電腦視覺的原理與應用
FOUNDATIONS OF COMPUTER VISION

張家瑋 博士
新漢股份有限公司工業4.0創新中心 顧問

HOW WE TEACH COMPUTERS TO UNDERSTAND PICTURES

https://youtu.be/40riCqvRoMs

電腦視覺

- 利用攝影機和電腦代替人眼對目標進行識別、跟蹤和測量等機器視覺，並做圖像處理，用電腦處理為更適合人眼觀察或傳送給儀器檢測的圖像。

- 人工智慧的主要研究問題是：如何讓系統具備「計劃」和「決策能力」，使之完成特定的動作，如移動機器人通過特定環境。
  - 此問題中，電腦視覺可作為感知器，為決策提供資訊。其中研究方向包括模式識別和機器學習，因此電腦視覺被看作人工智慧的分支。

電腦視覺應用

- 作為一個工程學科，電腦視覺基於相關理論來建立電腦視覺系統。這類系統的組成部分包括：
  1. 過程控制(Process Control)（如工業機器人和無人駕駛車）
  2. 事件監測(Event Monitoring)（如圖像監測）
  3. 資訊組織(Information Organization)（如圖像資料庫和圖像序列的索引建立）
  4. 物體與環境建模（如工業檢查，醫學圖像分析和拓撲建模）
  5. 交感互動（如人機互動的輸入裝置）

電腦視覺的相關領域 (1/2)

- **電腦視覺**的研究物件主要是對映到單幅或多幅圖像上的三維場景，如三維場景的重建。電腦視覺的研究很大程度上針對圖像的內容。
- **圖像處理**與**圖像分析**的研究物件主要是二維圖像，實現圖像的轉化，尤其針對像素級的操作，例如提高圖像對比度，邊緣提取，去雜訊和幾何變換如圖像旋轉。這一特徵表明無論是圖像處理還是圖像分析其研究內容都和圖像的具體內容無關。

電腦視覺的相關領域 (2/2)

- **機器視覺**主要是指工業領域的視覺研究，如自主機器人的視覺，用於檢測和測量的視覺。這表明在這一領域通過軟體硬體、圖像感知與控制理論與圖像處理緊密結合來實現高效的機器人控制或各種實時操作。
- **模式識別**使用各種方法從訊號中提取資訊，主要運用統計學的理論。此領域的一個主要方向便是從圖像資料中提取資訊。

電腦視覺的步驟 (1/3)

1. 圖像取得：數位圖像是由一或多個圖像傳感器產生，傳感器可以是各種攝錄影機，包括X-Ray斷層掃瞄，雷達，超聲波等，圖片可以是二維、三維圖組或者一個圖像序列。圖片像素值對應在一個或多個光譜上（如灰階圖）。
2. 預處理：對圖像提取某種特定的資訊，使圖像滿足後繼方法的要求。如：
   - 二次取樣保證圖像坐標的正確
   - 平滑去噪來濾除傳感器引入的裝置雜訊
   - 提高對比度來保證實現相關資訊可以被檢測到
   - 調整尺度空間使圖像結構適合局部應用

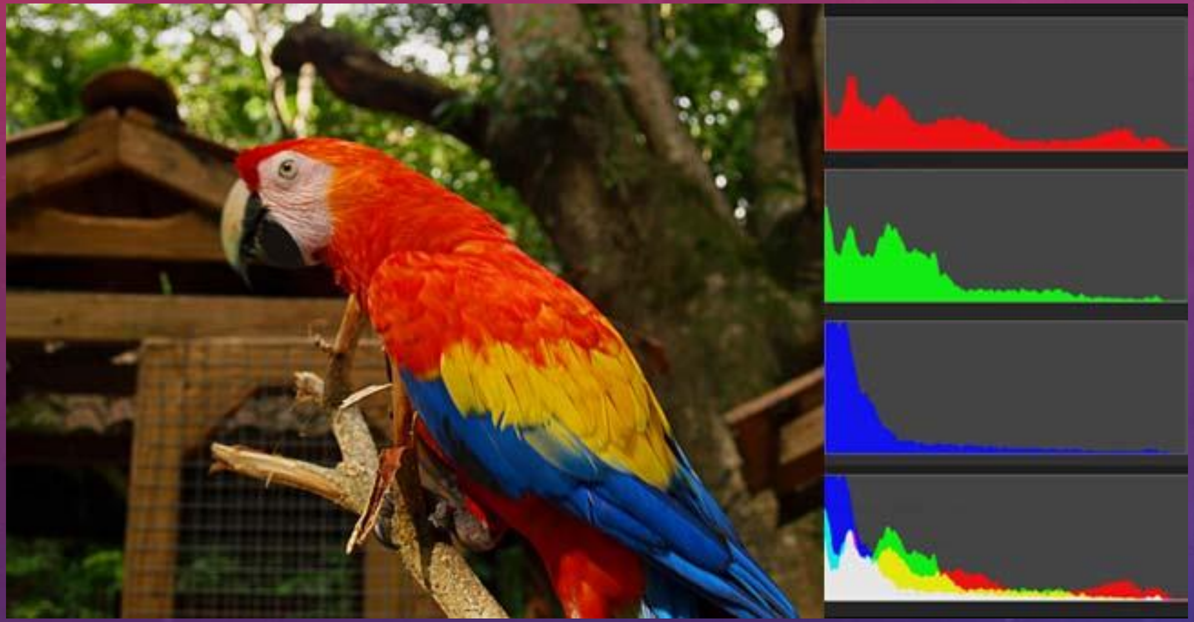
電腦視覺的步驟 (2/3)

3. 特徵提取：從圖像中提取各種複雜度的特徵。
    - 線、邊的提取
    - 局部的特徵點檢測，如邊角、斑點檢測
    - 更複雜的特徵可能與紋理或形狀有關。
4. 檢測/分割：對圖像分割並提取用於後繼處理的部分。
    - 篩選特徵點
    - 分割一或多幅圖片中含有特定目標的部分

電腦視覺的步驟 (3/3)

5. 進階處理：資料往往已經精煉到很小的數量，如含有目標物體的部分。
   - 驗證得到的資料是否符合前提要求
   - 估測特定係數，比如目標的姿態，體積
   - 對目標進行分類

COLOR HISTOGRAM



- 顏色都是由紅綠藍三原色（RGB）構成的，所以左圖共有4張直方圖（三原色直方圖 + 最後合成的直方圖）。
- 每種原色都可以取256個值，那麼整個顏色空間共有1600萬種顏色（256的三次方）。
- 針對1600萬種顏色比較直方圖，計算量太大，因此需要簡化。
  - 可以將0～255分成四個區：0～63為第0區，64～127為第1區，128～191為第2區，192～255為第3區。
  - 紅綠藍分別有4個區，總共可以構成64種組合（4的3次方）。

http://www.ruanyifeng.com/blog/2011/07/principle_of_similar_image_search.html

COLOR HISTOGRAM

红	绿	蓝	像素数量	红	绿	蓝	像素数量	红	绿	蓝	像素数量	红	绿	蓝	像素数量
0	0	0	7414	1	0	0	891	2	0	0	1146	3	0	0	11
0	0	1	230	1	0	1	13	2	0	1	0	3	0	1	0
0	0	2	0	1	0	2	0	2	0	2	0	3	0	2	0
0	0	3	0	1	0	3	0	2	0	3	0	3	0	3	0
0	1	0	8	1	1	0	592	2	1	0	2552	3	1	0	856
0	1	1	372	1	1	1	3462	2	1	1	9040	3	1	1	1376
0	1	2	88	1	1	2	355	2	1	2	47	3	1	2	0
0	1	3	0	1	1	3	0	2	1	3	0	3	1	3	0
0	2	0	0	1	2	0	0	2	2	0	0	3	2	0	0
0	2	1	0	1	2	1	101	2	2	1	8808	3	2	1	3650
0	2	2	10	1	2	2	882	2	2	2	53110	3	2	2	6260
0	2	3	1	1	2	3	16	2	2	3	11053	3	2	3	109
0	3	0	0	1	3	0	0	2	3	0	0	3	3	0	0
0	3	1	0	1	3	1	0	2	3	1	0	3	3	1	0
0	3	2	0	1	3	2	0	2	3	2	170	3	3	2	3415
0	3	3	0	1	3	3	0	2	3	3	17533	3	3	3	53929

- 左圖是某張圖片的顏色分佈表，將表中最後一欄提取出來，組成一個64維向量(7414, 230, 0, 0, 8, ..., 109, 0, 0, 3415, 53929) 。這個向量就是這張圖片的特徵值或者叫"指紋"。
- 尋找相似圖片就變成找出與其最相似的向量，可利用Pearson相關係數或Cosine相似度算出。

PERCEPTUAL HASH ALGORITHM

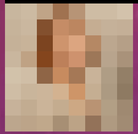
- 利用以下步驟對每張圖片生成“指紋”（fingerprint）字符串來比較不同圖片的指紋。
  1. 縮小尺寸：只保留圖片的結構，明暗等基本資訊，避免不同尺寸的差異。如縮成 8*8。
  2. 簡化色彩：將圖片轉為64級灰度，所有像素點總共只有64種顏色。
  3. 計算平均值：計算所有64個像素的灰度平均值。
  4. 比較像素的灰度：將每個像素的灰度與平均值比較，>=平均值，記為1、<平均值，記為0。
  5. 計算 Hash 值： 這64位的整數組合，就是圖片的指紋。組合的次序並不重要，只要保證所有圖片都採用同樣次序就行了，在此以 Hash 加密方法來組成。
     - 雜湊演算法(Hash Alogrithm)是一種從資料中建立「數位指紋(Digital fingerprint)」的方法，可以將任何長度的資料轉換成一個長度較短的「雜湊值(Hash value)」，又稱為「訊息摘要(MD：Message Digest)」。

http://www.ruanyifeng.com/blog/2011/07/principle_of_similar_image_search.html

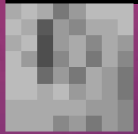
PERCEPTUAL HASH ALGORITHM

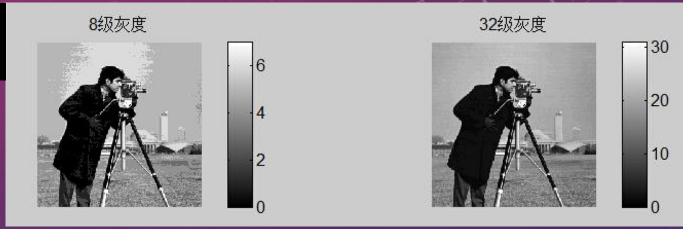
原圖

1. 縮小 (8*8)



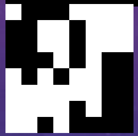
2. 簡化色彩 (64級灰度)





3. 計算64個像素的灰度平均值

4. 每個像素與灰度平均值比較



5. 得出 Hash 值

8f373714acfcf4d0

- 得到指紋以後，就可以對比不同的圖片，看看64位中有多少位是不一樣的。如果不相同的數據位不超過5，就說明兩張圖片很相似；如果大於10，就說明這是兩張不同的圖片。

MACHINE LEARNING

機器學習的基本概念與背景知識

BIG DATA 的沿革 (1/3)

- Data Mining
  - 資料探勘是利用分析技術來發掘資料間未知的關聯性與規則。
  - 少女未婚懷孕　購物商場比老爸還早知道？！
    - https://www.nownews.com/news/20120223/42676

DATA MINING

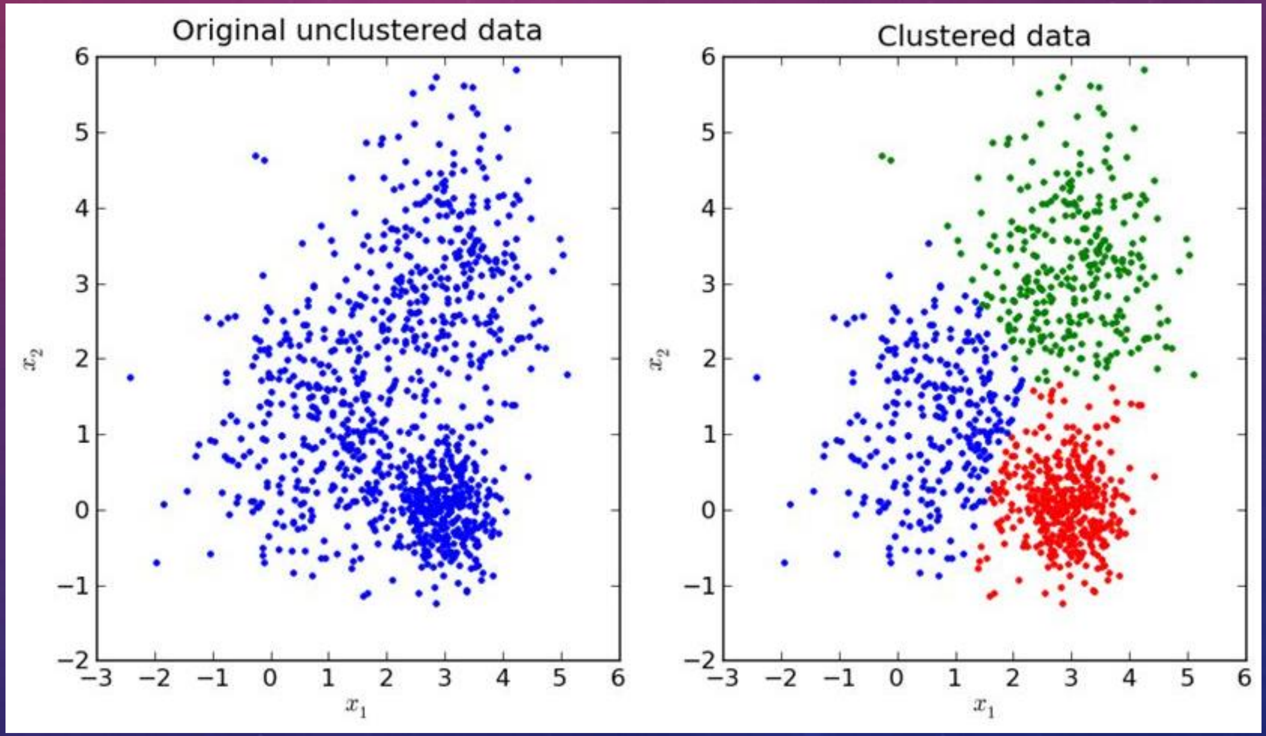
- ✓ 分群
  - 用於沒有標籤的資料，又通常為非監督式演算法。
- ✓ 分類
  - 用於有標籤的資料，又通常為監督式演算法。
- ✓ 關聯式法則
  - 有序性規則的資料

DATA MINING

```
RespondentId,StartDate,CompletedDate,LanguageCode,Question1,Question2,Question3,Question4,Question5,Question6,Question7,Questi
27357,2006.11.27 15:6,2006.11.27 15:7,en,Denmark,Financial Services,6 - 12 months,26-100,4,4,2,"cvbcvb",2,3,3,1,Opinio,1,0,0,1
27359,2006.11.27 15:7,2006.11.27 15:8,en,Italy,Hardware Vendor,1 - 2 years,26-100,3,5,4,,1,3,3,4,Opinio,0,0,0,0,1,0,0,1,0,,,,0
27360,2006.11.27 15:8,2006.11.27 15:8,en,Lithuania,Retail,6 - 12 months,6-10,4,1,4,"this is a random other text",2,2,2,2,Opini
27361,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Retail,6 - 12 months,6-10,4,1,4,"this is a random other text",2,2,2,2,Opinio,0
27362,2006.11.27 15:8,2006.11.27 15:8,en,Djibouti,Manufacturing,6+ years,101-250,0,4,0,"another random text",5,5,5,5,Opinio,1,
27363,2006.11.27 15:8,2006.11.27 15:8,en,Tanzania,Retail,1 - 2 years,1001-5000,1,1,1,"123456",2,2,2,2,Opinio,0,1,1,1,1,1,1,1
27364,2006.11.27 15:8,2006.11.27 15:8,en,Vanuatu,Other,1 - 2 years,1001-5000,6,5,6,"123456",6,6,6,6,Opinio,0,0,1,1,1,0,1,1,"
27365,2006.11.27 15:8,2006.11.27 15:8,en,Angola,Government,1 - 2 years,11-25,4,2,4,"123456",3,3,3,3,Opinio,0,0,1,1,1,1,1,0,,
27366,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Manufacturing,<6 months,1-5,1,4,1,"hey",5,5,5,5,Opinio,0,1,0,0,0,1,0,0,0,,"hey
27367,2006.11.27 15:8,2006.11.27 15:8,en,Norway,Education,2 - 5 years,5001-10000,6,0,6,"£6{[]}+åøæ'´'¨*-/+\",1,1,1,1,Opinio,1,
27368,2006.11.27 15:8,2006.11.27 15:8,en,Bermuda,Software Vendor,1 - 2 years,11-25,0,2,0,"123456",3,3,3,3,Opinio,1,0,1,0,0,1,0
27369,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Transportation,1 - 2 years,11-25,5,4,5,"123456",5,5,5,5,Opinio,0,1,0,0,0,1,0,0
27370,2006.11.27 15:8,2006.11.27 15:8,en,Maldives,Other,6+ years,10001 or more,2,5,2,"another random text",6,6,6,6,Network Pro
27371,2006.11.27 15:8,2006.11.27 15:8,en,Kyrgyzstan,Medical,2 - 5 years,26-100,3,5,3,"£6{[]}+åøæ'´'¨*-/+\",6,6,6,6,Network Pro
27372,2006.11.27 15:8,2006.11.27 15:8,en,Antigua and Barbuda,Government,6 - 12 months,501-1000,6,2,6,"this is a random other t
27373,2006.11.27 15:8,2006.11.27 15:8,en,Belarus,Financial Services,6+ years,10001 or more,2,1,2,"another random text",2,2,2,2
27374,2006.11.27 15:8,2006.11.27 15:8,en,Vatican City,Non-profit,1 - 2 years,11-25,0,0,0,"123456",1,1,1,1,Network Probe,1,0,0,
27375,2006.11.27 15:8,2006.11.27 15:8,en,Georgia,Financial Services,6+ years,10001 or more,6,1,6,"another random text",2,2,2,2
27376,2006.11.27 15:8,2006.11.27 15:8,en,Tokelau,Transportation,1 - 2 years,11-25,2,4,2,"123456",5,5,5,5,Network Probe,0,1,0,0
27377,2006.11.27 15:8,2006.11.27 15:8,en,Chad,Software Vendor,<6 months,1-5,6,2,6,"hey",3,3,3,3,Network Probe,1,1,1,1,1,1,1,1,
27378,2006.11.27 15:8,2006.11.27 15:8,en,Turkey,Software Vendor,6 - 12 months,501-1000,1,2,1,"this is a random other text",3,3
27379,2006.11.27 15:8,2006.11.27 15:8,en,East Timor,Transportation,<6 months,1-5,0,4,0,"hey",5,5,5,5,Opinio,1,1,0,0,1,0,1,1,0,
27380,2006.11.27 15:8,2006.11.27 15:8,en,Nicaragua,Medical,6 - 12 months,6-10,5,5,5,"this is a random other text",6,6,6,6,Opin
27381,2006.11.27 15:8,2006.11.27 15:8,en,Equatorial Guinea,Software Vendor,6+ years,101-250,6,2,6,"another random text",3,3,3,
27382,2006.11.27 15:8,2006.11.27 15:8,en,Zambia,Retail,<6 months,251-500,1,1,1,"hey",2,2,2,2,Surveyor,0,1,0,0,0,0,0,1,0,,"hey"
27383,2006.11.27 15:8,2006.11.27 15:8,en,French Southern and Antarctic Lands,Retail,1 - 2 years,1001-5000,2,1,2,"123456",2,2,2
27384,2006.11.27 15:8,2006.11.27 15:8,en,Guinea-Bissau,Hardware Vendor,2 - 5 years,26-100,6,3,6,"£6{[]}+åøæ'´'¨*-/+\",4,4,4,4,
27385,2006.11.27 15:8,2006.11.27 15:8,en,Viet Nam,Medical,2 - 5 years,26-100,4,5,4,"£6{[]}+åøæ'´'¨*-/+\",6,6,6,6,Opinio,1,1,1,
27386,2006.11.27 15:8,2006.11.27 15:8,en,Reunion,Medical,1 - 2 years,1001-5000,2,5,2,"123456",6,6,6,6,Opinio,1,1,1,1,1,1,1,1,1
27387,2006.11.27 15:8,2006.11.27 15:8,en,Puerto Rico,Non-profit,<6 months,1-5,0,0,0,"hey",1,1,1,1,Opinio,1,1,1,1,0,1,1,1,0,,"h
27388,2006.11.27 15:8,2006.11.27 15:8,en,East Timor,Financial Services,6 - 12 months,6-10,1,1,1,"this is a random other text",
27389,2006.11.27 15:8,2006.11.27 15:8,en,Northern Mariana Islands,Software Vendor,<6 months,1-5,2,2,2,"hey",3,3,3,3,Opinio,1,0
```

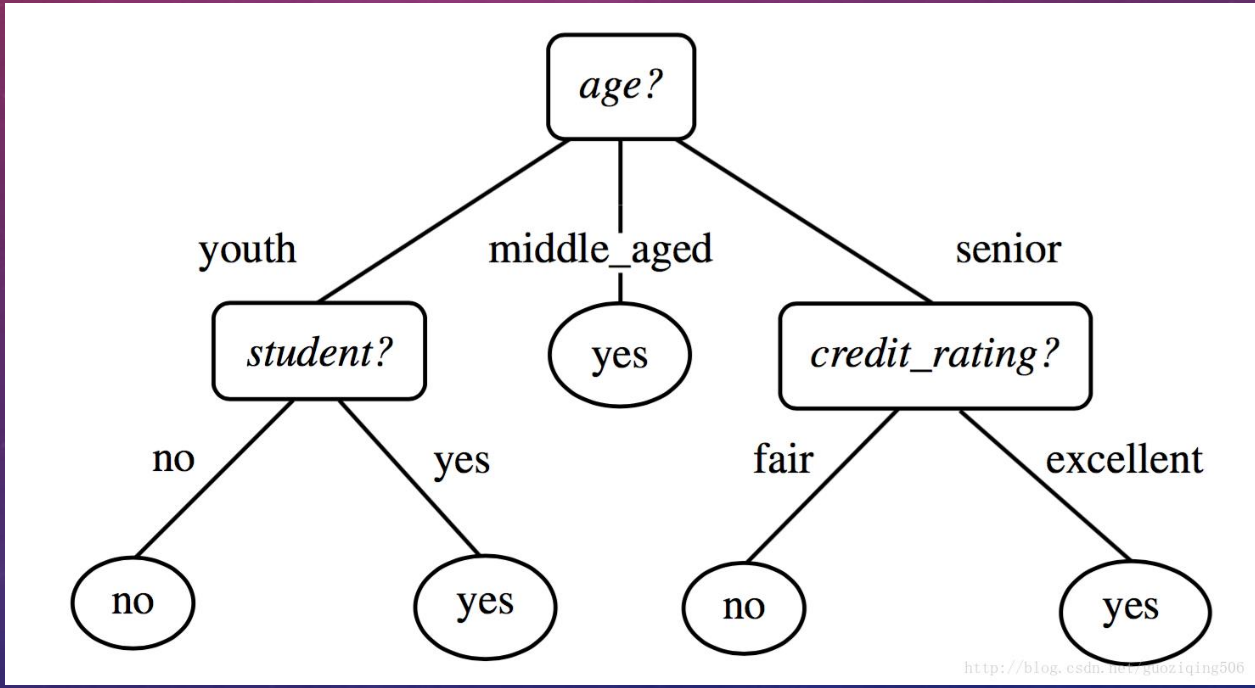
DATA MINING

- 分群
  - 用於沒有標籤的資料，又通常為非監督式演算法。



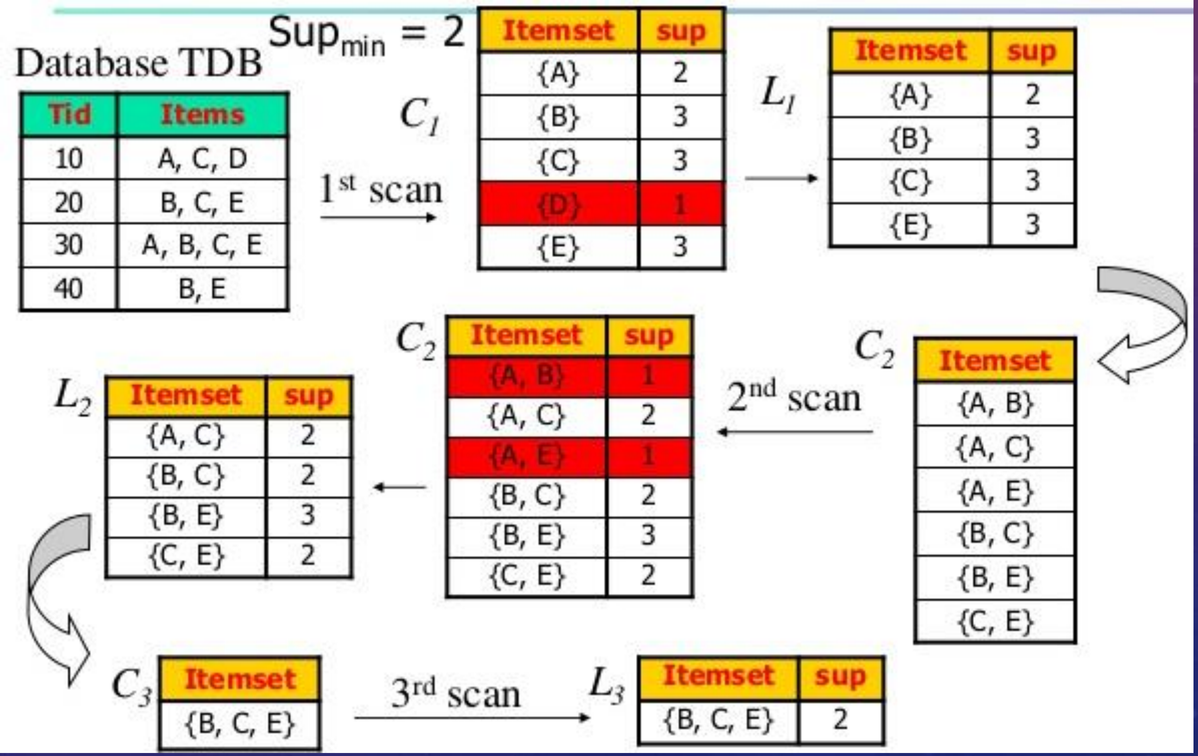
DATA MINING

- 分類
  - 用於有標籤的資料，又通常為監督式演算法。



DATA MINING

- 關聯式法則
  - 有序性 (尿布與啤酒)

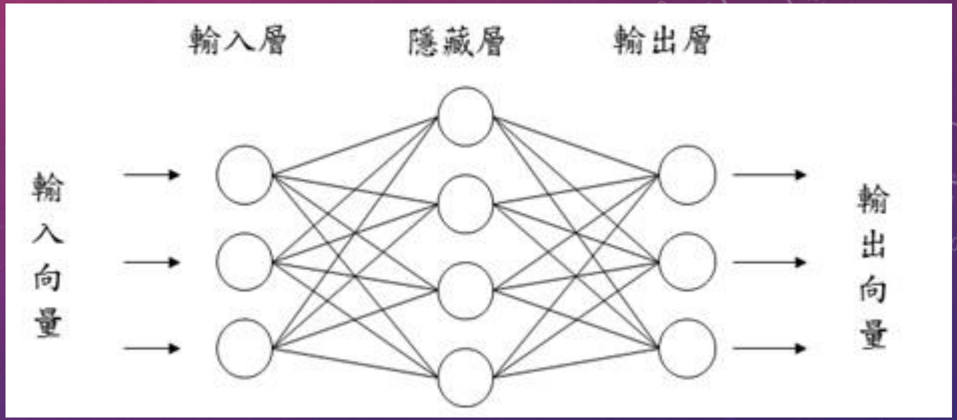


BIG DATA 的沿革 (2/3)

- Machine Learning
  - 人工智慧的分支，可用於資料探勘。
  - 讓機器可以自動學習、從巨量資料中找到規則，進而有能力做出分類或預測。
    - 判斷出類別
    - 估計出數值

BIG DATA 的沿革 (3/3)

- Deep Learning
  - 是機器學習的分支
  - 類神經網路的文藝復興
  - 從大規模未標記資料中建立更好的預測模型
  - 建立強 AI 的可能性



資料分析的基本步驟

1. 資料清除：去除極端、遺失值資料、不重要的屬性
2. 資料整合：因應用目的或特性，整合不同來源的資料
3. 資料選擇：揀選重要的屬性來逼近目的之最佳成效
4. 資料轉換：基於領域知識進行特徵縮放、數值類別轉換等
5. 資料探勘：選用合適的分析演算法得到目的之結果
6. 樣式評估：評估結果的樣式，是否如預期
7. 知識表示：因應目的將樣式轉換成合適的表達方法

資料分析的演算法重點

- 預處理（Preprocessing）
- 降維（Dimensionality Reduction）
- 模型選擇（Model Selection）
  - 監督式學習（Supervised learning）
    - 分類（Classification）：機器給出一個類別
    - 迴歸（Regression）：機器給出一個數值
  - 非監督式學習（Unsupervised learning）
    - 分群（Clustering）

降維（DIMENSIONALITY REDUCTION）

- 奇異值分解
  - Singular Value Decomposition (SVD)

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

=

book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

*

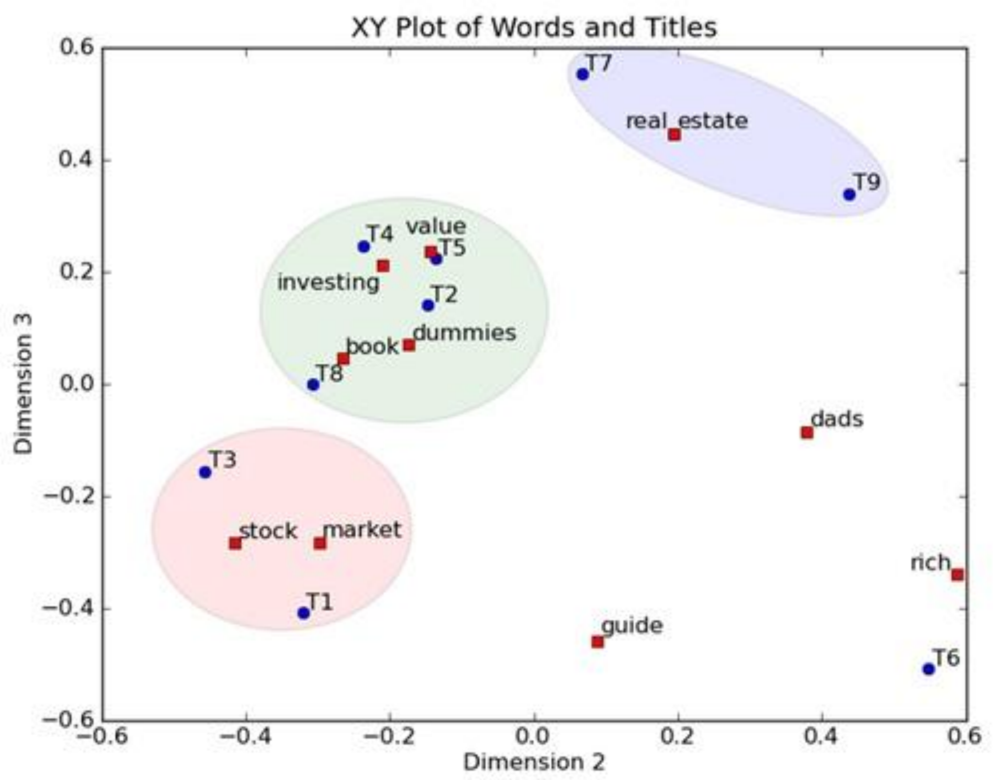
3.91	0	0
0	2.61	0
0	0	2.00

*

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

降維（DIMENSIONALITY REDUCTION）

- 奇異值分解
  - Singular Value Decomposition (SVD)



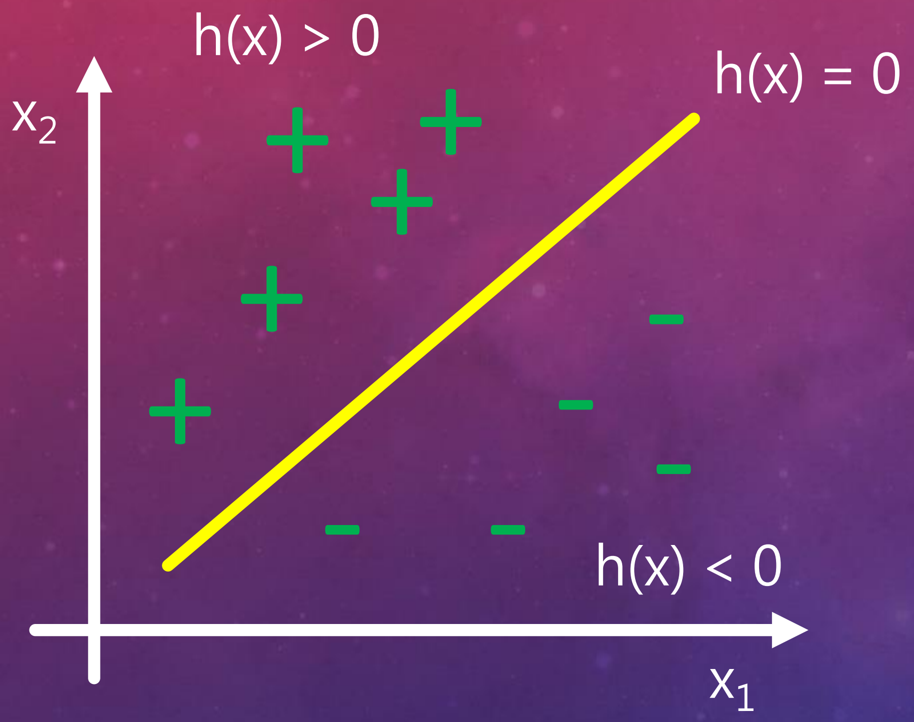
Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

DEEP LEARNING

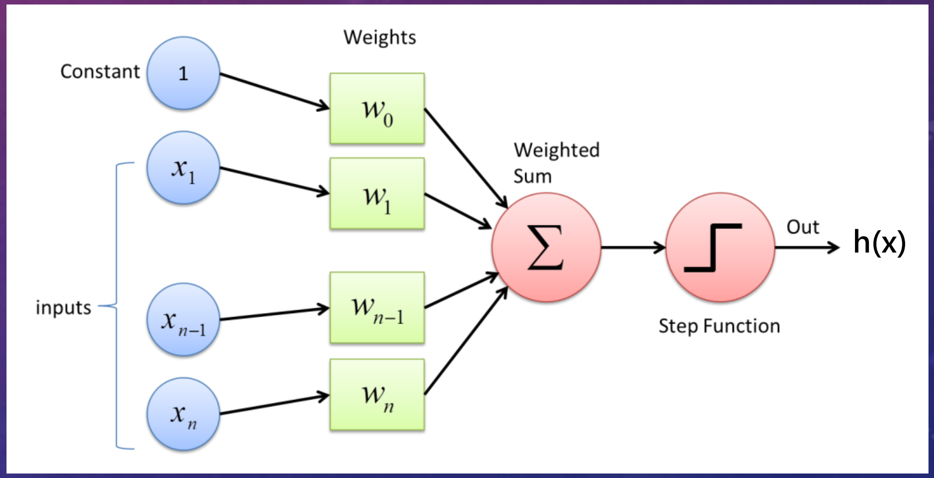
人工神經網路

ARTIFICIAL NEURAL NETWORK

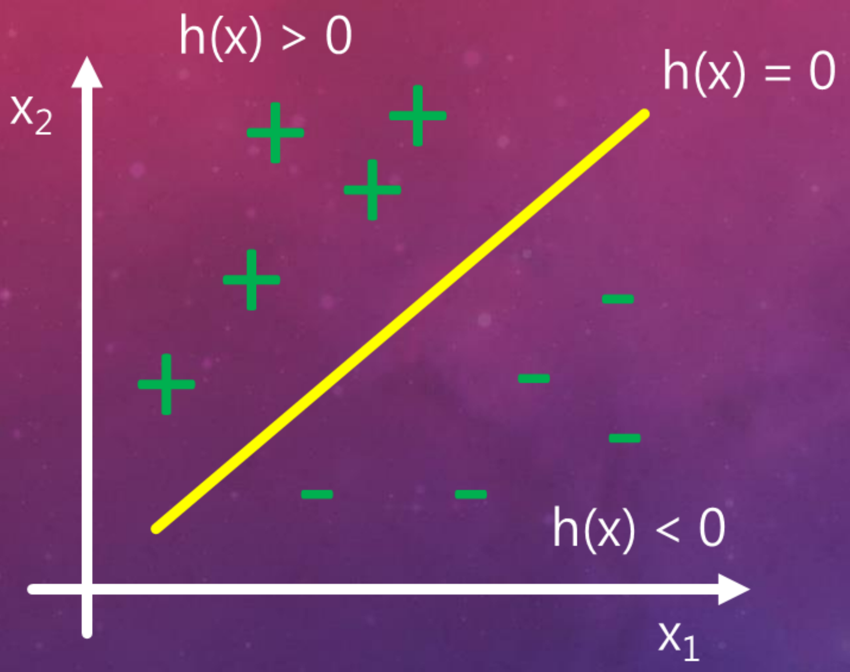
Perceptron Linear Algorithm



- Features: $x = (x_1, x_2)$
- Target: $y = +1$ or -1
- $h(x) = w_0 + w_1x_1 + w_2x_2$



Perceptron Linear Algorithm



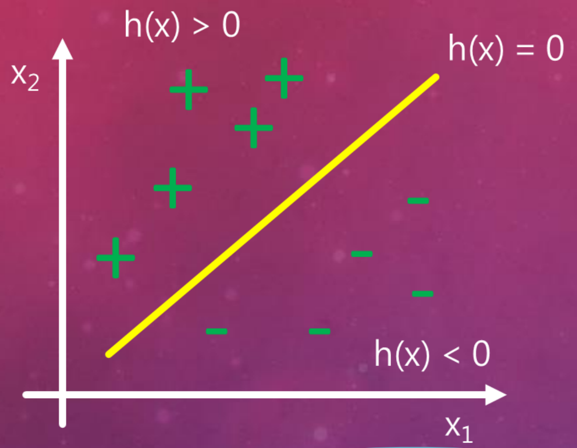
$$h(x) = w_0 + w_1x_1 + w_2x_2$$

$$scores = \sum_{i}^{N} w_i x_i + b$$

$$scores = \sum_{i}^{N+1} w_i x_i$$

- 若 $scores \geq 0$ ，则 $\hat{y} = 1$
- 若 $scores < 0$ ，则 $\hat{y} = -1$

Perceptron Linear Algorithm



- 若 $scores \geq 0$ ，则 $\hat{y} = 1$
- 若 $scores < 0$ ，则 $\hat{y} = -1$

\+ -

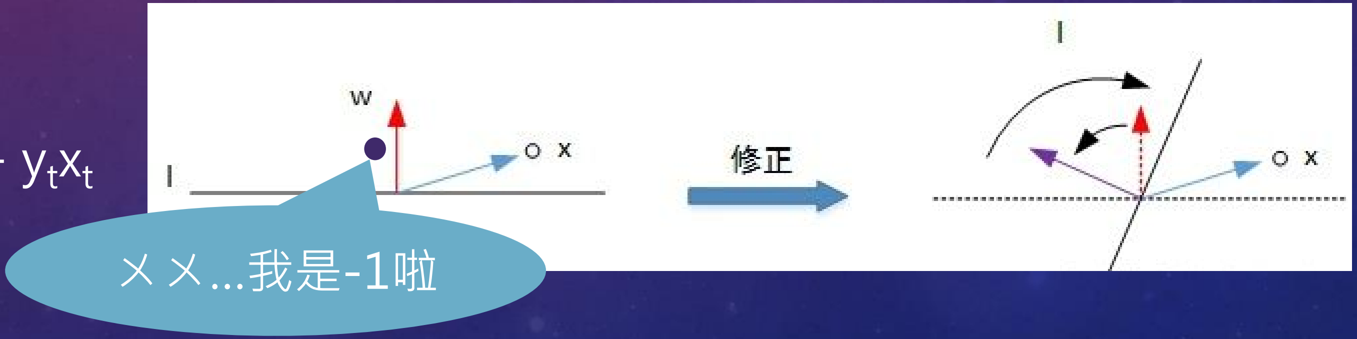
$w_{t+1} = w_t + y_t x_t$

[Case 1]

y = 1 錯分成 y = -1

\- +

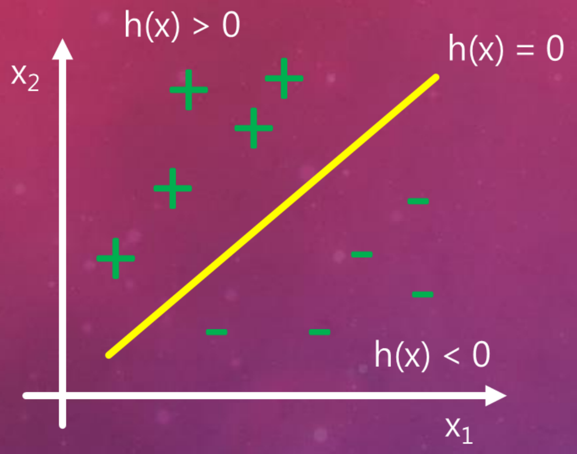
$w_{t+1} = w_t + y_t x_t$



[Case 2]

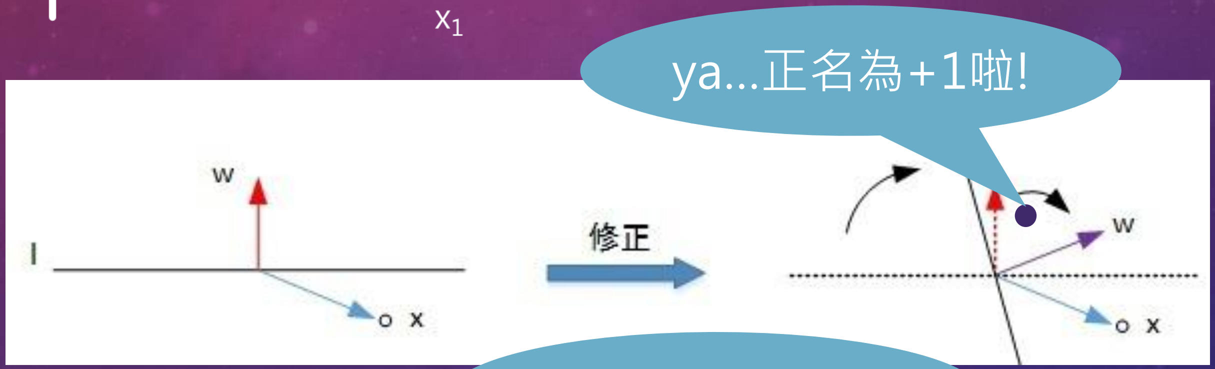
y = -1 錯分成 y = 1

Perceptron Linear Algorithm



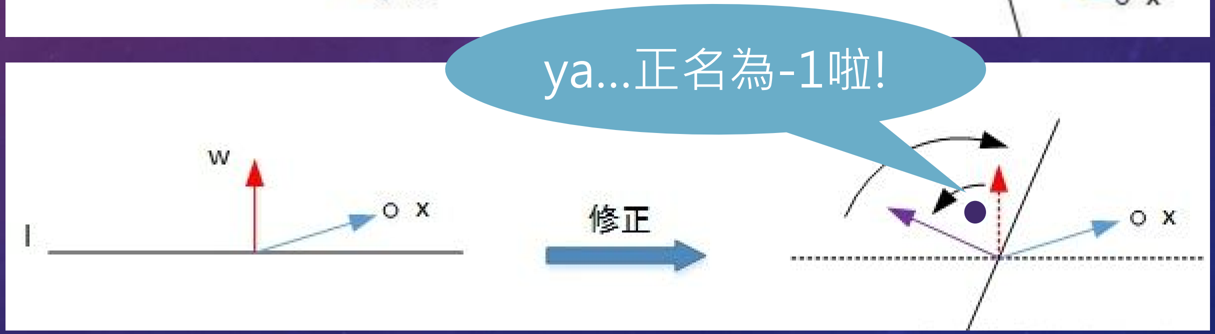
- 若 $scores \geq 0$ ，则 $\hat{y} = 1$
- 若 $scores < 0$ ，则 $\hat{y} = -1$

$$\overset{+}{w_{t+1}} = \overset{-}{w_t} + \overset{+}{y_t x_t}$$

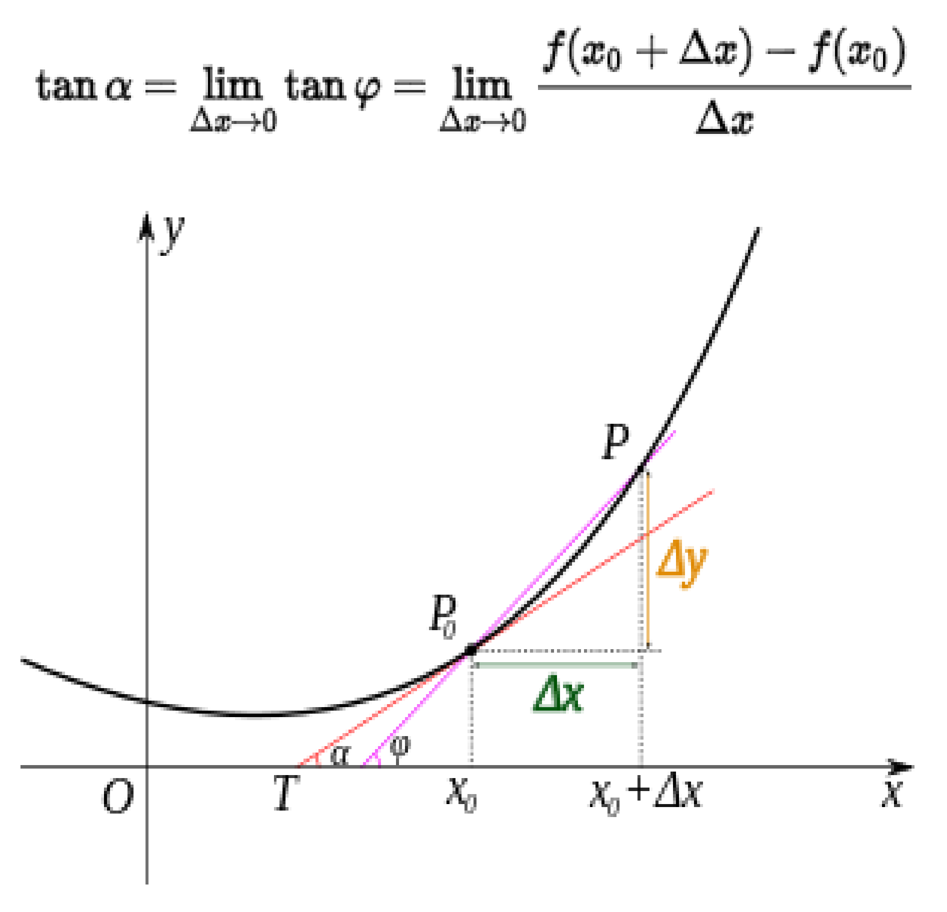


[Case 1]
y = 1 錯分成 y = -1

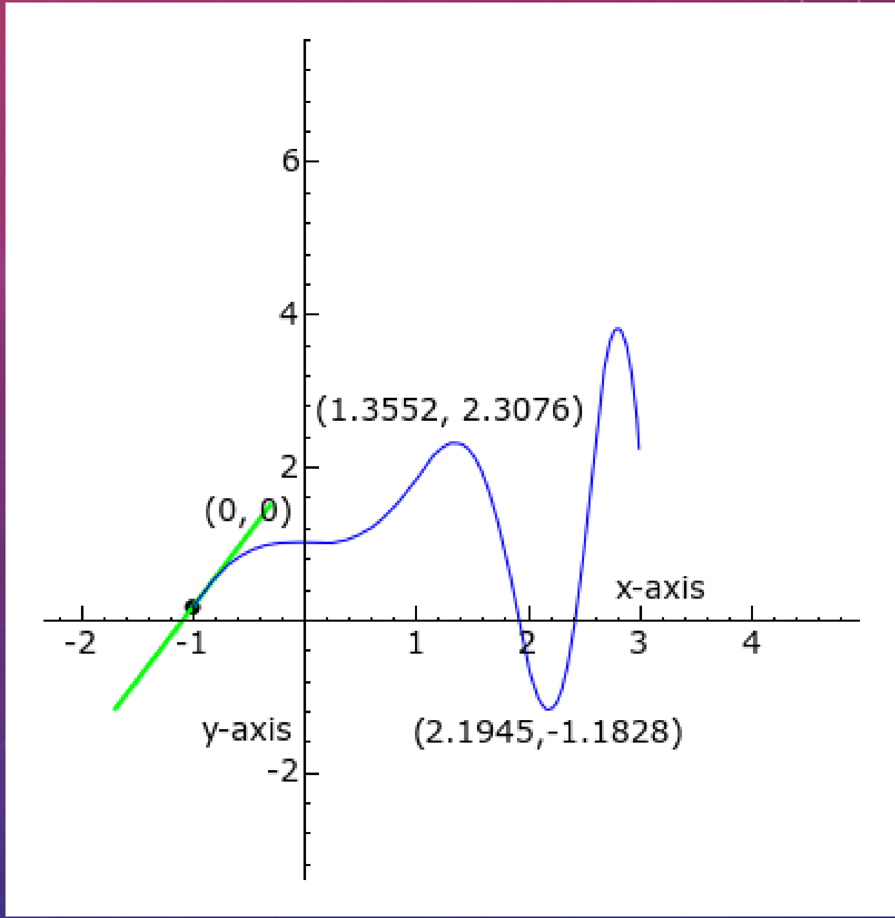
$$\overset{-}{w_{t+1}} = \overset{+}{w_t} + \overset{-}{y_t x_t}$$



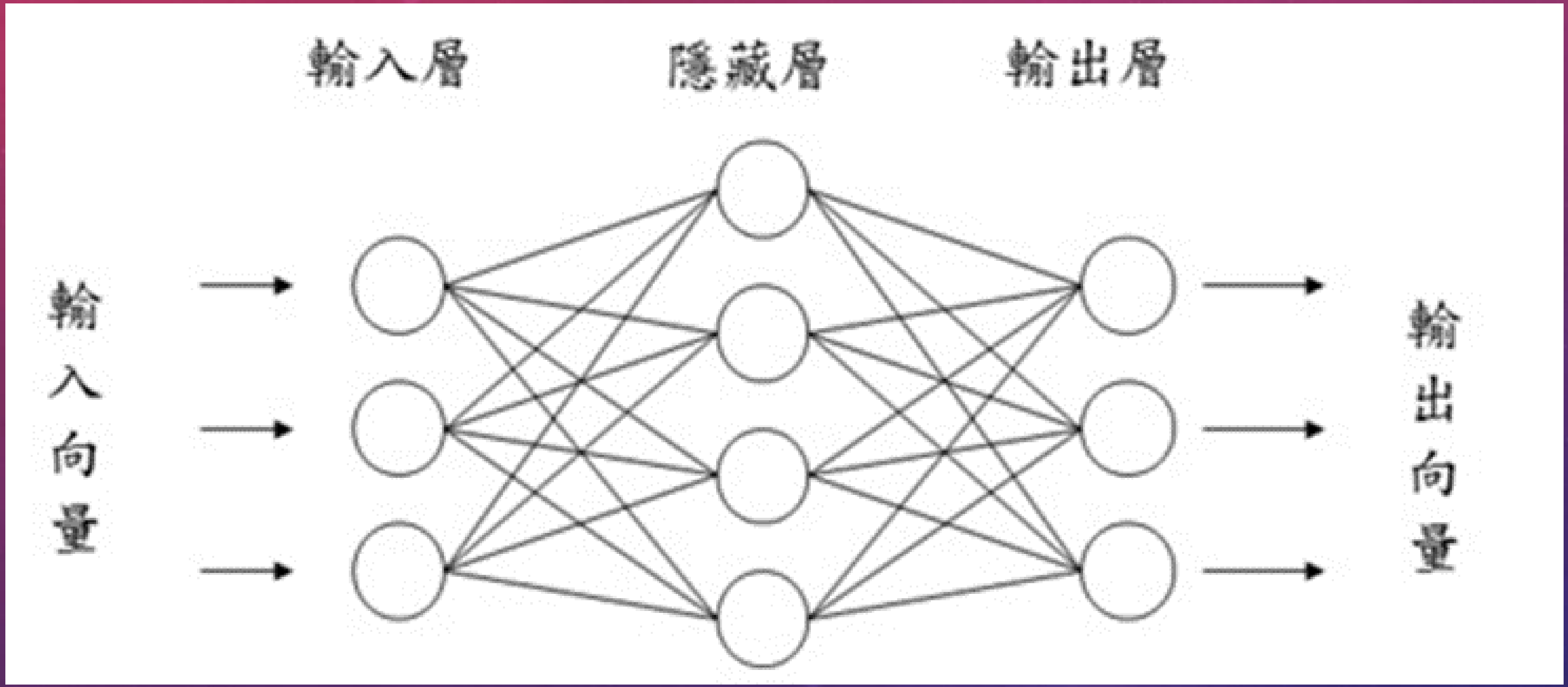
[Case 2]
y = -1 錯分成 y = 1

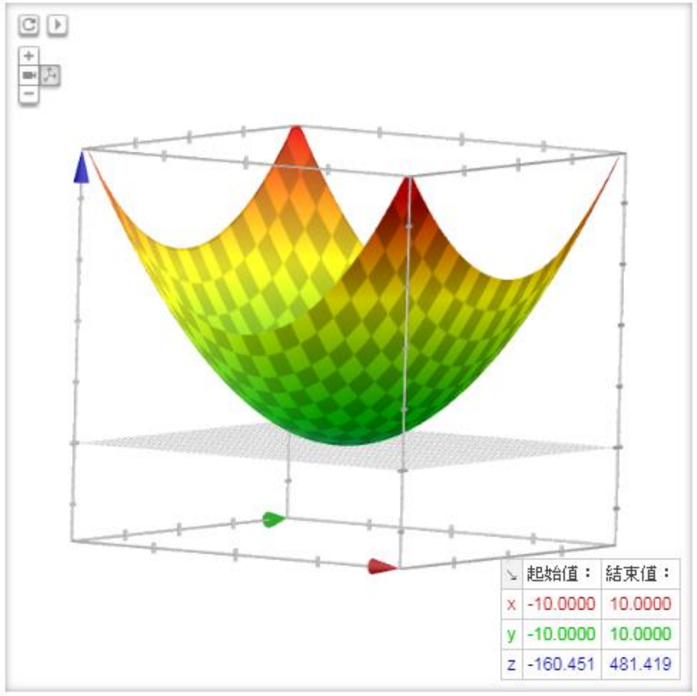


$$\tan\alpha = \lim_{\Delta x \to 0} \tan\varphi = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x) - f(x_0)}{\Delta x}$$
y
P
P_0
Δy
Δx
α
φ
O
T
x_0
$x_0+\Delta x$
x



6
4
2
-2
(1.3552, 2.3076)
(0, 0)
x-axis
-2
-1
1
2
3
4
y-axis
(2.1945,-1.1828)


輸入層
隱藏層
輸出層
輸入向量
輸出向量


起始值：
結束值：
x -10.0000 10.0000
y -10.0000 10.0000
z -160.451 481.419

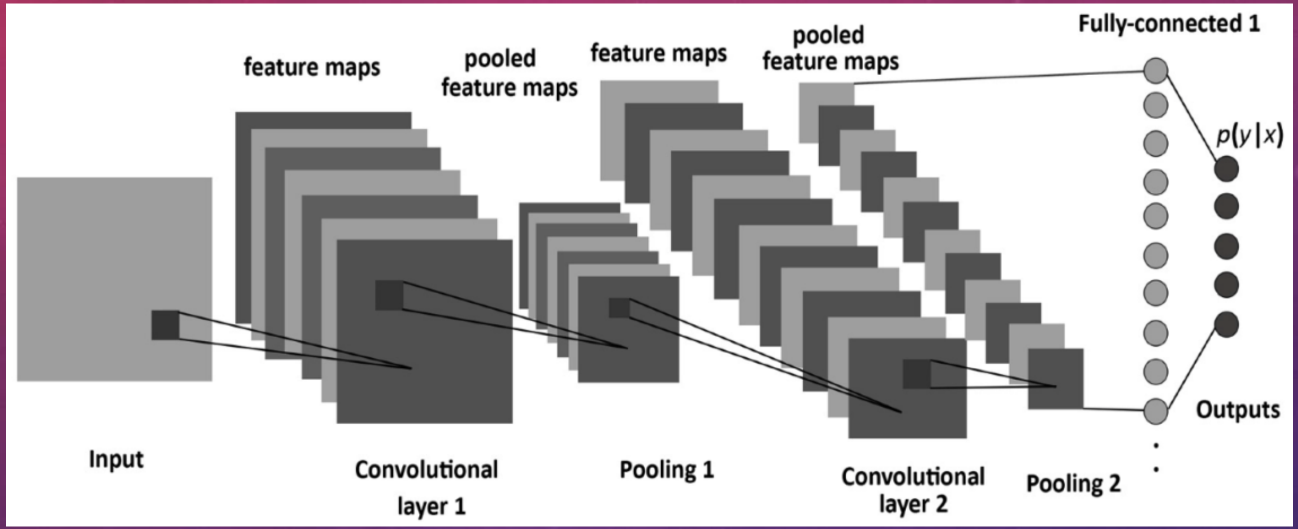
卷積神經網路 CONVOLUTIONAL NEURAL NETWORK

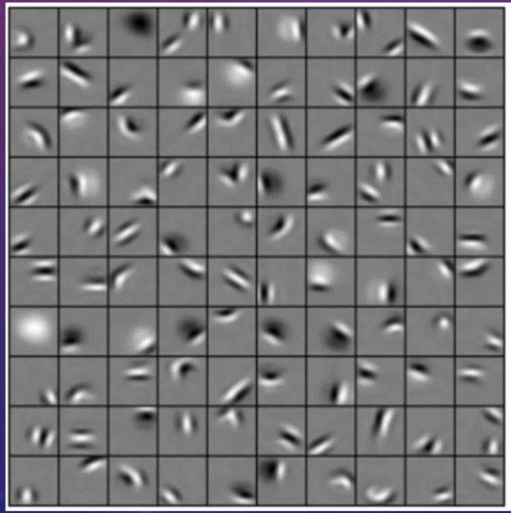
參考文獻

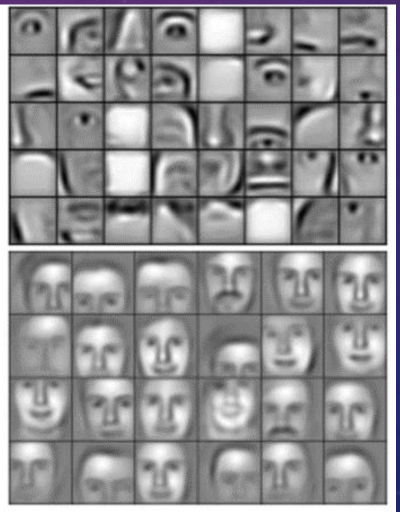
HTTPS://GOO.GL/FJVJG1

HTTPS://GOO.GL/Q5YKPK

CONVOLUTIONAL NEURAL NETWORK





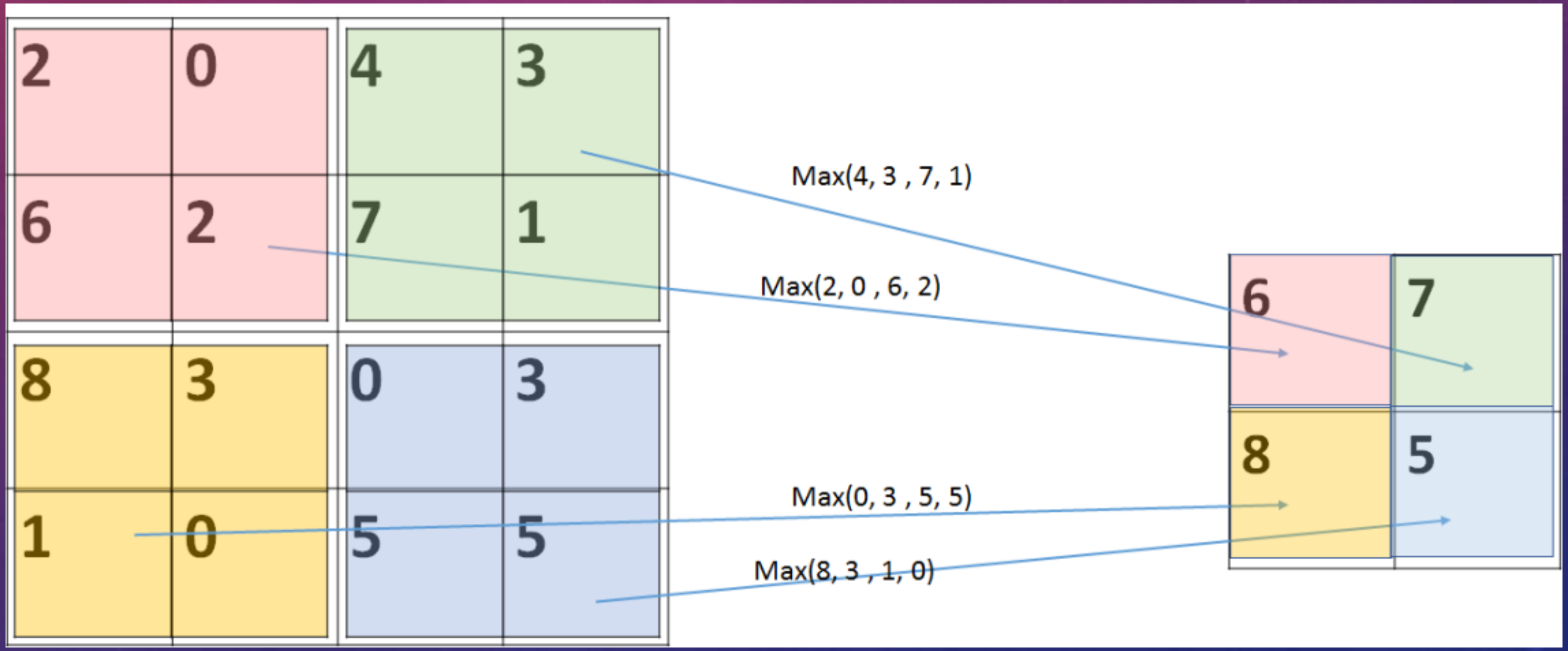


CONVOLUTIONAL NEURAL NETWORK

卷積層的運作

CONVOLUTIONAL NEURAL NETWORK

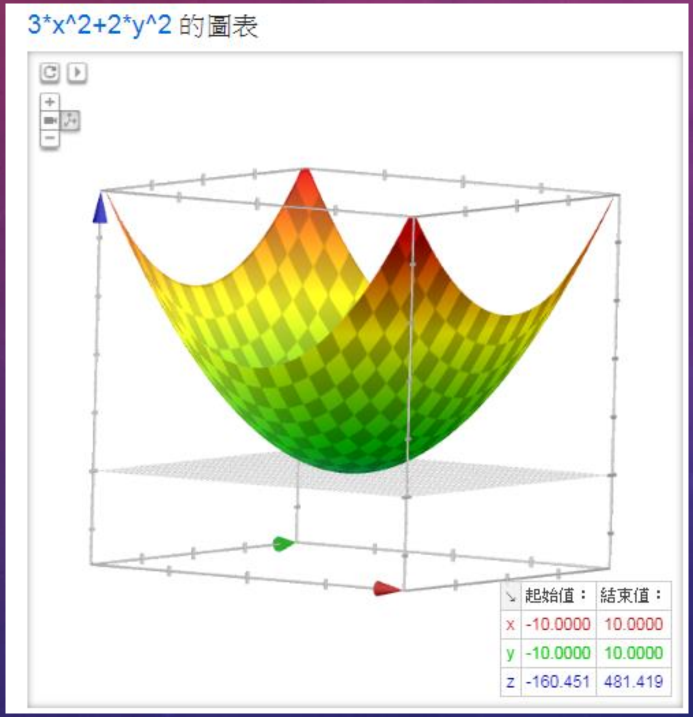
最大池化層的運作

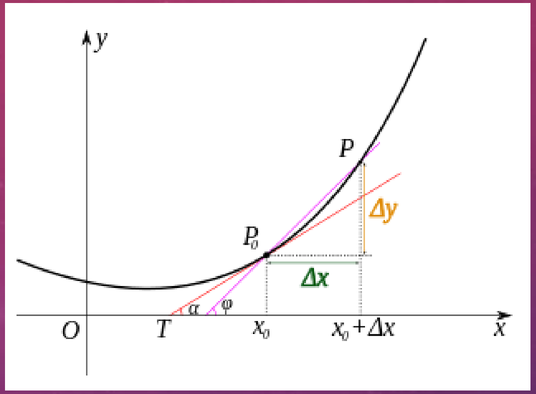


CONVOLUTIONAL NEURAL NETWORK

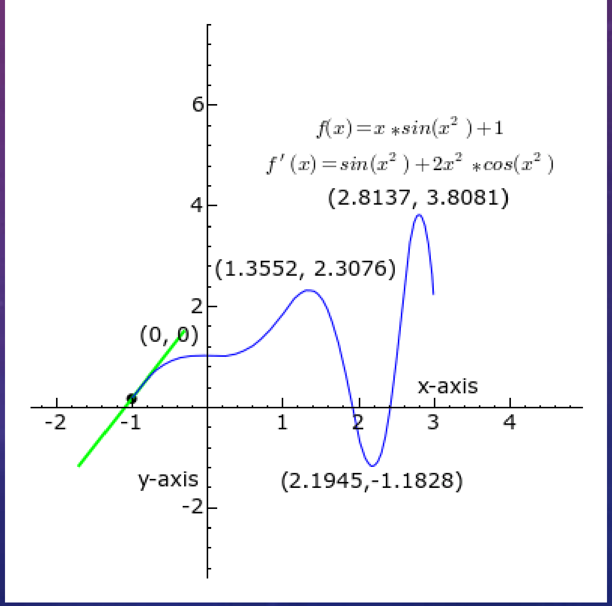
全連接層的運作

$3x^2+2y^2$

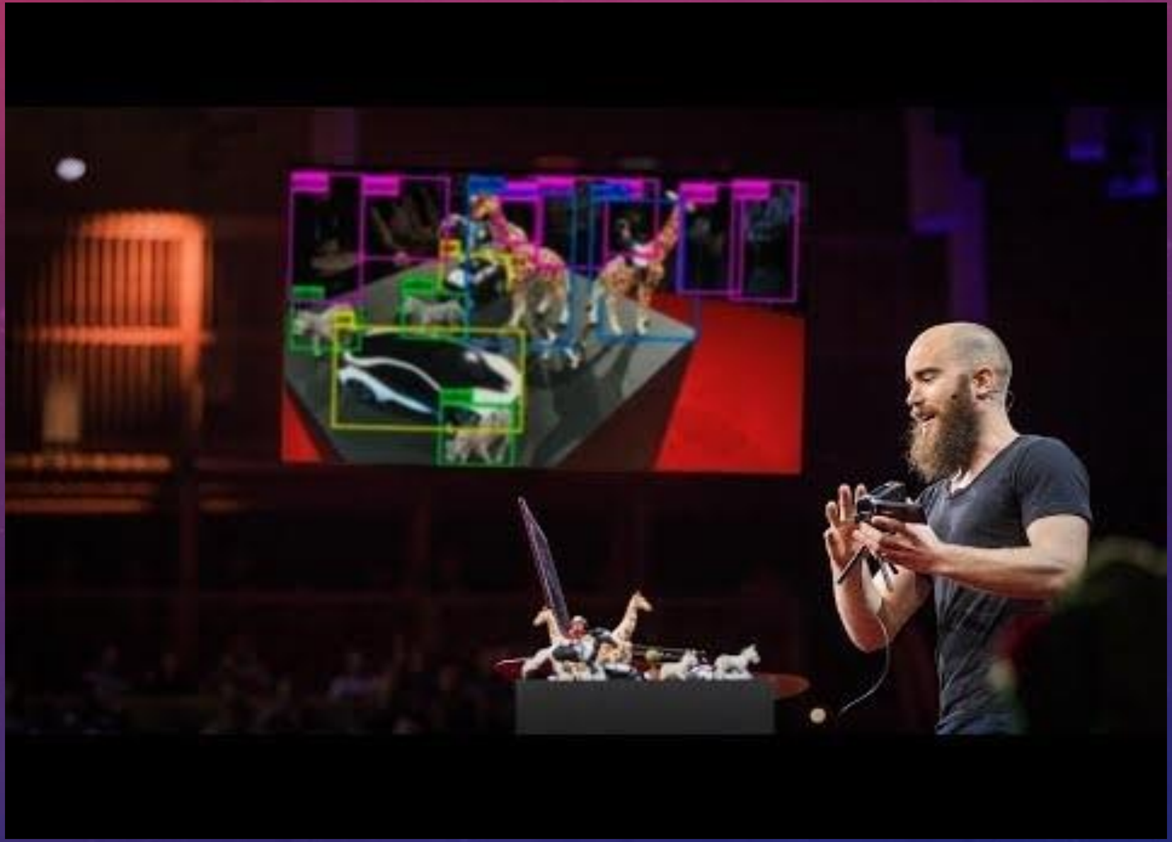




$$\tan\alpha = \lim_{\Delta x \to 0} \tan\varphi = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$$



HOW COMPUTERS LEARN TO RECOGNIZE OBJECTS INSTANTLY

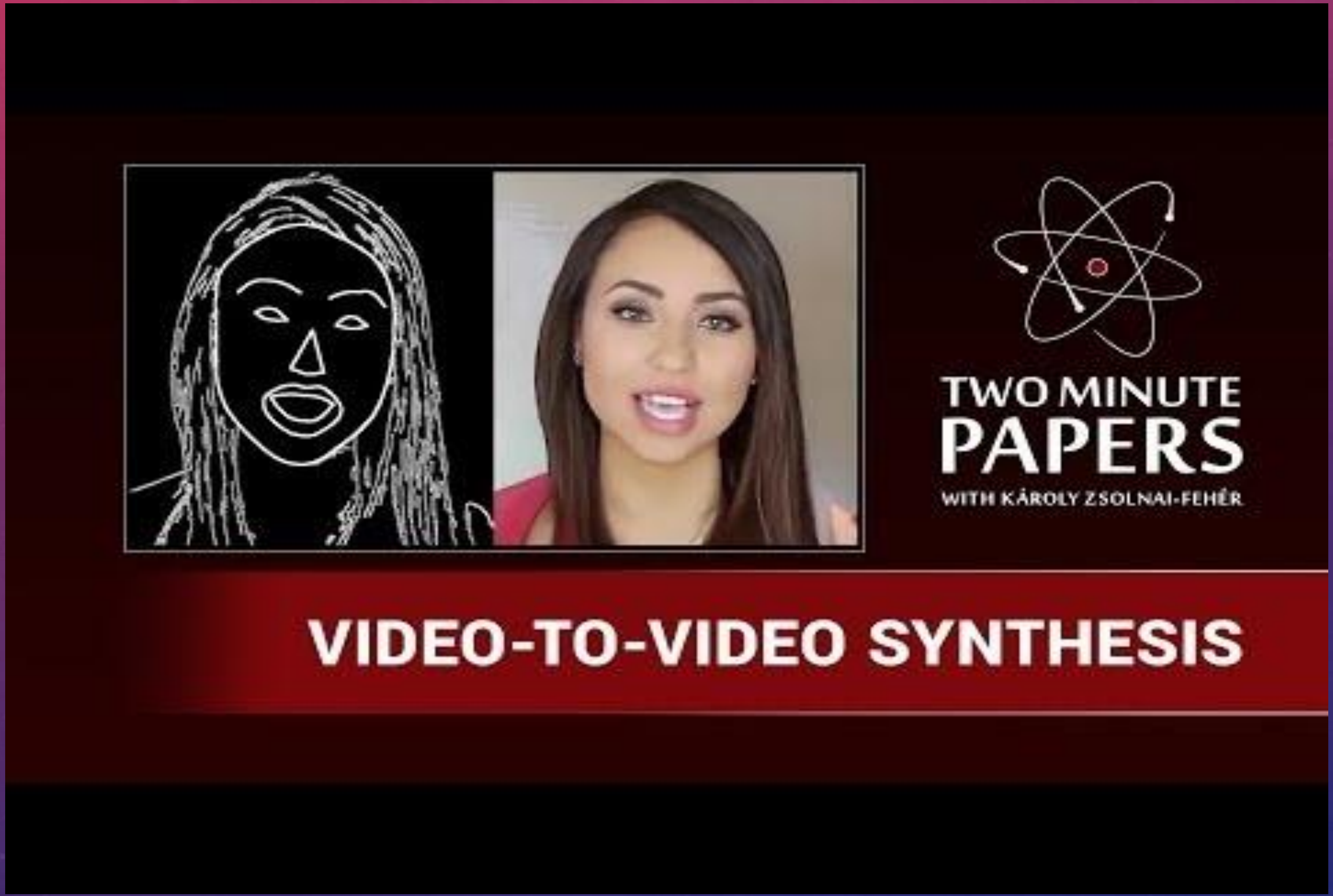


https://youtu.be/Cgxsv1riJhI

http://mropengate.blogspot.com/2018/06/yolo-yolov3.html

實際案例

VIDEO TO VIDEO



The paper "Video-to-Video Synthesis" and its source code is available here:
https://tcwang0509.github.io/vid2vid/
https://github.com/NVIDIA/vid2vid

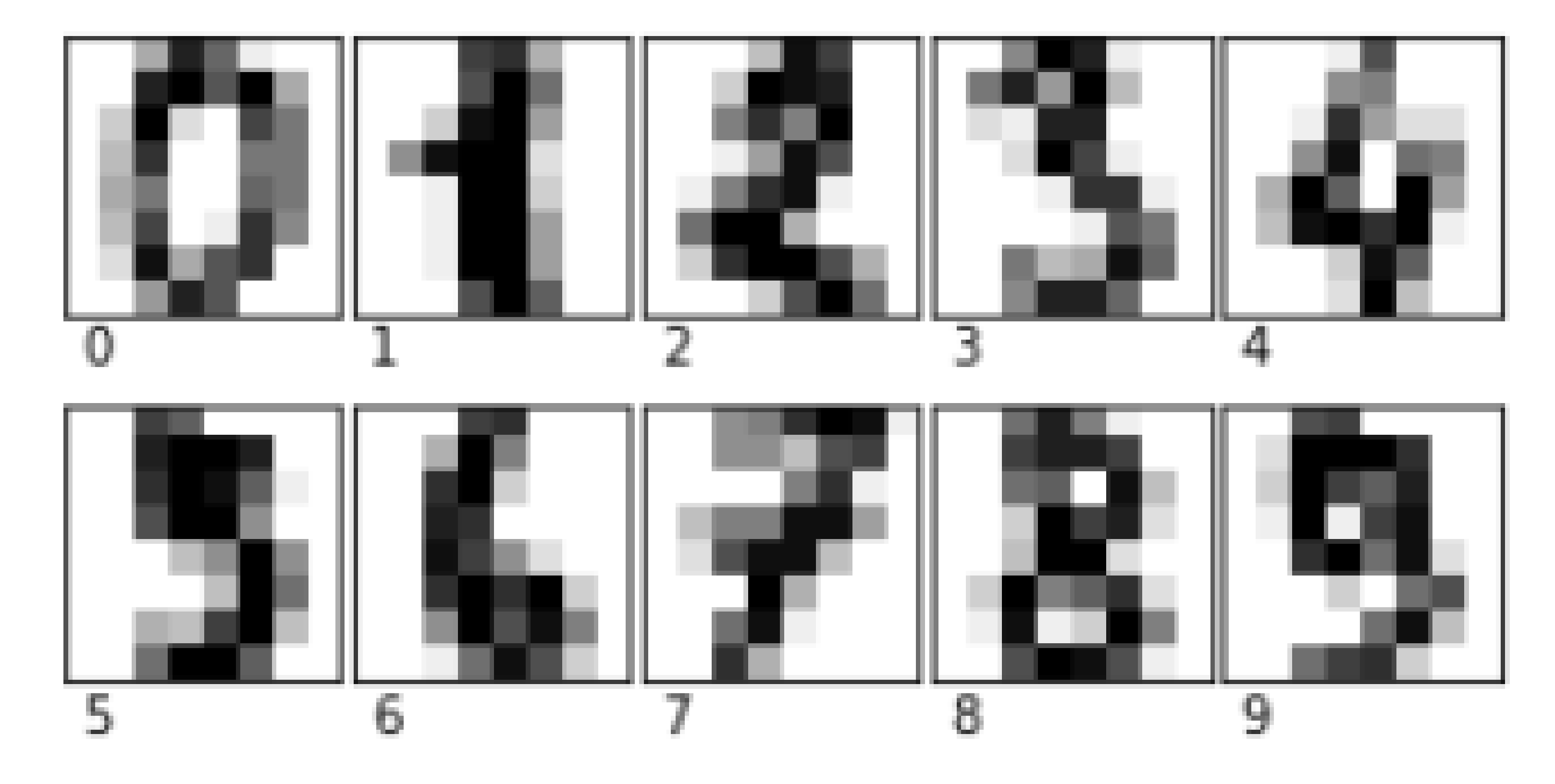
視覺整合信號轉換的體感操控機械手臂

https://youtu.be/s-rRrXRH-ks

實作練習

數字辨識

手寫數字辨識



以 SKLEARN 實作 LINEARSVC

```python
from sklearn.datasets import load_digits  # 載入預設手寫資料庫
from sklearn.model_selection import train_test_split  # 切割資料為訓練與測試集
from sklearn.preprocessing import StandardScaler  # 標準化
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report  # 預測結果的分析工具
import matplotlib.pyplot as plt

# 載入 `digits` --- Step 1. 圖像取得
digits = load_digits()

# 設定圖形的大小（寬, 高）
fig = plt.figure(figsize=(4, 2))

# 調整子圖形
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

# 把前 8 個手寫數字顯示在子圖形
for i in range(10):
    # 在 2 x 4 網格中第 i + 1 個位置繪製子圖形，並且關掉座標軸刻度
    ax = fig.add_subplot(2, 5, i + 1, xticks = [], yticks = [])
    # 顯示圖形，色彩選擇灰階
    ax.imshow(digits.images[i], cmap = plt.cm.binary) # Step 2. 預處理
    # 在左下角標示目標值
    ax.text(0, 9, str(digits.target[i]))

# 顯示圖形
plt.show();
```

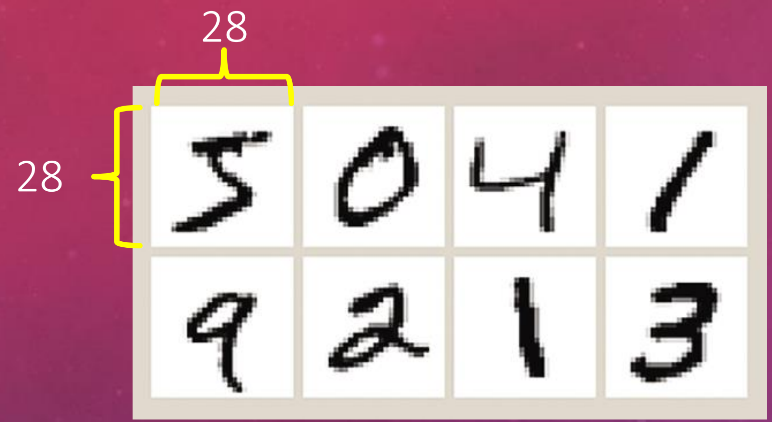
以 SKLEARN 實作 LINEARSVC

```

# 分割數據
X_train, X_test, Y_train, Y_test = train_test_split(digits.data, digits.target, test_size=0.25)

ss = StandardScaler() #標準化方法 Step 2. 預處理 (原始值-均值)/標準差
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)

lsvc = LinearSVC()
lsvc.fit(X_train, Y_train) # Step 3. & 4. 特徵提取與檢測

Y_predict = lsvc.predict(X_test) # Step 5. 分類

print (classification_report(Y_test, Y_predict, target_names=digits.target_names.astype(str))) # Step 5. 驗證
```

手寫數字辨識

	precision	recall	f1-score	support
0	1.00	0.98	0.99	41
1	0.91	0.95	0.93	42
2	0.94	0.92	0.93	37
3	0.89	0.95	0.92	44
4	0.96	1.00	0.98	49
5	0.96	0.91	0.93	47
6	0.98	1.00	0.99	45
7	0.98	0.98	0.98	60
8	0.87	0.85	0.86	40
9	0.98	0.91	0.94	45
avg / total	0.95	0.95	0.95	450

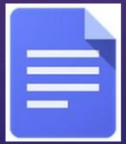
MNIST 手寫數字辨識

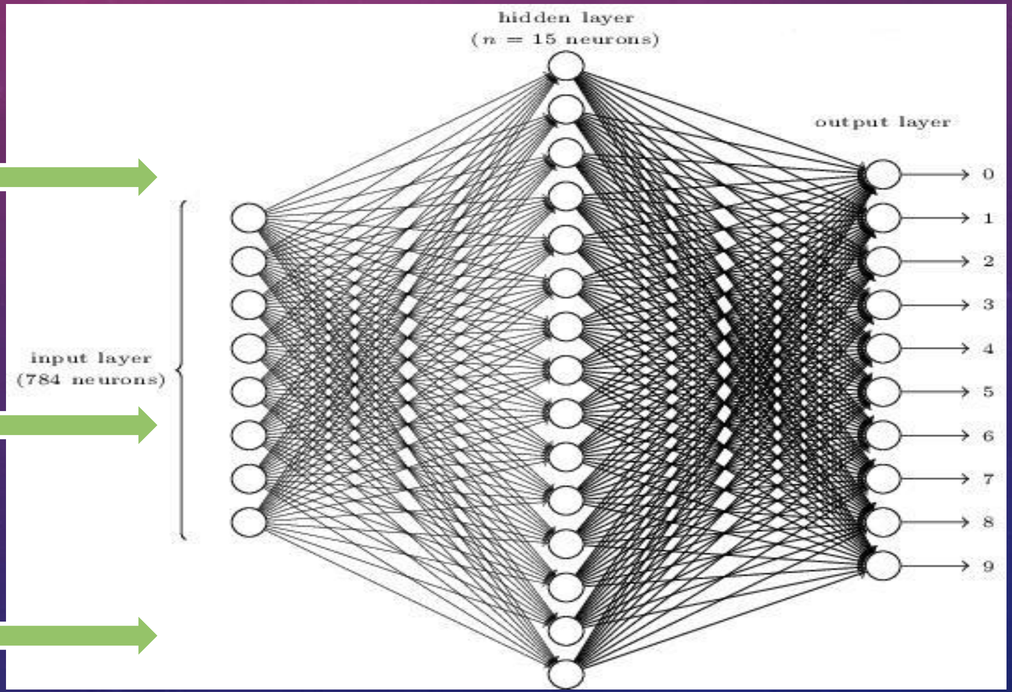


訓練資料

驗證資料

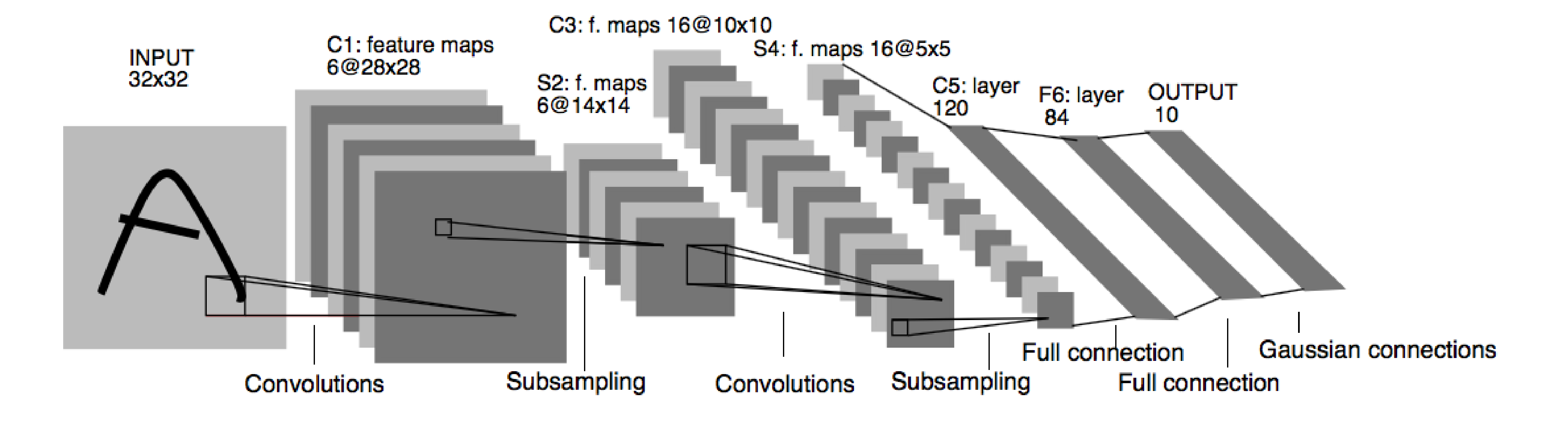
實際資料

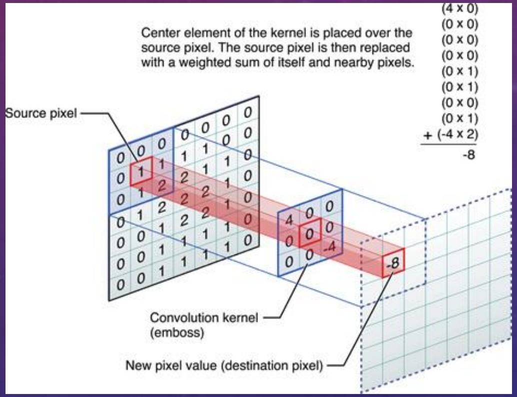




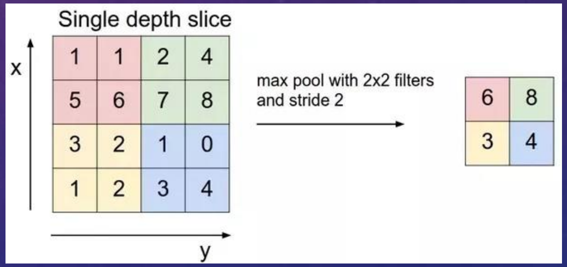
以最簡單的類神經網路架構，可達 **91%** 辨識率。若使用CNN則可高達 **99%** 辨識率。

卷積類神經網路





Convolution



Max Pooing

以 KERAS 實作 CNN

- Step 1. 載入必要函式庫

```
import numpy as np
import matplotlib.pyplot as plt

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import Conv2D, MaxPool2D, Flatten
from keras.utils import np_utils
```

- Step 2. 下載 MNIST 數據

```
nb_classes = 10
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(type(x_train))
print("x_train shape", x_train.shape)
print("y_train shape", y_train.shape)
```

以 KERAS 實作 CNN

- Step 3. 顯示圖片

```
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(x_train[0], cmap="binary",
interpolation="none")
plt.title("image" + str(y_train[0]))
plt.subplot(2,1,2)
plt.hist(x_train[0].reshape(784))
plt.title("Pixel Values")
plt.show()
```

- Step 4. 準備訓練資料

```
img_size_x, img_size_y = 28, 28
x_train = x_train.reshape(x_train.shape[0], img_size_x, img_size_y, 1)
x_test = x_test.reshape(x_test.shape[0], img_size_x, img_size_y, 1)
input_shape = (img_size_x, img_size_y, 1)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
x_train /= 255
x_test /= 255
```

以 KERAS 實作 CNN

- Step 5. 轉換為 One hot encoding

```
y_train = np_utils.to_categorical(y_train,nb_classes)
y_test = np_utils.to_categorical(y_test,nb_classes)
```

- Step 6. 定義類神經網路模型

Sequential可以讓我們按照順序將神經網路路串串起。深度學習為隱藏層有兩兩層或兩兩層以上.

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation="relu",
input_shape=input_shape))
model.add(Conv2D(64, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))
```

Loss:
https://keras.io/losses/
Optimizer:
https://keras.io/optimizers/

- Step 7. Compile

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

以 KERAS 實作 CNN

- Step 8. 訓練模型

```
history = model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=2,
validation_data=(x_test, y_test))
```

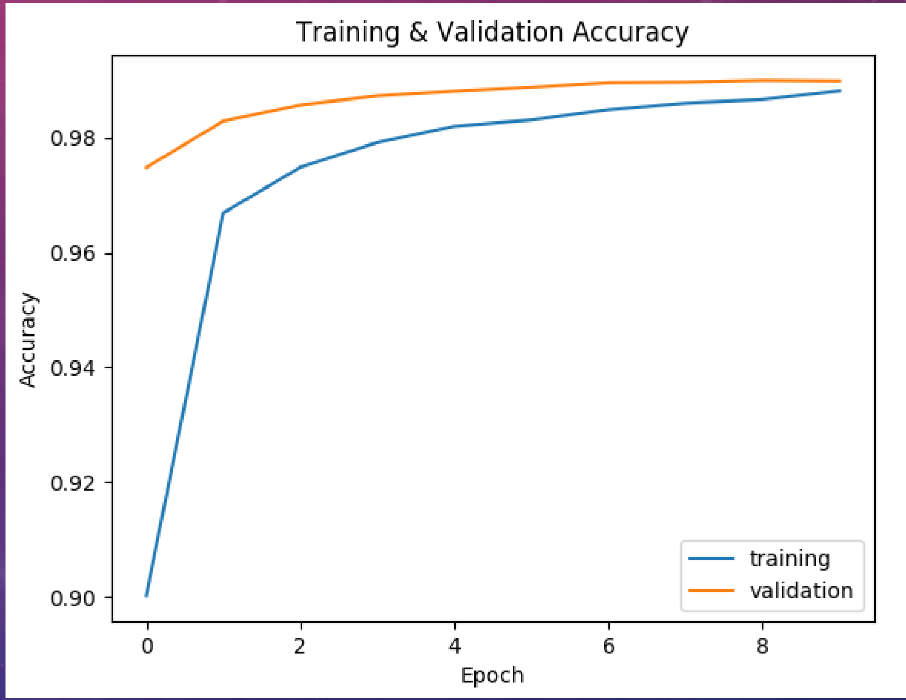
```
Epoch 1/10
125s - loss: 0.3322 - acc: 0.9002 - val_loss: 0.0789 - val_acc: 0.9748
Epoch 2/10
121s - loss: 0.1125 - acc: 0.9669 - val_loss: 0.0519 - val_acc: 0.9829
Epoch 3/10
123s - loss: 0.0844 - acc: 0.9748 - val_loss: 0.0424 - val_acc: 0.9857
Epoch 4/10
127s - loss: 0.0714 - acc: 0.9792 - val_loss: 0.0378 - val_acc: 0.9873
Epoch 5/10
124s - loss: 0.0617 - acc: 0.9820 - val_loss: 0.0364 - val_acc: 0.9881
Epoch 6/10
123s - loss: 0.0570 - acc: 0.9831 - val_loss: 0.0308 - val_acc: 0.9888
Epoch 7/10
124s - loss: 0.0506 - acc: 0.9849 - val_loss: 0.0294 - val_acc: 0.9896
Epoch 8/10
125s - loss: 0.0466 - acc: 0.9860 - val_loss: 0.0291 - val_acc: 0.9897
Epoch 9/10
124s - loss: 0.0441 - acc: 0.9867 - val_loss: 0.0286 - val_acc: 0.9900
Epoch 10/10
123s - loss: 0.0396 - acc: 0.9881 - val_loss: 0.0300 - val_acc: 0.9899
```

From 98% to 99%

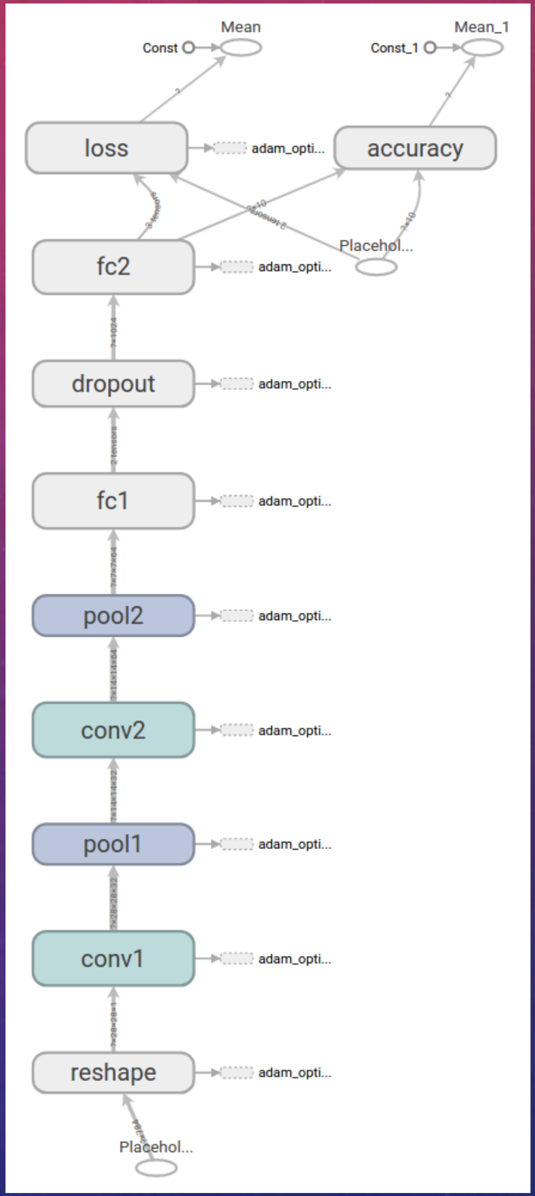
以 KERAS 實作 CNN

- Step 9. 檢查準確度

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training data')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['training','validation'],loc='lower right')
plt.show()
```



以 TENSORFLOW 實作 CNN



以 TENSORFLOW 實作 CNN

- Step 1. 準備副程式

Create tensor of shape, and the weights are normal-distribution. the input is the kernel filter size [height, width , channel, number]

```
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

strides stands for the moving of the kernel filter. [batch, height, width, channel]. padding='SAME' means the output shape = input shape

以 TENSORFLOW 實作 CNN

- Step 2. 載入數據並準備 PlaceHolder

```
# Load MNIST Data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

# Start TensorFlow InteractiveSession
sess = tf.InteractiveSession()

x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

以 TENSORFLOW 實作 CNN

- Step 3. 建立 Computation Graph

the input is the kernel filter size [height, width , channel, number]

```
# First Convolutional Layer
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1, 28, 28, 1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)


# Second Convolutional Layer
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

```
# Densely Connected Layer
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

# Dropout
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# Readout Layer
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
```

以 TENSORFLOW 實作 CNN

- Step 4. 開始訓練並測試準確度

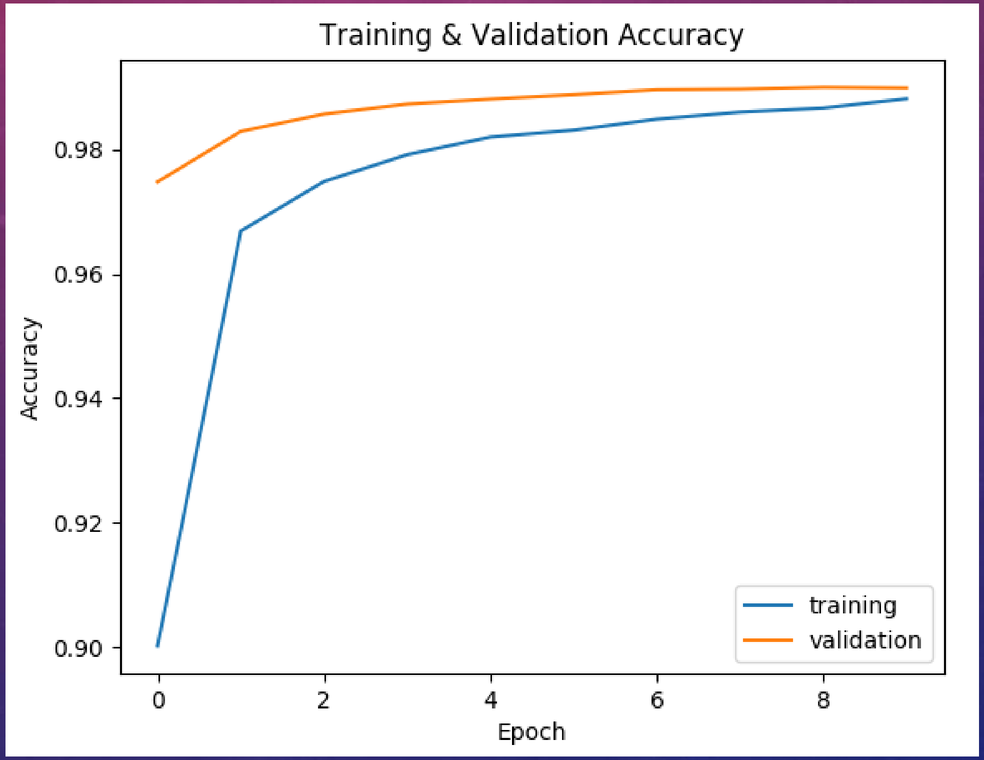
```python
# Train and Evaluate the Model
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(20000):
        batch = mnist.train.next_batch(50)
        if i % 100 == 0:
            train_accuracy = accuracy.eval(feed_dict={x: batch[0], y_: batch[1], keep_prob: 1.0})
            print('step %d, training accuracy %g' % (i, train_accuracy))
        train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

    print('test accuracy %g' % accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

tf.argmax:
when axis=0, it returns the max value row index each column
when axis=1, it returns the max value column index each row

tf.reduce_man: 計算平均值
'x' is [[1., 1.]
[2., 2.]]
tf.reduce_mean(x) ==> 1.5
tf.reduce_mean(x, 0) ==> [1.5, 1.5]
tf.reduce_mean(x, 1) ==> [1., 2.]

CNN模型準確度

CNN (2 Layers & Dropout)

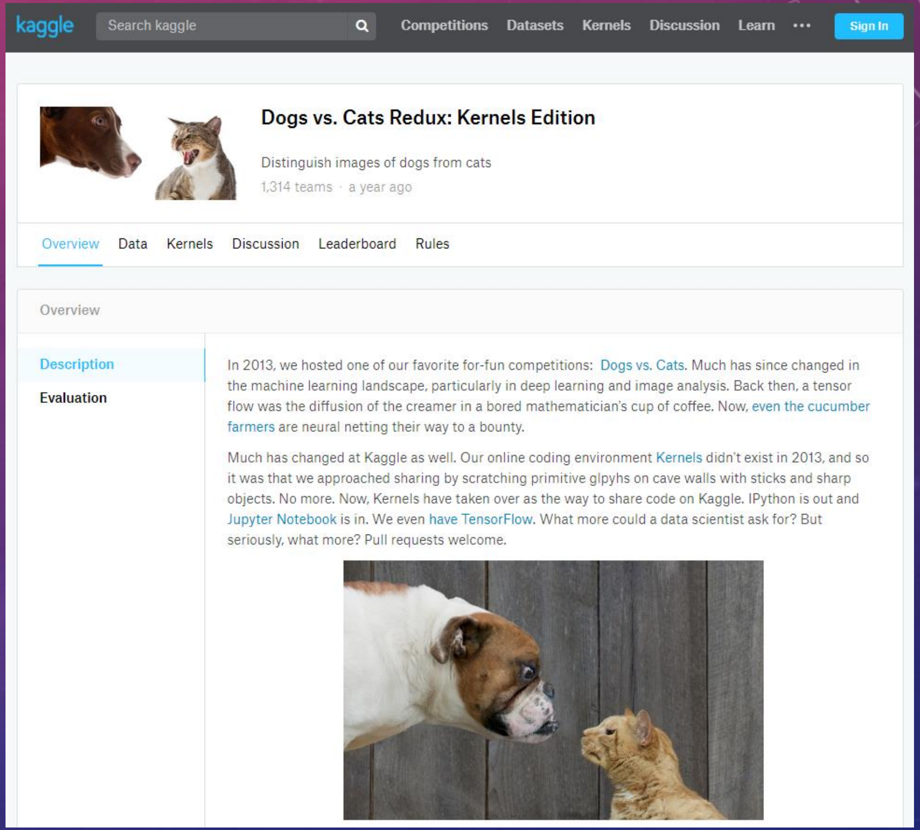


實作練習

貓狗辨識

DOGS VS. CATS

- https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
  - Training data: 25000 images
  - Test data: 12500 images
  - For each image in the test set, you should predict a probability that the image is a dog (1 = dog, 0 = cat).



貓狗辨識的結果

```
[0.14723705 0.98256    ]
[0.04469623 0.99915826]
[0.99998796 0.00834211]
[0.99993527 0.01661933]
[0.99877125 0.05415697]
[0.43036035 0.6431105 ]
[0.20522958 0.95668554]
[0.8880429  0.28058085]
[0.08643436 0.9956601 ]
[0.99908304 0.04826429]
[0.9999168  0.01841162]
[0.1397599  0.98483086]
[0.99558544 0.08876048]
[0.43750185 0.6269166 ]
[0.10178897 0.9934009 ]
[0.10858331 0.9922093 ]
```

THANK YOU