● Noise and Error

1. ideal mini-target

首先這裡我令

$$\text{sign}(0) = 1$$

從題目給的式子可以知道

$$f_{0/1}(\mathbf{x}) = \underset{y \in \{-1,+1\}}{\text{argmax}} P(y|\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right)$$

等式右邊在檢查 +1 的機率有沒有大於或等於 $\frac{1}{2}$。會有這樣的等式我們可以從「期望錯誤」來看看，首先假設$P(+1|\mathbf{x}) = p$，p 介於 0 到 1 之間，此時如果我們得到一個 $\mathbf{x}$ ，去猜他的y是$+1$跟-1的期望錯誤是：

$$\begin{cases} 1-p & 猜 y = +1 \\ p & 猜 y = -1 \end{cases}$$

所以如果我們希望猜$y = -1$的期望錯誤大於等於猜$y = +1$，可以得到：

$$p \geq 1 - p \Rightarrow p \geq \frac{1}{2}$$

也就是說，只要$P(+1|\mathbf{x}) \geq \frac{1}{2}$，去猜$y = +1$可以得到較小的期望錯誤，就可以得到上面公式中的：

$$\text{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right)$$

對於 CIA 的 Error Function，題目希望 False Positive 比 False Negative 更重要 1000 倍，所以一樣假設$P(+1|\mathbf{x}) = p$，p 介於 0 到 1 之間，此時如果我們得到一個 $\mathbf{x}$ ，去猜他的y是$+1$跟-1的期望錯誤是：

$$\begin{cases} 1000 \times (1-p) & 猜 y = +1 \\ p & 猜 y = -1 \end{cases}$$

可以發現因為 False Positive 比 False Negative 更重要 1000 倍，也就是說猜$y = +1$的期望錯誤也就要多乘上那 1000 倍；最後跟上面一樣，我們希望猜$y = -1$的期望錯誤大於等於猜$y = +1$，可以得到：

$$p \geq 1000 \times (1-p)$$

$$\Rightarrow p \geq \frac{1000}{1001}$$

因此我們可以知道：

$$f_{CIA}(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \frac{1000}{1001}\right)$$

2. a noisy test environment

原本在無干擾(noise)的環境中：

$$E_{out}(g) = \underset{\mathbf{x}\sim P(\mathbf{x})}{\mathcal{E}} [\![g(\mathbf{x}) \neq f(\mathbf{x})]\!]$$

或者可以說：

$$Correct_{out}(g) = \underset{\mathbf{x}\sim P(\mathbf{x})}{\mathcal{E}} [\![g(\mathbf{x}) = f(\mathbf{x})]\!] = 1 - E_{out}(g)$$

此時換到有干擾的環境，有 ϵ 的機率會翻轉原本的結果，因此可以知道其錯誤就會變成：

$$\underbrace{E_{out}(g)(1-\epsilon)}_{\text{原本對的有}(1-\epsilon)\text{變成錯的}} + \underbrace{\big(1 - E_{out}(g)\big)\epsilon}_{\text{原本的部分只剩下}(1-\epsilon)\text{保持錯誤}}$$

$$= E_{out}(g) - E_{out}(g)\epsilon + \epsilon - E_{out}(g)\epsilon = E_{out}(g) + \epsilon - 2\epsilon E_{out}(g)$$

● Linear Regression

3. $h(x) = wx$

$$\frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(wx_n - y_n)^2$$

接著對 w 微分並找到等於 0 的值：

$$\frac{d}{dw}\left(\frac{1}{N}\sum_{n=1}^{N}(wx_n - y_n)^2\right) = \frac{1}{N}\sum_{n=1}^{N}2(wx_n - y_n)x_n = 0$$

$$\Rightarrow \sum_{n=1}^{N} wx_n^2 - \sum_{n=1}^{N} x_n y_n = 0 \Rightarrow w\sum_{n=1}^{N} x_n^2 = \sum_{n=1}^{N} x_n y_n$$

$$\Rightarrow w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$$

此時再對 w 微分：

$$\frac{d}{dw}\left(\frac{1}{N}\sum_{n=1}^{N}2(wx_n - y_n)x_n\right) = \frac{d}{dw}\left(\frac{1}{N}\sum_{n=1}^{N}2wx_n^2 - 2x_n y_n\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}2x_n^2$$

因為題目有確保 $\sum_{n=1}^{N} x_n^2$ 不等於 0，所以可以知道二階微分恆大於 0，也就是說 $w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$ 確實是極小值。

4. 小標題二

5. output transformation

根據講義的內容：

$$E_{in}(\mathbf{w}_{\mathrm{LIN}}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = \frac{1}{N}(\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y})$$

$$\nabla E_{in}(\mathbf{w}_{LIN}) = \frac{2}{N}(\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y})$$

$$\mathbf{w}_{LIN} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

將 $y_n' = ay_n + b$ 代入式子：

$$\mathbf{w}_{LIN}' = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}(\mathbf{ay} + \mathbf{b})$$

$$\Rightarrow \mathbf{w}_{LIN}' = a\mathbf{w}_{LIN} +$$

- More on Linear Models

6. Hessian Matrix

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1 + \exp(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})\right)$$

如果只求一次微分可以知道：

$$\frac{\partial}{\partial w_i}E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\frac{-y_n x_{n,i}}{1 + \exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})}$$

接著使用微積分把海森矩陣展開：

$$\mathrm{Hessian}\big(E_{in}(\mathbf{w})\big) =$$

$$\begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N}\frac{\exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})\big(y_n^2 x_{n,0}x_{n,0}\big)}{[1+\exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})]^2} & \cdots & \frac{1}{N}\sum_{n=1}^{N}\frac{exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})\big(y_n^2 x_{n,d}x_{n,0}\big)}{[1+exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})]^2} \\ \vdots & \ddots & \vdots \\ \frac{1}{N}\sum_{n=1}^{N}\frac{exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})\big(y_n^2 x_{n,0}x_{n,d}\big)}{[1+exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})]^2} & \cdots & \frac{1}{N}\sum_{n=1}^{N}\frac{exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})\big(y_n^2 x_{n,d}x_{n,d}\big)}{[1+exp(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x_n})]^2} \end{bmatrix}$$

使用 Logistic Function 的特性、$y_1^2 = 1$，以及將 Σ 的部分改成用矩陣的方式來表示，把矩陣化簡：

中間的 D 矩陣為：

$$\begin{bmatrix} \frac{1}{N}\theta(y_1\mathbf{w}^{\mathrm{T}}\mathbf{x_1})\theta(-y_1\mathbf{w}^{\mathrm{T}}\mathbf{x_1}) & \cdots & 0 \\ \vdots & \frac{1}{N}\theta(y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)\theta(-y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i) & \vdots \\ 0 & \cdots & \frac{1}{N}\theta(y_N\mathbf{w}^{\mathrm{T}}\mathbf{x}_N)\theta(-y_N\mathbf{w}^{\mathrm{T}}\mathbf{x}_N) \end{bmatrix}$$

可以注意到$\theta(y_i\mathbf{w^T x}_i)\theta(-y_i\mathbf{w^T x}_i)$不管 y_i 是+1還是−1，$\theta(\mathbf{w^T x}_i)$ 跟 $\theta(-\mathbf{w^T x}_i)$ 兩個都一定會出現，所以可以把 y_i 省略掉；接著換成題目所要的形式：

$$\begin{bmatrix} \frac{1}{N}h_t(\mathbf{x}_1)h_t(-\mathbf{x}_1) & \cdots & 0 \\ \vdots & \frac{1}{N}h_t(\mathbf{x}_i)h_t(-\mathbf{x}_i) & \vdots \\ 0 & \cdots & \frac{1}{N}h_t(\mathbf{x}_N)h_t(-\mathbf{x}_N) \end{bmatrix}$$

而位於左右兩邊的 $\mathbf{X}^T$跟 $\mathbf{X}$：

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_{1,0}} & \cdots & \mathbf{x}_{\mathrm{N},\mathbf{0}} \\ \vdots & \ddots & \vdots \\ \mathbf{x_{1,d}} & \cdots & \mathbf{x}_{\mathrm{N},\mathbf{d}} \end{bmatrix}$$

7. truncated squared loss

$$\nabla \mathrm{err}(\mathbf{w^T x}, y) = \begin{cases} 0 & 1 \le y\boldsymbol{w^T x} \\ 2(y\mathbf{w^T x} - 1)\mathbf{xy} & 1 > y\mathbf{w^T x} \end{cases}$$

所以可以知道 truncated squared loss 的 SGD 長的如下：

$$\begin{cases} \mathbf{w}_{\boldsymbol{t+1}} \leftarrow \mathbf{w}_{\boldsymbol{t}} + \boldsymbol{\eta} \times 0 \times (\mathbf{x}_{\boldsymbol{n}}\boldsymbol{y}_{\boldsymbol{n}}) & 1 \le y\boldsymbol{w^T x} \\ \mathbf{w}_{\boldsymbol{t+1}} \leftarrow \mathbf{w}_{\boldsymbol{t}} + \boldsymbol{\eta} \times\ 2(1 - y_n\mathbf{w}_{\boldsymbol{t}}^{\mathbf{T}}\mathbf{x}_{\boldsymbol{n}})(\mathbf{x}_{\boldsymbol{n}}\boldsymbol{y}_{\boldsymbol{n}}) & 1 > y\mathbf{w^T x} \end{cases}$$

而原版的 PLA 是：

$$\mathbf{w}_{\boldsymbol{t+1}} \leftarrow \mathbf{w}_{\boldsymbol{t}} + 1 \times [\![\boldsymbol{y}_{\boldsymbol{n}} \neq \boldsymbol{sign}(\mathbf{w}_{\boldsymbol{t}}^{\mathbf{T}}\mathbf{x}_{\boldsymbol{n}})]\!](\mathbf{x}_{\boldsymbol{n}}\boldsymbol{y}_{\boldsymbol{n}})$$

不同之處：

- truncated squared loss 的 SGD 是 $y\boldsymbol{w^T x} \ge \mathbf{1}$ 才不會更新，但是 PLA 只要 $y\boldsymbol{w^T x} \ge 0$ 就不會更新。
- truncated squared loss 的 SGD 只要$y\boldsymbol{w^T x} < \mathbf{1}$就會開始更新但是 PLA 要等到 $y\boldsymbol{w^T x} < 0$ 才開始更新。
- truncated squared loss 的 SGD 只要$y\boldsymbol{w^T x}$ 越小，$\boldsymbol{\eta} \times\ 2(1 - y_n\mathbf{w}_{\boldsymbol{t}}^{\mathbf{T}}\mathbf{x}_{\boldsymbol{n}})$的部分就越大，但是 PLA 都固定是 1。

相同之處：

- 只要 $y\boldsymbol{w^T x} \ge \mathbf{1}$ ，truncated squared loss 的 SGD 跟 PLA 就都不會更新，不像 square error 或 cross entropy 會更新。

- Multinomial Logistic Regression

8. Multinomial Logistic Regression

$$E_{in}(W) = \frac{1}{N}\sum_{n=1}^{N}\left(-\sum_{k=1}^{K}\left([\![\boldsymbol{y_n} = \boldsymbol{k}]\!]\ln\frac{\exp(\mathbf{w_y^T x_n})}{\sum_{i=1}^{K}\exp(\mathbf{w_i^T x_n})}\right)\right)$$

先將 E_{in} 化簡：

$$E_{in}(W) = \frac{1}{N}\sum_{n=1}^{N}\left(ln\left(\sum_{i=1}^{K}\exp(\mathbf{w_i^T x_n})\right) - \mathbf{w_y^T x_n}\right)$$

然後我們先只對第 i 個權重做偏微分：

$$\frac{\partial}{\partial \mathbf{w_i}}E_{in}(W) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\mathbf{w_i^T x_n})\mathbf{x_n}}{\sum_{i=1}^{K}\exp(\mathbf{w_i^T x_n})} - [\![\boldsymbol{y_n} = i]\!]\mathbf{x_n}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(h_i(\mathbf{x_n}) - [\![\boldsymbol{y_n} = i]\!]\right)\mathbf{x_n}$$

所以可以知道：

$$\begin{bmatrix} \dfrac{\partial}{\partial \mathbf{w_1}} \\ \vdots \\ \dfrac{\partial}{\partial \mathbf{w_K}} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N}\left(h_1(\mathbf{x_n}) - [\![\boldsymbol{y_n} = 1]\!]\right)\mathbf{x_n} \\ \vdots \\ \frac{1}{N}\sum_{n=1}^{N}\left(h_K(\mathbf{x_n}) - [\![\boldsymbol{y_n} = K]\!]\right)\mathbf{x_n} \end{bmatrix}$$

上面的矩陣是 K 個 row，$d+1$個 column，由於題目要求的$\nabla E_{in}(W)$是要轉 90 度，所以我們給他轉 90 度：

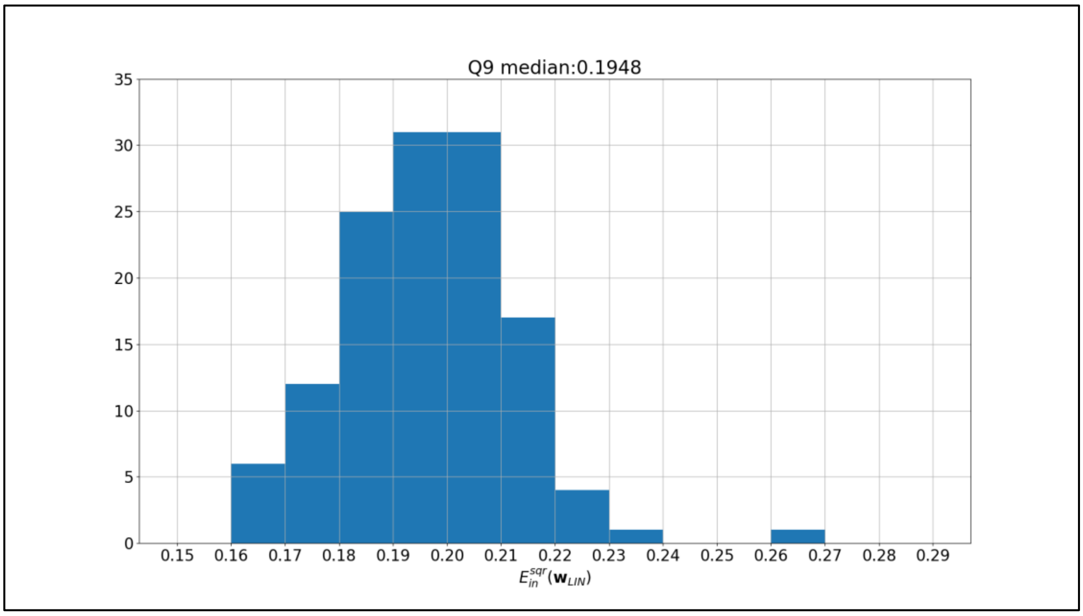
$$\nabla E_{in}(W) = \begin{bmatrix} \vdots & \cdots & \vdots \\ \dfrac{\partial}{\partial \mathbf{w_1}} & \cdots & \dfrac{\partial}{\partial \mathbf{w_K}} \\ \vdots & \cdots & \vdots \end{bmatrix} =$$

$$\begin{pmatrix} \vdots & \cdots & \vdots \\ \dfrac{1}{N}\sum_{n=1}^{N}\left(h_1(\mathbf{x_n}) - [\![\boldsymbol{y_n} = 1]\!]\right)\mathbf{x_n} & \cdots & \dfrac{1}{N}\sum_{n=1}^{N}\left(h_K(\mathbf{x_n}) - [\![\boldsymbol{y_n} = K]\!]\right)\mathbf{x_n} \\ \vdots & \cdots & \vdots \end{pmatrix}$$

這樣一來$\nabla E_{in}(W)$的維度就是$(d+1)\times K$，跟 W 一樣了。

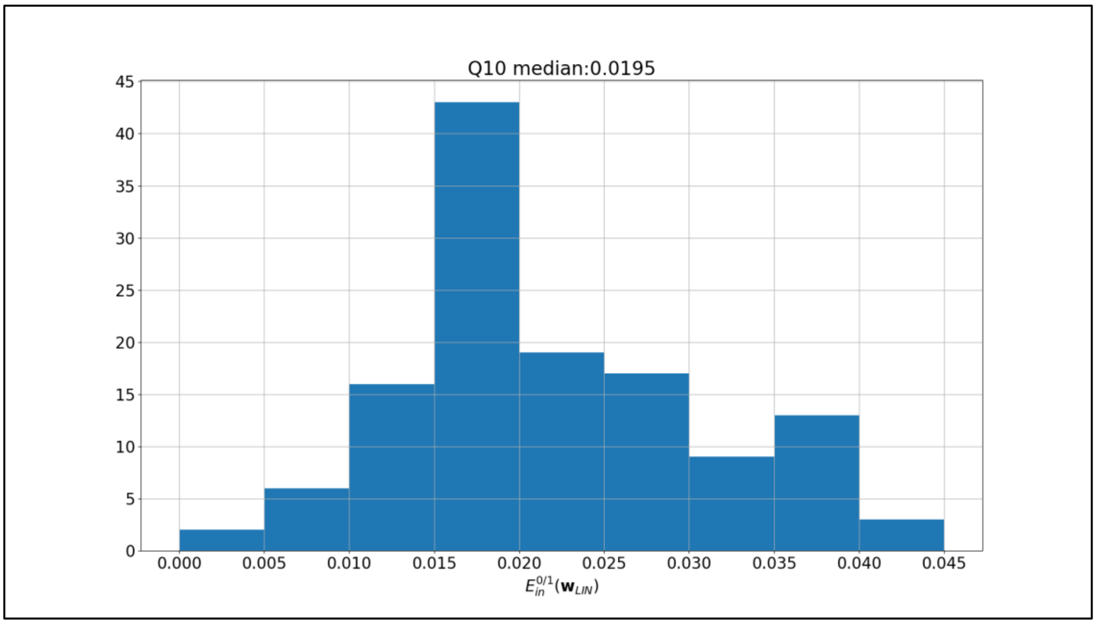
● Multinomial Logistic Regression

9. linear regression with square error



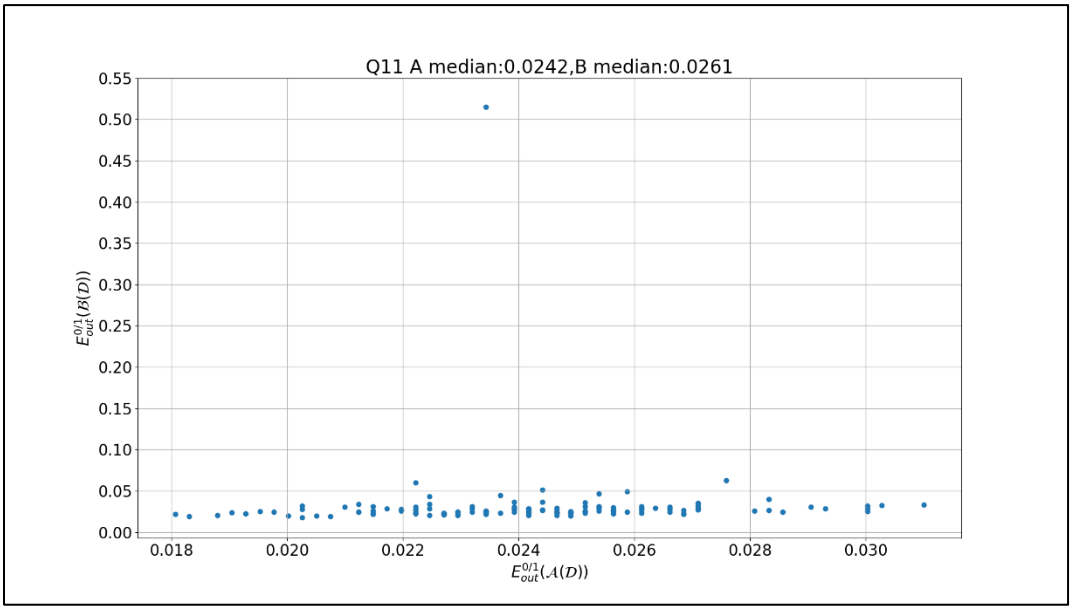
$E_{in}^{sqr}(\mathbf{w}_{LIN})$中位數是 0.1948。

10. linear regression with 0/1 error



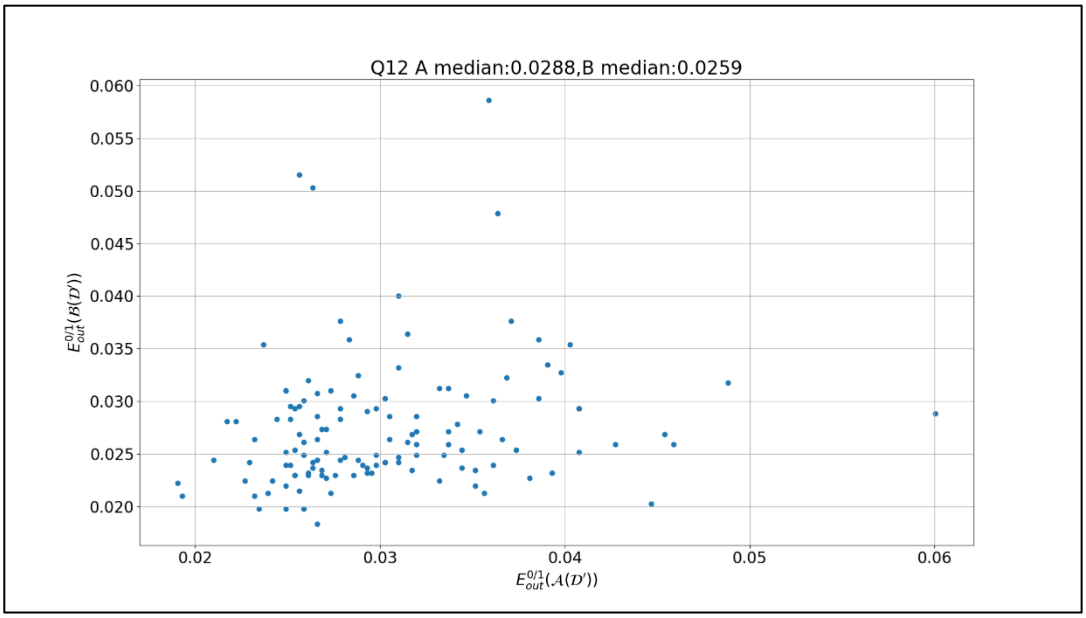
$E_{in}^{0/1}(\mathbf{w}_{LIN})$中位數是 0.0195。

11. compare LIN and LOG



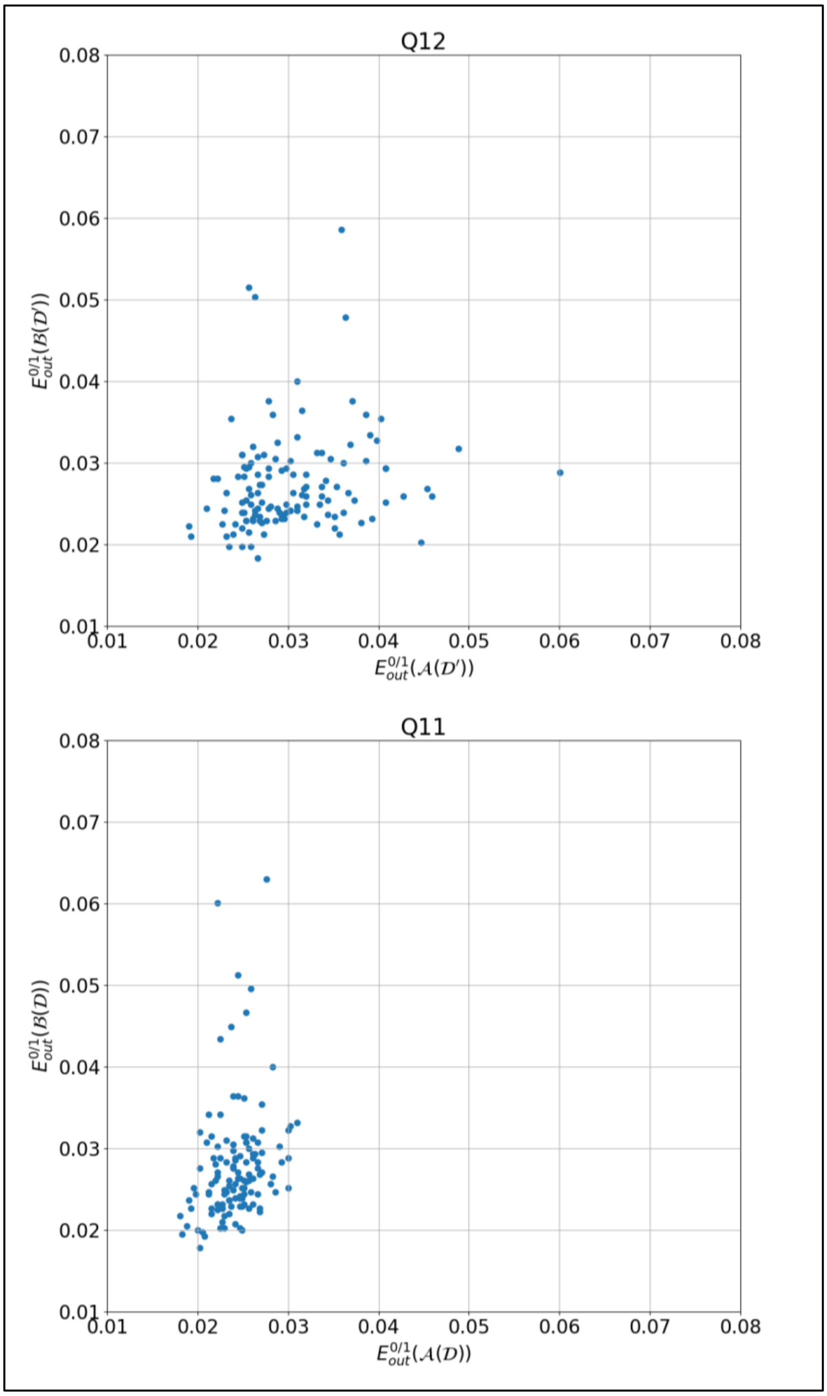
$E_{out}^{0/1}(\mathcal{A}(\mathcal{D}))$跟$E_{out}^{0/1}(\mathcal{B}(\mathcal{D}))$的中位數分別是 0.0242，0.0261

12. compare LIN and LOG with outlier



$E_{out}^{0/1}(\mathcal{A}(\mathcal{D}'))$跟$E_{out}^{0/1}(\mathcal{B}(\mathcal{D}'))$的中位數分別是 0.0288，0.0259

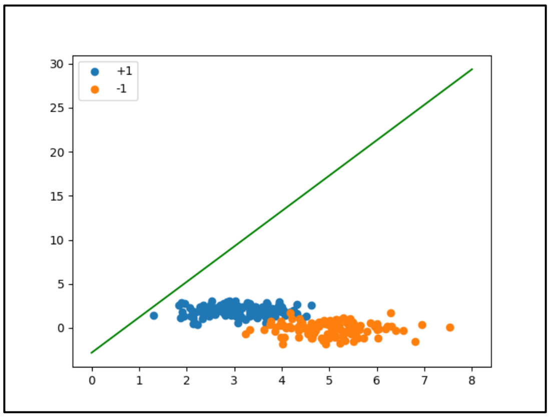
上面的圖因為 outlier 導致圖片變形很嚴重，所以我限定了範圍後得到兩軸比例一樣的圖：

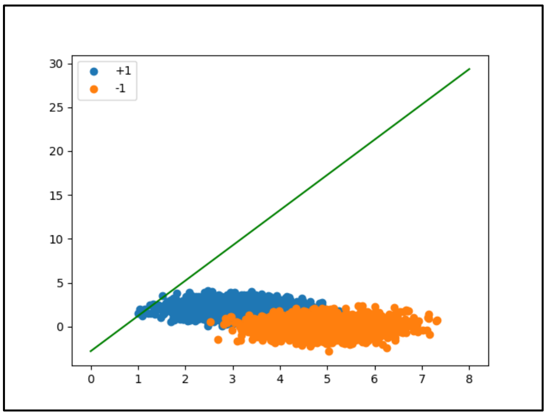


可以看到在一定的範圍內，其實兩者是相當接近的，或者說大致坐落於 $y = x$ 這條線附近，這也跟中位數相近的結果相符合。

- logistic regression 在加入 outlier 前就已經有許多點的錯誤率容易比 linear regression 來的大了，在 11 題的圖中還可以看到有一個點錯誤率比 0.5 還高，個人暫時推測應該是尚未找到最佳權重。
- 可以看到加入 outlier 後，linear regression 的錯誤率也跟著提高，顯示出 outlier 有影響到最佳權重的值。

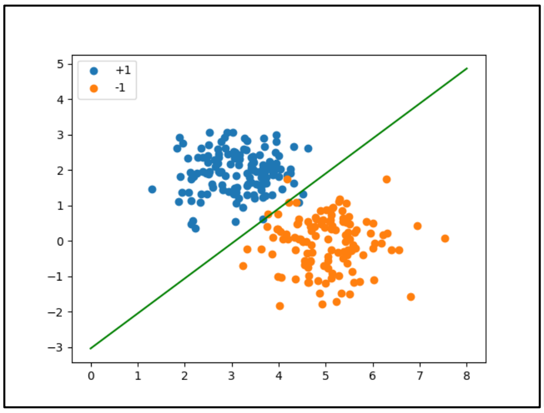
為了印證第一點，我先從尚未加入 outlier 的資料找出那個錯誤率大於 0.5 的資料出來，發現他的圖長的如下：

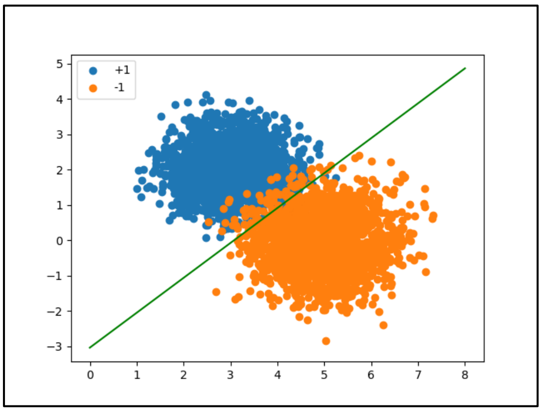




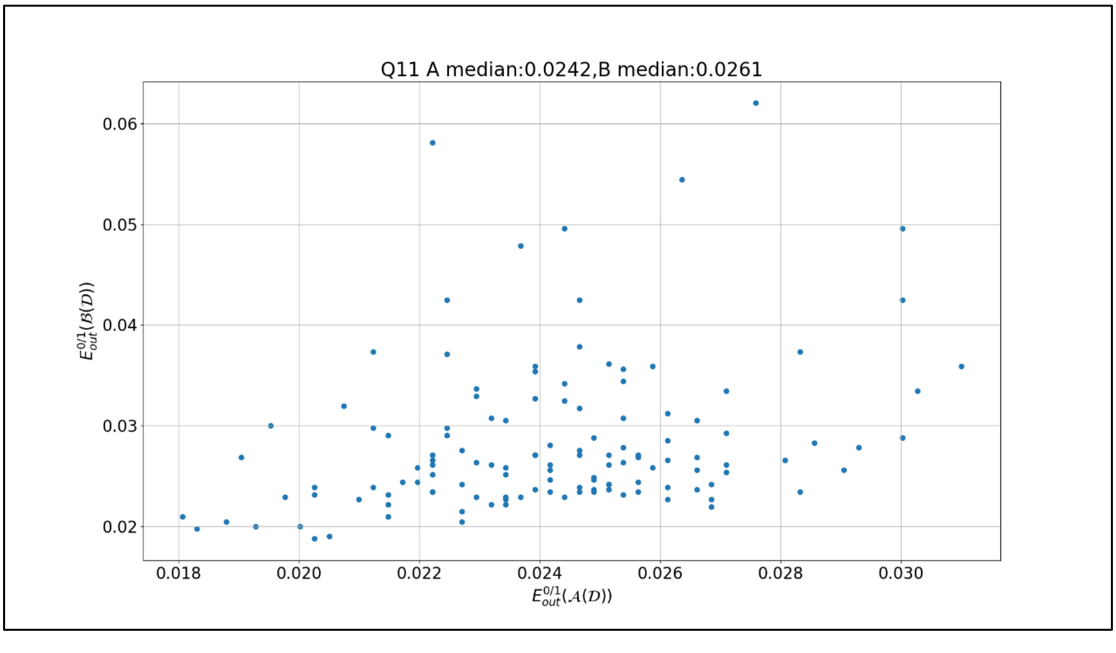
左邊是訓練資料，右邊是測試資料。

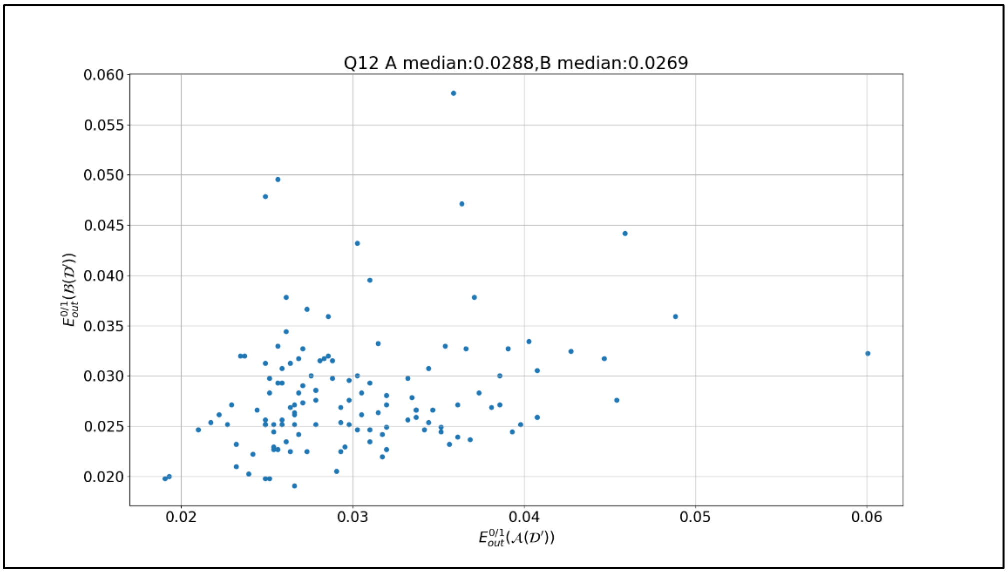
我覺得這應該是尚未找到最佳的權重所導致，於是我將訓練的次數從 500 調整到 5000，$\eta = 0.1$保持不變，訓練跟測試的結果分別如下：

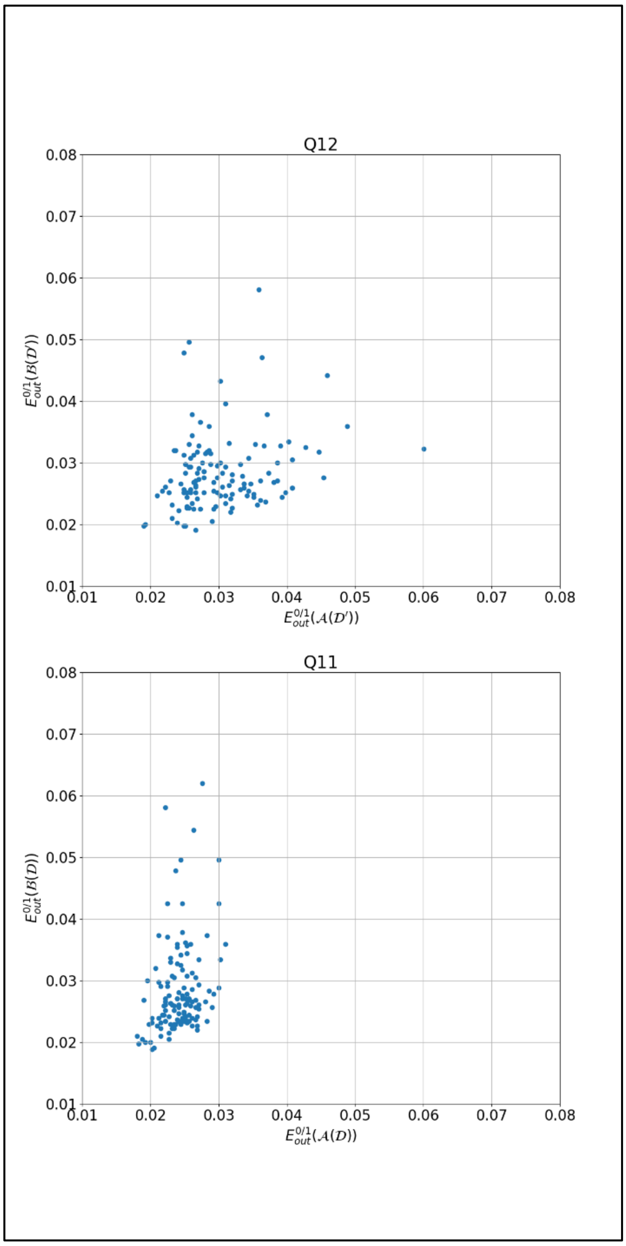




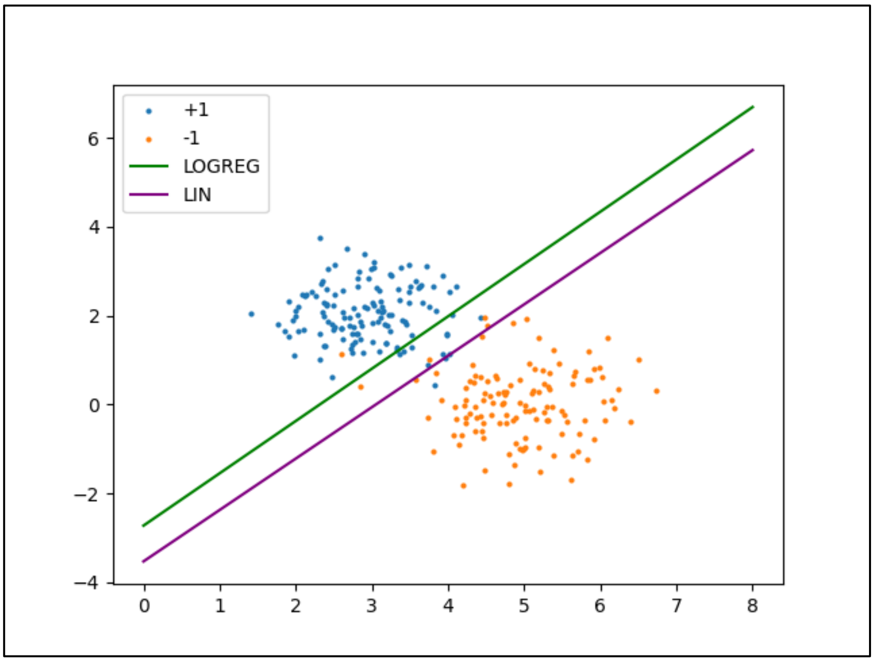
確實印證了我的猜測。所以我重新做了第 11 題跟第 12 題的模擬，將訓練次數改成 5000 次，得到如下的圖形：



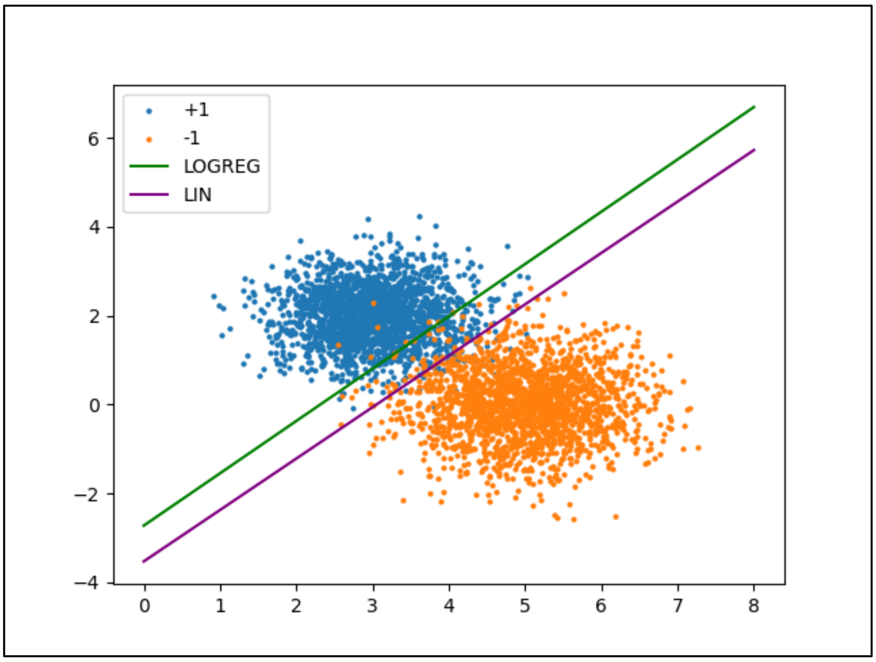

Q12 A median:0.0288,B median:0.0269
$E_{out}^{0/1}(\mathcal{B}(\mathcal{D}'))$
$E_{out}^{0/1}(\mathcal{A}(\mathcal{D}'))$


Q12
$E_{out}^{0/1}(\mathcal{B}(\mathcal{D}'))$
$E_{out}^{0/1}(\mathcal{A}(\mathcal{D}'))$
Q11
$E_{out}^{0/1}(\mathcal{B}(\mathcal{D}))$
$E_{out}^{0/1}(\mathcal{A}(\mathcal{D}))$

但是依舊發現 logistic regression 在加入 outlier 前就已經有許多點的錯誤率容易比 linear regression 來的大了，所以應該只是剛好那個錯誤率大於 0.5 的點沒有學好而已。所以接著我再從尚未加入 outlier 的資料找出一個錯誤率大於 0.05 的資料出來，發現他的圖長的如下：



可以發現在訓練資料中，綠色線雖然跟紫色線有點距離，但是他好像也能不錯的分開兩種點。這時再看測試資料：



會發現綠色線雖然在訓練資料中看起來還不錯，但是到了測試資料就有過多的點判斷錯誤了。

也就是說 logistic regression 在加入 outlier 前的錯誤率比較高，是訓練的結果，而不是訓練不夠。

那現在問題就變成為何 logistic regression 在訓練資料中會得到跟 linear regression 如此不同的權重。我認為主要是 E_{in} 的不同所導致，logistic regression 尋求的是最大化 Likelihood，或者說要最小化：

$$\frac{1}{N}\sum_{\mathrm{n=1}}^{\mathrm{N}} -ln\Big(\theta(y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n})\Big)$$

其中的 $y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n}$ 是未標準化的與超平面的距離，這個距離會再代入 logistic function 投射到 0 到 1 的範圍，再取 log，全部加起來取負號。此時先來看 linear regression 要最小化的 E_{in}：

$$\frac{1}{N}\sum_{\mathrm{n=1}}^{\mathrm{N}}(\mathbf{w}^{\mathbf{T}}\mathbf{x_n}-y_n)^2$$

這裡我讓括弧內上下同乘y_n：

$$\frac{1}{N}\sum_{\mathrm{n=1}}^{\mathrm{N}}\left(\frac{y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n}-y_n^2}{y_n}\right)^2=\frac{1}{N}\sum_{\mathrm{n=1}}^{\mathrm{N}}(y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n}-1)^2$$

一樣可以看到「未標準化的與超平面的距離」的身影

- logistic regression 會去最大化$\frac{1}{N}\sum_{\mathrm{n=1}}^{\mathrm{N}} ln\Big(\theta(y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n})\Big)$，也就是說他除了要求點要分對($\theta(y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n})$的值大於 0.5)，也會傾向於讓$\theta(y_n\mathbf{w}^{\mathbf{T}}\mathbf{x_n})$值越大越好，換句話說就是比較喜歡讓分對的點離超平面更遠一些。
- linear regression 則是追求各個點到超平面之間的距離減 1 後的平方總和越小越好。

對於同樣的一組訓練資料，logistic regression 比較偏好讓分對的點離超平面更遠一些的權重，而 linear regression 則是偏好讓分對的點離超平面(到某個位置)更近一些的權重；但是由於今天資料產生的方式是選取兩個中心點，以該中心向外輻射的常態分布，而 logistic regression 的選取方式使他容易受訓練資料的分佈影響，因而偏向其中一個中心，所以才會在測試資料中有較差的表現。

- Bonus

13. 123