* Noise and Error
  1. ideal mini-target

首先這裡我令

從題目給的式子可以知道

等式右邊在檢查 +1 的機率有沒有大於或等於 。會有這樣的等式我們可以從「期望錯誤」來看看，首先假設， 介於0到1之間，此時如果我們得到一個**，**去猜他的是跟的期望錯誤是：

所以如果我們希望的期望錯誤大於等於的期望錯誤，可以得到：

也就是說，只要，去猜可以得到較小的期望錯誤，就可以得到上面公式中的：

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， 介於0到1之間，此時如果我們得到一個**，**去猜他的是跟的期望錯誤是：

因為False Positive比False Negative更重要1000倍，所以猜的期望錯誤就要多乘上那1000倍；最後跟上面一樣，我們希望的期望錯誤大於等於，可以得到：

因此我們可以知道：

* 1. a noisy test environment

原本在無干擾(noise)的環境中：

或者可以說：

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

* Linear Regression

接著對微分並找到等於0的值：

此時再對微分：

因為題目有確保 不等於0，所以可以知道二階微分恆大於0，也就是說 確實是極小值。

1. optimal approximation

首先列出Squared Error：

接著取期望值：

在偏微分之前，先列出期望值的值：

再帶入就可以化簡成：

接著就可以開始偏微分了：

接著開始解聯立方程組：

1. 第2條乘
2. 第3條減第一條

於是可以知道最佳的權重

1. output transformation

根據講義的內容：

將 代入式子，可以得到：此時我們先將細部放大來看：

代表的第j個element；為了方便觀察，我將n改成從0開始。

所以可以知道應該會等於：

此時我們把移項到權重向量那邊：

這時我們再細部查看：

這時我們先回顧，在轉換之前，從原本的解可以知道甚麼：

從上面的等式可以找到對應的項，所以這時再回顧我們需要的內容：

可以注意到，粉紅色可以拆成左右兩個部分，其中左邊剛好就是原本的再多乘上倍，所以我們可以知道：

那接下來就是要看怎麼湊出右半部分：

這時我們可以回顧一個特性，是我們幫補上的常數項，並且值是1，所以這時候再回顧剛剛的矩陣：

可以注意到其實第一個column右邊都是1，所以可以化簡寫成：

所以我們就只取第一個column，就可以湊出我們想要的**：**

也就是第一個值為b，其他值為0的的矩陣。

因此最終我們可以知道：

* More on Linear Models

1. Hessian Matrix

如果只求一次微分可以知道：

接著使用微積分把海森矩陣展開：

使用Logistic Function的特性、，以及將的部分改成用矩陣乘法來表示，把矩陣化簡只有對角線有值的對角矩陣**D**：

可以拿掉的原因是在推導牛頓法的時候，會有一個步驟是同時移項，此時就會可以移除掉。可以注意到不管是還是， 跟 兩個都一定會出現，所以可以把省略掉；接著換成題目所要的形式：

而位於左右兩邊的 跟：

1. truncated squared loss

所以可以知道truncated squared loss的SGD長的如下：

而原版的PLA是：

不同之處：

* truncated squared loss的SGD是 才不會更新，但是PLA只要 就不會更新。
* truncated squared loss的SGD只要就會開始更新但是PLA要等到 才開始更新。
* truncated squared loss的SGD只要越小，

的部分就越大，但是PLA都固定是1。

相同之處：

* 只要，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

1. Multinomial Logistic Regression

先將化簡：

然後我們先只對第個權重做偏微分：

所以可以知道：

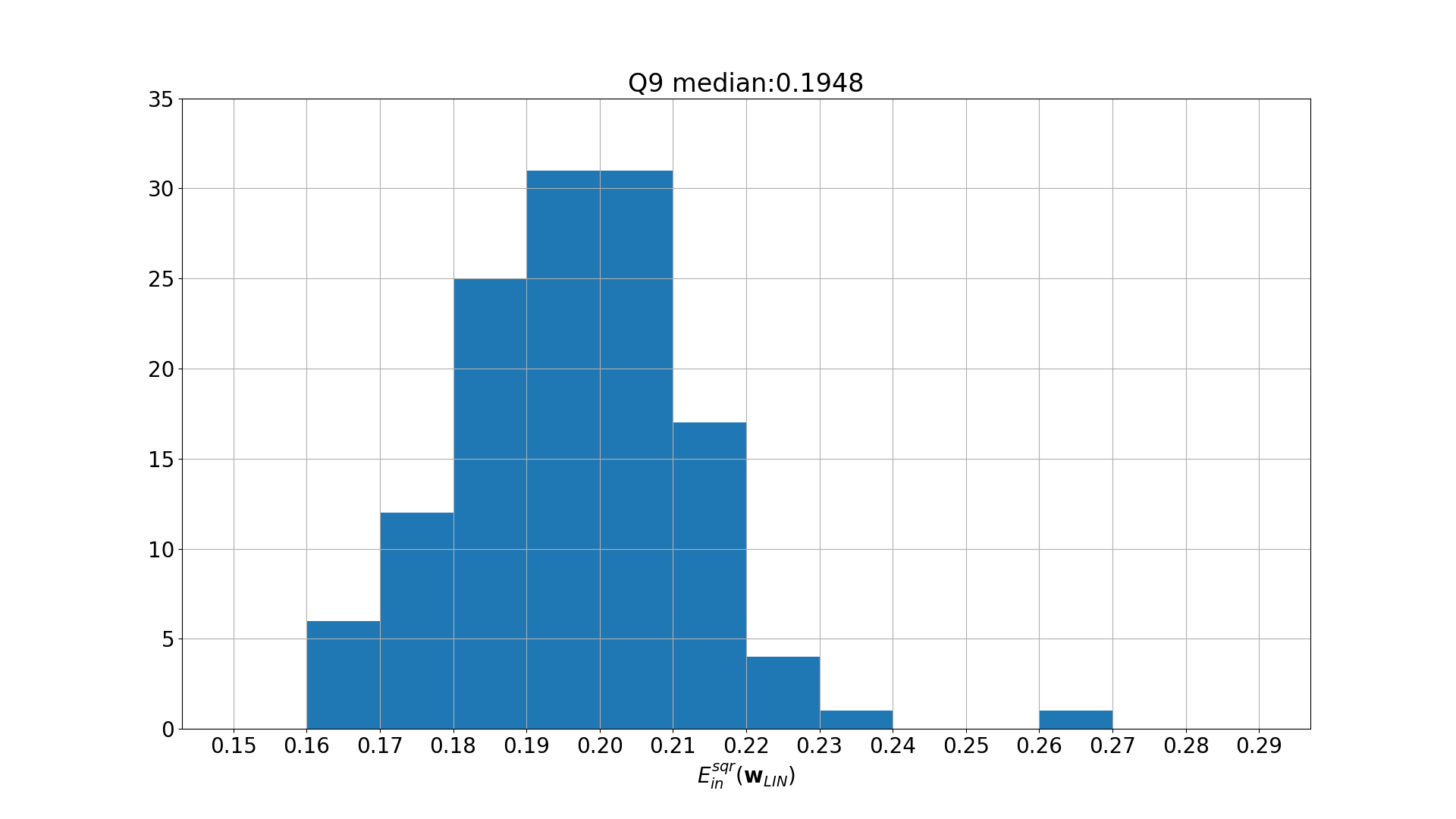
上面的矩陣是K個row，個column，由於題目要求的是要轉90度，所以我們給他轉90度：

=

這樣一來的維度就是，跟一樣了。

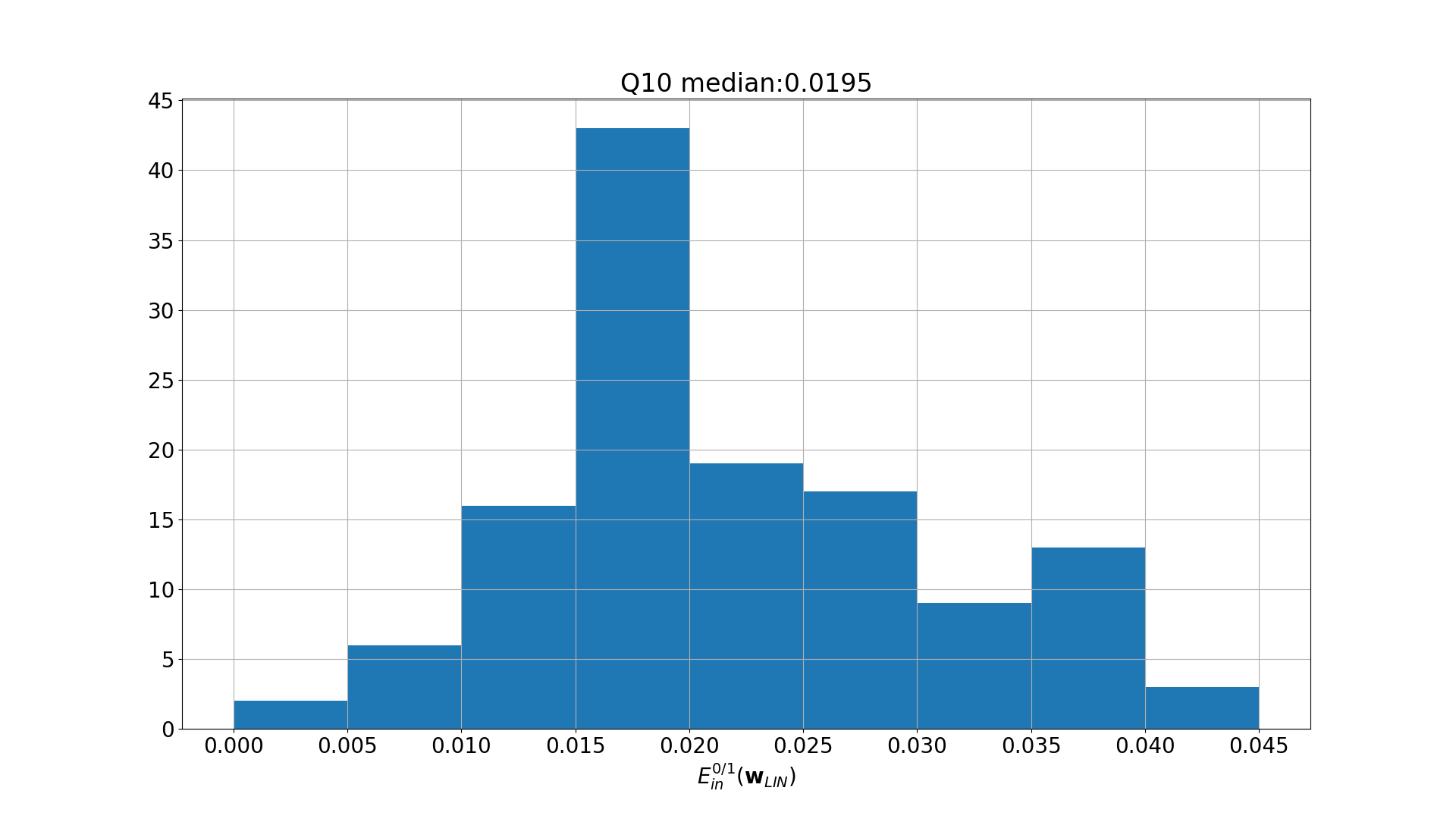
* Multinomial Logistic Regression

1. linear regression with square error



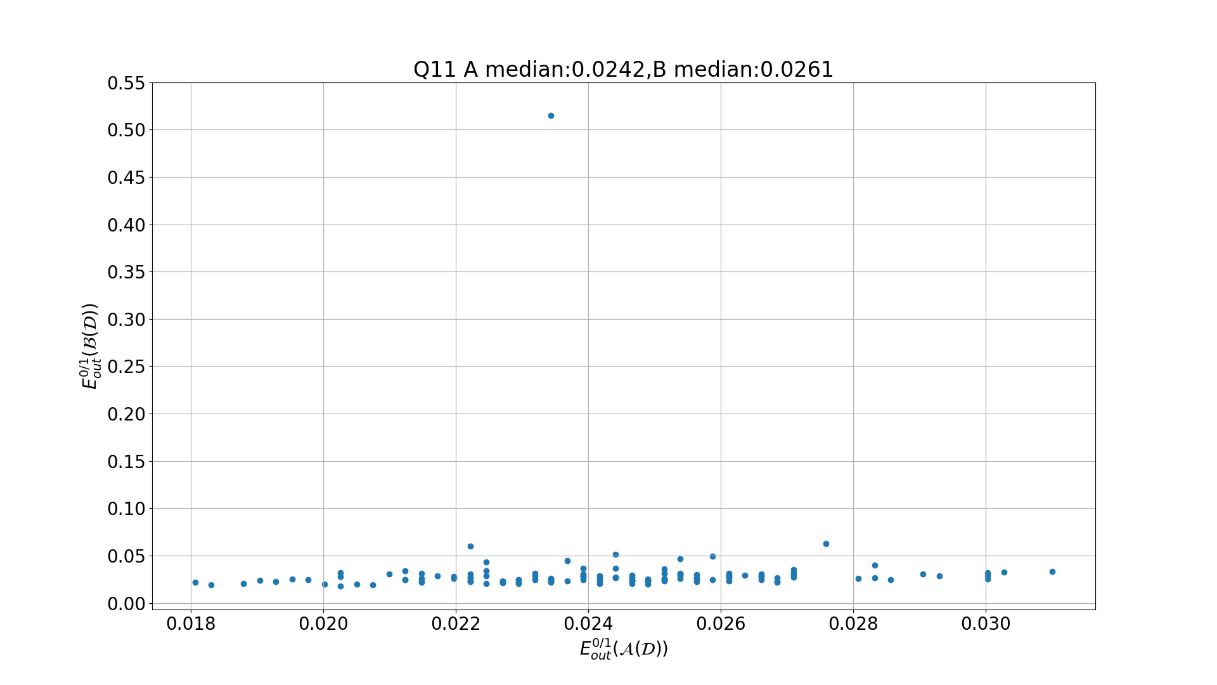
中位數是0.1948。

1. linear regression with 0/1 error



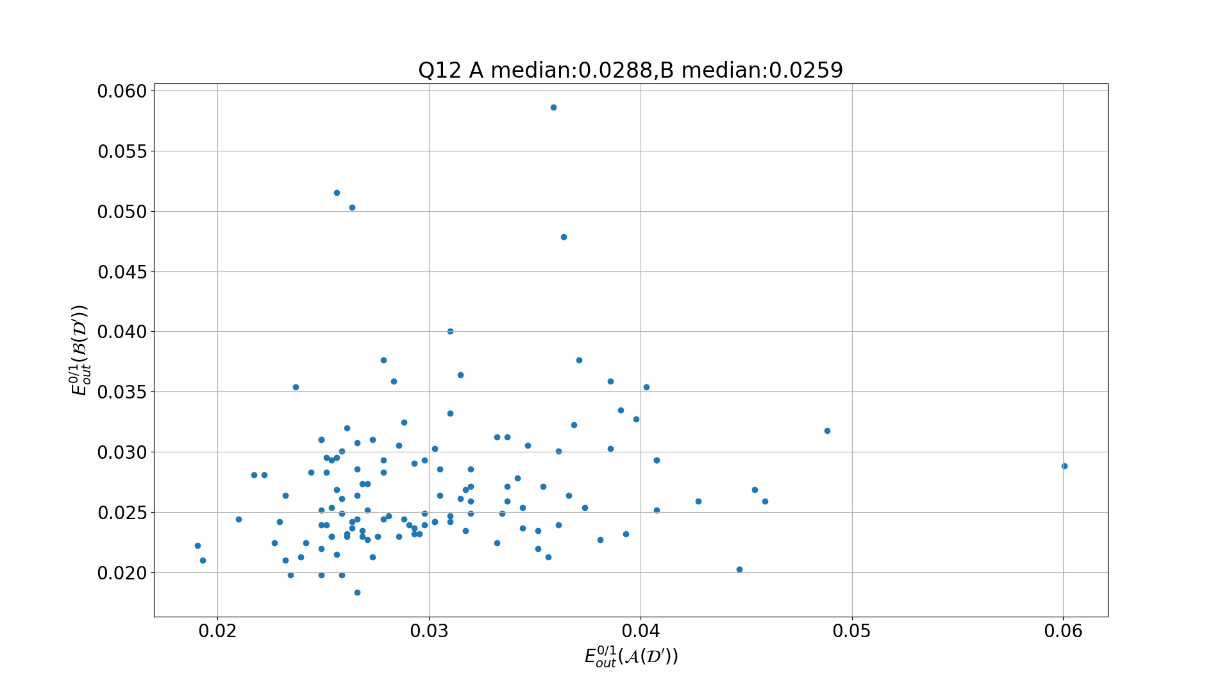
中位數是0.0195。

1. compare LIN and LOG



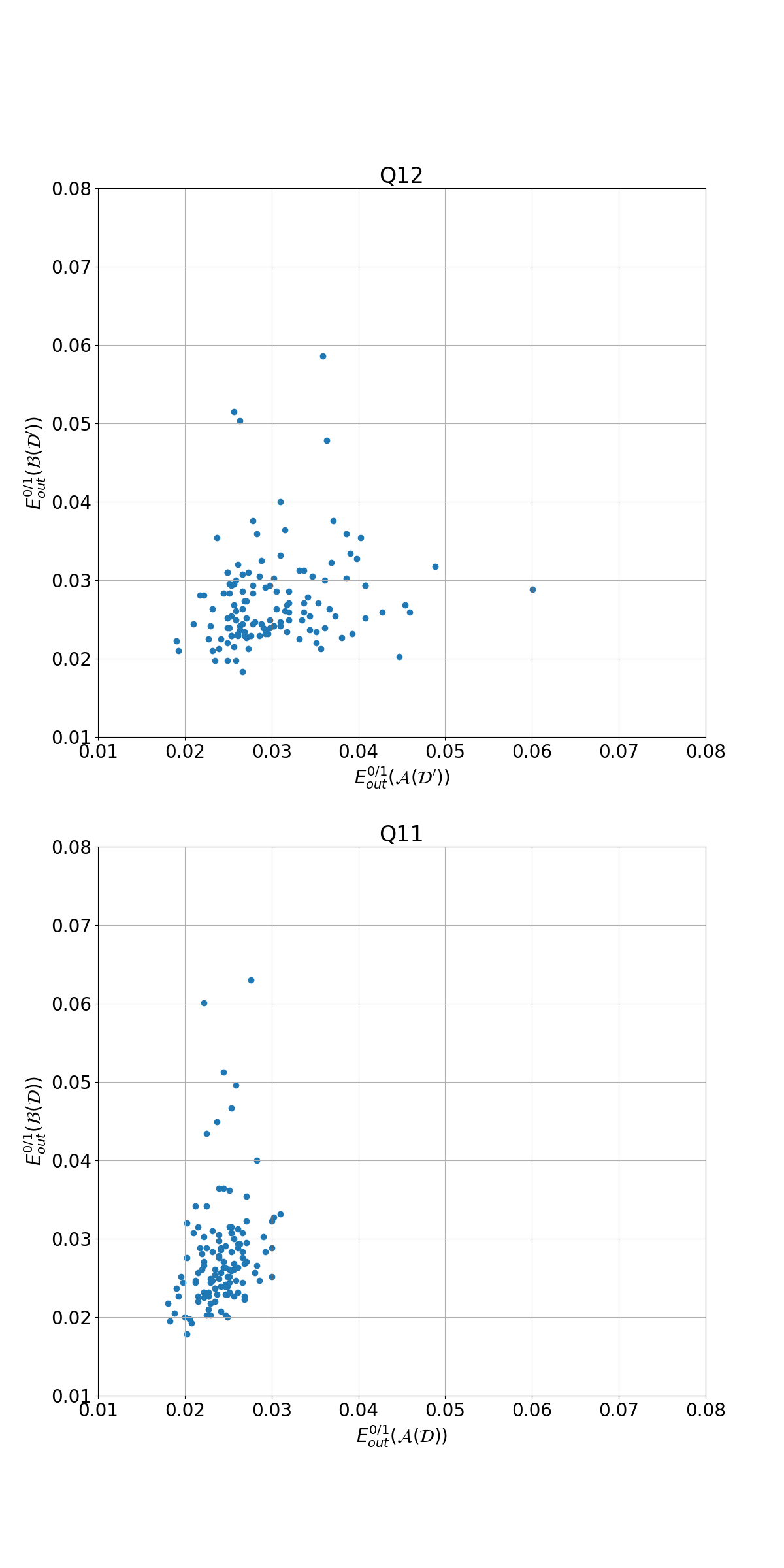
跟的中位數分別是 0.0242，0.0261

1. compare LIN and LOG with outlier



跟的中位數分別是 0.0288，0.0259

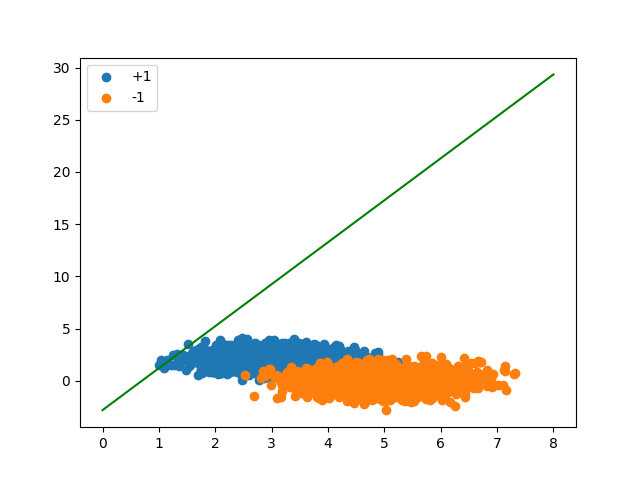
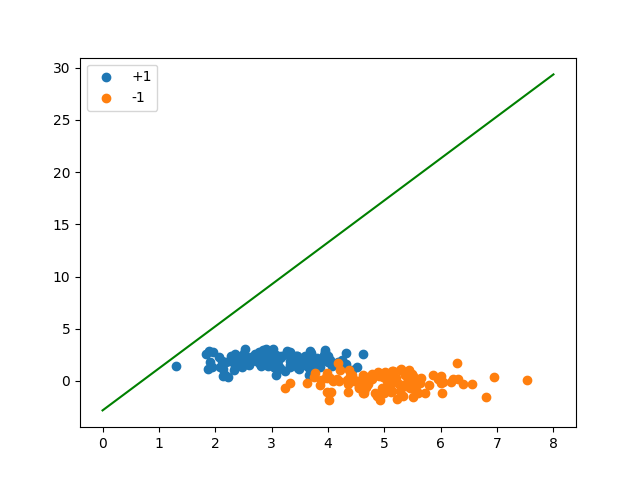
上面的圖因為outlier導致圖片變形很嚴重，所以我限定了範圍後得到兩軸比例一樣的圖：



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。

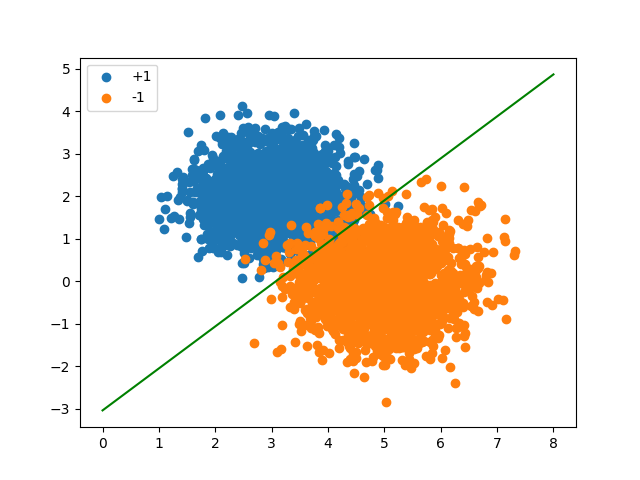
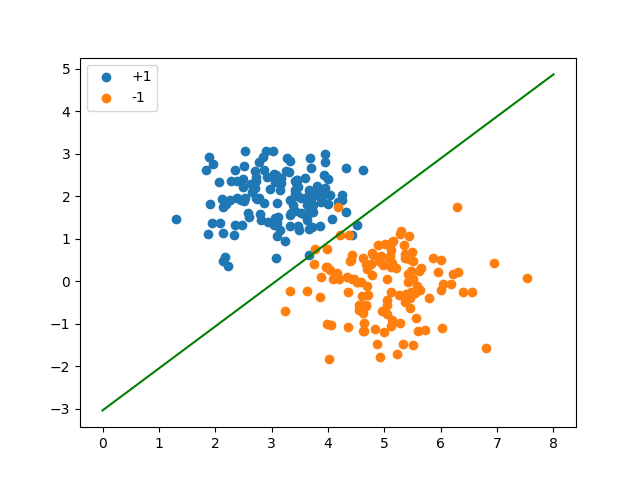
* logistic regression在加入outlier前就已經有許多點的錯誤率容易比linear regression來的大了，在11題的圖中還可以看到有一個點錯誤率比0.5還高，個人暫時推測應該是尚未找到最佳權重。
* 可以看到加入outlier後，linear regression的錯誤率也跟著提高，顯示出outlier有影響到最佳權重的值。

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：

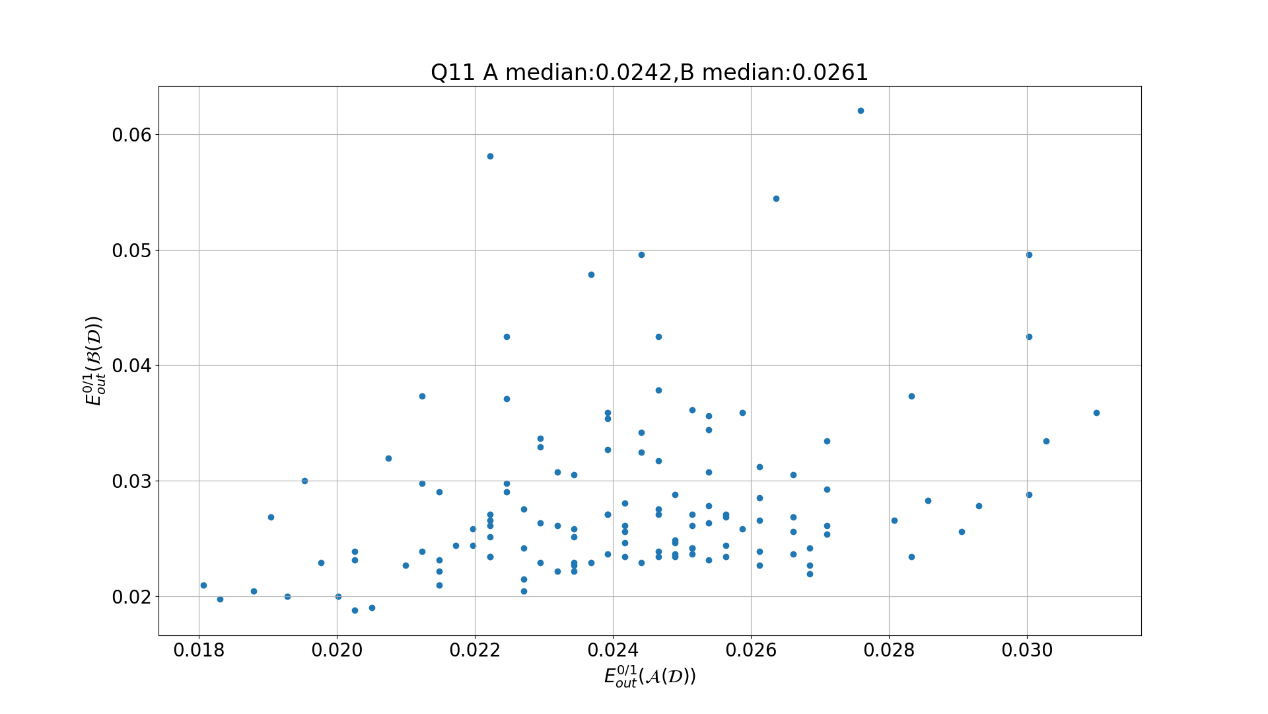


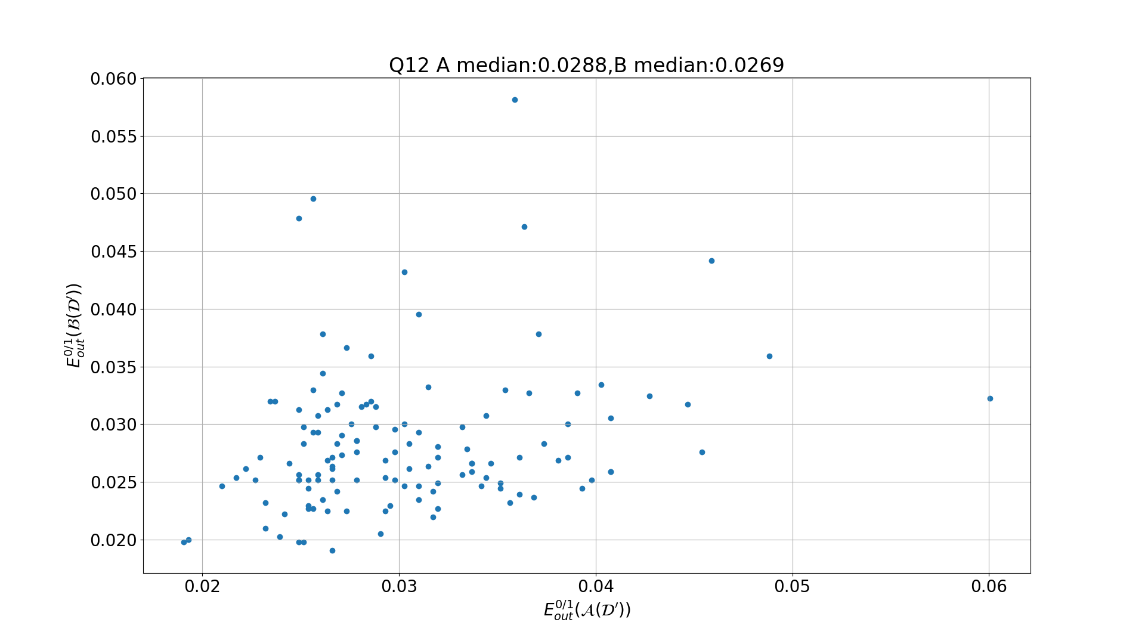
左邊是訓練資料，右邊是測試資料。

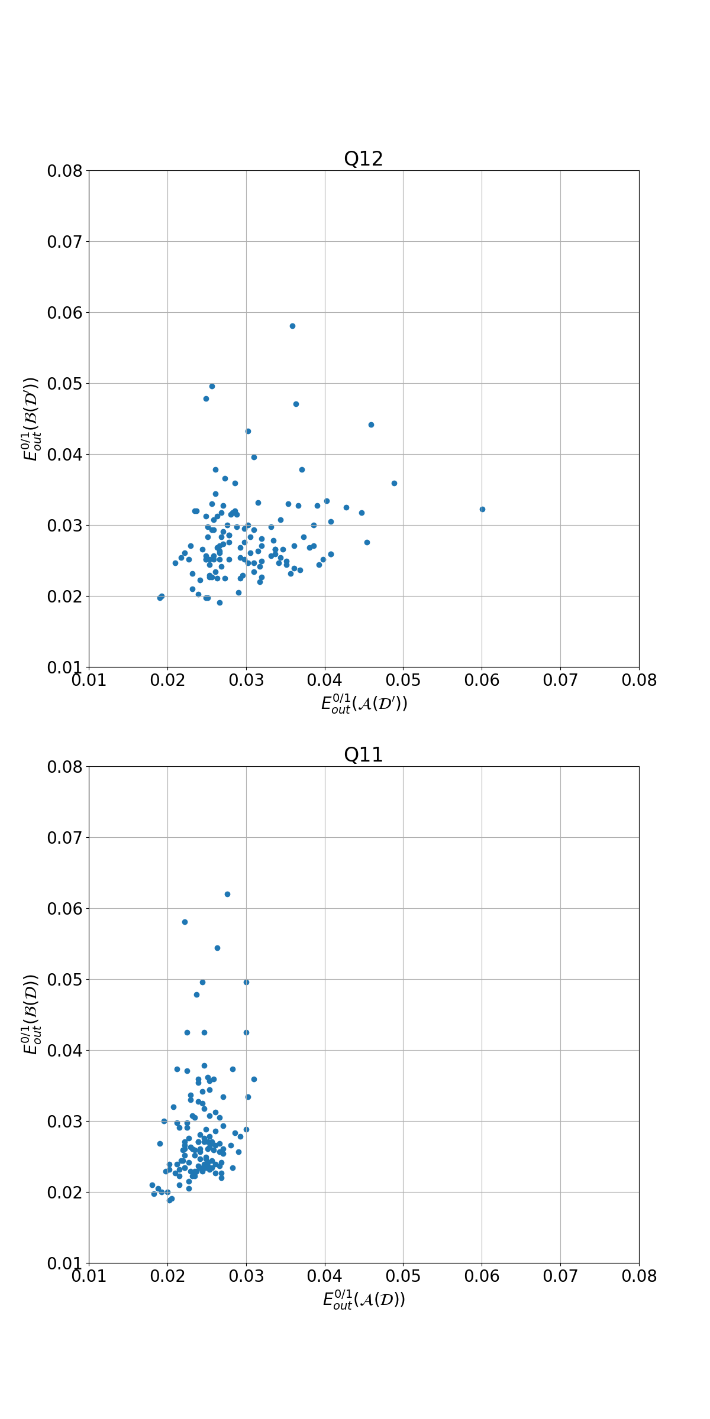
我覺得這應該是尚未找到最佳的權重所導致，於是我將訓練的次數從500調整到5000，保持不變，訓練跟測試的結果分別如下：



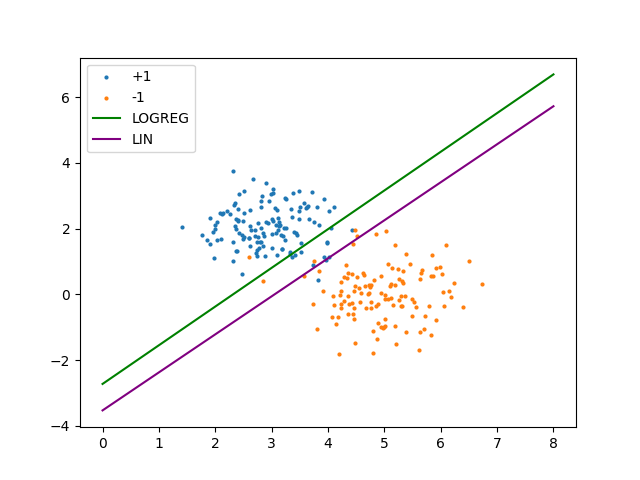
確實印證了我的猜測。所以我重新做了第11題跟第12題的模擬，將訓練次數改成5000次，得到如下的圖形：

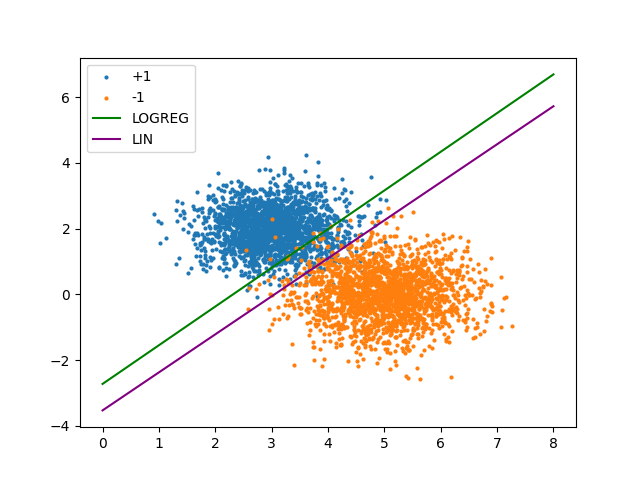






但是依舊發現logistic regression在加入outlier前就已經有許多點的錯誤率容易比linear regression來的大了，所以應該只是剛好那個錯誤率大於0.5的點沒有學好而已。所以接著我再從尚未加入outlier的資料找出一個錯誤率大於0.05的資料出來，發現他的圖長的如下：



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。我認為主要是的不同所導致，logistic regression尋求的是最大化Likelihood，或者說要最小化：

其中的 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：

這裡我讓括弧內上下同乘：

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，也就是說他除了要求點要分對(的值大於0.5)，也會傾向於讓值越大越好，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

1. internal linear regression

令第6題的**D**矩陣中的，可以得到：

由於要將轉成 的形式，加上**D**又是正定對角矩陣，所以我們可以直接給**D**開根號：

所以可以知道

接著處理，他應該要長得像下面兩個矩陣乘積後的結果：

拿掉的理由一樣是因為移項的時候可以消掉。

而所對應的形式為，所以我們可以知道這兩個矩陣的乘積應該要長的像下面這樣：

所以可以知道 的長相：