

The background features a dark blue-to-purple gradient with abstract white and yellow circular patterns. These patterns include concentric circles, arcs, and degree markings (40, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) that suggest a technical or scientific theme.

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



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斷層掃描，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）。

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

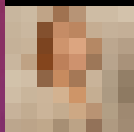
- 利用以下步驟對每張圖片生成“指紋”（fingerprint）字符串來比較不同圖片的指紋。
 1. 縮小尺寸：只保留圖片的結構，明暗等基本資訊，避免不同尺寸的差異。如縮成 8×8 。
 2. 簡化色彩：將圖片轉為64級灰度，所有像素點總共只有64種顏色。
 3. 計算平均值：計算所有64個像素的灰度平均值。
 4. 比較像素的灰度：將每個像素的灰度與平均值比較， \geq 平均值，記為1、 $<$ 平均值，記為0。
 5. 計算 Hash 值：這64位的整數組合，就是圖片的指紋。組合的次序並不重要，只要保證所有圖片都採用同樣次序就行了，在此以 Hash 加密方法來組成。
 - 雜湊演算法(Hash Algorithm)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)」。

PERCEPTUAL HASH ALGORITHM

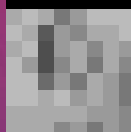
原圖



1. 縮小 (8*8)



2. 簡化色彩 (64級灰度)



8級灰度



32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

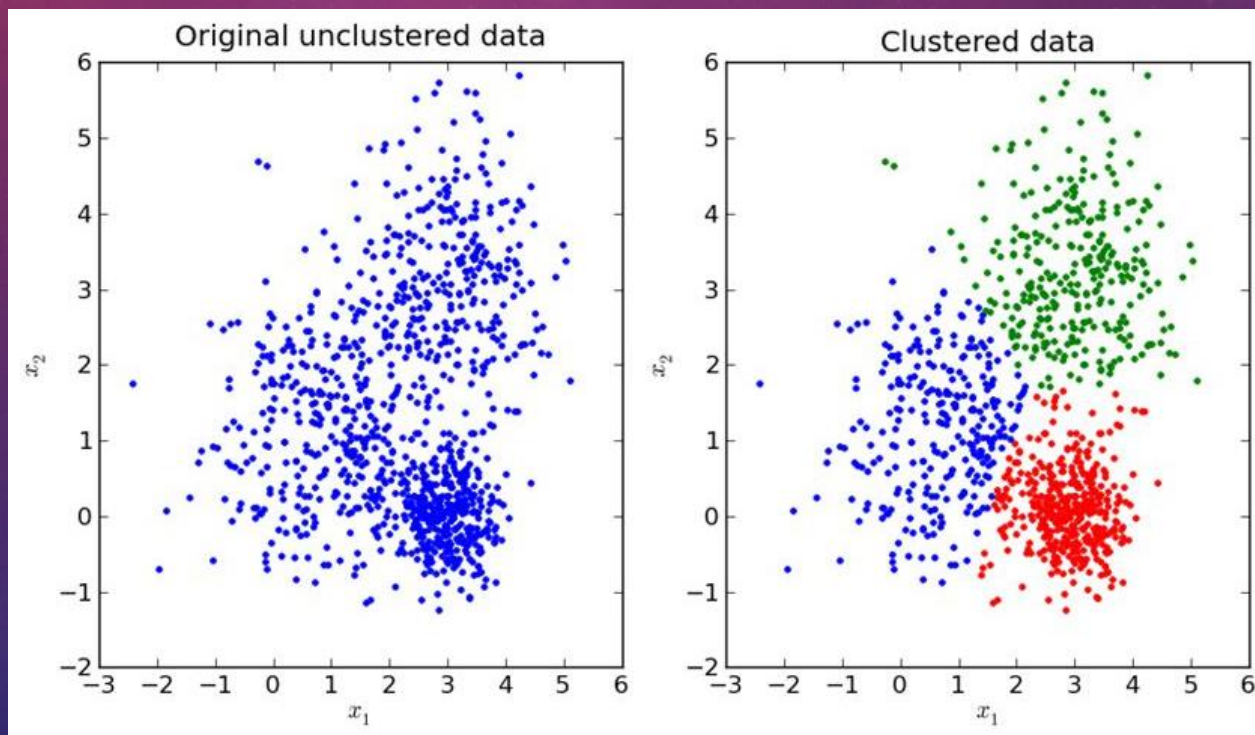
- 有序性規則的資料

DATA MINING

ResponseId	StartDate	CompletedDate	LanguageCode	Question1	Question2	Question3	Question4	Question5	Question6	Question7	Question8
27357	2006.11.27	2006.11.27	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	2006.11.27	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	2006.11.27	en	Lithuania	Retail	6 - 12 months	6-10	4,1,4,"this is a random other text"	2,2,2,2	Opinio	1,0,0,1
27361	2006.11.27	2006.11.27	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	2006.11.27	en	Djibouti	Manufacturing	6+ years	101-250	0,4,0,"another random text"	5,5,5,5	Opinio	1
27363	2006.11.27	2006.11.27	en	Tanzania	Retail	1 - 2 years	1001-5000	1,1,1,"123456"	2,2,2,2	Opinio	0,1,1,1,1,1,1,1,1
27364	2006.11.27	2006.11.27	en	Vanuatu	Other	1 - 2 years	1001-5000	6,5,6,"123456"	6,6,6,6	Opinio	0,0,1,1,1,1,0,1,1,
27365	2006.11.27	2006.11.27	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	2006.11.27	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	2006.11.27	en	Norway	Education	2 - 5 years	5001-10000	6,0,6,"f6{[]]+&#x'' '*-/\\	1,1,1,1	Opinio	1,
27368	2006.11.27	2006.11.27	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	2006.11.27	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	2006.11.27	en	Maldives	Other	6+ years	10001 or more	2,5,2,"another random text"	6,6,6,6	Network Pro	
27371	2006.11.27	2006.11.27	en	Kyrgyzstan	Medical	2 - 5 years	26-100	3,5,3,"f6{[]]+&#x'' '*-/\\	6,6,6,6	Network Pro	
27372	2006.11.27	2006.11.27	en	Antigua and Barbuda	Government	6 - 12 months	501-1000	6,2,6,"this is a random other t			
27373	2006.11.27	2006.11.27	en	Belarus	Financial Services	6+ years	10001 or more	2,1,2,"another random text"	2,2,2,2		
27374	2006.11.27	2006.11.27	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	2006.11.27	en	Georgia	Financial Services	6+ years	10001 or more	6,1,6,"another random text"	2,2,2,2		
27376	2006.11.27	2006.11.27	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	2006.11.27	en	Chad	Software Vendor	<6 months	1-5	6,2,6,"hey"	3,3,3,3	Network Probe	1,1,1,1,1,1,1,1,1
27378	2006.11.27	2006.11.27	en	Turkey	Software Vendor	6 - 12 months	501-1000	1,2,1,"this is a random other text"	3,3		
27379	2006.11.27	2006.11.27	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	2006.11.27	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	2006.11.27	en	Equatorial Guinea	Software Vendor	6+ years	101-250	6,2,6,"another random text"	3,3,3,		
27382	2006.11.27	2006.11.27	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	2006.11.27	en	French Southern and Antarctic Lands	Retail	1 - 2 years	1001-5000	2,1,2,"123456"	2,2,2		
27384	2006.11.27	2006.11.27	en	Guinea-Bissau	Hardware Vendor	2 - 5 years	26-100	6,3,6,"f6{[]]+&#x'' '*-/\\	4,4,4,4,		
27385	2006.11.27	2006.11.27	en	Viet Nam	Medical	2 - 5 years	26-100	4,5,4,"f6{[]]+&#x'' '*-/\\	6,6,6,6	Opinio	1,1,1,
27386	2006.11.27	2006.11.27	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	2006.11.27	en	Puerto Rico	Non-profit	<6 months	1-5	0,0,0,"hey"	1,1,1,1	Opinio	1,1,1,0,1,1,1

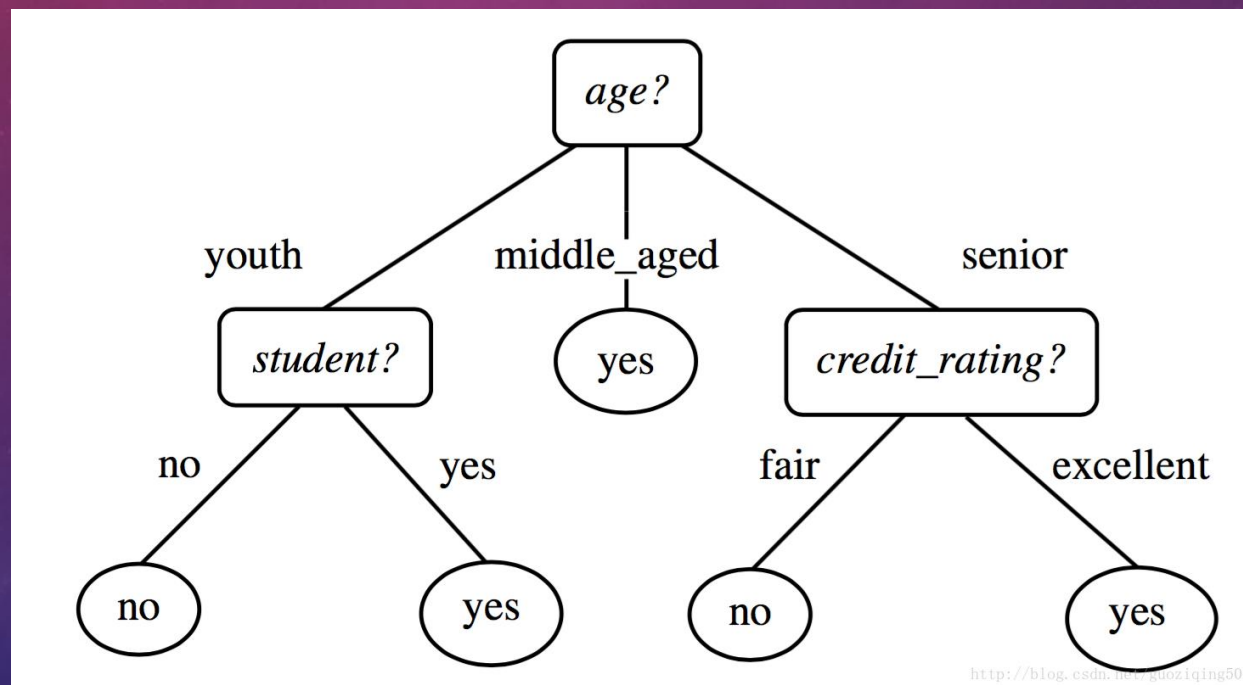
DATA MINING

- 分群
 - 用於沒有標籤的資料，又通常為非監督式演算法。



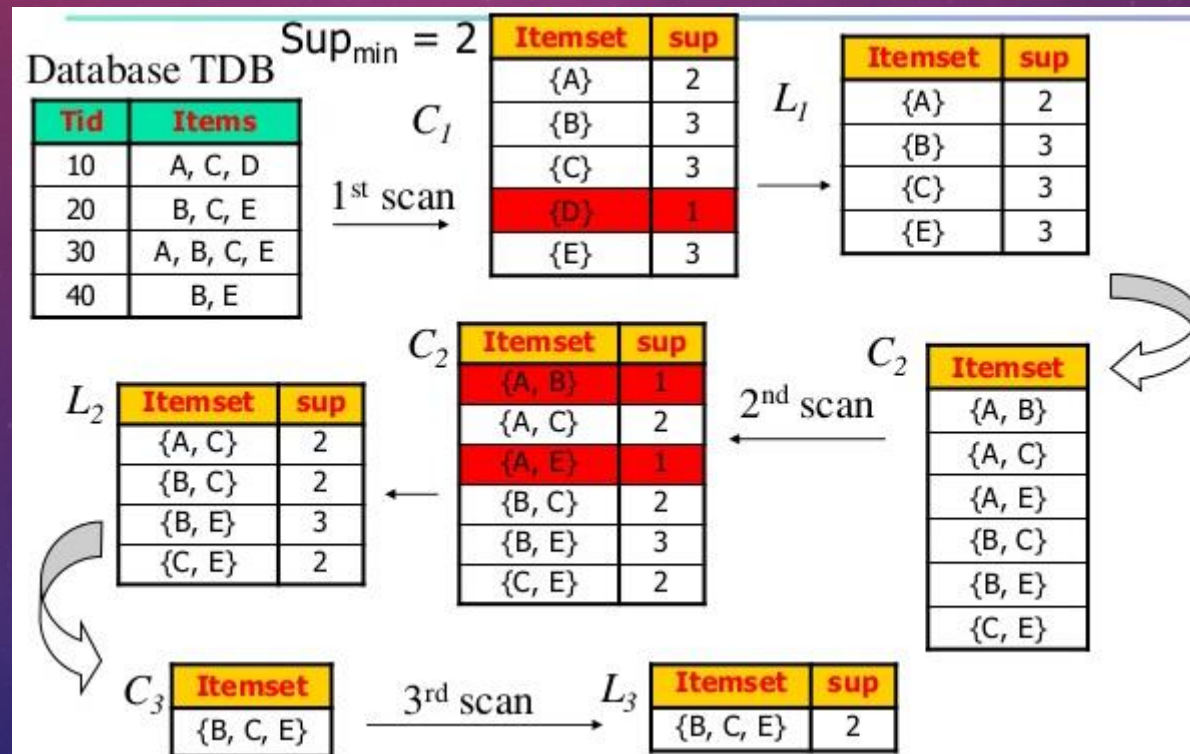
DATA MINING

- 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)

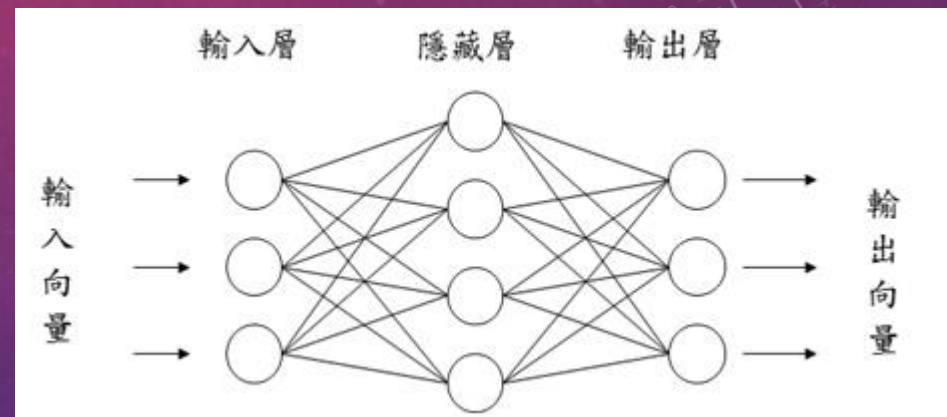
- Deep Learning

- 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				

=

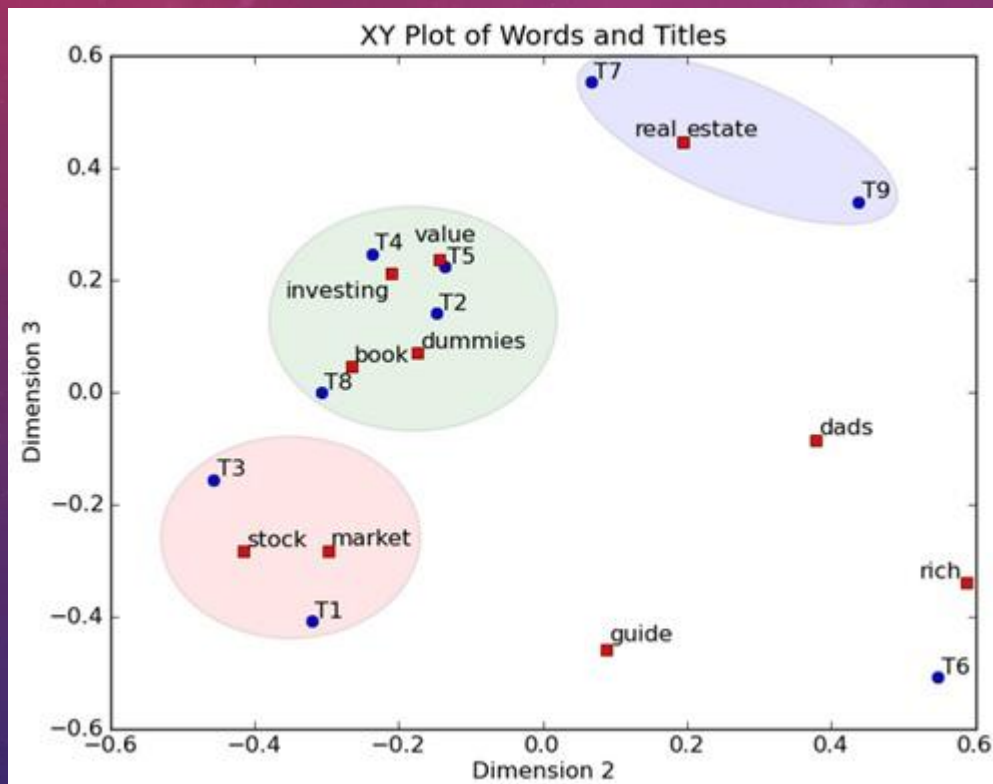
book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

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)



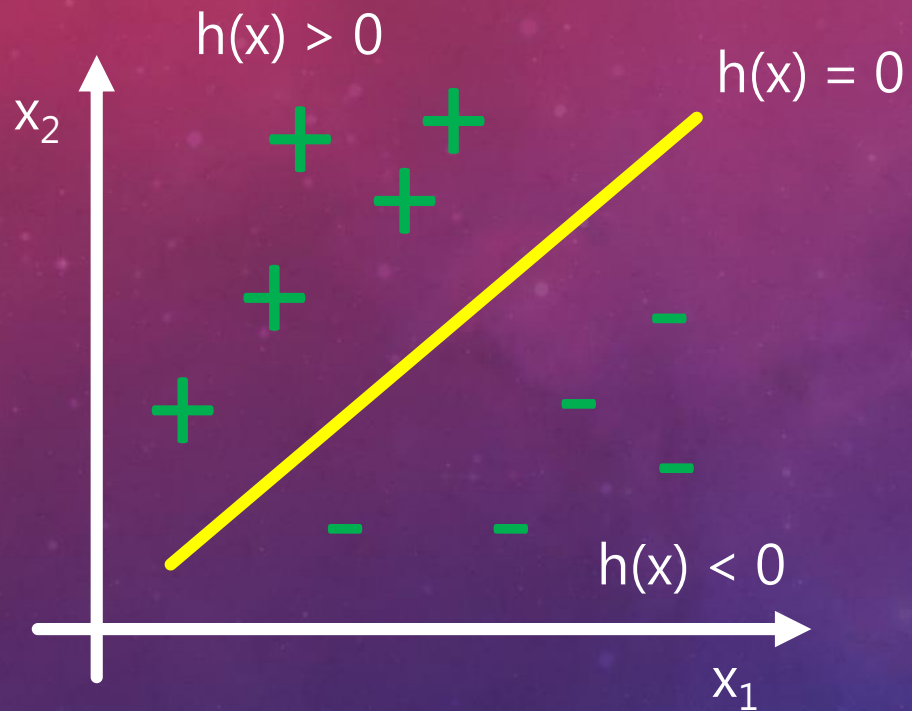
Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				



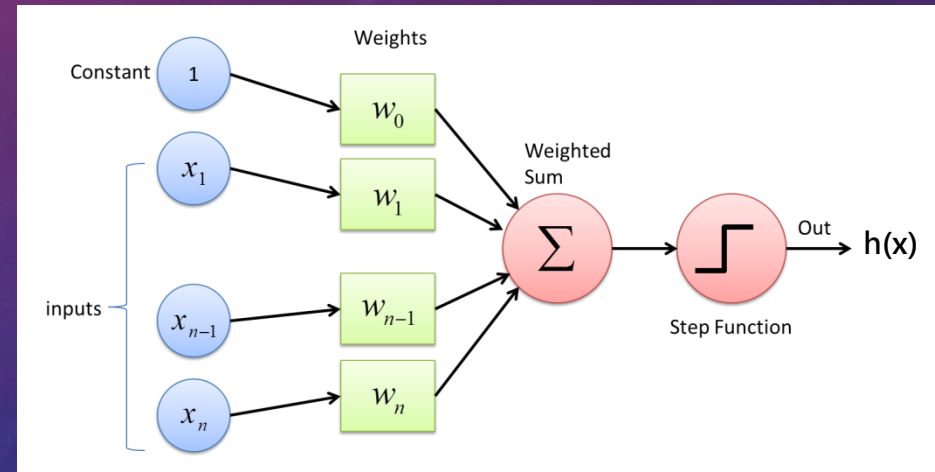
DEEP LEARNING

人工神經網路
ARTIFICIAL NEURAL NETWORK

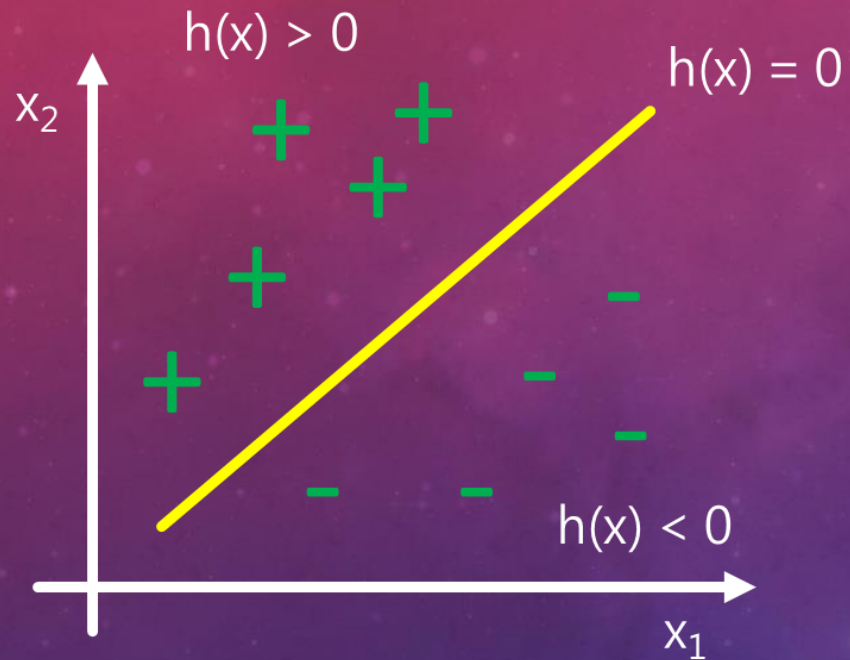
Perceptron Linear Algorithm



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



Perceptron Linear Algorithm



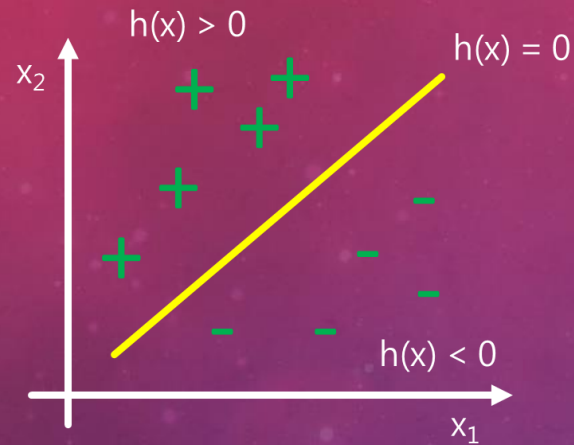
$$h(x) = w_0 + w_1x_1 + w_2x_2$$

$$scores = \sum_i^N w_i x_i + b$$

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

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

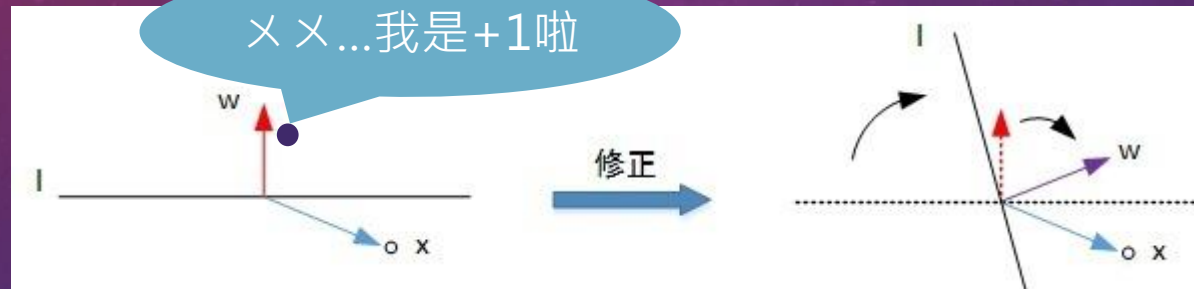
Perceptron Linear Algorithm



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

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

(Note: The '+' sign in the original image is yellow, and the '-' sign is black.)



[Case 1]
 $y = 1$ 錯分成 $y = -1$

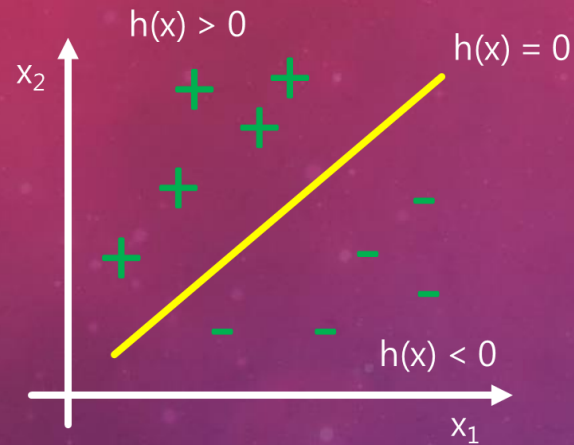
$$w_{t+1} = w_t + y_t x_t$$

(Note: The '-' sign in the original image is black, and the '+' sign is yellow.)



[Case 2]
 $y = -1$ 錯分成 $y = 1$

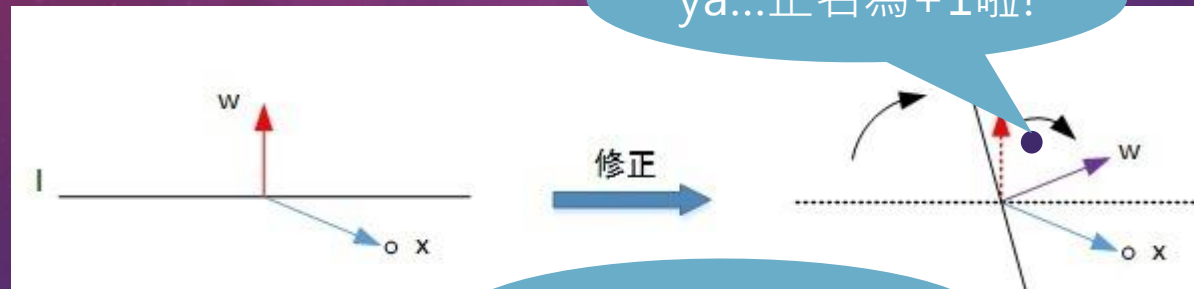
Perceptron Linear Algorithm



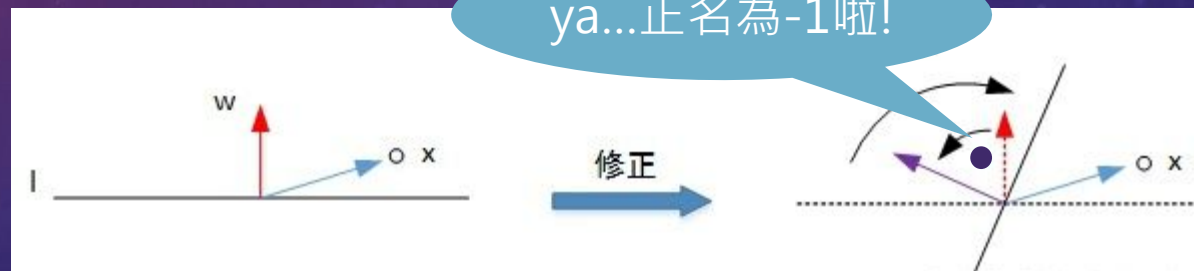
- 若 $scores \geq 0$, 則 $\hat{y} = 1$
- 若 $scores < 0$, 則 $\hat{y} = -1$

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

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

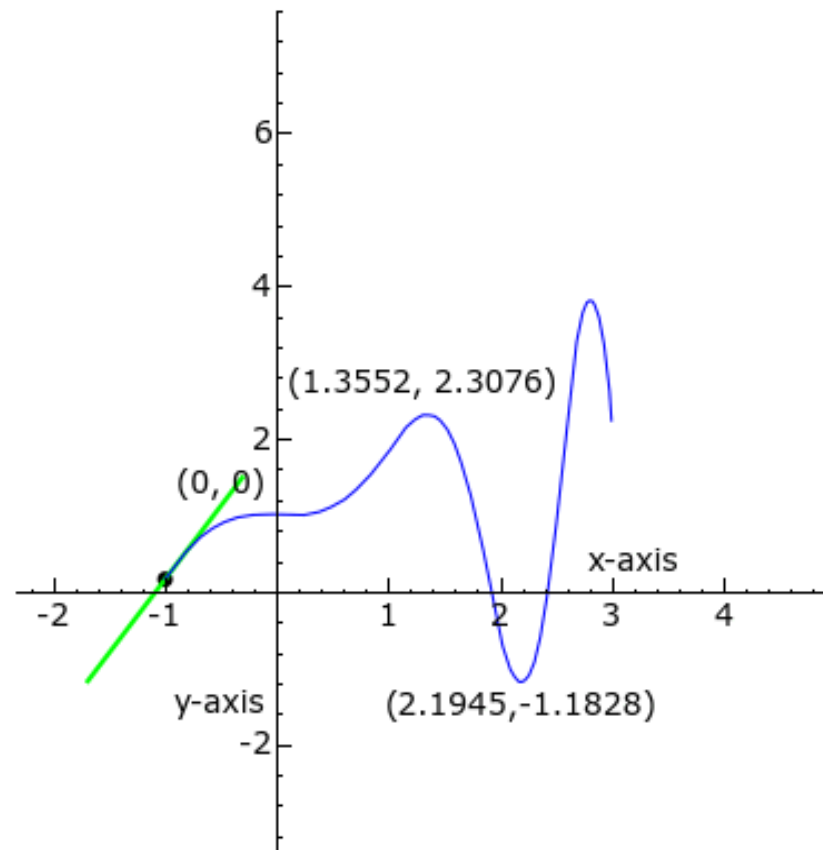
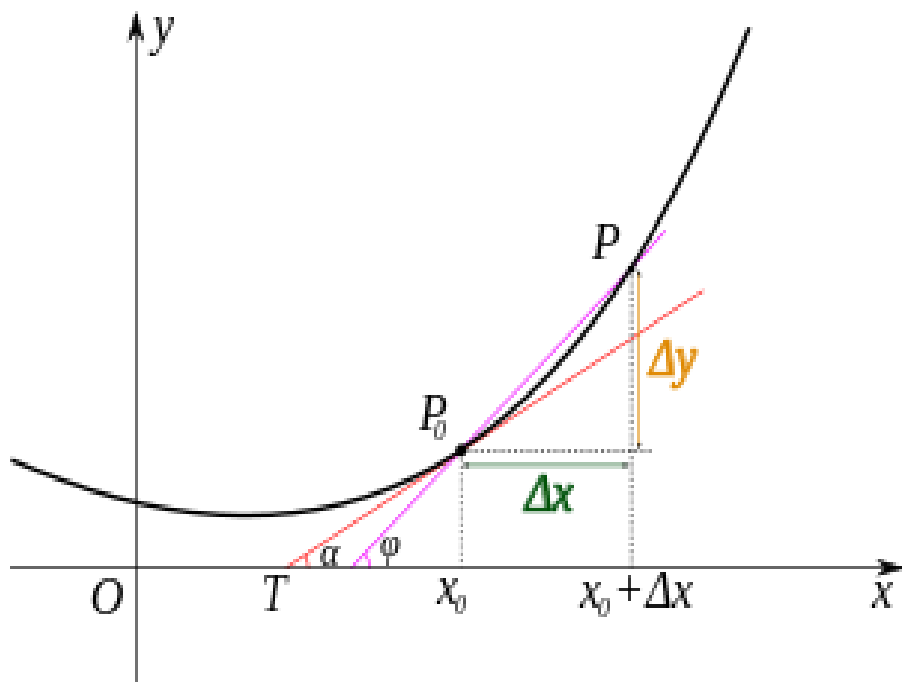


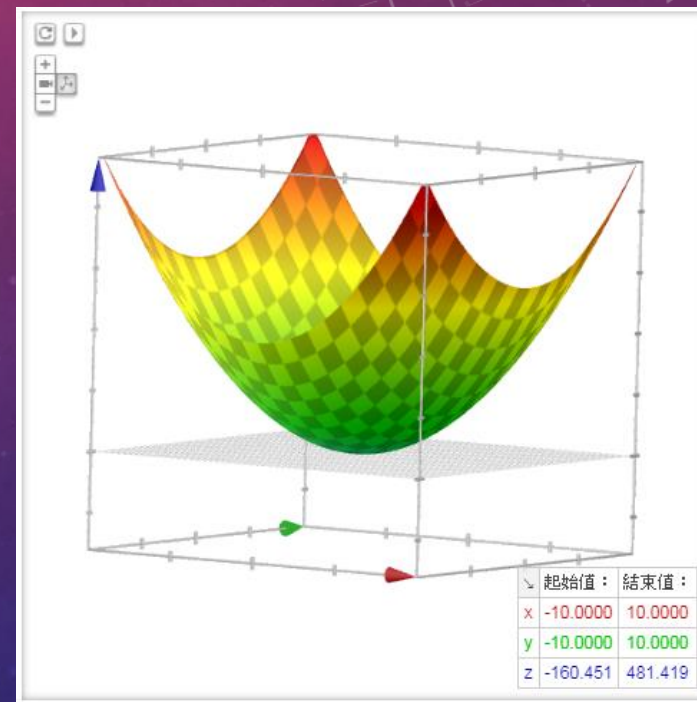
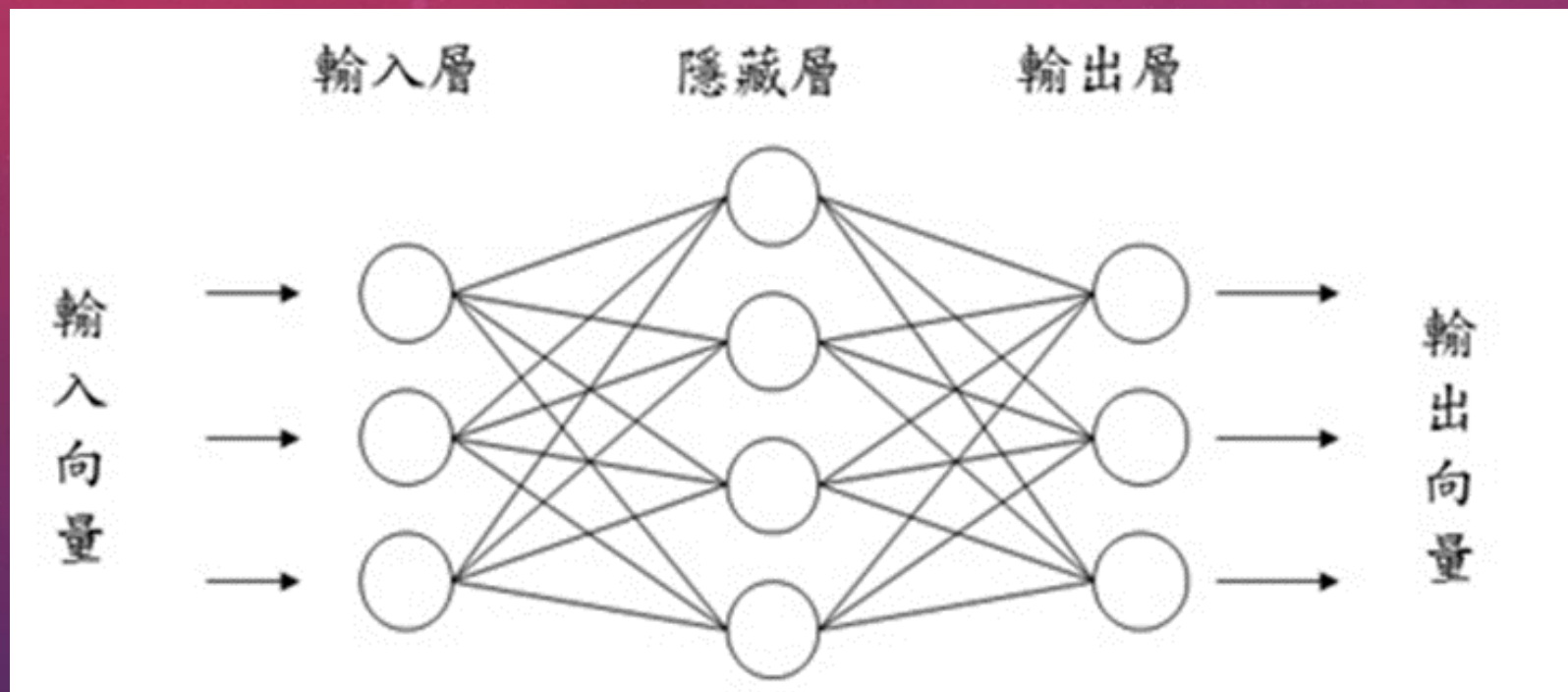
[Case 1]
 $y = 1$ 錯分成 $y = -1$



[Case 2]
 $y = -1$ 錯分成 $y = 1$

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







卷積神經網路

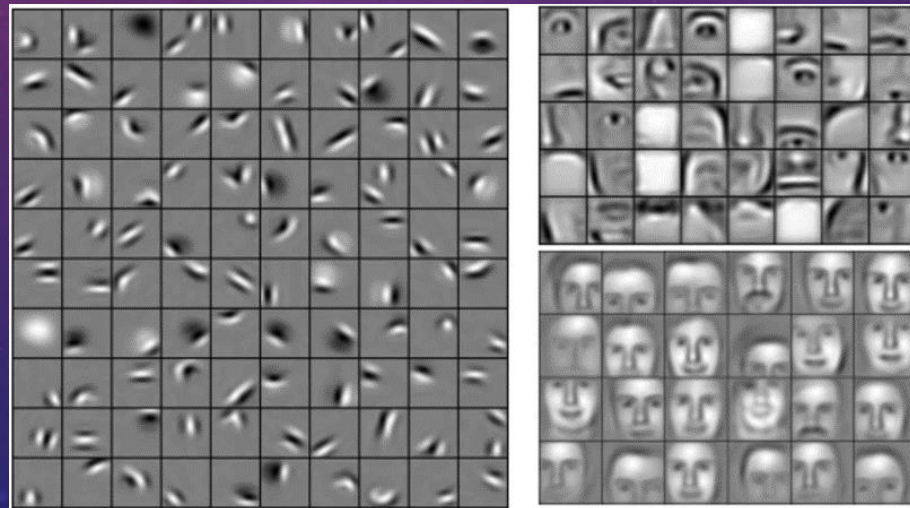
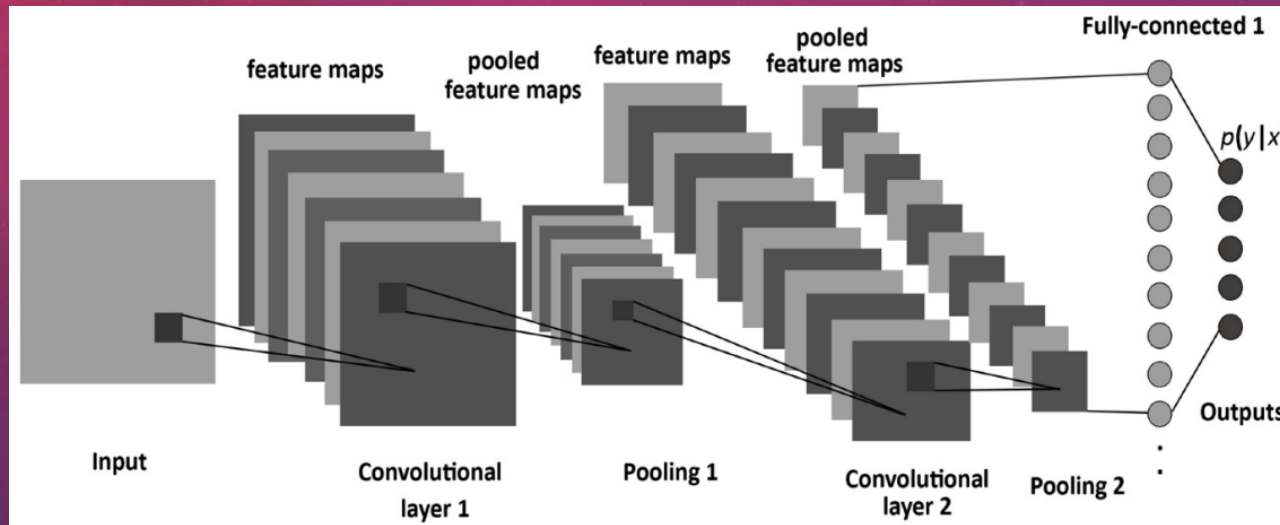
CONVOLUTIONAL NEURAL NETWORK

參考文獻

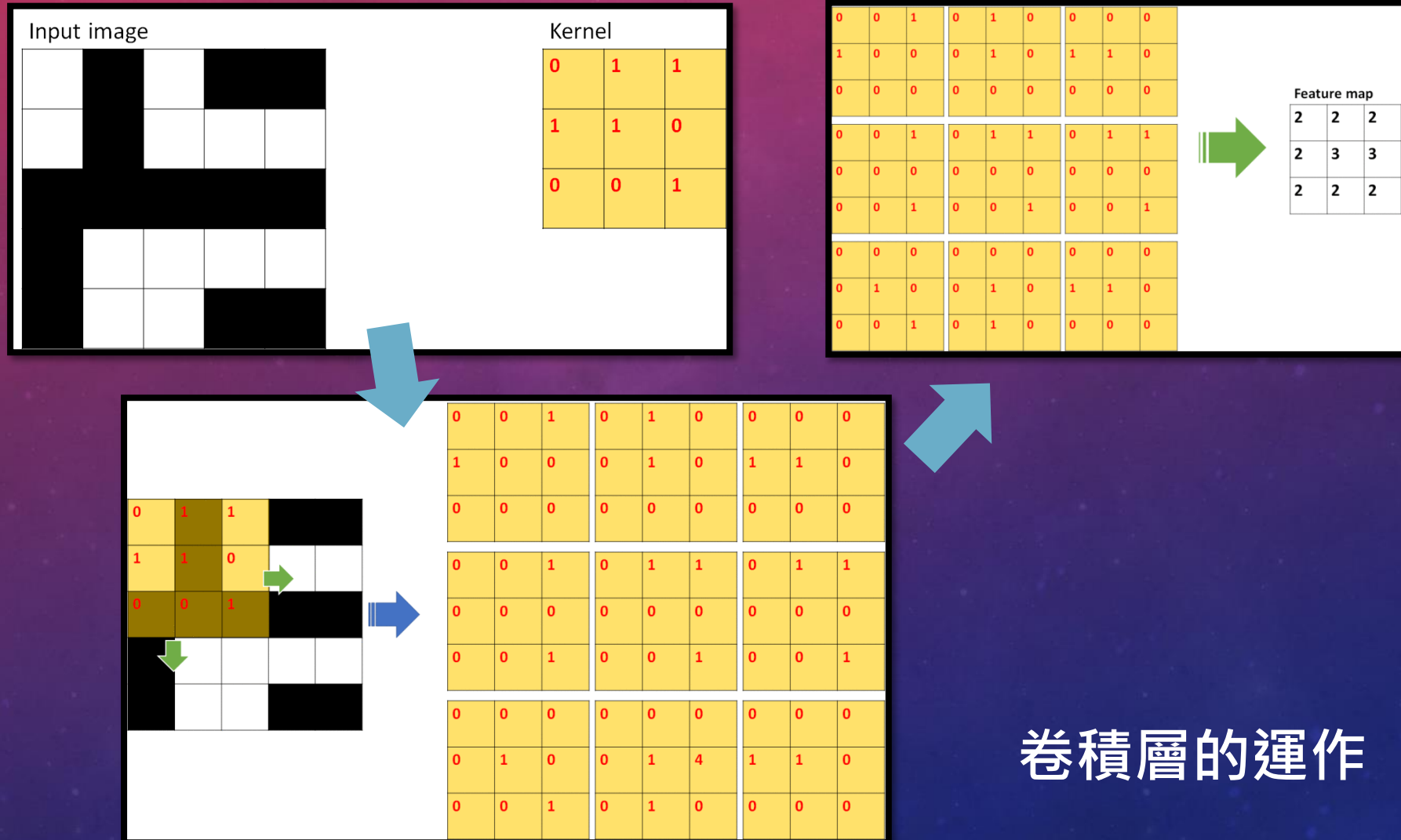
[HTTPS://GOO.GL/FJVJG1](https://goo.gl/FJVJG1)

[HTTPS://GOO.GL/Q5YKPK](https://goo.gl/Q5YKPK)

CONVOLUTIONAL NEURAL NETWORK



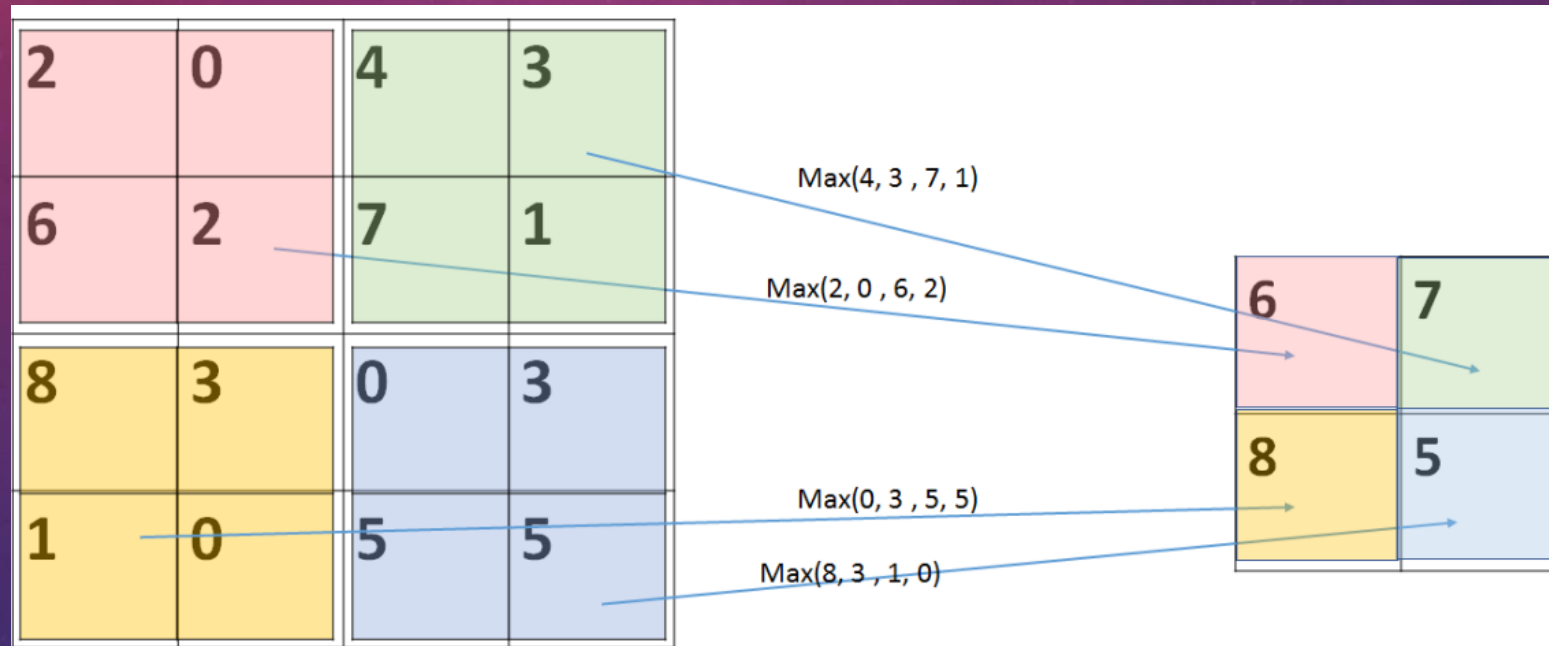
CONVOLUTIONAL NEURAL NETWORK



卷積層的運作

CONVOLUTIONAL NEURAL NETWORK

最大池化層的運作

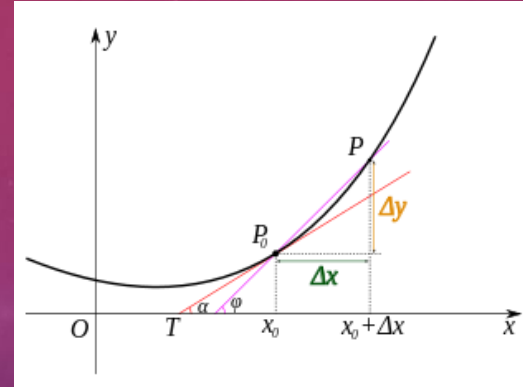
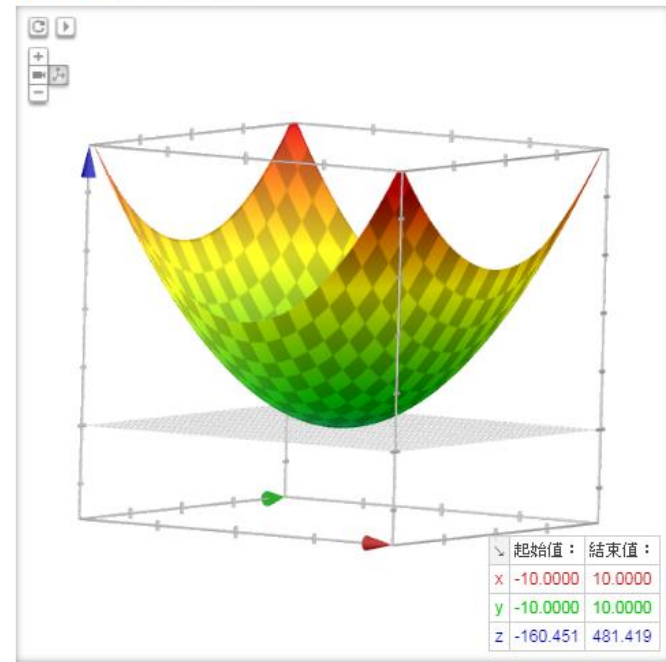


CONVOLUTIONAL NEURAL NETWORK

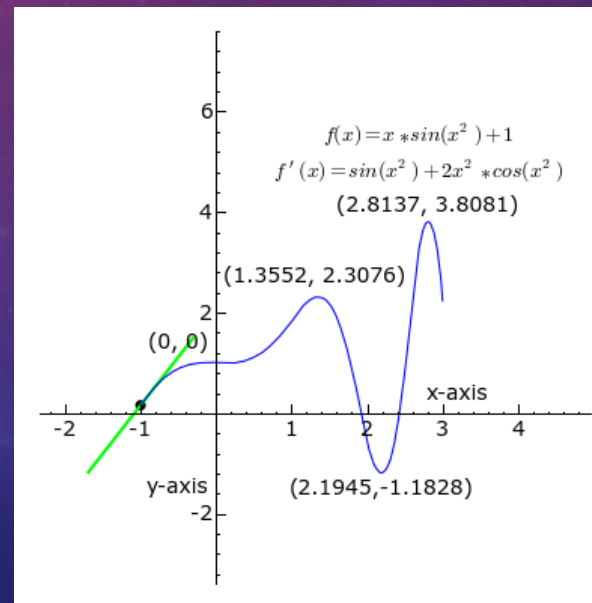
全連接層的運作

$$3x^2 + 2y^2$$

$3x^2 + 2y^2$ 的圖表



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



HOW COMPUTERS LEARN TO RECOGNIZE OBJECTS INSTANTLY



<https://youtu.be/Cgxsv1riJhI>

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

The background features a gradient from deep red at the top to dark blue at the bottom, speckled with white dots resembling stars. Overlaid on this are several faint, white circular and semi-circular lines, some with arrows indicating a clockwise direction. A large circular scale with numerical markings from 140 to 260 is visible on the left side.

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

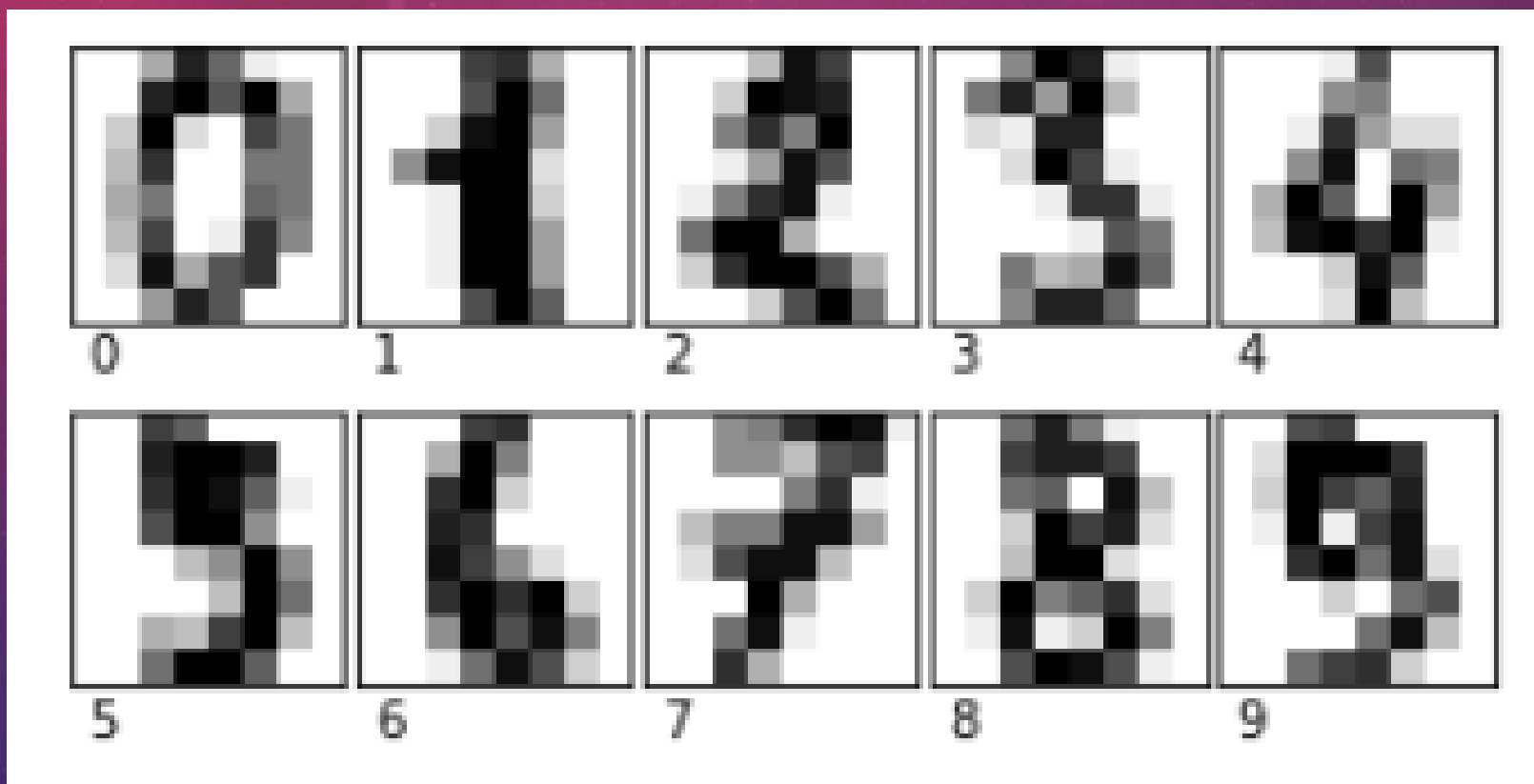


The background features a gradient from deep red at the top to dark blue at the bottom, speckled with white dots resembling stars. Overlaid on this are several faint, white circular and semi-circular lines, some with arrows indicating a clockwise direction. A prominent circular scale on the left side has numerical markings from 140 to 260 in increments of 10.

實作練習

數字辨識

手寫數字辨識



以 SKLEARN 實作 KNN

```
1 from sklearn.datasets import load_digits # 載入預設手寫資料庫
2 from sklearn.model_selection import train_test_split # 切割資料為訓練與測試集
3 from sklearn.preprocessing import StandardScaler # 標準化
4 from sklearn.neighbors import KNeighborsClassifier
5 #from sklearn.svm import LinearSVC
6 from sklearn.metrics import classification_report # 預測結果的分析工具
7 import matplotlib.pyplot as plt
8
9 # 載入 `digits` --- Step 1. 圖像取得
10 digits = load_digits()
11
12 # 設定圖形的大小 (寬, 高)
13 fig = plt.figure(figsize=(4, 2))
14
15 # 調整子圖形
16 fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
17
18 # 把前 8 個手寫數字顯示在子圖形
19 for i in range(10):
20     # 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
21     ax = fig.add_subplot(2, 5, i + 1, xticks = [], yticks = [])
22     # 顯示圖形, 色彩選擇灰階
23     ax.imshow(digits.images[i], cmap = plt.cm.binary) # Step 2. 預處理
24     # 在左下角標示目標值
25     ax.text(0, 9, str(digits.target[i]))
26
27 # 顯示圖形
28 plt.show();
```

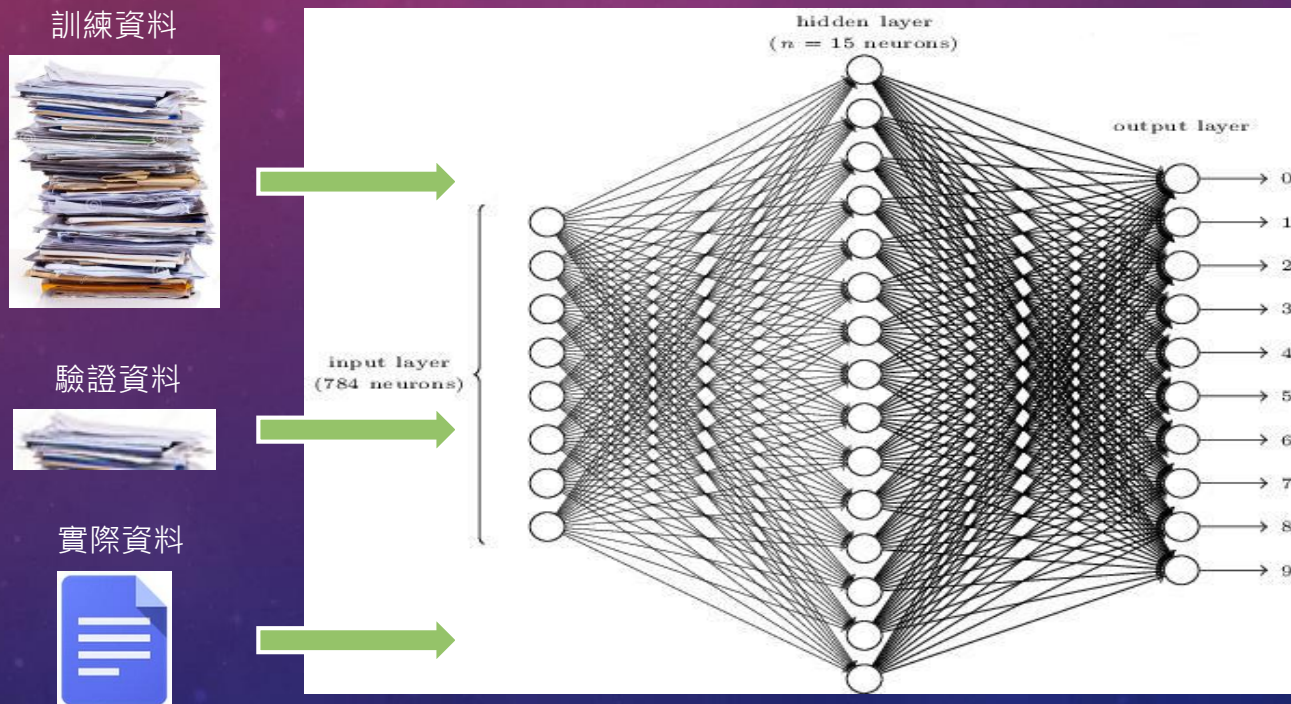
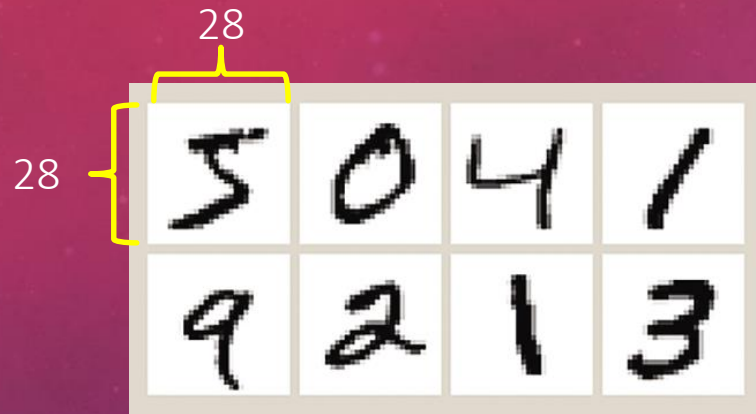
以 SKLEARN 實作 KNN

```
30 # 分割數據
31 X_train, X_test, Y_train, Y_test = train_test_split(digits.data, digits.target, test_size=0.25)
32
33 ss = StandardScaler() #標準化方法 Step 2. 預處理 (原始值-均值)/標準差
34 X_train = ss.fit_transform(X_train)
35 X_test = ss.transform(X_test)
36
37 kNN = KNeighborsClassifier(n_neighbors=3)
38 kNN = fit(X_train, Y_train)
39 Y_predict = kNN.predict(X_test)
40
41 #lsvc = LinearSVC()
42 #lsvc.fit(X_train, Y_train) # Step 3. & 4. 特徵提取與檢測
43 #Y_predict = lsvc.predict(X_test) # Step 5. 分類
44
45 print (classification_report(Y_test, Y_predict, target_names=digits.target_names.astype(str))) # Step 5. 驗證
```

手寫數字辨識

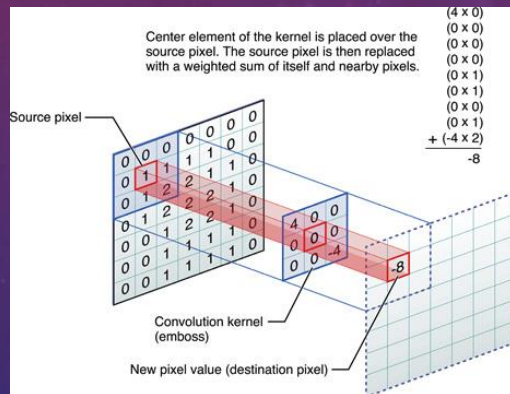
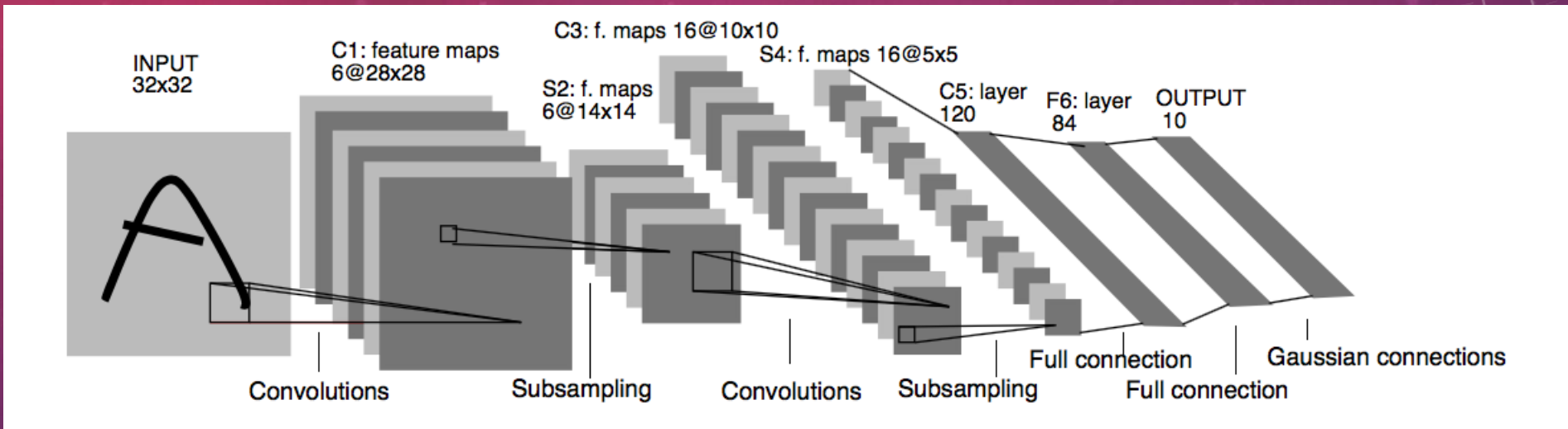
	precision	recall	f1-score	support
0	1.00	0.98	0.99	41
1	0.91	0.95	0.93	42
2	0.94	0.92	0.93	37
3	0.89	0.95	0.92	44
4	0.96	1.00	0.98	49
5	0.96	0.91	0.93	47
6	0.98	1.00	0.99	45
7	0.98	0.98	0.98	60
8	0.87	0.85	0.86	40
9	0.98	0.91	0.94	45
avg / total	0.95	0.95	0.95	450

MNIST 手寫數字辨識

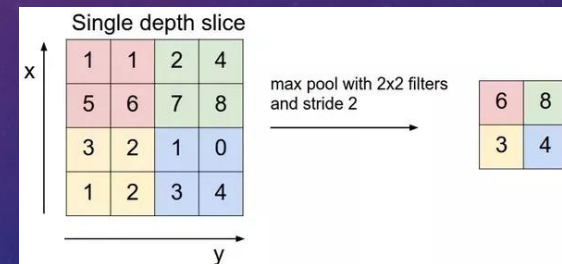


以最簡單的類神經網路架構，可達 **91%** 辨識率。若使用CNN則可高達 **99%** 辨識率。

卷積類神經網路



Convolution



Max Pooling

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))
```

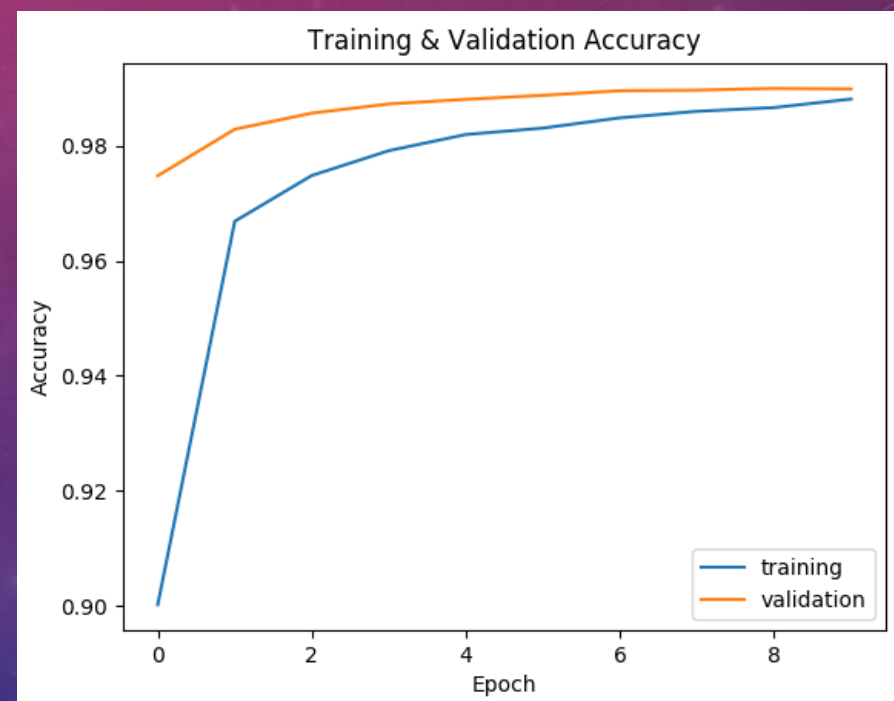
```
Epoch 1/10  
125s - loss: 0.3322 - acc: 0.9002 - val_loss: 0.0789 - val_acc: 0.9748  
Epoch 2/10  
121s - loss: 0.1125 - acc: 0.9669 - val_loss: 0.0519 - val_acc: 0.9829  
Epoch 3/10  
123s - loss: 0.0844 - acc: 0.9748 - val_loss: 0.0424 - val_acc: 0.9857  
Epoch 4/10  
127s - loss: 0.0714 - acc: 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()
```





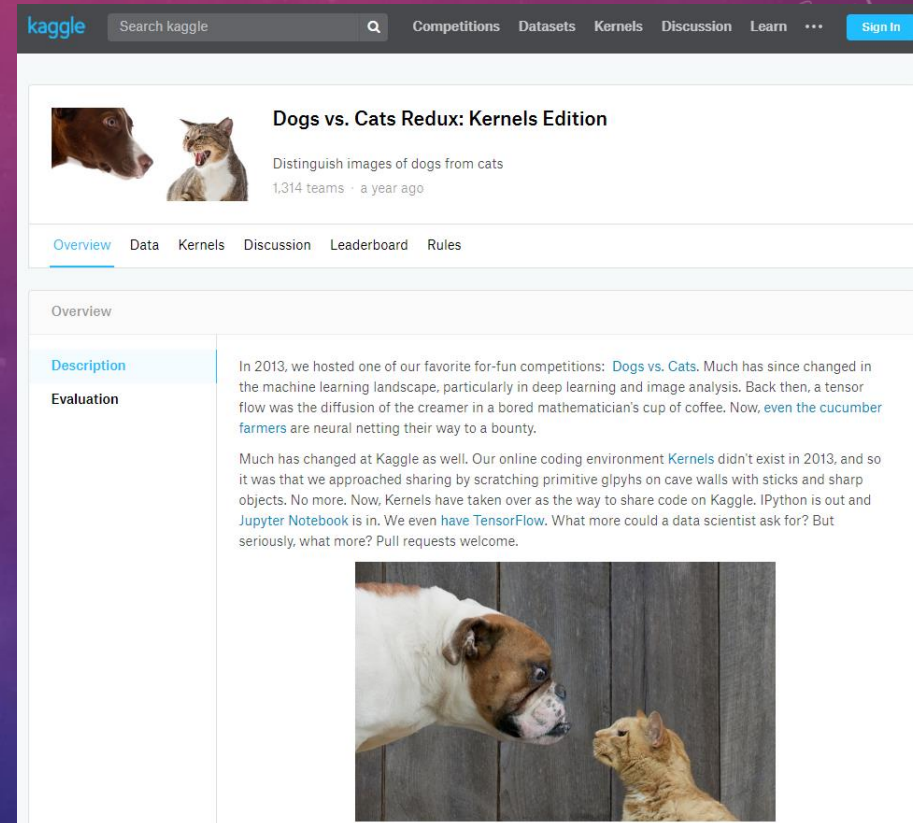
實作練習

貓狗辨識

[LINK](#)

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).



The screenshot shows the Kaggle website interface for the 'Dogs vs. Cats Redux: Kernels Edition' competition. The header includes the Kaggle logo, a search bar, and navigation links for Competitions, Datasets, Kernels, Discussion, Learn, and a Sign In button. The competition title is 'Dogs vs. Cats Redux: Kernels Edition', with a subtitle 'Distinguish images of dogs from cats' and statistics '1,314 teams · a year ago'. Below the title are tabs for Overview, Data, Kernels, Discussion, Leaderboard, and Rules. The 'Overview' section is active, showing a 'Description' and 'Evaluation' tab. The description text reads: 'In 2013, we hosted one of our favorite for-fun competitions: [Dogs vs. Cats](#). Much has since changed in the machine learning landscape, particularly in deep learning and image analysis. Back then, a tensor flow was the diffusion of the creamer in a bored mathematician's cup of coffee. Now, [even the cucumber farmers](#) are neural netting their way to a bounty. Much has changed at Kaggle as well. Our online coding environment [Kernels](#) didn't exist in 2013, and so it was that we approached sharing by scratching primitive glyphs on cave walls with sticks and sharp objects. No more. Now, Kernels have taken over as the way to share code on Kaggle. IPython is out and [Jupyter Notebook](#) is in. We even [have TensorFlow](#). What more could a data scientist ask for? But seriously, what more? Pull requests welcome.' Below the text is an image of a white and brown dog and a golden retriever looking at each other.

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]



以 TFLEARN 實作 CNN

```
1 import cv2 # pip install opencv-python
2 import numpy as np
3 import os
4 from random import shuffle # 隨機資料庫
5 from tqdm import tqdm # 輸出進度條
6 import matplotlib.pyplot as plt # 繪圖
7
8 train_dir = 'Pictures/train/'
9 test_dir = 'Pictures/test1/'
10 img_size = 50
11 lr = 1e-3
12
13 def label_img(img):
14     word_label = img.split('.')[ -3]
15     if word_label == 'cat': return [1,0]
16     elif word_label == 'dog': return [0,1]
17
18 def create_train_data():
19     training_data = []
20     for img in tqdm(os.listdir(train_dir)):
21         if (not img.endswith('.jpg')):
22             continue
23         label = label_img(img)
24         path = os.path.join(train_dir, img)
25         img = cv2.imread(path, cv2.IMREAD_GRAYSCALE) # 讀取圖片並轉為灰階
26         img = cv2.resize(img, (img_size, img_size)) # 將圖片轉為統一的大小
27         training_data.append([np.array(img), np.array(label)])
28     shuffle(training_data)
29     return training_data
30
31 train_data = create_train_data()
```

以 TFLEARN 實作 CNN

```
32
33 ▽ def process_test_data():
34     testing_data = []
35     for img in tqdm(os.listdir(test_dir)):
36         if (not img.endswith('.jpg')):
37             continue
38         path = os.path.join(test_dir, img)
39         img_num = img.split('.')[0]
40         img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
41         img = cv2.resize(img, (img_size, img_size))
42         testing_data.append([np.array(img), img_num])
43
44     shuffle(testing_data)
45     return testing_data
```


以 TFLEARN 實作 CNN

```
47 import tflearn # 需要安裝tensorflow，並安裝 tflearn
48 from tflearn.layers.conv import conv_2d, max_pool_2d # 2維 CNN 以及最大池化
49 from tflearn.layers.core import input_data, dropout, fully_connected # 輸入層，dropout，全連接層
50 from tflearn.layers.estimator import regression # cross entropy層
51 import tensorflow as tf
52
53 tf.reset_default_graph()
54 convnet = input_data(shape = [None, img_size, img_size, 1], name = 'input')
55
```

以 TFLEARN 實作 CNN

```
57 convnet = conv_2d(convnet, 64, 5, activation='linear') # the number of convolutional filters, filter_size
58 convnet = max_pool_2d(convnet, 5)
59
60 convnet = conv_2d(convnet, 32, 5, activation='linear')
61 convnet = max_pool_2d(convnet, 5)
62
63 convnet = conv_2d(convnet, 16, 5, activation='linear')
64 convnet = max_pool_2d(convnet, 5)
65
66
67 convnet = fully_connected(convnet, 16, activation = 'linear')
68 convnet = dropout(convnet, 0.5)
69
70 convnet = fully_connected(convnet, 8, activation = 'linear')
71
72 convnet = fully_connected(convnet, 4, activation = 'relu')
73
74 convnet = fully_connected(convnet, 2, activation='sigmoid')
75 convnet = regression(convnet, optimizer='adam', learning_rate = lr, loss='categorical_crossentropy', name='targets')
76
77 model = tflearn.DNN(convnet, tensorboard_dir='log')
```

以 TFLEARN 實作 CNN

```
78
79 trainNum = -5000 #len(train_data) * -0.2
80 #print(trainNum)
81
82 train = train_data[:trainNum]
83 test = train_data[trainNum:]
84
85 X = np.array([i[0] for i in train], dtype=np.float64).reshape(-1, img_size, img_size, 1)
86 y = np.array([i[1] for i in train], dtype=np.float64)
87 Xtest = np.array([i[0] for i in test], dtype=np.float64).reshape(-1, img_size, img_size, 1)
88 ytest = np.array([i[1] for i in test], dtype=np.float64)
89
90 model.fit({'input': X}, {'targets': y}, n_epoch=30, batch_size=500, validation_set=({'input': Xtest}, {'targets': ytest}),
91         |snapshot_step=500, show_metric=True, run_id='model' )
92
93 test_data = process_test_data()
```


以 TFLEARN 實作 CNN

```
94
95 fig = plt.figure()
96 for num,data in enumerate(test_data[:16]):
97     img_num = data[1]
98     img_data = data[0]
99     y = fig.add_subplot(4, 4, num+1)
100     orig = img_data
101     data = img_data.reshape(img_size, img_size, 1)
102     model_out = model.predict([data])[0]
103     print(model_out)
104     if np.argmax(model_out) == 1:
105         label = 'Dog'
106     else:
107         label = 'Cat'
108
109     y.imshow(orig, cmap='gray')
110     plt.title(label)
111     y.axes.get_xaxis().set_visible(False)
112     y.axes.get_yaxis().set_visible(False)
113
114 plt.tight_layout()
115 plt.show()
```

以TFLEARN 實作 CNN

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

```
Training Step: 799 | total loss: 0.36109 | time: 33.383s
| Adam | epoch: 010 | loss: 0.36109 - acc: 0.8478 -- iter: 19750/20000
Training Step: 800 | total loss: 0.36053 | time: 36.618s
| Adam | epoch: 010 | loss: 0.36053 - acc: 0.8462 | val_loss: 0.48558 - val_acc: 0.7870 -- iter: 20000/20000
--
```



THANK YOU