1-yp_1) - \frac{yt_2}{yp_2}\cdot(-yp_1\cdot yp_2)$$

$$= yt_0\cdot yp_1 - yt_1(1-yp_1) + yt_2\cdot yp_2$$

$$= -yt_1 + yp_1(yt_0 + yt_1 + yt_2)$$

$$= yp_1 - yt_1 \qquad yt_0 + yt_1 + yt_2 = 1$$

$$\frac{\partial L}{\partial u_i} = yp_i - yt_i \quad (i = 0,\ 1,\ 2)$$

定義誤差向量 $\boldsymbol{yd}$：

$$\boldsymbol{yd} = \boldsymbol{yp} - \boldsymbol{yt}$$

$$\frac{\partial L}{\partial u_i} = yd_i \quad (i = 0,\ 1,\ 2)$$

$$\frac{\partial L}{\partial w_{12}} = x_2\frac{\partial L}{\partial u_1} = x_2\cdot yd_1$$

一般化之後　只考慮 1 個訓練樣本的情況

$$\frac{\partial L}{\partial w_{ij}} = x_j\cdot yd_i$$

M 筆訓練資料　完整的算式

$$\frac{\partial L}{\partial w_{ij}} = \frac{1}{M}\sum_{m=0}^{M-1} x_j^{(m)}\cdot yd_i^{(m)}$$

梯度下降法的運用

【上下標】

k：迭代運算次數 *index*

m：資料樣本 *index*

i、j：向量與矩陣的下標

【常數】

M：資料樣本總數 (150 筆)

N：分類的類別數 (3 類)

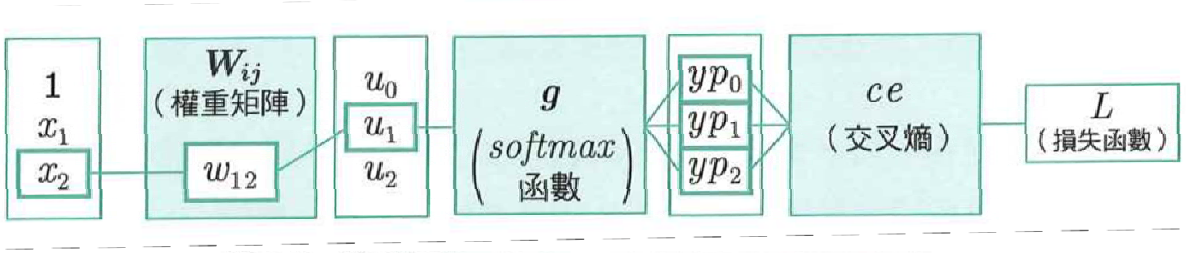


圖 9-7　權重矩陣、*Softmax* 函數與損失函數的關係

權重矩陣與輸入資料做內積得到 $u^{(k)(m)}$　　$\boldsymbol{u}^{(k)(m)} = \boldsymbol{W}^{(k)} \cdot \boldsymbol{x}^{(m)}$

將 $u^{(k)(m)}$送入 *Softmax* 函數 h 算出預測值向量 $yp^{(k)(m)}$　　$\boldsymbol{yp}^{(k)(m)} = \boldsymbol{h}(\boldsymbol{u}^{(k)(m)})$

Softmax 函數定義　　$$h_i = \frac{\exp(u_i)}{\sum_{j=0}^{N-1} \exp(u_j)}$$

由預測值向量與實際值向量計算誤差向量　　$\boldsymbol{yd}^{(k)(m)} = \boldsymbol{yp}^{(k)(m)} - \boldsymbol{yt}^{(m)}$

利用誤差算出下降的梯度，更新權重矩陣的值　　$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} yd_i^{(k)(m)} \cdot x_j^{(m)}$$

Chapter 10: 深度學習

重點 實現深度學習所需概念	第 1 章 迴歸 1	第 7 章 迴歸 2	第 8 章 二元分類	第 9 章 多類別分類	第 10 章 深度學習
1 損失函數	○	○	○	○	○
3.7 矩陣運算				○	○
4.5 梯度下降法		○	○	○	○
5.5 Sigmoid 函數			○		○
5.6 Softmax 函數				○	○
6.3 概似函數與最大概似估計法			○	○	○
10 反向傳播					○

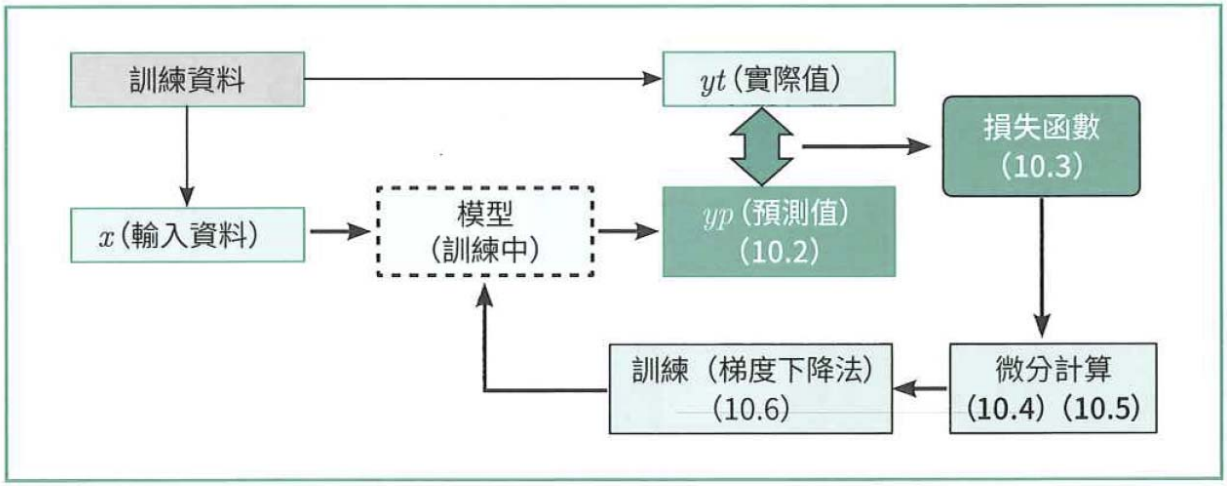


圖 10-1 本章學習地圖

範例問題設定

- **mnist手寫數字資料集**(The MNIST database of handwritten digits):
  - 7萬張解析度為28x28的手寫數字影像資料
  - 使用其中6萬張作為訓練資料集。
  - 使用其它1萬張作為驗證(測試)資料集。
- 將解析度28x28的影像資料轉成有784 (28x28)個分量的向量資料(也可稱為784維的 1D張量)，並建構以此為輸入資料的模型。
- 此向量各分量皆為1(黑)~255(白)的灰階值。
- 深度學習中也有將影像以矩陣(即2D張量)處理的方法，稱為卷積神經網路 (Convolutionαl neural networks, CNN)

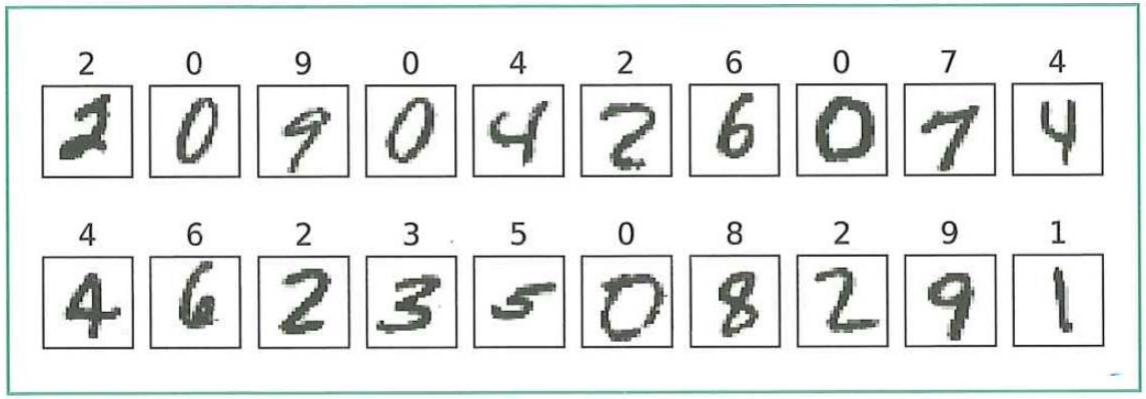


圖 10-2 *mnist* 資料

美國國家標準暨技術研究院 (National Institute of Standards and Technology, 簡寫為NIST)

模型的架構與預測函數

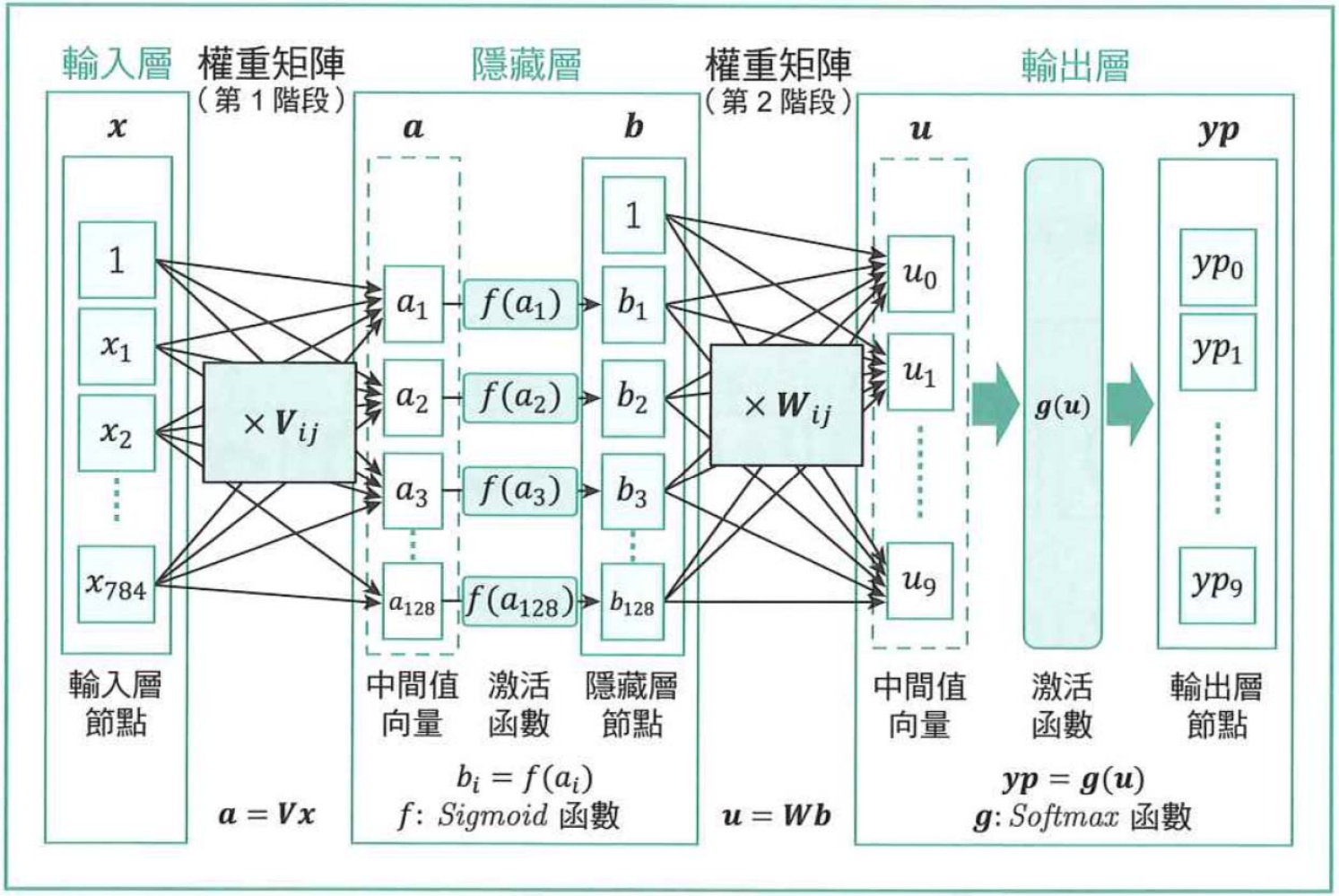


圖 10-3　3 層神經網路的架構圖

模型的架構與預測函數

- 隱藏層及輸出層各是由「中間值向量」、「激活函數」、「結果節點(隱藏層節點/輸出層節點)」這三者構成:
  - **中間值向量**: 用來稱呼前一層的節點與權重矩陣相乘之後得到的向量。(例如: $\boldsymbol{u}$)。
  - **激活函數**: 將中間值向量代入此函數，用以求出各層最終值(結果節點)。(例如: $g(\boldsymbol{u})$, sigmoid, softmax)。
  - **結果節點**: 由激活函數計算結果得到最終值的節點。(例如 : yp)。

	隱藏層	輸出層
中間值向量	$\boldsymbol{a}$	$\boldsymbol{u}$
激活函數	$Sigmoid$ 函數 $f(a_i)$	$Softmax$ 函數 $g(u)$
結果節點	$\boldsymbol{b}$（隱藏層節點）	$\boldsymbol{yp}$（輸出層節點）

表 10-1　各層與構成元素之間的關係

第1階段: 輸入層x到隱藏層b的關係

- 輸入層節點$\boldsymbol{x}$會增加1個的虛擬變數(x_0)，因此$\boldsymbol{x}$的維數會由原本的784維(28x28)變成785維。
- 輸入層各節點與第 1階段的權重矩陣 $\boldsymbol{V}_{ij}$相乘後，做為隱藏層的輸入資料。
- 若設定隱藏層節點$\boldsymbol{b}$的維數為128(此為經驗值)，則 $\boldsymbol{V}$會是一個 785x128的矩陣。
- 由輸入層$\boldsymbol{x}$求出中間值向量$\boldsymbol{a}$的方程式為: $\boldsymbol{a} = \boldsymbol{V}\boldsymbol{x}$。
- 再將$\boldsymbol{a}$的每個分量a_i用激活函數$f(x)$ 計算出隱藏層$\boldsymbol{b}$對應的b_i: $b_i = f(a_i)$。
- 此處令 $f(x)$ 為 Sigmoid函數，則連結輸入層節點$\boldsymbol{x}$與隱藏層節點$\boldsymbol{b}$的算式為: $$f(a_i) = \frac{1}{1 + \exp(-a_i)}$$

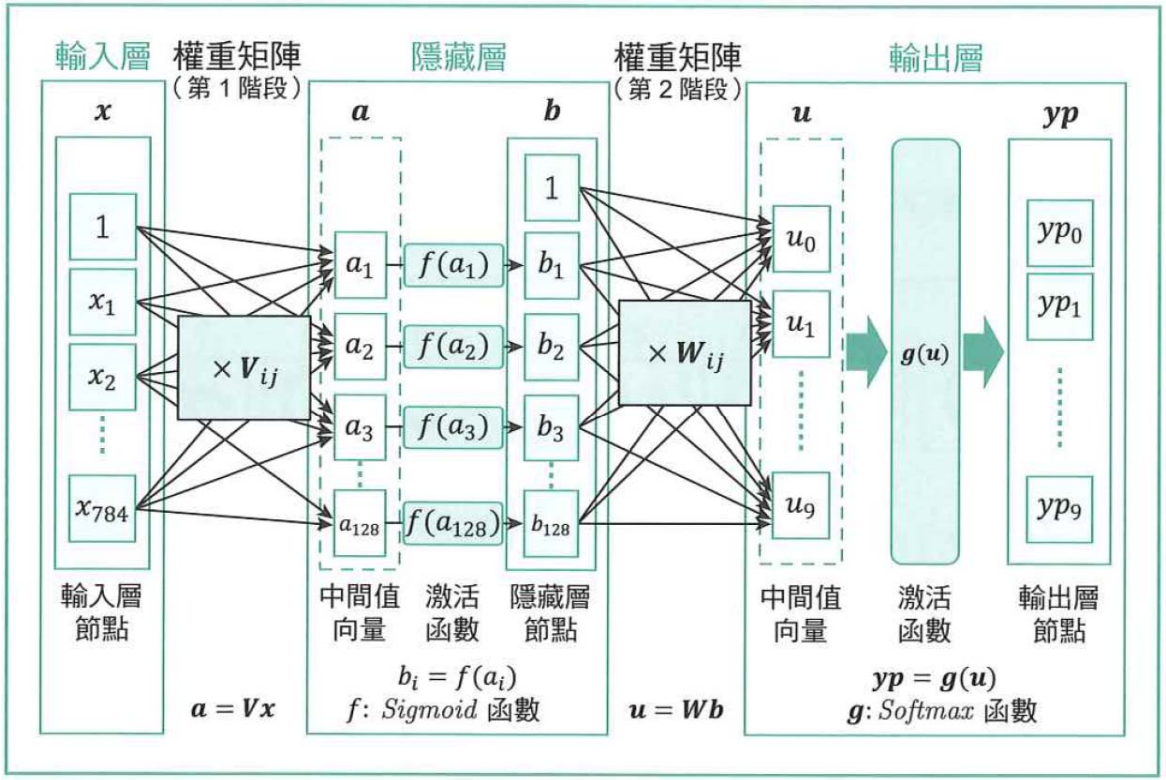


圖 10-3　3 層神經網路的架構圖

第2階段: 隱藏層b到輸出層yp的關係

- $\boldsymbol{b}$與權重矩陣$\boldsymbol{W}$相乘，可得到中間值向量$\boldsymbol{u}$: $\boldsymbol{u} = \boldsymbol{Wb}$。
- 將向量$\boldsymbol{u}$代入Softmax函數$g(u)$，得到輸出的預測值$\boldsymbol{yp}$: $\boldsymbol{yp} = \boldsymbol{g(u)}$。

(式中的 N 是指分類的類別數，此例是要區分出 0~ 9共10種類別。這也就是為何採用Softmax函數做多類別分類的原因。)

$$g_i(\boldsymbol{u}) = \frac{\exp(u_i)}{\sum_{k=0}^{N-1} \exp(u_k)}$$

「前向傳播(feedforward)法」, 或「前饋式神經網路」

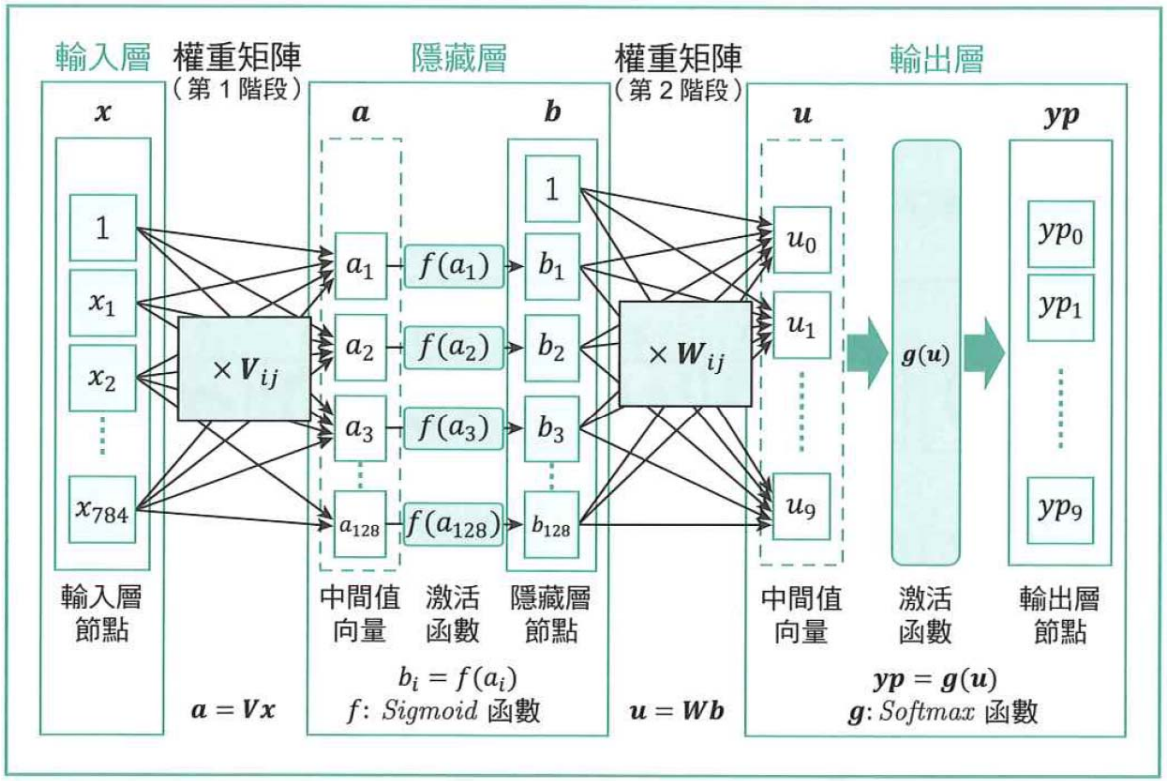


圖 10-3　3 層神經網路的架構圖

損失函數

$$L(\boldsymbol{W}) = -\frac{1}{M}\sum_{m=0}^{M-1}\sum_{i=0}^{N-1}(yt_i^{(m)}\log(yp_i^{(m)}))$$

比對一下 (9.5.1) 式

M：資料樣本的筆數

N：分類的類別數 (此範例中為 10)

$yt_i{}^{(m)}$：實際值 (第 i 個分類器對第 m 筆資料樣本的正確解答)

$yp_i{}^{(m)}$：預測值 (第 i 個分類器對第 m 筆資料樣本的輸出)

簡化算式

$$L(\boldsymbol{W}) = -\sum_{i=0}^{N-1} yt_i \log(yp_i)$$

損失函數的微分

- 對損失函數做微分，是為了之後的梯度下降法做準備。
- 簡化圖10-3，由資料輸入開始，一路到算出損失函數值的整個過程:

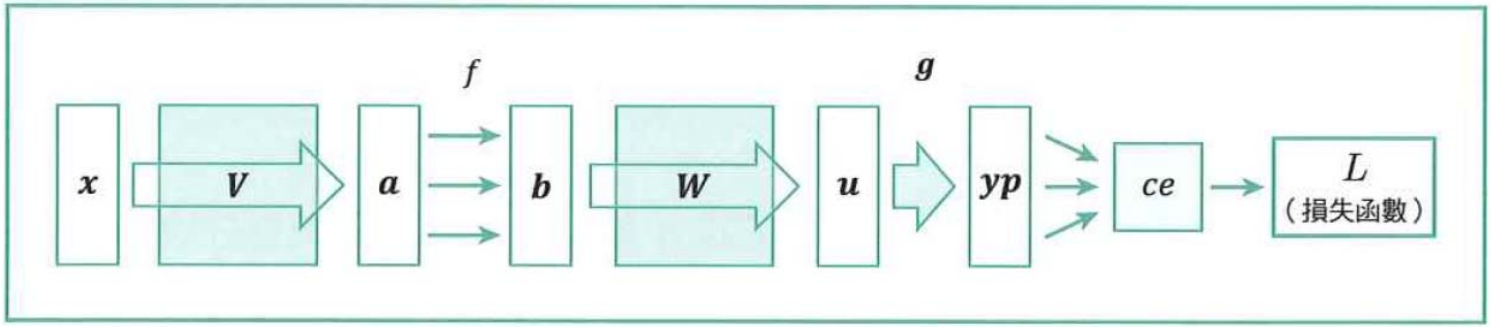


圖 10-4　由輸入資料到損失函數的關係

$$b_i = f(a_i)$$

$f(x)$：*Sigmoid* 函數

$$\boldsymbol{u} = \boldsymbol{W}\,\boldsymbol{b}$$

$$\boldsymbol{yp} = \boldsymbol{g}(\boldsymbol{u})$$

$g(\boldsymbol{u})$：*Softmax* 函數

$$L = \mathrm{ce} = -\sum_{i=0}^{N-1} yt_i \log(yp_i)$$

從損失函數到隱藏層: 對第2階段w權重矩陣各參數偏微分

與 9-6 節的推導結果完全相同

$$\boldsymbol{yd} = \boldsymbol{yp} - \boldsymbol{yt} \qquad (10.4.1) \leftarrow (9.6.10)$$

$$\frac{\partial L}{\partial u_i} = yd_i \qquad (10.4.2) \leftarrow (9.6.11)$$

$$\frac{\partial L}{\partial w_{ij}} = b_j \cdot yd_i \qquad (10.4.3) \leftarrow (9.6.13)$$

此處將 x_j 換成 b_j 了

從隱藏層到輸入層: 對第1階段 V 權重矩陣各參數偏微分

- 計算損失函數$\boldsymbol{L}$對$\boldsymbol{V}$的每個參數做偏微分，以 v_{12}為例說明，之後再改寫為v_{ij}的一般式。
- v_{12} 的變化(就是偏微分)，在隱藏層中間值向量 $\boldsymbol{a}$ 中，只有 a_1 會受影響，其他的$a_2 \sim a_{128}$皆無關。
- 因此$\boldsymbol{L}$對v_{12}偏微分，可用鏈鎖法則寫成下式：

$$\frac{\partial L}{\partial v_{12}} = \frac{\partial L}{\partial a_1} \cdot \frac{\partial a_1}{\partial v_{12}}$$

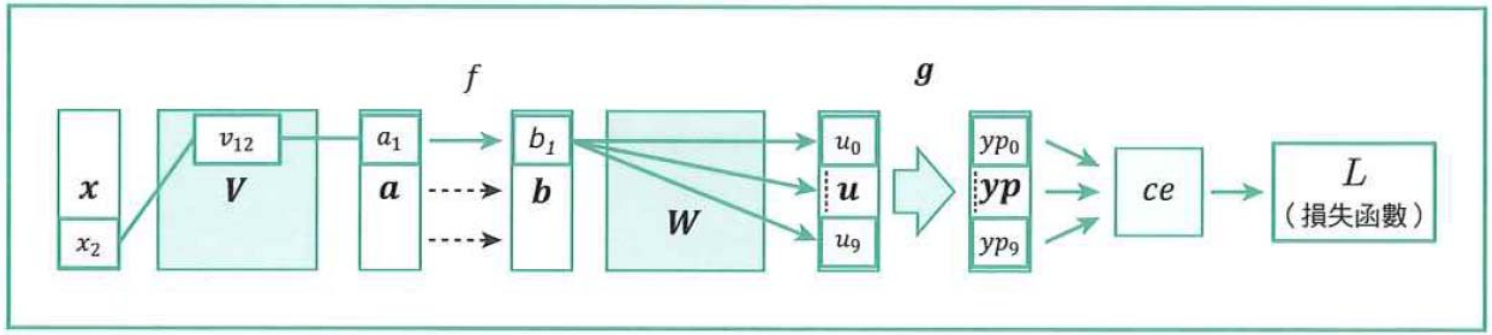


圖 10-5　會因 v_{12} 變化而受到影響的關係圖

計算 $\frac{\partial L}{\partial v_{ij}}$: $\frac{\partial a_1}{\partial v_{12}}$、$\frac{\partial L}{\partial a_1}$、$\frac{\partial L}{\partial b_1}$、$\frac{\partial u_l}{\partial b_1}$

$$\frac{\partial L}{\partial v_{12}} = \frac{\partial L}{\partial a_1} \cdot \frac{\partial a_1}{\partial v_{12}} \quad \Rightarrow \quad \frac{\partial L}{\partial v_{12}} = x_2 \cdot \frac{\partial L}{\partial a_1}$$

計算 $\frac{\partial a_1}{\partial v_{12}}$

$$\boldsymbol{a} = \boldsymbol{V}\boldsymbol{x}$$

$$a_1 = v_{10}x_0 + v_{11}x_1 + v_{12}x_2 + v_{13}x_3 + \cdots \quad \Rightarrow \quad \frac{\partial a_1}{\partial v_{12}} = x_2$$

計算 $\frac{\partial L}{\partial a_1}$

a_1的變化只會影響到b_1，與$b_2 \sim b_{128}$皆無關。因此$\boldsymbol{L}$對a_1的偏微分，可由鏈鎖法則:

$$\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial b_1} \cdot \frac{db_1}{da_1}$$

b_1 只與 a_1 有關，因此使用常微分符號

因為 $b_1 = f(a_1)$，因此 b_1對a_1微分即為函數$f(a_1)$對a_1的微分，因此可得:

$$\frac{db_1}{da_1} = f'(a_1)$$

計算 $\frac{\partial L}{\partial b_1}$

b_1 的變化對$\boldsymbol{u}$中所有的u_i都有影響，因此要用全微分鏈鎖法則來算出$\boldsymbol{L}$對b_1的偏微分。

$$\frac{\partial L}{\partial b_1} = \sum_{l=0}^{N-1} \frac{\partial L}{\partial u_l} \frac{\partial u_l}{\partial b_1}$$

$$\frac{\partial L}{\partial u_l} = y d_l$$

計算 $\frac{\partial L}{\partial v_{ij}}$: $\frac{\partial a_1}{\partial v_{12}}$、$\frac{\partial L}{\partial a_1}$、$\frac{\partial L}{\partial b_1}$、$\frac{\partial u_l}{\partial b_1}$

計算 $\frac{\partial u_l}{\partial b_1}$

$\boldsymbol{u = Wb}$

以 u_2 為例

$$u_2 = w_{20}b_0 + w_{21}b_1 + w_{22}b_2 + w_{23}b_3 + \cdots$$

$$\frac{\partial u_2}{\partial b_1} = w_{21}$$

改寫為一般式

$$\frac{\partial u_l}{\partial b_1} = w_{l1}$$

$$\frac{\partial L}{\partial b_1} = \sum_{l=0}^{N-1} \frac{\partial L}{\partial u_l}\frac{\partial u_l}{\partial b_1}$$

(前一頁)

$$\frac{\partial L}{\partial u_l} = yd_l$$

$$\frac{\partial u_l}{\partial b_1} = w_{l1}$$

$$\frac{\partial L}{\partial b_1} = \sum_{l=0}^{N-1} yd_l \cdot w_{l1}$$

$$\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial b_1} \cdot \frac{db_1}{da_1}$$

(前一頁)

$$\frac{\partial L}{\partial a_1} = f'(a_1) \sum_{l=0}^{N-1} yd_l \cdot w_{l1}$$

$$\frac{\partial L}{\partial a_i} = f'(a_i) \sum_{l=0}^{N-1} yd_l \cdot w_{li}$$

$$\frac{\partial L}{\partial v_{ij}} = x_j \cdot \frac{\partial L}{\partial a_i}$$

損失函數L對第1階段權重短陣 V偏微分的結果。所以L對第 1階段權重矩陣 V和第2階段權重短陣W的偏微分都算好了。

反向傳播

- 兩個權重矩陣W、V的偏微分，都是從輸出層節點反向計算回去，並使誤差值下降到設定的程度，即可得到最佳化的權重矩陣。這種運算方法稱為「**反向傳播(backpropagation)法**，或稱倒傳遞神經網路」。
- 「反向傳播的核心就是梯度下降法與鏈鎖法則」。

權重矩陣 W 的偏微分算式：

$$\frac{\partial L}{\partial w_{ij}} = b_j \cdot \frac{\partial L}{\partial u_i}$$

$$\frac{\partial L}{\partial u_i} = yd_i$$

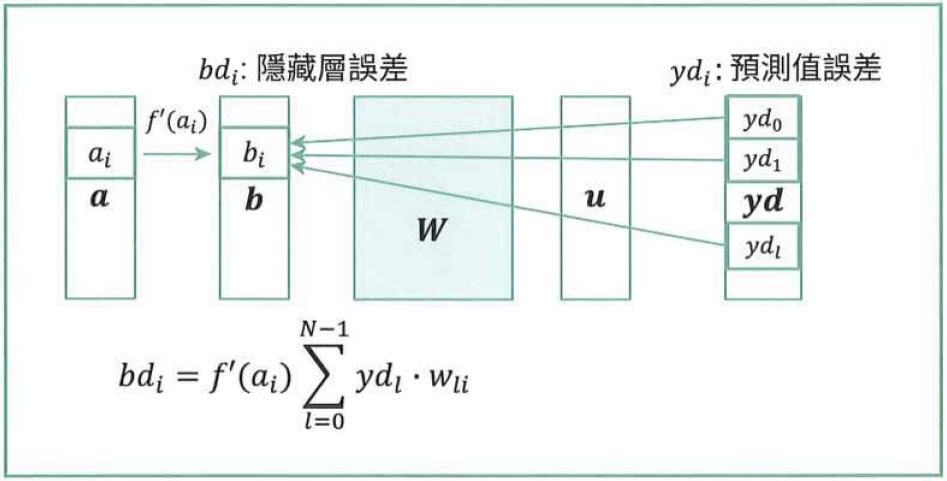


圖 10-6　隱藏層的誤差計算

權重矩陣 V 的偏微分算式：

$$\frac{\partial L}{\partial v_{ij}} = x_j \cdot \frac{\partial L}{\partial a_i}$$

$$\frac{\partial L}{\partial a_i} = f'(a_i) \sum_{l=0}^{N-1} yd_l \cdot w_{li}$$

將隱藏層 b 的誤差定義為 bd

$$bd_i = \frac{\partial L}{\partial a_i} = f'(a_i) \sum_{l=0}^{N-1} yd_l \cdot w_{li}$$

簡化

$$\frac{\partial L}{\partial v_{ij}} = x_j \cdot bd_i$$

$$\frac{\partial L}{\partial a_i} = bd_i$$

梯度下降法的運用

- 損失函數的權重矩陣偏微分已推導出來，用於實作梯度下降法。

變數與算式整理:

- **上下標**
  - k: 迭代運算次數 index。
  - m: 資料樣本 index。
  - i、j、l: 向量及矩陣的下標。

- **變數**
  - M: 資料樣本的總數。
  - N: 分類的類別數。
  - H: 隱藏層節點的維數。

- **演算法:**
  「函數定義」、「預測值計算」、「誤差計算」、「梯度計算」等四個部分。

隱藏層為1層時需要的式子

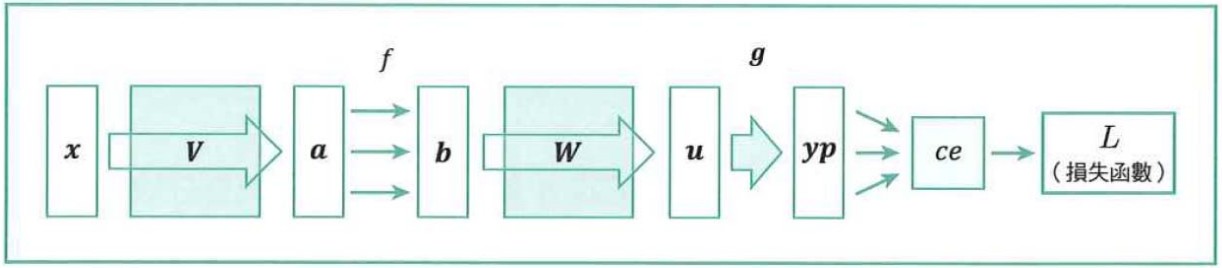


圖 10-4　由輸入資料到損失函數的關係

函數定義

Sigmoid 函數

$$f(x) = \frac{1}{1 + \exp(-x)}$$

Softmax 函數

$$g_i(\boldsymbol{u}) = \frac{\exp(u_i)}{\sum_{j=0}^{N-1} \exp(u_j)}$$

預測值計算

輸入層節點與第 1 階段權重矩陣之內積：

$$\boldsymbol{a}^{(k)(m)} = \boldsymbol{V}^{(k)} \boldsymbol{x}^{(m)}$$

將內積結果代入 *Sigmoid* 函數，並令其為隱藏層節點之值：

$$b_i^{(k)(m)} = f(a_i^{(k)(m)})$$

隱藏層節點與第 2 階段權重矩陣之內積：

$$\boldsymbol{u}^{(k)(m)} = \boldsymbol{W}^{(k)} \boldsymbol{b}^{(k)(m)}$$

將內積結果代入 *Softmax* 函數，並令其為預測值：

$$\boldsymbol{yp}^{(k)(m)} = \boldsymbol{g}(\boldsymbol{u}^{(k)(m)})$$

隱藏層為1層時需要的式子

誤差計算

預測值誤差：

$$\boldsymbol{yd}^{(k)(m)} = \boldsymbol{yp}^{(k)(m)} - \boldsymbol{yt}^{(m)}$$

由預測值誤差計算隱藏層之誤差：

$$bd_i^{(k)(m)} = f'(a_i^{(k)(m)}) \sum_{l=0}^{N-1} yd_l^{(k)(m)} w_{li}^{(k)}$$

梯度計算並修正權重參數

由預測值誤差計算第 2 階段權重矩陣之梯度並修正權重參數：

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} b_j^{(k)(m)} yd_i^{(k)(m)}$$

由隱藏層誤差計算第 1 階段權重矩陣之梯度並修正權重參數：

$$v_{ij}^{(k+1)} = v_{ij}^{(k)} - \frac{\alpha}{M} \sum_{m=0}^{M-1} x_j^{(m)} bd_i^{(k)(m)}$$

將隱藏層的層數增加，運算原則都是不變的。