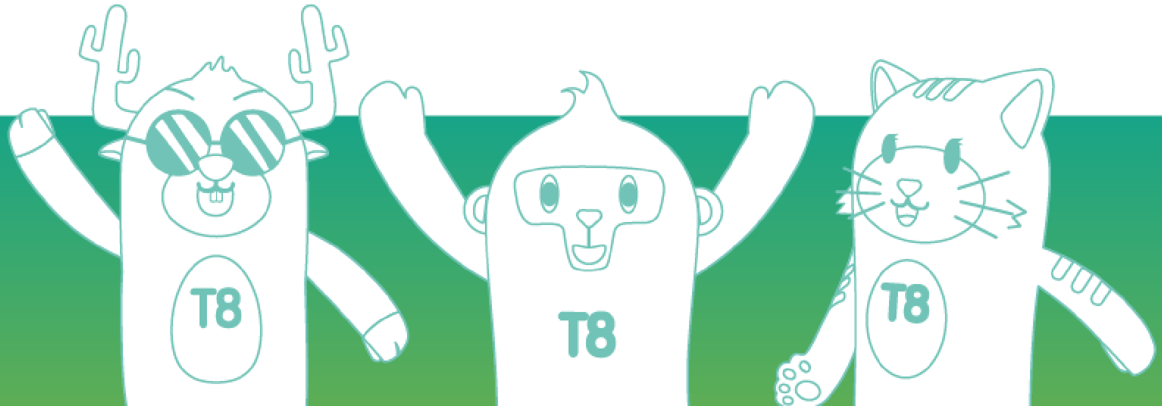
機器學習教學小計畫

授課講師　陳少君

教材編寫　陳少君



學習目標：

- 了解教學方式與進度

Module 0.
教學目的方式與進度

目的方式與進度

資料探勘養成班教學目的

為何而學

機器學習利用正確有效的資料訓練模型，並利用驗證資料及測試資料調整其參數，持續改進模型，使其分類、分群、迴歸等演算法能被有效應用

機器學習的一個分支：類神經網路，將成為未來深度學習的基礎。其他各演算法在各行各業也有其廣泛的應用領域。

承先啟後

先備知識：數學概念，Python/R 程式語言，資料探勘

進階：深度學習及應用

資料科學知識示意地圖

先備知識

- **數學基礎** (線性代數 機率統計 微積分)
- 程式語言 **Python**
- 資料庫，網際網路，IT系統架構

資料探勘

- 輸入 **ETL Pandas**
- 關聯規則分析
- **EDA** 特徵擷取
- 處理 **NumPy**
- 機器學習演算法
- 圖表 **Matplotlib**
- **A/B** 測試

機器學習

- 迴歸/預測
- 組合 **(Ensemble)**
- 分類
- 感知器/神經網路
- 演算法 **Scikit Learn**
- 分群

深度學習

- 激勵函數 損失函數 優化
- **DNN/CNN/RNN** 自然語言分析
- **GAN/VAE** 強化學習

養成班教學進度

第一天(9月8日)：迴歸與分類演算法
- 線性回歸探索
- Entropy(熵)與決策樹
- Log loss 與 羅吉斯迴歸
- 樸素+貝葉斯
- 支持向量機(SVM)

第二天(9月14日) ：神經網路，集成與分群
- 類神經網路簡介
- Ensemble集成演算法
- Enron案例
- 分群演算法

第一天

2 小時

線性回歸
Entropy熵

2 小時

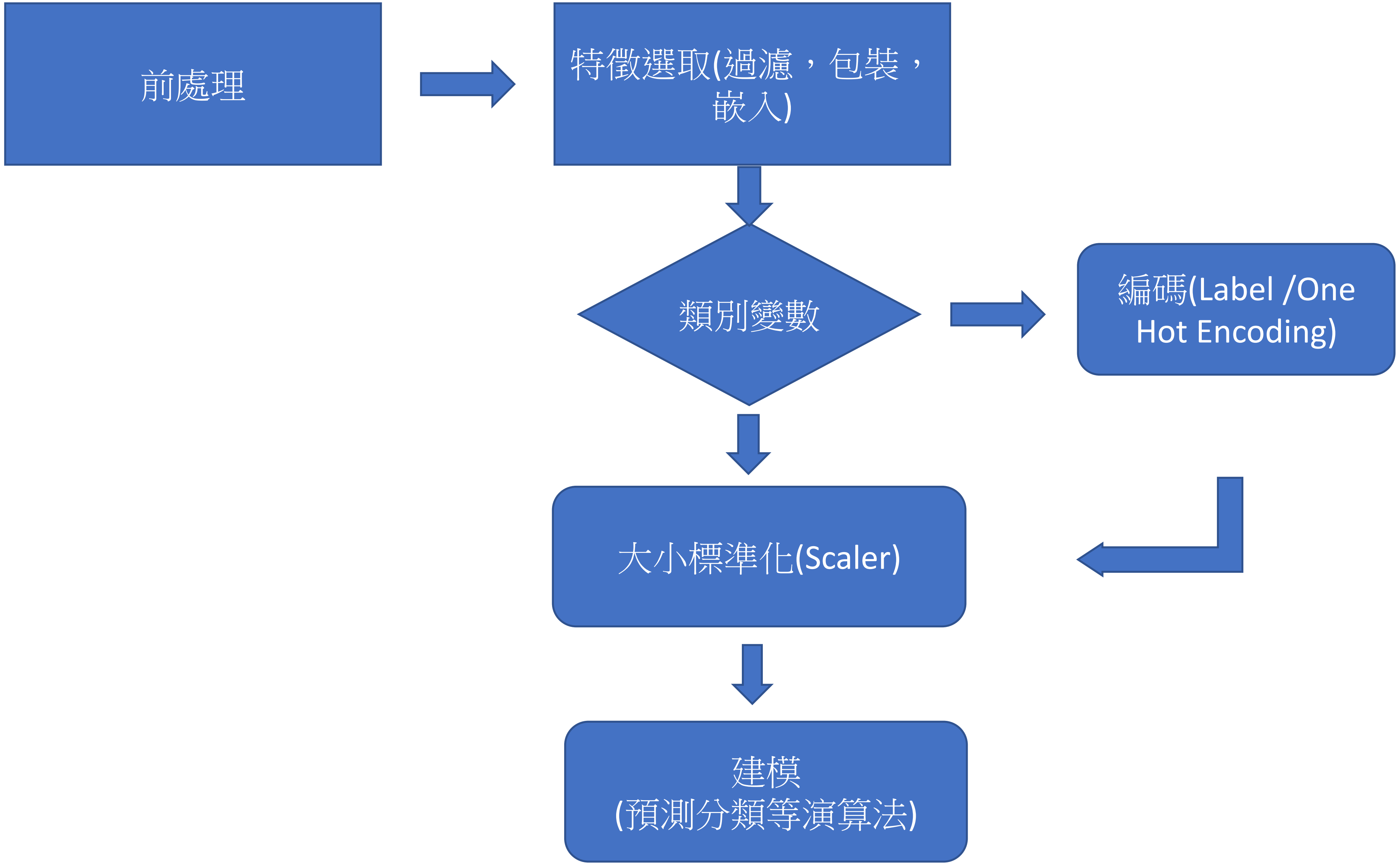
決策樹
羅吉斯回歸

2 小時

樸素貝葉斯
SVM

預習：回歸分類演算法
作業：作業一

從特徵到建模


前處理
特徵選取(過濾，包裝，嵌入)
類別變數
編碼(Label /One Hot Encoding)
大小標準化(Scaler)
建模
(預測分類等演算法)

線性迴歸之假設

1. 目標值(Y)的變異數不隨預測值X的不同而改變(才能用MSE計算)
2. 殘差(residuals)本身呈常態分佈 ：**繪直方圖檢驗**
3. 殘差 vs 目標值(or 預測值)是隨機分布/獨立的：**繪散點圖檢驗**

解釋力看R2
顯著與否看F

Entropy Calculator

線上Shannon Entropy 計算器

[Shannon Entropy Calculator | Information Theory (omnicalculator.com)](omnicalculator.com)

Log 可以有 底(Base)為2(Shannon)，底為歐拉數 e (nat)，或者底為10 (dit)，Entropy Calculate 出來的值也會不同，但可輕易互換

Dit Value = Log_{10} A = Log_2 A / Log_2 10 = Shannon Value * Log_{10} 2

Dit Value = Shannon Value * 0.301

ID3 – Pick Outlook first, if Sunny, then….

Outlook	Temperature	Humidity	Windy	Play
Sunny	Med	High	False	Y
Sunny	Cool	Normal	False	Y
Sunny	Cool	Normal	True	N
Sunny	Med	Normal	False	Y
Sunny	Cool	High	True	N

E(3,2) = 0.971

Entropy: 2/5 * E(2,0)+3/5*E(1,2) > 0

Temp	Play	Not Play	
Med	2	0	2
Cool	1	2	3
			5

Entropy: 2/5 * E(1,1)+3/5*E(2,1) > 0

Humidity	Play	Not Play	
High	1	1	2
Norm	2	1	3
			5

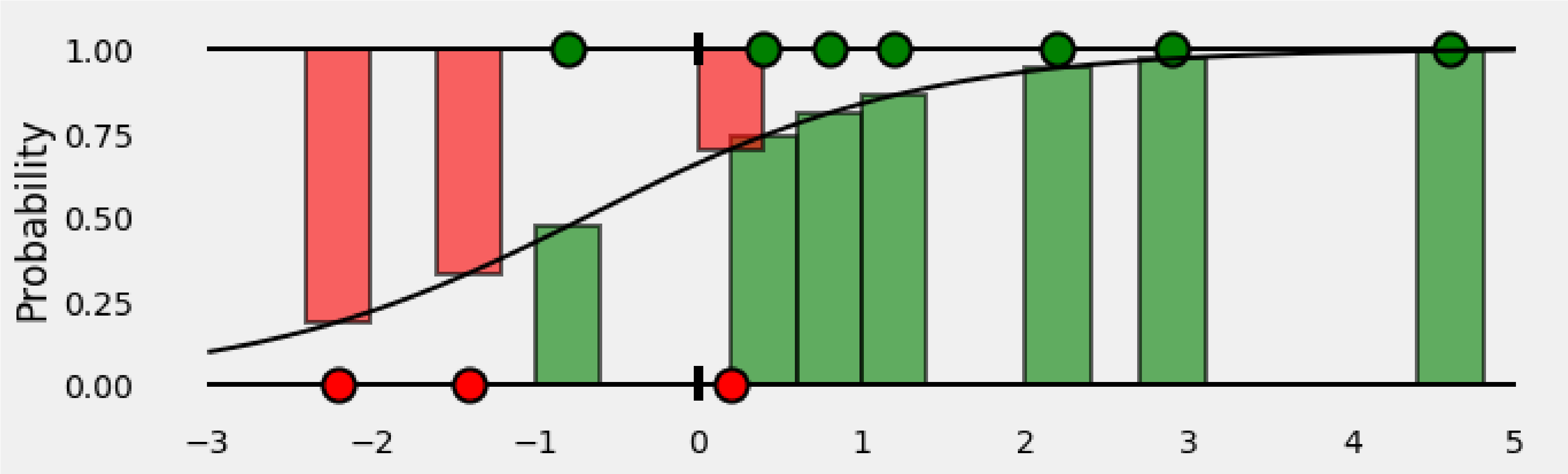
Windy	Play	Not Play	
False	3	0	3
True	0	2	2
			5

Entropy: 3/5 * E(3,0) + 2/5 *E(0,2) = 0

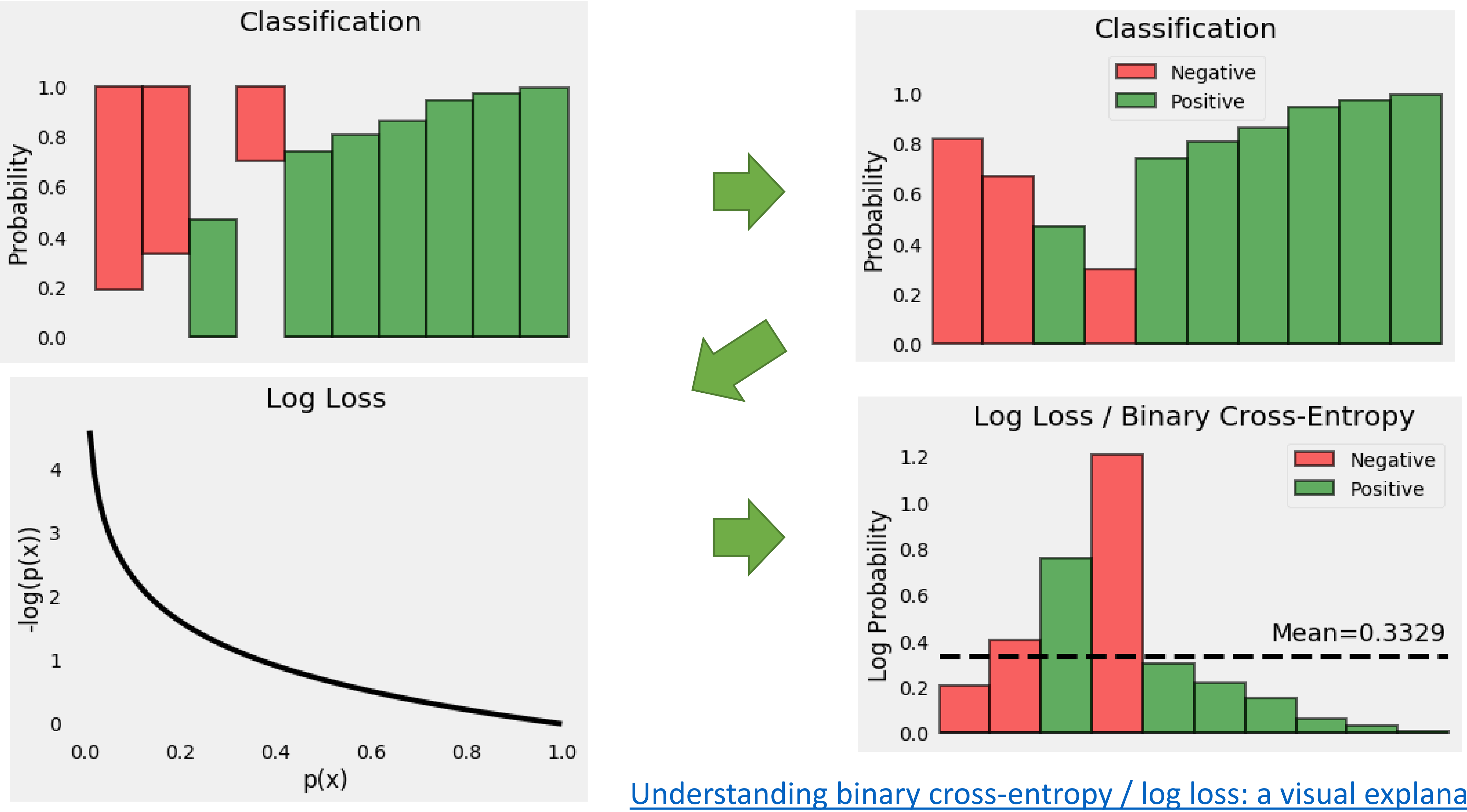
Pick Windy!

Log Loss (Cross Entropy)計算

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

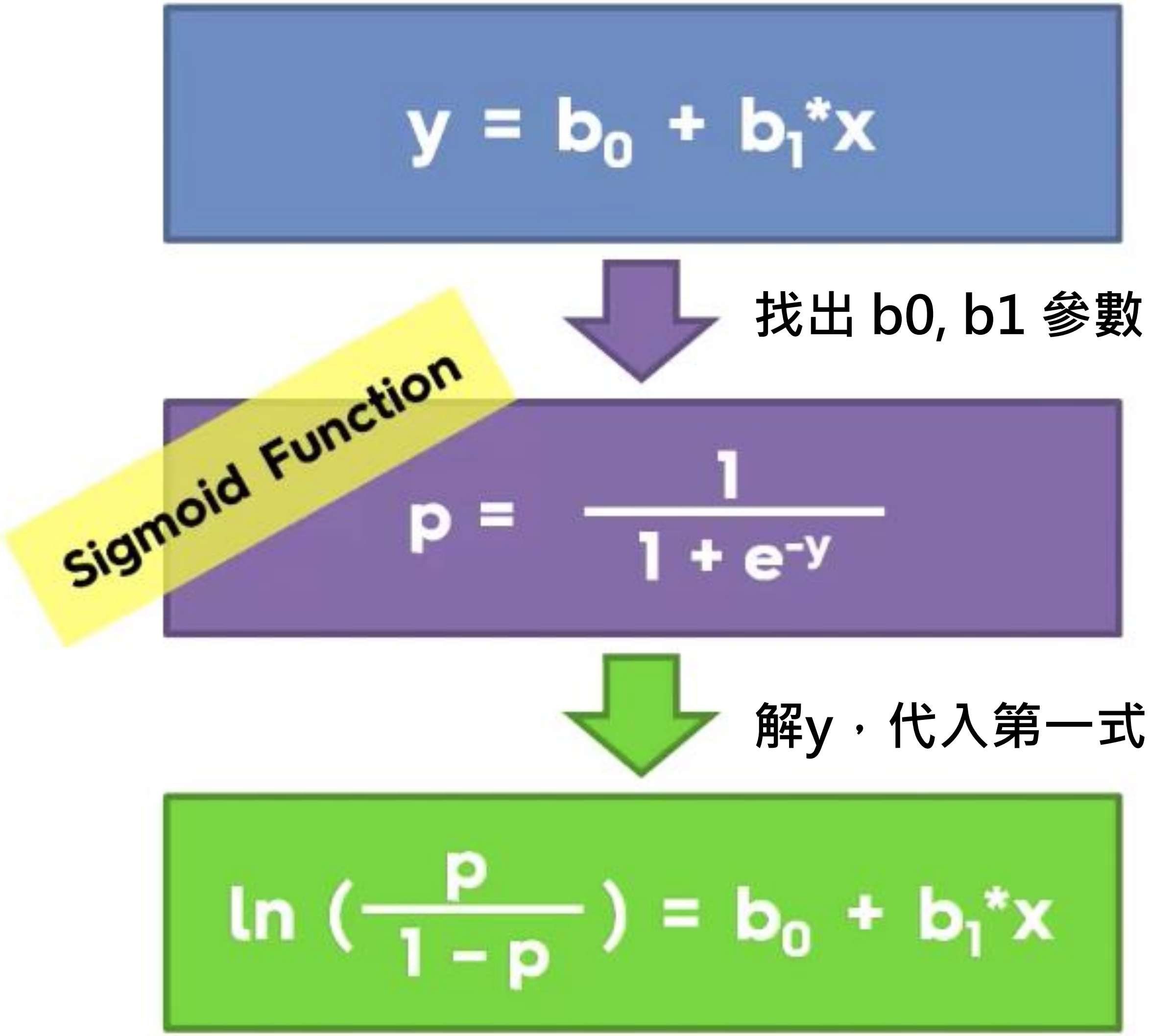


Log Loss (Cross Entropy) 計算



[Understanding binary cross-entropy / log loss: a visual explanation | by Daniel Godoy | Towards Data Science]

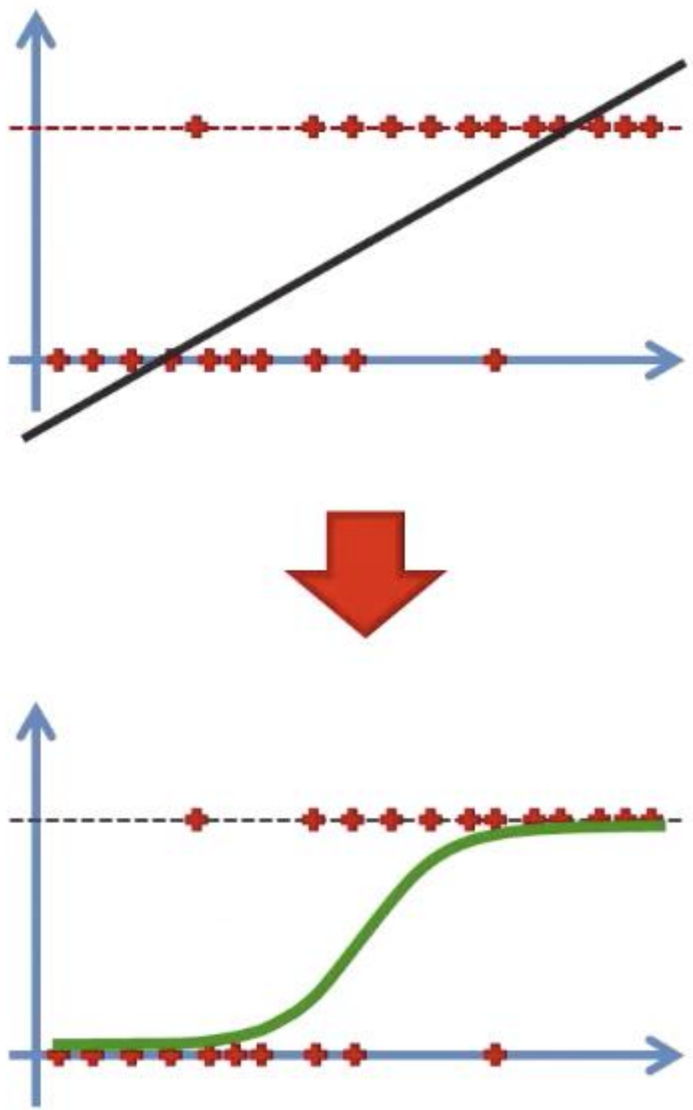
由Linear Regression 導出 Logistic Regression



找出 b0, b1 參數

解y，代入第一式左邊

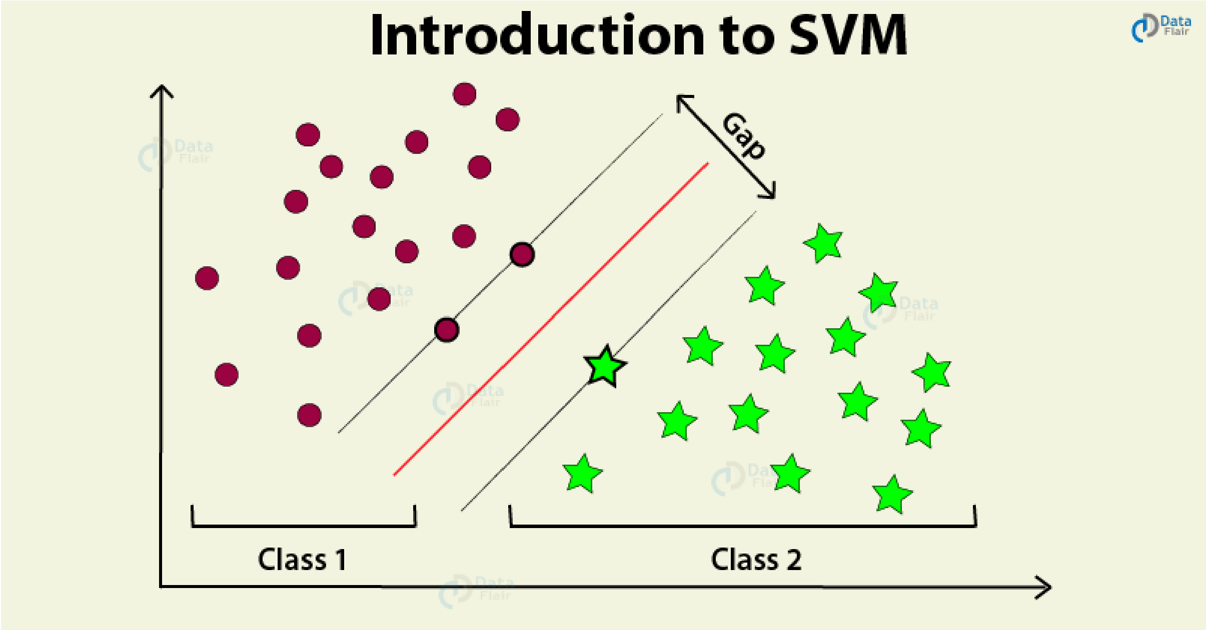
此公式即可預測p

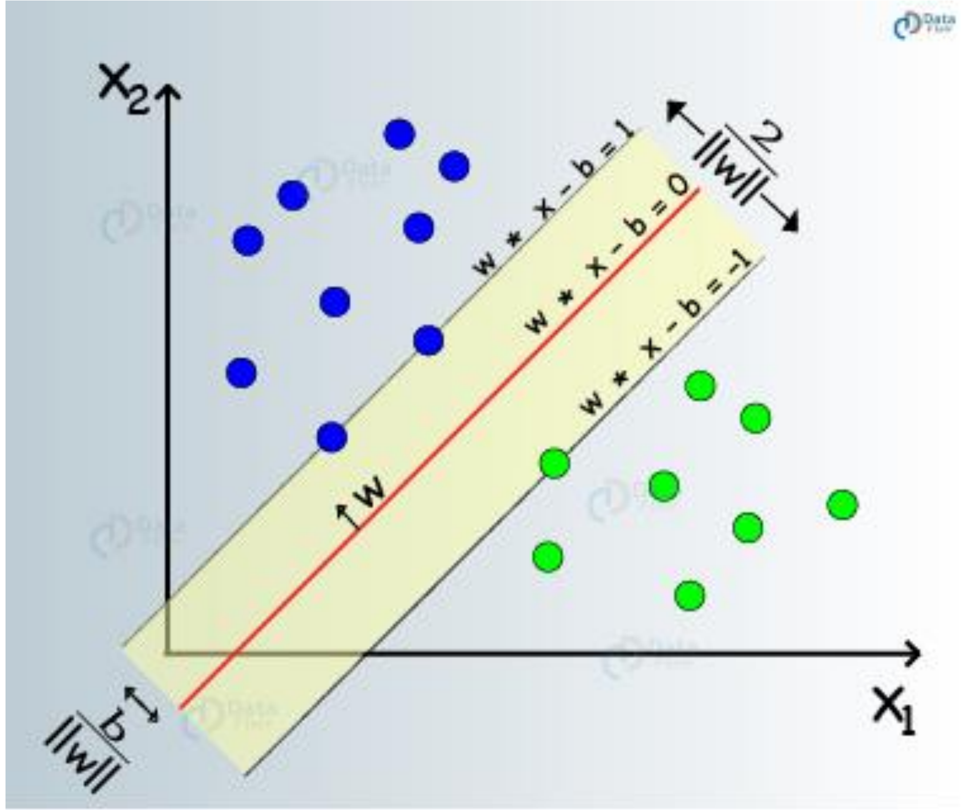


[Linear to Logistic Regression, Explained Step by Step - KDnuggets]

SVM 步驟

1. 定義Class Label 為 1 and -1
2. 定義兩個Class 的支持超平面(Support Hyperplane)
3. 在兩支持超平面中間作為分類器(Classifier)
4. 定義目標函數為 margin = 2 / ||w||
5. 限制兩邊距離至少為1
6. 結合目標函數與限制式求解 w





SVM 說明

一個點 (x_0,y_0) 到一條線 : $Ax+By+c = 0$ 的距離是:
$|Ax_0 + By_0 + c|/sqrt(A^2+B^2)$,

所以H0 到 H1 的距離就是: $|w \bullet x+b|/||w||=1/||w||$,
所以H1 到 H2 的全距是: $2/||w||$ (1.目標函數)

求極大值就是求$||w||$極小值.
因為假設H1 和 H2之間無其他點，所以:

$x_i \bullet w+b \geq +1$ when $y_i = +1$
$x_i \bullet w+b \leq -1$ when $y_i = -1$

合併為: $y_i (x_i \bullet w-b) \geq 1$ (2.限制條件)
有了1與2，以Lagrange Multiplier方式求解

Bayes’ Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

教室裡有24人，8人熱愛運動，14人是男生，男生且熱愛運動有5人，假設某A喜歡運動，此人為男生的機率是多少？

演算法：貝氏定理

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

$$P(熱愛運動|男) = \frac{P(男 \cap 熱愛運動)}{P(男)} = \frac{5/24}{14/24} = 5/14$$

$$P(男|熱愛運動) = \frac{P(熱愛運動|男)\,P(男)}{P(熱愛運動)} = \frac{5/14 * 14/24}{8/24} = (5/24)/(8/24) = 5/8$$

Laplace Smoothing

以Book, Price在 stmt class 為例

Before LS, Book = 2 / 15 = 0.1333

Laplace Smoothing =

$$\frac{\text{該字出現次數} + 1}{\text{該條件總字數} + \text{文件集獨特字總數}}$$

After LS, Book = (2 + 1) / (15 + 21) = 3/36 = 0.8333 (被稀釋)

已存在者(book)：1/21 (最小可能數) < 3/36 (LS) < 2/15 (原來的值)，

不存在者(Price) ：當出現次為 0 時 至少有 1/(15+21)=0.0277 而非 0

第二天

2 小時

類神經網路
集成學習

→

2 小時

集成學習
Enron Case

→

2 小時

分群(Clustering)

預習：Perceptron，Ensemble Learning, Clustering
作業：作業二

感知器(Perceptron)與內積

```
#產生0/1的激活函數，其實就是 輸入(X)和權重(WEIGHT)的內積(DOT PRODUCT)
import numpy as np  # for matrix multiplication

def booleanActivation(input, weights):
    # input and weights are arrays of values
    input = [1] + input  # add bias input
    z = np.dot(input, weights).sum()  # h_w(x)

    if z > 0:
        return 1
    else:
        return 0
```

X1	NOT(X)
0	1
1	0

NOT(X): $[1, X]\begin{bmatrix}1\\-1\end{bmatrix}$ = 1 * 1 + (-1)X = 1 – X. (IF X = 0 NOT(X) = 1, IF X=1, NOT(X) = 0)

X1	X2	X1 or X2
0	0	0
0	1	1
1	0	1
1	1	1

OR(X): $[1, X1, X2]\begin{bmatrix}-0.5\\1\\1\end{bmatrix}$ = 1 * -0.5 + 1*X1 + 1*X2 以右邊的真值表描述

Adaboost 如何 assign weights (I)

#Sample weight = 1/N
#Where N = Number of records
#example是一個 Data Frame

```
#Initially assign same weights to each records in the dataset
example['probR1'] = 1/(example.shape[0])
```

#replace= True 取完放回，每次機率都一樣
#weights 是被取到的機率，剛開始1/N，每一筆可能不同

```
#simple random sample with replacement
random.seed(10)
example1 = example.sample(len(example), replace = True, weights = example['probR1'])
```

#example1 變成 X_train y_train，建模訓練

```
#fitting the DT model with depth one
clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=1)
clf = clf_gini.fit(X_train, y_train)
```

Adaboost 如何 assign weights(II)

判斷錯誤的 weight 將改變，供下一輪使用

```
#error calculation
e1 = sum(example['misclassified'] * example['probR1'])
```

```
#calculation of alpha (performance)
alpha1 = 0.5*log((1-e1)/e1)
```

```
#update weight
new_weight = example['probR1']*np.exp(-1*alpha1*example['Label']*example['pred1'])
```

```
#normalized weight
z = sum(new_weight)
normalized_weight = new_weight/sum(new_weight)
```

#最後的預測

```
#final prediction
t = alpha1 * example['pred1'] + alpha2 * example['pred2'] + alpha3 * example['pred3'] + alpha4 * example['pred4']

#sign of the final prediction
np.sign(list(t))
```

Blending 步驟

1.The train set is split into two parts, viz-training and validation sets.
2.Model(s) are fit on the training set.
3.The predictions are made on the validation set and the test set.
4.The validation set and its **predictions are used as features** to build a new model.
5.This model is used to make final predictions on the test and meta-features.

The difference between stacking and blending is that **Stacking uses out-of-fold predictions for the train set of the next layer (i.e meta-model)**, and Blending uses a validation set (let’ s say, 10-15% of the training set) to train the next layer.

演算法中的正則化(Regularization)

目的：經由調整Loss Function (增加 L1, L2等)達到降低過適(Overfitting)。

L1: (LASSO) $||\mathbf{w}||_1 = |w_1| + |w_2| + ... + |w_N|$

L2: (RIDGE) $\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$

$$\hat{y} = w_1x_1 + w_2x_2 + ... + w_Nx_N + b$$

Logistic Regression Loss Function: L(y_hat,y) = y log y_hat + (1 - y)log(1 - y_hat)

$$Loss = Error(y, \hat{y})$$

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

可參考：

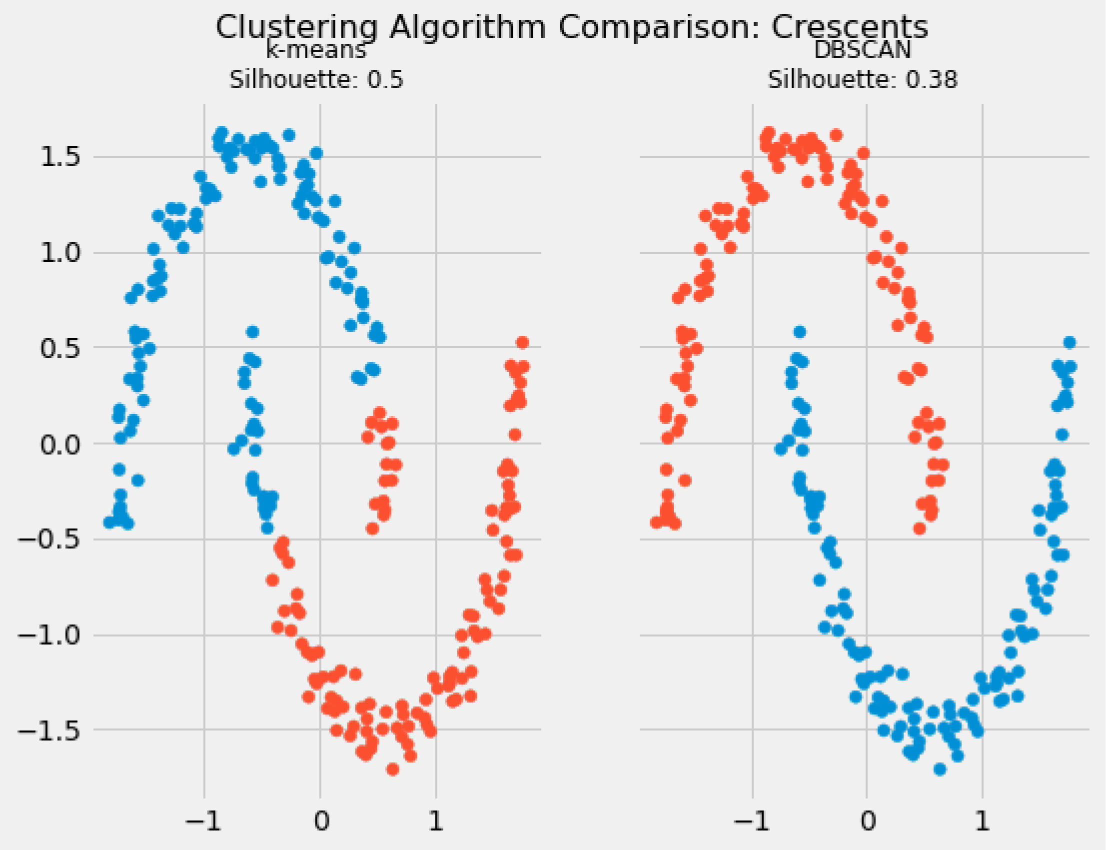
[Regularization in Machine Learning - GeeksforGeeks]

K-means 與 DBSCAN 之優劣

可以用SSE找出膝蓋點(Knee)，建議 K-means 的 K 數

輪廓係數(silhouette coefficients)比較分群好壞
K-means =0.5, DBSCAN = 0.39

用ARI (Adjusted Rand Score) 比較 K-means 和 DBSCAN 如右圖
K-means = 0.47 DBSCAN = 1.0



機器學習演算法的優劣

https://www.itread01.com/content/1541334303.html

https://www.hackingnote.com/en/machine-learning/algorithms-pros-and-cons

https://www.ipshop.xyz/5950.html