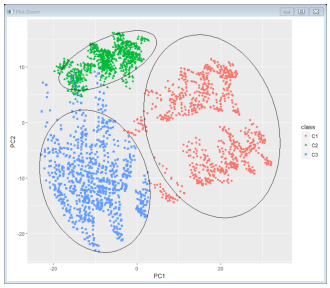
ML HW2

Multi-class Probabilistic Generative Model

Balance data

以 $\text{Traing}：\text{Validation} = 9：1$ 的比例做資料切割。切分的方式則依照編號個位數的數字，共作 10 次 cross-validation。 (例如：第一次，將照片編號尾數為 1 的作為 Validation data，其餘為 Training data，故各有 300、2700 筆)

首先，利用 Training data 做主成分分析 (PCA)，萃取出前兩大的主成分，作為我們的 features，分別命名為 PC1、PC2。將三組的分數畫出來後，得到右圖，雖然由圖可知共變異數矩陣並不相同，但為求計算方便，在這邊假設三組相同，並假設 $p(x|C_k)$ 為高斯分配，利用 training data 計算各組平均數 μ_k、共變異數矩陣 Σ 後，帶入下面的公式



$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)\}$$

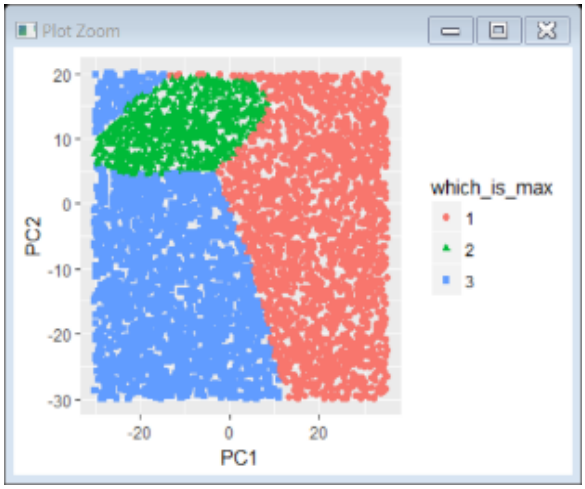
接著乘上 prior ，並做 normalization，得到 $p(C_k|x)$，Model 就建好了。(由於這邊是 balance data，故 prior $P(C_1) = P(C_2) = P(C_3) = \frac{1}{3}$)

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)}$$

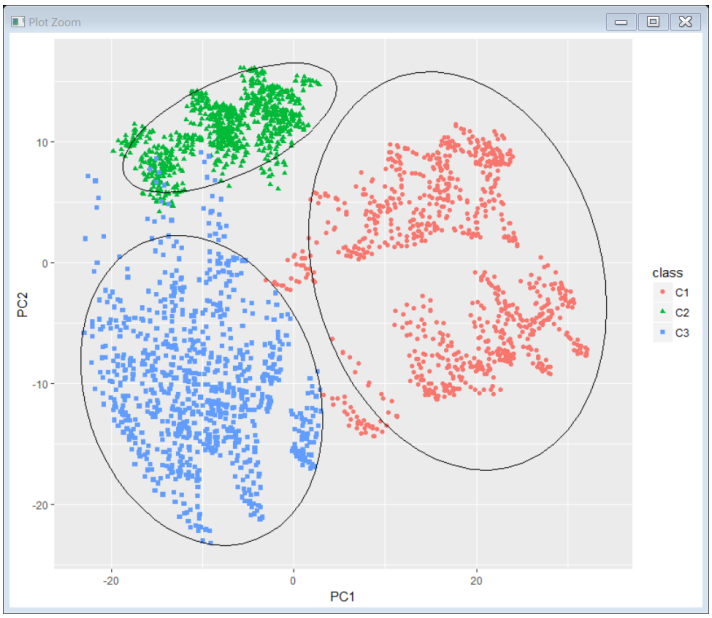
建完 model 後，就可以把 validation data 拿來測試了。

首先，利用 training model 的 PCA 參數，把 validation data 的 PC1、PC2 算出來，再帶入$p(C_k|x)$，看哪一組的分數最大，就分類到該組。

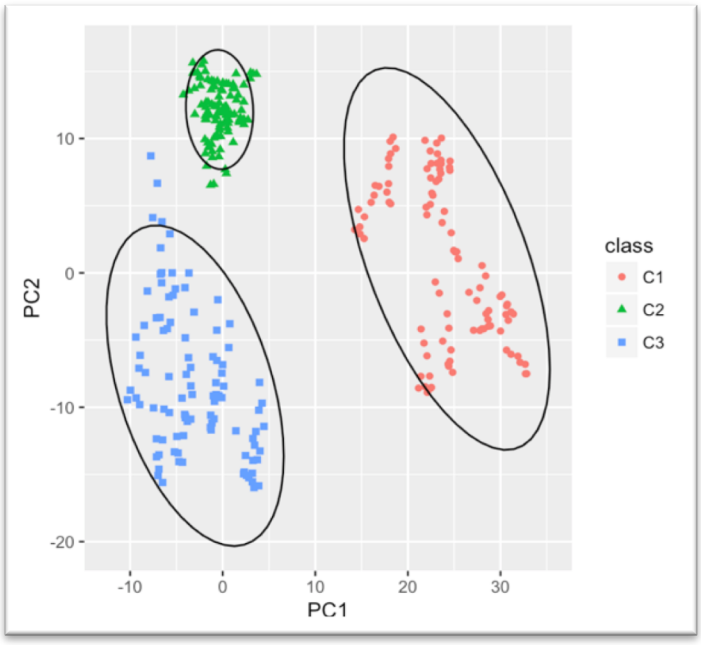
這邊用了 5000 個假資料，帶入 model，畫出 decision region，可以看出來，第一組與第三組有明顯的線區隔，並延伸到更大的區間、而第二組則集中在[-30~10 , 5~20] 這個區間。



↑ Decision region



↑ Training data



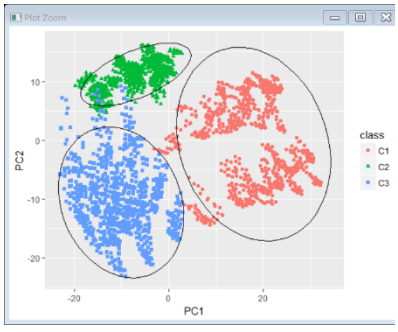
↑ Validation data

接著進行 cross validation 的驗證

train	valid	如何切	Hit rate (%)	Error rate (%)
900,900,900	100,100,100	尾數=1	97.0	3.0
900,900,900	100,100,100	尾數=2	98.6	1.4
900,900,900	100,100,100	尾數=3	99.3	0.6
900,900,900	100,100,100	尾數=4	99.0	1.0
900,900,900	100,100,100	尾數=5	98.6	1.3
900,900,900	100,100,100	尾數=6	96.6	3.3
900,900,900	100,100,100	尾數=7	97.6	2.3
900,900,900	100,100,100	尾數=8	99.0	1
900,900,900	100,100,100	尾數=9	99.3	0.6
900,900,900	100,100,100	尾數=0	98.3	1.6
		平均	98.3	1.7

註：900,900,900 表示(C1, C2, C3)各取 900 份作為 training data。

由上表可知，整體來說判斷準確率很高，猜想可能是我們先做 PCA，將 900-dim 轉化成兩個變異度最大的 dim，所以可以容易地將三組分開（由圖可知，雖然不是線性可分，但三組間有明顯的分隔區域）。



Unbalance data

以下探討 unbalance data 對 model 的影響。根據作業要求，示範將第一組的 training data 減少為 100 筆，其餘兩組保持 900 筆。發現平均 error rate 上升至 11.2%，大約增加了六倍，且大多數錯誤發生在「將第一組資料錯誤分類」，如右表。(推測原因為第一組的 training data 相對少)。

	1	2	3
C1	81	17	2
C2	3	97	0
C3	1	0	99

train	valid	如何切	Hit rate (%)	Error rate (%)
100,900,900	100,100,100	尾數=1	92.3	7.6
100,900,900	100,100,100	尾數=2	93.3	6.6
100,900,900	100,100,100	尾數=3	84.3	15.6
100,900,900	100,100,100	尾數=4	87.3	12.6
100,900,900	100,100,100	尾數=5	85.0	15.0
100,900,900	100,100,100	尾數=6	96.0	4.0
100,900,900	100,100,100	尾數=7	94.3	5.6
100,900,900	100,100,100	尾數=8	84.3	15.6
100,900,900	100,100,100	尾數=9	88.0	12.0
100,900,900	100,100,100	尾數=0	84.0	16.0
		平均	88.8	11.2

Multi-class Probabilistic Discriminative Model

在 Multi-class 中，不能用 2 -class 的 sigmoid function，而要改用 soft-max 的方式，公式如下：

Multiclass Logistic Regression

$$p(\mathcal{C}_k|\boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}$$

這邊需使用 Newton-Raphson method，透過 Gradient、Hessian matrix，疊代出新的 W。公式如下：

***Newton-Raphson* method**

$$g(\theta) \approx g(\theta_0) + \frac{dg(\theta)}{d\theta}(\theta - \theta_0) \quad \Rightarrow \quad \theta_1 = \theta_0 - \frac{g(\theta_0)}{\left.\frac{dg(\theta)}{d\theta}\right|_{\theta=\theta_0}}$$

$$\mathbf{w}^{(\mathrm{new})} = \mathbf{w}^{(\mathrm{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

H = $\nabla\nabla E(\mathbf{w})$ Hessian matrix

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

cross-entropy error function *for the multiclass classification problem*

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\,\boldsymbol{\phi}_n$$

IRLS algorithm

$$\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_k) = -\sum_{n=1}^{N} y_{nk}(\mathbf{I}_{kj} - y_{nj})\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T$$

分割資料的方式如前一題，由於我有 2700 筆 training data、2 個 Feature (PC1、PC2)，所以我的 Φ 為 2700 X 2 的矩陣；由於本題有 3 個 class，所以 Y 和 T 皆為 2700 X 3 的矩陣、又因為有 3 個 class、2 個 feature，所以 gradient 為 6 X 1 的矩陣、Hessian 為 6 X 6 的矩陣。

然而，有時候 Hessian matrix (H) 沒辦法取反矩陣 (已經確認過 det(H) ≠ 0，為線性獨立，但就是無法取反矩陣)， 如下例子(使用的程式語言為 R)：

```
              PC1           PC2           PC1           PC2           PC1           PC2
PC1  1.129045e+05 -2.795630e-11 -5.645226e+04  1.620482e-11 -5.645226e+04  1.620482e-11
PC2 -2.798117e-11  6.214986e+04  1.700817e-11 -3.107493e+04  1.700817e-11 -3.107493e+04
PC1 -5.645226e+04  1.620482e-11  1.129045e+05 -2.795630e-11 -5.645226e+04  1.620482e-11
PC2  1.700817e-11 -3.107493e+04 -2.798117e-11  6.214986e+04  1.700817e-11 -3.107493e+04
PC1 -5.645226e+04  1.620482e-11 -5.645226e+04  1.620482e-11  1.129045e+05 -2.795630e-11
PC2  1.700817e-11 -3.107493e+04  1.700817e-11 -3.107493e+04 -2.798117e-11  6.214986e+04
```

```
> det(H)
[1] -0.006699497

> solve(H)
Error in solve.default(H) :
  system is computationally singular: reciprocal condition number = 8.40882e-18
```

後來上網找資料，有人提到可以改變對線性相依偵測的 tolerance，於是我的 H 可以取反矩陣了，但事情也沒這麼簡單，他們倆個相乘居然不是單位矩陣... orz，也就是 $HH^{-1} \neq I$，如下所示：

```
> H%*% solve(H, tol=1e-18)
```

	PC1	PC2	PC1	PC2	PC1	PC2
PC1	1	-0.4497436	0.000000e+00	-0.4497436	0.000000e+00	0.0502564
PC2	0	1.0000000	0.000000e+00	0.5000000	0.000000e+00	0.0000000
PC1	0	1.5502564	1.000000e+00	1.5502564	4.440892e-16	0.5502564
PC2	0	-1.5000000	0.000000e+00	-0.5000000	0.000000e+00	-1.0000000
PC1	0	0.3636684	4.440892e-16	0.3636684	1.000000e+00	0.3636684
PC2	0	1.0000000	0.000000e+00	0.0000000	0.000000e+00	2.0000000

也因為這樣，造成我的 w 無法收斂，Newton 做幾次就發散了...。(教練，我想打球阿.....)

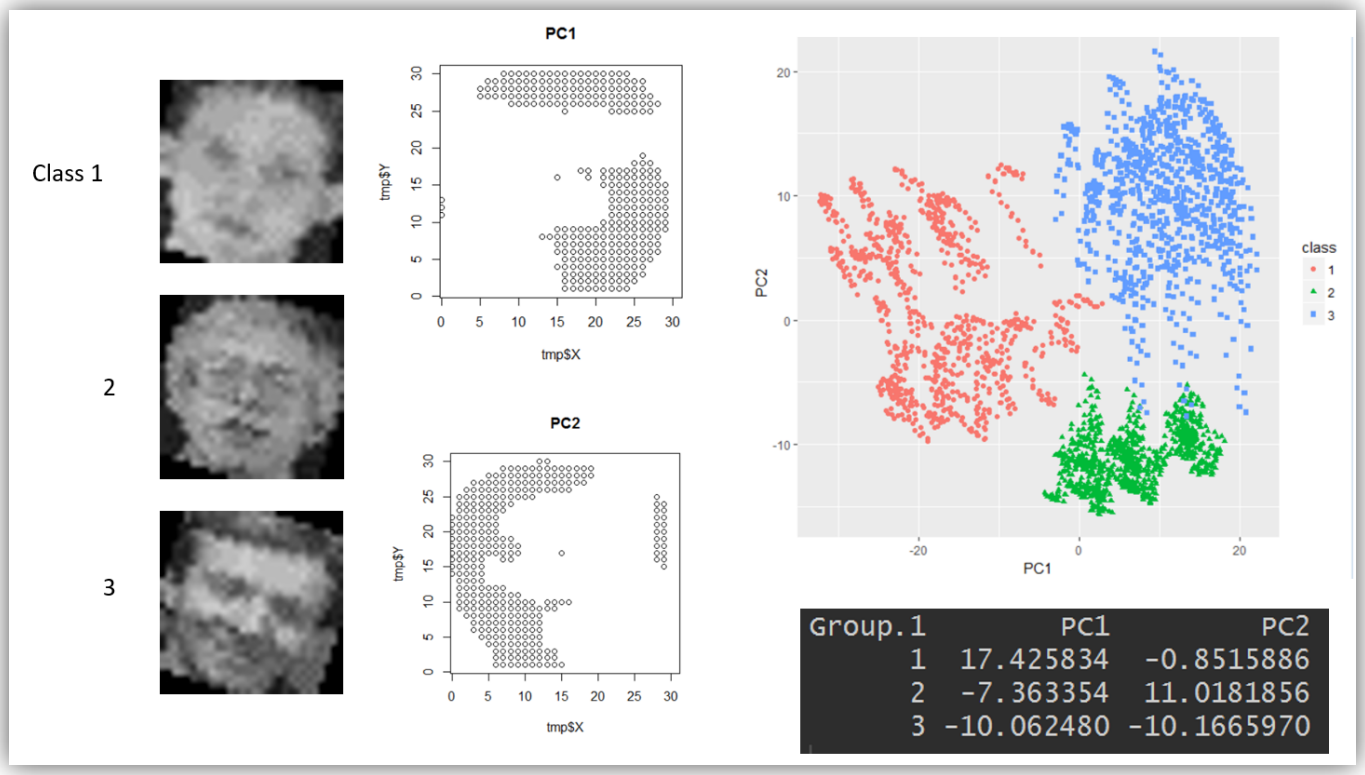
雖然沒辦法順利的取得我的 W，但我還是說明一下如何進行後續動作。假設經過足夠多次疊代後，W 收斂了，接著拿這個 W 去乘 validation data 的 Φ，再做 soft-max，看哪個組的分數最高，就分到該組。

接著與第一題一樣，計算 error rate 及 cross-validation，並探討在 unbalance data 的情況下，會有甚麼影響。

Other discussion

本次作業有用到 PCA，由於以前有在其他課堂上學過，所以想把這個拿出來討論。

取出變異量最大的兩個主成分(Principle Component)後，分別命名為 PC1、PC2，我想反推回去到底這 900 個 pixel 中，哪幾個 pixel 的 PC1、PC2 分數最高，於是我對這 900 個取了前 200 名，反推回去圖片的位置，得到結果如下圖：



Group.1	PC1	PC2
1	17.425834	-0.8515886
2	-7.363354	11.0181856
3	-10.062480	-10.1665970

可以發現，平均來說，PC1 表示在這三組中，「左前額及左臉」的位置的深淺變異程度最大；PC2 表示在這三組中，「右前額及右臉」的位置的深淺變異程度最大。也因為如此，所以利用這兩個分數，可以將三組輕鬆地分開。