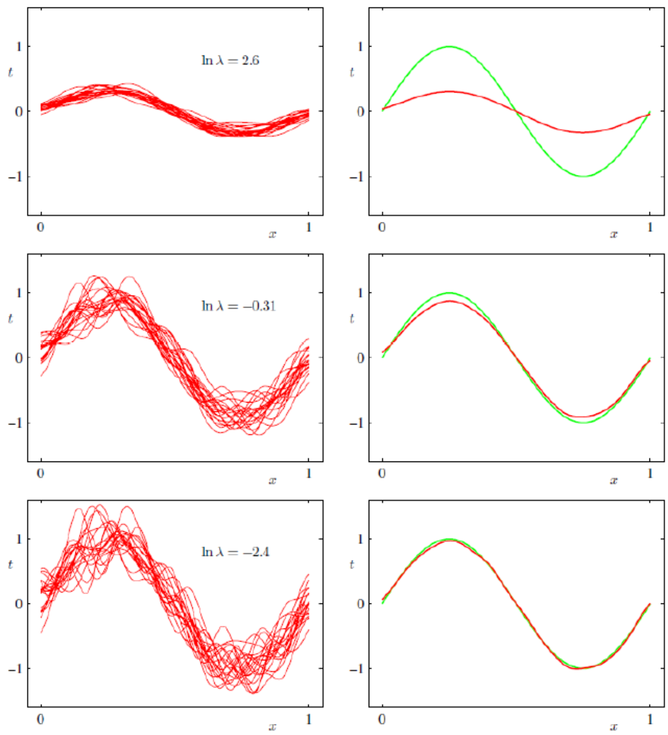
Bias 與 variance 之間的權衡

若是我們希望 bias 小，也就是我們算出來的參數要和母體參數的誤差要很小，常常會讓 variance 很大，換句話說，就是 sensitivity 很高，容易因為幾筆數據的不同而讓結果有大幅度的改變，又或是我們重複執行試驗數次，可能每一次得到的結果都不太一樣。

之前我們也有拿課本的圖說明這樣的概念



當 λ 越大，可看到多條紅線幾乎是重疊，不會有不穩定情形發生，但是隨之而來的就是誤差較大，可看到右圖，紅線和綠線差非常多。

當 λ 越小，不同的資料就會產生不同的結果，不會有不穩定情形發生，但是隨之而來的就是誤差較大，可看到右圖，紅線和綠線差非常多。

當然的，我們希望母體的參數和我們推得的越接近越好，也就是 $\min(E(\hat{\theta}-\theta)^2)$ ，在這之後我們

令 $MSE(\hat{\theta}) = E(\hat{\theta}-\theta)^2$ ，希望找到使得 $MSE(\hat{\theta})$ 最小的 $\hat{\theta}$

但是，我們推算參數時，其實我們是不知到母體參數的，如果知道的話，我們就直接用母體參數了，幹嘛還要那麼累推導那麼多 XD(就像如果我已經知道全球的男女比，我幹嘛還要慢慢抽樣去推算)，所以這個方法其實是不管用的。

接下來要推導的，就是最佳的 MSE 不是在最小的 bias 或是最小 variance 的地方，所以 bias 不是越小越好，variance 也是一樣。

note:

MSE 和 bias 有些許不同，$\boxed{MSE(\hat{\theta}) = E(\hat{\theta}-\theta)^2}$，是母體參數和估計參數差值的平方的期望值，

$bias = \theta - E(\hat{\theta})$ ，是母體參數和估計參數期望值的差

Bias- variance decomposition

$$E(\hat{\theta}-\theta)^2$$
$$=E((\hat{\theta}-E(\theta))-(\theta-E(\theta)))^2$$
$$=E((\hat{\theta}-E(\theta))^2-2(\hat{\theta}-E(\theta))(\theta-E(\theta))+(\theta-E(\theta))^2)$$

我們將第二項單獨拿出來看

$$2(\hat{\theta}-E(\theta))(\theta-E(\theta))$$
$$=2(\theta-E(\theta))E(\hat{\theta}-E(\theta))=2(\theta-E(\theta))(E(\hat{\theta})-E(E(\theta)))$$

這裡的 $E(\theta)$ 可想成之前的 μ ，則 $E(\hat{\mu})=\mu, E(E(\mu))=E(\mu)=\mu$

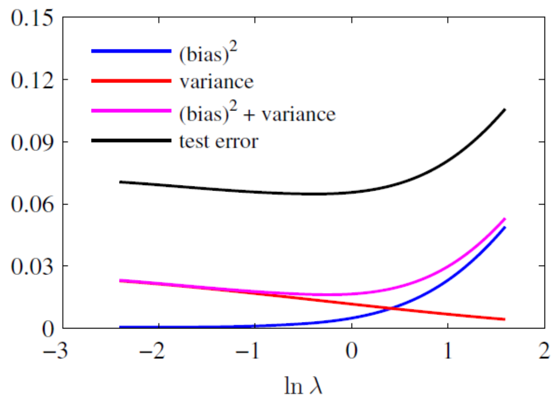
故

$$2(\hat{\theta}-E(\theta))(\theta-E(\theta))$$
$$=2(\theta-E(\theta))(E(\hat{\theta})-E(E(\theta))$$
$$=2(\theta-E(\theta))(E(\theta)-E(\theta))=0$$

故

$$E(\hat{\theta}-\theta)^2=E((\hat{\theta}-E(\theta))^2-2(\hat{\theta}-E(\theta))(\theta-E(\theta))+(\theta-E(\theta))^2)$$
$$=E(\hat{\theta}-E(\theta))^2+E(\theta-E(\theta))^2$$
$$=E(\hat{\theta}-E(\hat{\theta}))^2+E(\theta-E(\hat{\theta}))^2$$
$$=\mathrm{variance}(\hat{\theta}) \quad + \quad bias^2(\hat{\theta})$$

我們試著透過增大 λ 來觀察上式中 variance 和 bias 的關係



課本 p151

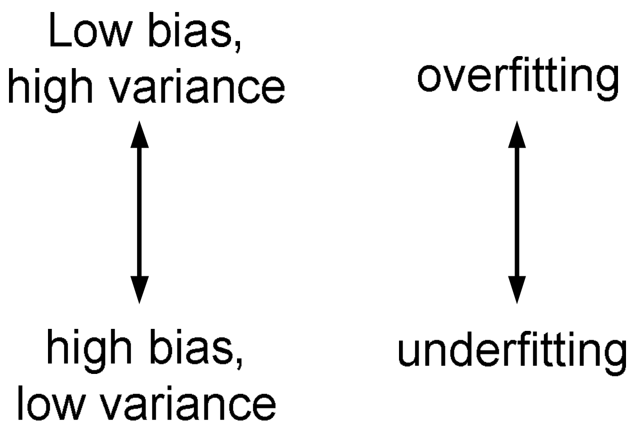
當加大 λ 時，因為 $\lambda=\frac{b}{a}$ ，當然的 a，也就是 variance 會變小，而當 variance 變小時，bias 隨

之而來的會變大，而帶入上面的公式時，我們剛剛推得的 MSE 和實際的 test error 是吻合的！

note:

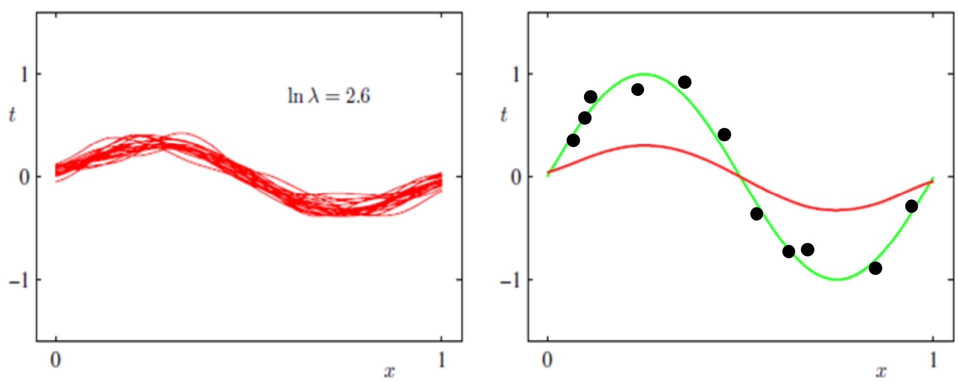
由上式推導我們可知道，雖然我們沒辦法直接得到最小的 MSE，但如果我們將 variance， 像是 lession10-11 時我們給予的參數 a，通常我們可以給予我們估計的 model variance，有些人可能就會有些疑惑，給 a=0 不是最好嗎？如果我們持續做 online learning，通常到最後收斂時 variance 都會接近 0，這樣對於我們估計出來的模型應該是很有信心的啊！怎麼可能會有人沒事假設一個很大的 a，讓最後就算收斂了，我們的信心仍然不足。前面也解釋過了，通常給予較小的 variance 會造成較大的 bias，但是較大的 bias 不見得是壞事，像上圖所示，最佳的 MSE 落在的區間不是最小的 variance 或是最小的 bias，而是要有所折衷。

Bias variance tradeoff



underfitting

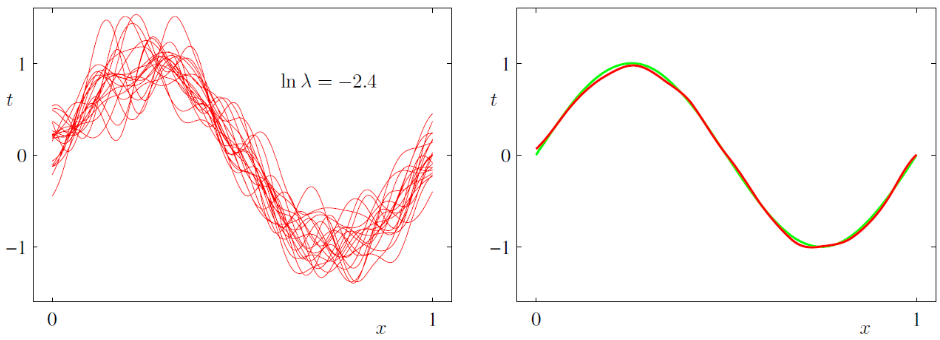
訓練出來的 model 連自身的 test data 都有著很大的誤差，就像是下圖的 case，train data 是黑點，當發生 underfitting 時，train 出來的結果如左圖，連本身的 train data 誤差都極大



overfitting

訓練出來的結果太貼近自身的 test data，下圖是因為取樣點夠多，所以不會發生 overfitting 的

現象，否則若是取樣點不夠，會離我們理想的 model 差距很大。



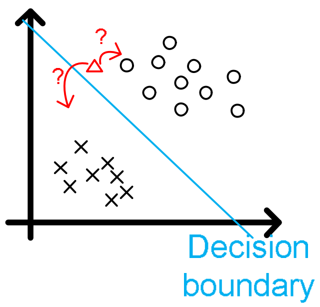
low bias:
e.g. LSE, k-nn algorithm

low variance:
e.g. rLSE, Naïve bayes classifier

classification

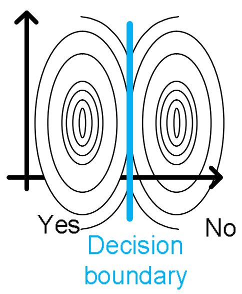
我們等會會說明，分類其實就是在做 regression

decision boundary



決定一個新 data 是在哪一個分類的依據，若是在 的左半邊就屬於 X 分類，若在 右半邊就屬於 O 分類。

e.g. Naïve bayes 分類器就是一個很顯著的例子



在 decision boundary 上的點，無法分類其為哪一類，因為屬於兩類的機率皆相同

Confusion matrix

用來分類各種分類的狀況，以表達分類的準確性，不只是分類分析的準不準而已，還分成四種狀況

		實際	
		YES (positive)	NO (negative)
預測	YES	true positive(TP)	false positive(FP)
	NO	false negative(FN)	true negative(TN)

true/false 代表預測的成功與否
positive/negative 代表預測發生的情形為何

分析的準確性有兩種指標，sensitivity 及 specificity

sensitivity 指的是所有實際為 yes 的結果中，有多少被我們判斷出來，也就是 $\frac{TP}{TP+FN}$

specificity 指的是我們將實際為 no 的結果判斷錯誤的機率，也就是 $\frac{FP}{FP+TN}$

ROC 空間

ROC curve 是可以用肉眼就看出分類的準確性的圖，X 軸是偽陽性率(false positive rate)，也就是預測為 Yes 但預測錯誤的機率，又稱為假警報率，就是上面提的 specificity。Y 為真陽性率(true positive rate)，又稱為敏感度(sensitivity)，我們在意的都只是實際上預測的狀況，所以分母都是實際狀況。

X 軸越小，Y 軸越大，代表我們預測的準確率越高，X 軸越小代表我們越不容易將實際為 NO 的事件預測為 YES，Y 軸越大代表我們越容易將實際為 YES 的事件預測為 YES。故在圖表中越左上角代表預測越準確。如果在最左上角，代表完美預測。

我們會將 ROC curve 圖做一(0,0)到(1,1)的對角線，越左上角代表預測越準確，越右下角代表越不準確，若是落在對角線上，代表完全沒有識別率(也就是你是亂猜的)。

例子我懶得想，直接拿維基百科的例子來看

A

TP=63	FP=28	91
FN=37	TN=72	109
100	100	200

TPR = 0.63
FPR = 0.28
ACC = 0.68

B

TP=77	FP=77	154
FN=23	TN=23	46
100	100	200

TPR = 0.77
FPR = 0.77
ACC = 0.50

C

TP=24	FP=88	112
FN=76	TN=12	88
100	100	200

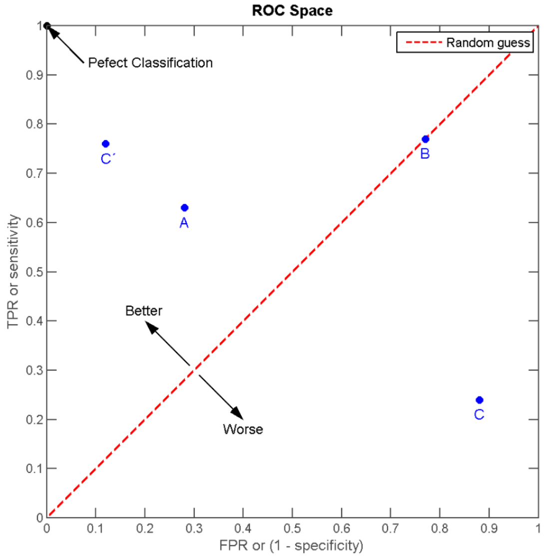
TPR = 0.24
FPR = 0.88
ACC = 0.18

C'

TP=76	FP=12	88
FN=24	TN=88	112
100	100	200

TPR = 0.76
FPR = 0.12
ACC = 0.82

將各點對應到 ROC 空間中如下圖



A 有最佳的識別率，B 完全沒識別率，C 有最差的識別率，但是如果我們將 C 每次的預測結果反過來看，其擁有最佳的識別率，所以 C 也是算有識別率的(就像是我們常講的反指標)。

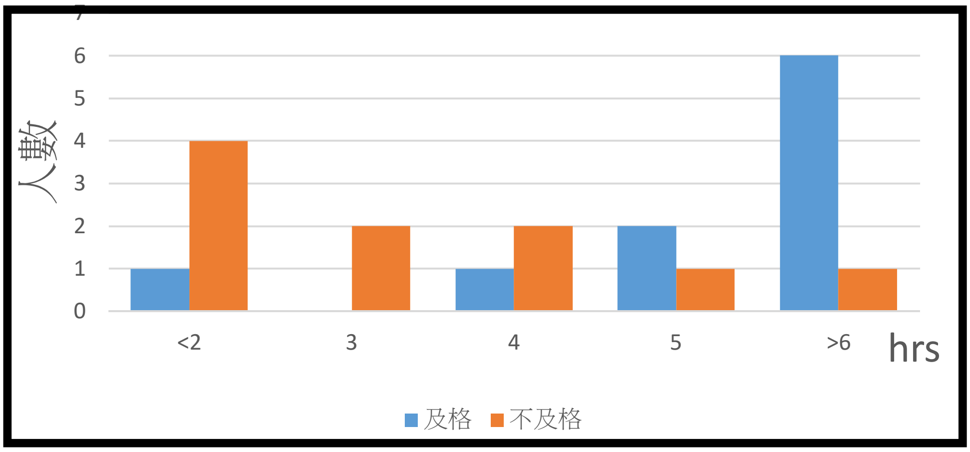
ROC curve

　　我們常常喜歡用一個閥值(threshold)來區隔發生事件的歸類，例如一天喝 6000c.c.以上的水會水中毒，BMI 超過多少就是過胖…，ROC 就是利用閥值的不同，來判斷我們預測採用的方法是不是好的。

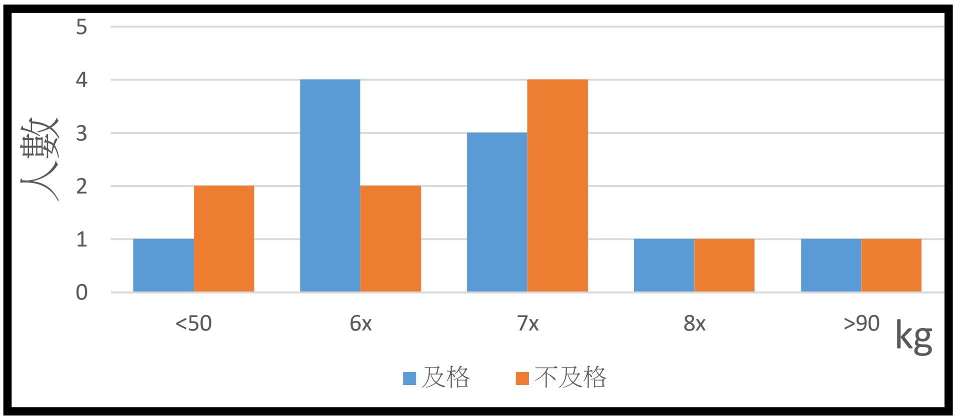
　　以下舉個例子應該會比較好懂

　　假設我們有兩個方法預測一個學生考試成績好不好，一個是花在讀書方面的時間，一個是該學生的體重。我們找十個考試及格和十個考試不及格的學生來做測試。

以讀書時間來看



以體重來看

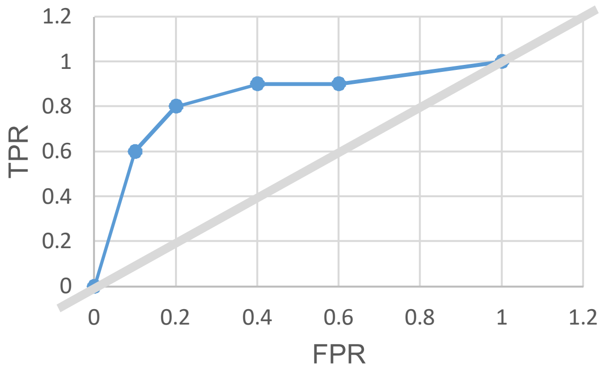


我們可以針對不同的閥值繪製出一張 ROC curve，例如以讀書時間來做預測為例，假定我們定義”只要一天讀書 4 小時以上就一定可以及格”，這樣的預測可得到 TPR 為 0.9，FPR 為 0.4，

我們將讀書時間閥值從<2 小時一直做到>6 小時，可得到下表

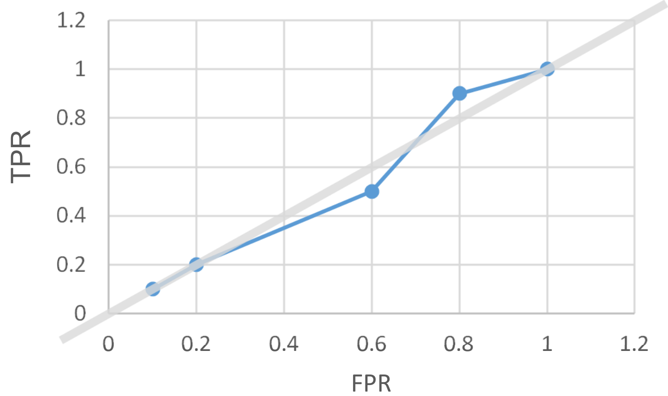
	TPR	FPR
<2	1	1
3	0.9	0.6
4	0.9	0.4
5	0.8	0.2
>6	0.6	0.1
>10	0	0

可得到 ROC curve

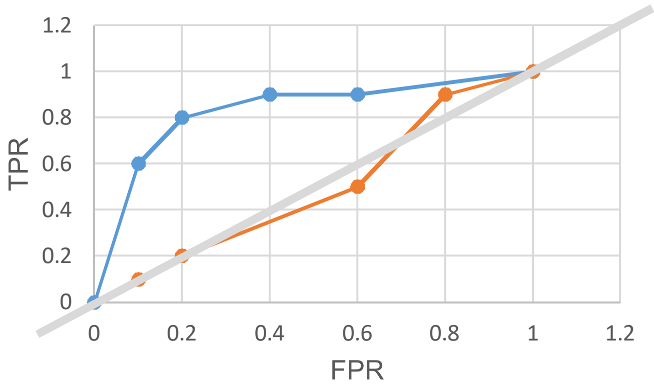


我們再做體重閥值的 ROC
先得下表

	TPR	FPR
<50	1	1
6x	0.9	0.8
7x	0.5	0.6
8x	0.2	0.2
>90	0.1	0.1
>120	0	0



我們將兩者因素結果疊起來



我們要如何判斷哪種判別標準比較好，就是使用 AUC(area under the curve of ROC)
AUC 是判別一個判別哪種標準正確率較好的標準，像是上圖，使用讀書時間得到的 AUC 比體重得

到的 AUC 還要高，故採用讀書時間的標準比較準確。

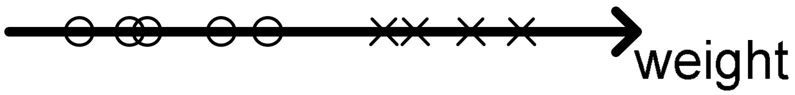
Regression -> classification

　　這邊描述的是 supervise learning(監督學習)，所謂的監督學習就是給予每個 train data 一個 label 供分類

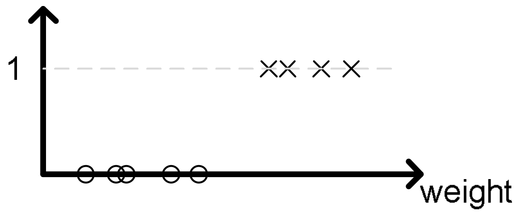
one-coding

one dimension

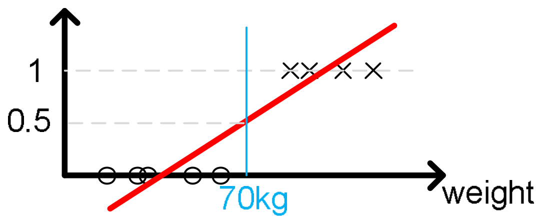
　e.g.
我們想分別男生和女生，以體重來做區分



將男生視為 label 1，女生視為 label 0(就是所謂的 indicator function)得



我們將上圖做 LSE，再找出 LSE 直線中縱軸為 0.5 的 weight，該 weight 為 decision boundary

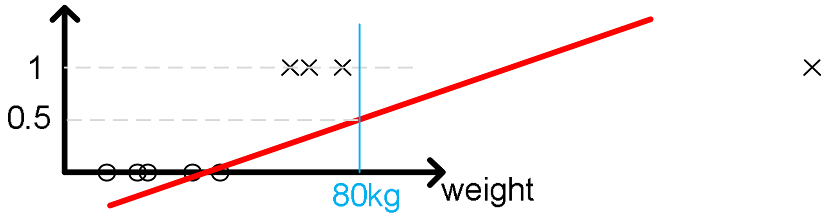


我們就可以歸納出>70kg 是男生這個結論，在目前的 test case 下看起來也沒甚麼問題

缺點：

　　易受極端值影響，容易使 LSE 直線偏移而使得 decision boundary 偏移。

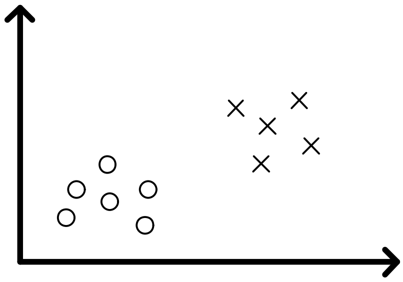
e.g.



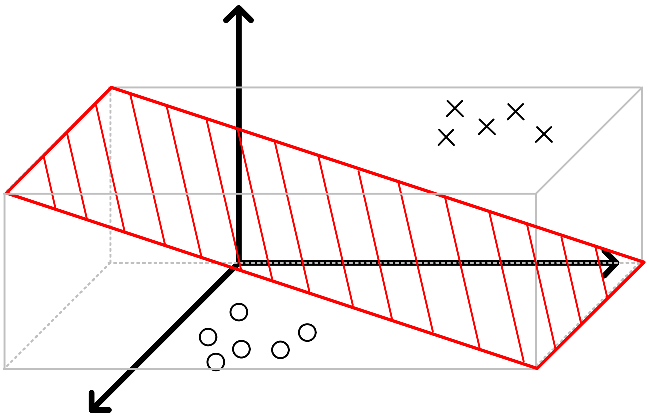
如果我們歸納出”>80kg 才是男生”，那麼 test case 下就有三個男生被我們歸類錯誤

two dimension

其實沒啥大不了的，做個延伸而已



加入 indicator function

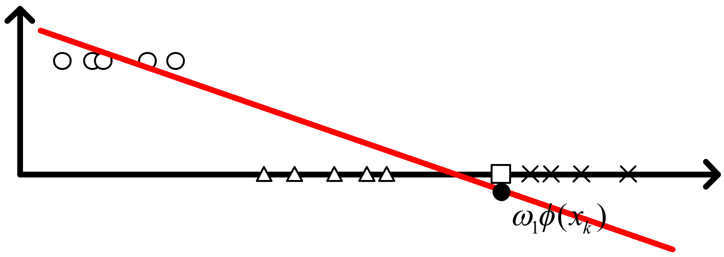


one-k-coding

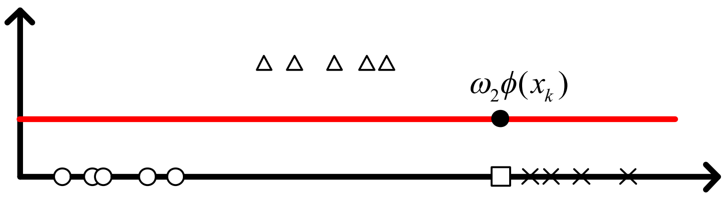
一次分類 k 個群組，我們一樣可以使用 one-coding 的方式，我們只需要分類其中一群組和其他所有群組，有 k 個群組，我們就需做$\binom{k}{2}$次

e.g.

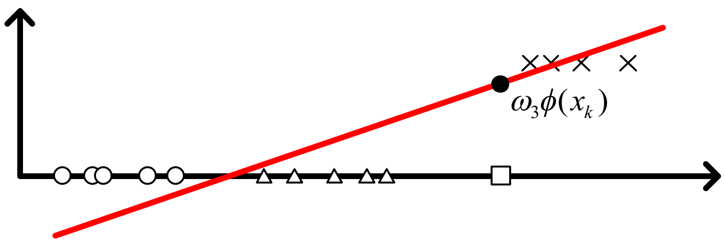
找分類 O



找分類 Δ



找分類 X



而□的最佳分類為 $\arg\max_{a}(\omega_a\phi(x_k))$ ，像是此例中□最佳分類為 X

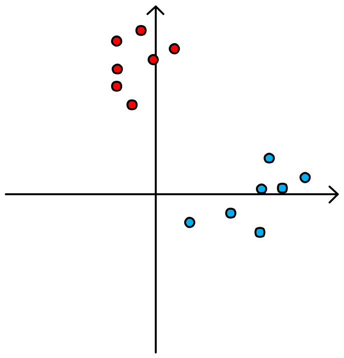
loss function

由於有易受極端值影響的缺點，故有多種不同的 loss function 被發明出來

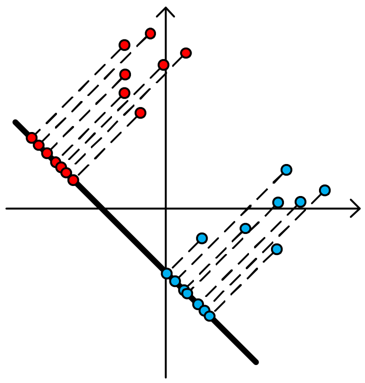
Fisher Linear Discriminant

又稱為 LDA(linear discriminant analysis)，是一種降維的分類方法，我會先說明他的概念，後面會有數學推導，數學推導上課沒講，有興趣再看就好

先來看一張圖



我們希望將這兩種顏色的點分群，Fisher Linear discriminant 是找出一個映射面，使得所有點映射到該面上後，相同的分類的點能夠盡可能靠近，不同的分群的點能夠盡可能疏離。下圖中的黑色粗線就是我們希望得到的映射面。



此映射面其實是某個空間的 eigenvector，下面就會推導該如何得到該映射平面，有興趣再看

Step1

我們的目標是找到一參數 w，則 Xw 為映射後的新平面，令 $y = xw$
我們希望將資料分成兩類，故我們需要能代表兩個分類的參數，這裡我們使用的是該分類的平均值

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

m_i 為第 i 類的 mean，Di 為該分類的集合

而投影後的平均值可表示為

$$\overline{m_i} = \frac{1}{n_i} \sum_{y \in Y_i} y = \frac{1}{n_i} \sum_{x \in D_i} xw = (\frac{1}{n_i} \sum_{x \in D_i} x)w = m_i W$$

Yi 為第 i 類經過投影後的資料點集合

Step2

我們希望同一分類中的點越集中越好，直覺上就是分類中的點的 variance 越小越好，分類中的 variance 可表示為

$$\sigma_i^2 = \sum_{y \in Y_i} (y - \overline{m_i})^2$$

同時，我們也希望兩個分類越遠越好，故我們希望最大化 $|\overline{m_1} - \overline{m_2}| = |m_1 w - m_2 w| = |(m_1 - m_2)w|$

我們將希望最大化的放分子，最小化的放分母，定義出一新的函式 $J(w)$

$$J(w) = \frac{|\overline{m_1} - \overline{m_2}|^2}{\sigma_1^2 + \sigma_2^2} = \frac{|(m_1 - m_2)w|^2}{\sum_{y \in Y_1} (y - \overline{m_1})^2 + \sum_{y \in Y_2} (y - \overline{m_2})^2} = \frac{((m_1 - m_2)w)^T ((m_1 - m_2)w)}{\sum_{x \in D_1} (xw - m_1 w)^2 + \sum_{x \in D_2} (xw - m_2 w)^2}$$

$$= \frac{w^T (m_1 - m_2)^T (m_1 - m_2) w}{\sum_{x \in D_1} w^T (x - m_1)^T (x - m_1) w + \sum_{x \in D_2} w^T (x - m_2)^T (x - m_2) w} = \frac{w^T (m_1 - m_2)^T (m_1 - m_2) w}{w^T (\sum_{x \in D_i} (x - m_1)^T (x - m_1) + \sum_{x \in D_2} (x - m_2)^T (x - m_2)) w}$$

令 $S_i = \sum_{x \in D_i} (x - m_i)^T (x - m_i)$，則

$$J(w) = \frac{w^T (m_1 - m_2)^T (m_1 - m_2) w}{w^T (S_1 + S_2) w} = \boxed{\frac{w^T S_B w}{w^T S_A w}}$$

我們希望找出 $J(w)$ 最大的 w，但是假設我們找到一個 w 有著最大的 $J(w)$，其常數倍數也都會是解 e.g.

$$J(w) = \frac{w^T S_B w}{w^T S_A w} = \frac{(2w)^T S_B (2w)}{(2w)^T S_A (2w)}$$

故我們設定一限制條件，分母固定為一常數 k，然後希望最大化分子，我們使用 Lagrange multiplier，得

$$L(w) = w^T S_B w - \lambda (w^T S_A w - k)$$

$$\frac{\partial}{\partial w} L(w) = 2 S_B w - 2\lambda S_A w = 0$$

$$\Rightarrow S_B w = \lambda S_A w$$

我們可以用特徵向量的方式求解 w，但前提是 S_A 可逆，其實 S_1,S 都是半正定矩陣，只要其任一個 variance 不為零就必定可逆，此時我們求解的就是下式的特徵向量

$$S_A^{-1} S_B w = \lambda w$$

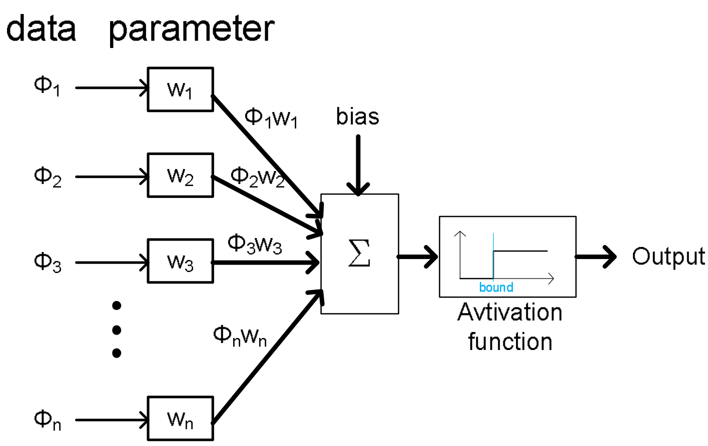
參考資料(寫得非常好且詳細)：http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/tutorials/LDA.pdf

perceptron(感知器)

參考資料(我都是看這個)：http://function1122.blogspot.tw/2010/10/perceptron-learning-algorithm.html

這裡只是將網站上的內容整理出來，但其實網站已經講的非常的白話且詳細，其實看網站就可以了。但我後面會補上個例子，想要有實際圖形化的感覺可以看我後面的例子。

perception 會根據 test data 來對一輸入 data 進行判斷的工具，其只會提供 Yes/No 兩種輸出結果，下圖是示意圖，將 input data 經過 Xw 後，若值超過一個 bound(通常都是設為 0)，輸出為 Yes(1)，若否，輸出 No(0)



圖片來源(重製)：http://aass.oru.se/~lilien/ml/seminars/2007_02_01b-Janecek-Perceptron.pdf

而因為 activation function 不是一個可微分的函式，故我們在 train data 時沒辦法使用之前使用的 MAP 方式，即將一 data 判斷為正確的分類，求其機率最大的 w，因為我們會在之中使用微分=0 的方法，這裡我們只能使用 gradient descent 方式。

因為我們還是沒辦法對 activate function 做微分，故我們只在 activation function 之前做 gradient descent，運作的方式簡單來說就是先隨便給定一個參數列(看 basis 有幾維就有幾個參數)，如此可得到一 regression function，每次輸入一筆 data，若預測成功就不會做任何事，如果預測和實際不符，就會利用 Gradient descent 來修正，此方法和牛頓法一樣，會逐漸往最佳的 regression 方向前進，最差也只是不會更動 regression function 而已。

判斷是否預測成功的方法就是代入 regression function，若大於 0 則輸出 Yes，若小於等於 0 就會輸出 No。(這裡就是一個 activate function)

假定我們的參數列為 $\mathbf{w}$ ，$\mathbf{w}_{\mathbf{n+1}}$ 為此次修正後的新參數列，$\mathbf{w}_{\mathbf{n}}$ 為之前求出的參數列

Gradient descent 式子為

$$\mathbf{w}_{\mathbf{n+1}} = \mathbf{w}_{\mathbf{n}} + \nabla f(a_n)$$

note:

Gradient descent 原本是想找山谷，所以公式原為 $\mathbf{w}_{\mathbf{n+1}} = \mathbf{w}_{\mathbf{n}} - \nabla f(a_n)$ ，但這裡是想找山峰(極大值)，故用+號

$f(a_n)$ 為我們的 regression function，我們只有在預測失敗時才需要修正參數列，故我們將 $f(a_n)$ 表示為 $f(a_n) = Xwt$

$$\nabla f(a_n) = \frac{\partial}{\partial w} f(a_n) = Xt$$

故

$$\mathbf{w}_{\mathbf{n+1}} = \mathbf{w}_{\mathbf{n}} - Xt$$

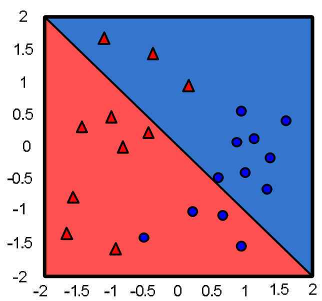
當預測失敗時，實際為 Yes 時，t=1，實際為 No 時，t=-1，預測成功時則不做修正

note:
想一下為何 t 要這樣設定，因為若我們猜測錯誤但實際為 Yes 時，代表代入 regression function 的值太小了(<0)，以至於輸出為 No，所以我們需要”往正面”方向移動，故 t 為 1，若猜測錯誤但實際為 No 時，要”往負面”方向移動，故 t 為-1

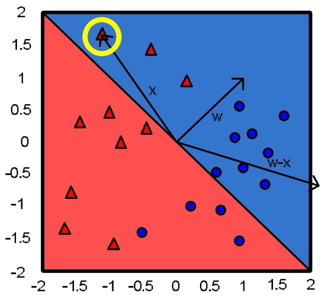
舉個例子應該更清楚，我以這個網站舉的例來說明 perceptron 的概念，但我會說明的詳細些
http://cpmarkchang.logdown.com/posts/189108-machine-learning-perceptron-algorithm

initial
假設空間為二維(有兩個評判標準 x,y)，假設 w1=1,w2=1，regression function 為

$w_1 x + w_2 y = x + y = 0$ 得到的分類為下圖

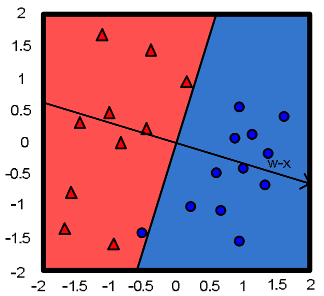


first iteration
隨便拿一預測錯誤的點出來，假設拿到分類錯誤的紅點



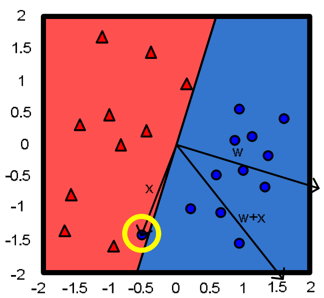
w 是我們的參數列向量，剛好就是 regression function 的法向量，x 是該 test data 的向量，因為預測為 No，$\mathbf{w}_{n+1}$ 為 $\mathbf{w}_n - X$

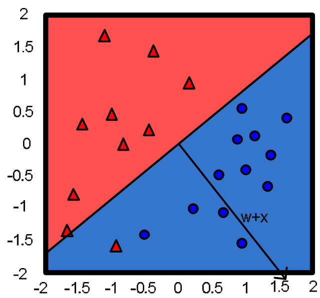
修正參數列



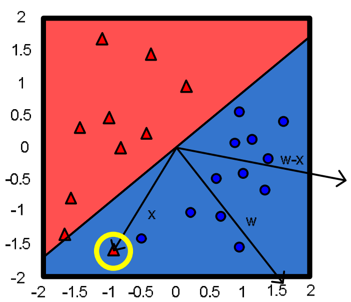
Second iteration

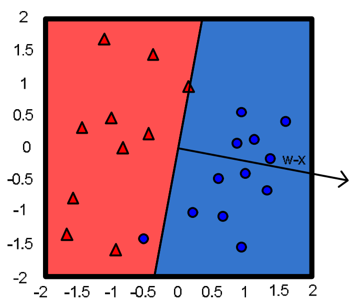
w 是我們的參數列向量，剛好就是 regression function 的法向量，x 是該 test data 的向量，因為預測錯誤且預測為 Yes，$\mathbf{w}_{n+1}$ 為 $\mathbf{w}_n + X$



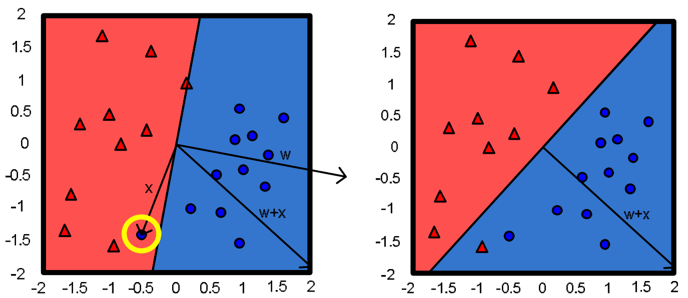


Third iteration

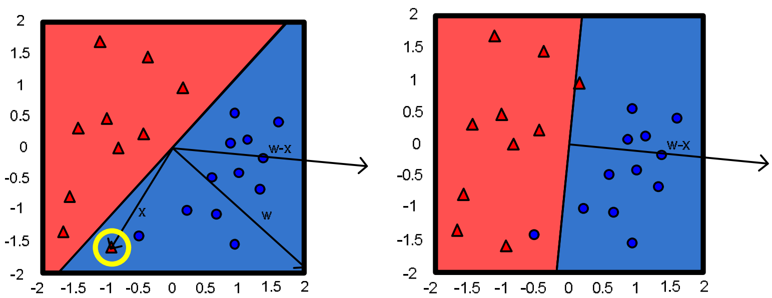




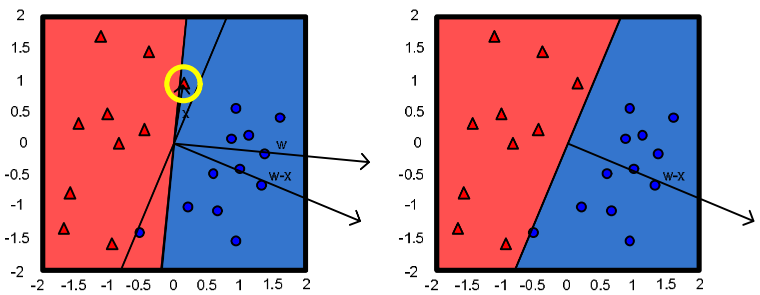
forth iteration



fifth iteration

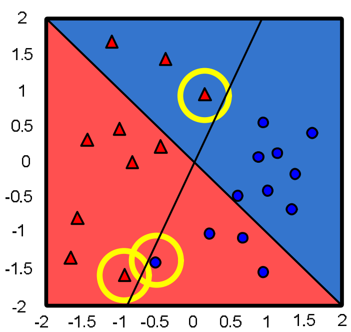


sixth



分類完成！

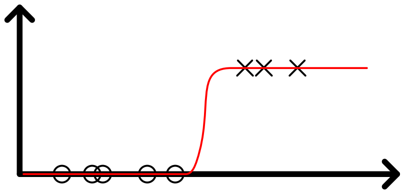
note(一張一張圖畫完的感想):重點是要將最 critical 的點都做過修正之後，通常就會分類成功了



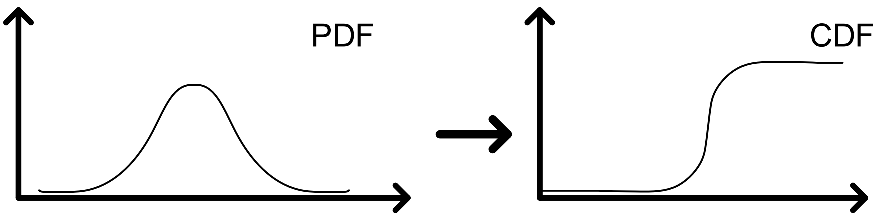
想要有代數感覺的話，可以看這份
http://aass.oru.se/~lilien/ml/seminars/2007_02_01b-Janecek-Perceptron.pdf

Logistic regression

因為 activate function 沒有好的數學性質，我們希望找到近似於 activate function 的函數，我們就可以做整塊的 MAP，這裡要介紹的就是 sigmoid function



1. CDF of Gaussian



Gaussian distribution 的 CDF 形式為 $\frac{1}{2}\left(1+erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$

其中

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

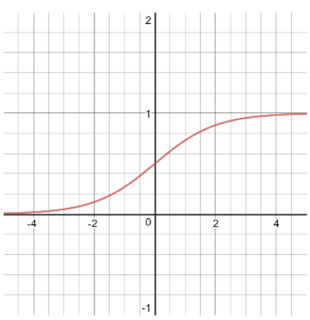
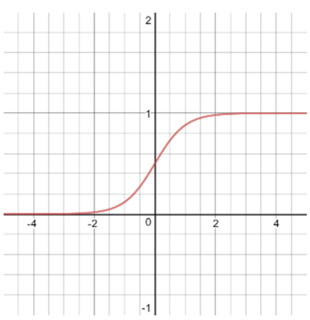
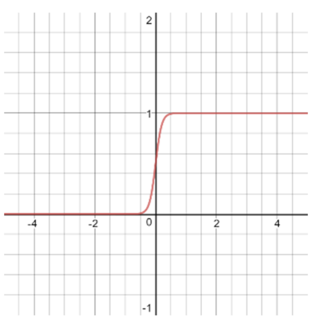
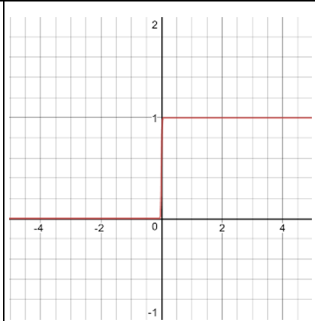
參數為 μ 及 σ ，注意，這裡的 μ 及 σ 都只是用來表示的 CDF function 而已，和 train data 沒有任何關係。是我們想要結果，我們可以給定我們想要的 μ 及 σ

2. logistic function

函數的外觀和 CDF 非常的類似，數學形式為

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

不同的 k 的 logistic function

k=1	k=2	k=10	k=100
			

Probability point of view

上課中只有推導要如何找 MLE，MAP 老師說太複雜就沒推導了
參考資料：
http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/logistic-regression-note.pdf
http://web.ntpu.edu.tw/~ccw/statmath/M_logistic.pdf

由於我們目前的分類都是二元的，故我們可以假設其為 Bernoulli 分布

$$y_i \sim Bernoulli(f(Xw))$$

MLE:

$$\arg\max_{\mathbf{w}} P(D|\theta)$$
$$= \prod_i \left(\frac{1}{1+e^{-x_i\mathbf{w}}}\right)^{y_i} \left(\frac{e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}}\right)^{(1-y_i)}$$

一樣的，我們先取 log 後，變成累加再微分=0 找極大值

$$J = \sum_{i=1}^{n} \left(y_i \log\left(\frac{1}{1+e^{-x_i\mathbf{w}}}\right) + (1-y_i)\log\left(\frac{e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}}\right)\right)$$

w 是一個 vector，形式為 $\begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_D \end{bmatrix}$，我們做微分時一次只看一個 term

for the first term

$$\frac{\partial}{\partial w_j}\log\left(\frac{1}{1+e^{-x_i\mathbf{w}}}\right) = \frac{-\partial}{\partial w_j}\log(1+e^{-x_i\mathbf{w}}) = \frac{-1}{1+e^{-x_i\mathbf{w}}}(-x_{ij})e^{-x_i\mathbf{w}} = \frac{x_{ij}e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}}$$

x_{ij} 為第 i 個 outcome 中的第 j 個 basis

會有 x_{ij} 出現是因為

$$\frac{\partial}{\partial w_j}x_i\mathbf{w} = \begin{bmatrix} 1 & x_i & \dots & x_i^{d-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix} = \frac{\partial}{\partial w_j}(w_1 + w_2x_i + w_3x_i^2 + \dots + w_jx_i^{j-1} + \dots) = \frac{\partial}{\partial w_j}w_jx_i^{j-1} = x_i^{j-1}$$

上式的 design matrix 只是隨便舉例，最後剩下的 x_i^{j-1} 是 x_i 中的第 j 項，故記為 x_{ij}

for the second term

$$\frac{\partial}{\partial w_j}(1-y_i)\log(\frac{e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}})=(1-y_i)\frac{\partial}{\partial w_j}(\log e^{-x_i\mathbf{w}}+\log(\frac{1}{1+e^{-x_i\mathbf{w}}}))=(1-y_i)(-x_{ij}+x_{ij}\frac{e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}})$$

$$=(1-y_i)\frac{-x_{ij}}{1+e^{-x_i\mathbf{w}}}$$

$$\frac{\partial J}{\partial w_j}=\sum_{i=1}^{n}(y_i\frac{x_{ij}e^{-x_i\mathbf{w}}}{1+e^{-x_i\mathbf{w}}}-(1-y_i)\frac{x_{ij}}{1+e^{-x_i\mathbf{w}}})$$
$$=\sum_{i=1}^{n}(y_ix_{ij}(1-\frac{1}{1+e^{-x_i\mathbf{w}}})-(1-y_i)\frac{x_{ij}}{1+e^{-x_i\mathbf{w}}})=\sum_{i=1}^{n}(y_ix_{ij}-\frac{y_ix_{ij}}{1+e^{-x_i\mathbf{w}}}-\frac{x_{ij}}{1+e^{-x_i\mathbf{w}}}+\frac{y_ix_{ij}}{1+e^{-x_i\mathbf{w}}})$$
$$=\sum_{i=1}^{n}(y_ix_{ij}-\frac{x_{ij}}{1+e^{-x_i\mathbf{w}}})=\sum_{i=1}^{n}(x_{ij}(y_i-\frac{1}{1+e^{-x_i\mathbf{w}}}))$$

我們欲求解 $\frac{\partial J}{\partial w_j}=\sum_{i=1}^{n}(x_{ij}(y_i-\frac{1}{1+e^{-x_i\mathbf{w}}}))=0$ ，如果寫成 matrix form

令 $\frac{\partial J}{\partial \mathbf{w}}=\begin{bmatrix}\frac{\partial J}{\partial w_1}\\ \frac{\partial J}{\partial w_2}\\ \dots\\ \frac{\partial J}{\partial w_D}\end{bmatrix}, A=\begin{bmatrix}x_{11} & x_{12} & \dots & x_{1d}\\ x_{21} & x_{22} & \dots & x_{2d}\\ \dots & \dots & \dots & \dots\\ x_{n1} & x_{n2} & \dots & x_{nd}\end{bmatrix}$

$$\frac{\partial J}{\partial \mathbf{w}}=\begin{bmatrix}\frac{\partial J}{\partial w_1}\\ \frac{\partial J}{\partial w_2}\\ \dots\\ \frac{\partial J}{\partial w_D}\end{bmatrix}=A^T(\frac{1}{1+e^{-X_i\mathbf{w}}}-\mathbf{y})$$

note: 找極值取 gradient=0 時，加一負號不影響找極值。

但此方程式是一非線性方程式，故我們無法得到 close form，我們還是只能用 steepest gradient descent 來逼近解。

再講一次 Gradient descent 步驟：
核心公式：

$$\mathbf{w_{n+1}}=\mathbf{w_n}+\nabla_{\mathbf{w}}J$$

$$\mathbf{w_{n+1}} = \mathbf{w_n} + \nabla_{\mathbf{w}} J = \mathbf{w_n} + A^T\left(\frac{1}{1+e^{-X_i\mathbf{w}}} - y_i\right)$$

一直重複直到 $\mathbf{w_{n+1}}$ 逼近至一定值，也就是 $\frac{\partial J}{\partial \mathbf{w}} = A^T\left(\frac{1}{1+e^{-X_i\mathbf{w}}} - y_i\right) = 0$ 那刻

下堂課會提到使用 Newton’s method 來更快的收斂，會用到 Hessian matrix。