Homework 2

108B (5403) Machine Learning　王傳鈞　0416047

本次作業所使用到的程式碼，都已經上傳到 GitHub 當中，若有需要參考其詳細內容，歡迎點擊連結直接前往網頁瀏覽。例如：「`Q1_plot.m`」的 MATLAB source code 代表是第一題所使用到的繪圖程式碼，「`Q2_a.m`」的 MATLAB code 則代表第二題的(a)小題所使用的程式碼，其餘命名方式請依此類推。

第一題

根據題目給定的數據，我們可以依據 posterior probability 由高到低做排序，並且依次計算出某筆數據之下 # of true positive instances (TP)、# of false positive instances (FP)、 # of true negative instances (TN) 、# of false negative instances (FN)、false positive rate (FPR)、true positive rate (TPR) 等各項數值(詳細請見表格一)，最後繪製成 FPR -TPR 平面圖即可得到 ROC curve(圖一)。

Rank	Label	TP	FP	TN	FN	FPR	TPR
1	P	1	0	10	9	0.0	0.1
2	P	2	0	10	8	0.0	0.2
3	P	3	0	10	7	0.0	0.3
4	P	4	0	10	6	0.0	0.4
5	N	4	1	9	6	0.0	0.4
6	P	5	1	9	5	0.1	0.5
7	P	6	1	9	4	0.1	0.6
8	N	6	2	8	4	0.1	0.6
9	P	7	2	8	3	0.2	0.7
10	N	7	3	7	3	0.2	0.7
11	P	8	3	7	2	0.3	0.8
12	N	8	4	6	2	0.3	0.8
13	P	9	4	6	1	0.4	0.9
14	N	9	5	5	1	0.4	0.9
15	N	9	6	4	1	0.5	0.9
16	N	9	7	3	1	0.6	0.9
17	N	9	8	2	1	0.7	0.9
18	P	10	8	2	0	0.8	1.0
19	N	10	9	1	0	0.9	1.0
20	N	10	10	0	0	1.0	1.0

Table 1 Ranking Method to Draw ROC curve and Calculate Area Under Curve

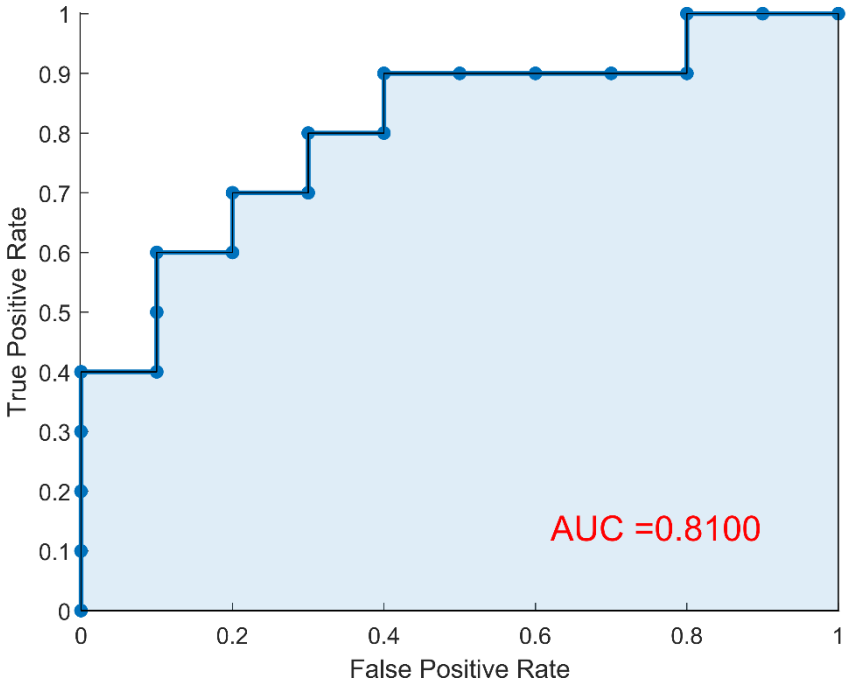

Figure 1 ROC curve to Question 1 and Its Area Under Curve

第二題

利用肉眼計算，我們可以很容易得知： $\mathrm{f}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix}1 & 0\\ 0 & 1000\end{bmatrix}\mathbf{x}$ 的最小值發生在 $\mathbf{x_{min}} = \begin{bmatrix}0\\0\end{bmatrix}$ 且 $\mathrm{f}(\mathbf{x_{min}}) = 0$。

(a) 利用我撰寫的 MATLAB code「Q2_a.m」，總共經過 9731 次疊代收斂在：

$\mathbf{x_{gradient}} \approx \begin{bmatrix}3.5299 \times 10^{-6}\\ -3.5299 \times 10^{-9}\end{bmatrix}$、$\mathrm{f}(\mathbf{x_{gradient}}) \approx 6.2362 \times 10^{-12}$。

以下表格二列出首三輪與尾三輪的詳細 $\mathbf{x}$ 疊代數值與變化量：

$\mathbf{x}^{\mathbf{iter}}$		$\mathrm{f}(\mathbf{x}^{\mathbf{iter}})$	$\left\|\mathbf{x}^{\mathbf{n+1}} - \mathbf{x}^{\mathbf{n}}\right\|_2$
$\mathbf{x}^{\mathbf{0}}$	$[1000 \quad 1]^{\mathrm{T}}$	500500	-
$\mathbf{x}^{\mathbf{1}}$	$[9.9800 \times 10^{2} \quad -0.9980]^{\mathrm{T}}$	4.9850×10^{5}	2.8256
$\mathbf{x}^{\mathbf{2}}$	$[9.9601 \times 10^{2} \quad 0.9960]^{\mathrm{T}}$	4.9651×10^{5}	2.8200
. . .			
$\mathbf{x}^{\mathbf{9729}}$	$[3.5440 \times 10^{-6} \quad -3.5440 \times 10^{-9}]^{\mathrm{T}}$	6.2863×10^{-12}	1.0034×10^{-8}
$\mathbf{x}^{\mathbf{9730}}$	$[3.5369 \times 10^{-6} \quad 3.5369 \times 10^{-9}]^{\mathrm{T}}$	6.2612×10^{-12}	1.0014×10^{-8}
$\mathbf{x}^{\mathbf{9731}}$	$[3.5299 \times 10^{-6} \quad -3.5299 \times 10^{-9}]^{\mathrm{T}}$	6.2362×10^{-12}	9.9940×10^{-9}

Table 2 Detailed Info. about First Three and Last Three Iterations (Gradient Descent)

(b) 利用我撰寫的 MATLAB code「Q2_b.m」，總共經過 1 次疊代收斂在：

$\mathbf{x}_{\mathbf{Newton}} = \begin{bmatrix}0\\0\end{bmatrix}$、$\mathrm{f}(\mathbf{x}_{\mathbf{Newton}}) = \mathbf{0}$。

以下表格三列出全部的詳細 $\mathbf{x}$ 疊代數值與變化量：

$\mathbf{x}^{\mathbf{iter}}$		$\mathrm{f}(\mathbf{x}^{\mathbf{iter}})$	$\mathbf{d}^{\mathbf{n}}$
$\mathbf{x}^{\mathbf{0}}$	$[1000 \quad 1]^{\mathrm{T}}$	500500	$[-1000 \quad -1]^{\mathrm{T}}$
$\mathbf{x}^{\mathbf{1}}$	$[0 \quad 0]^{\mathrm{T}}$	0	–

Table 3 Detailed Information about All Iterations (Newton's Method)

經過以上兩種不同的方法，我們可以發現牛頓法收斂的速度驚人地快速，幾乎是早就知道答案般地，直接從起始點跳到最終結果；相反地，梯度下降法因為受限於每一次最多只能行走「梯度」長度的下降量，因此要很久才收斂到理想水準。但是，此例並無法顯示牛頓法的一個缺點：遇到特殊函數，若起始點選的離最終答案並不接近，則有可能永遠無法收斂。

第四題

(a) 請參考(b)小題內容。

(b) 利用我撰寫的 MATLAB code「Q4_b.m」，可以得到：

hypothesis $\mathrm{f}(\mathbf{x}) = \sum_{\mathrm{i}=1}^{13} \alpha_{\mathrm{i}}\, \mathrm{y}_{\mathrm{i}} \langle \mathbf{x}^{\mathrm{i}}, \mathbf{x} \rangle + \mathrm{b}$，其中

$\boldsymbol{\alpha} = [10 \quad 1 \quad 0 \quad 0 \quad 0 \quad 5 \quad 5 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^{T}$

$\mathbf{y} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1]^{T}$

$\mathbf{x^1} = [0 \quad 0]^{T}$　　$\mathbf{x^6} = [0.5 \quad 0.5]^{T}$　　$\mathbf{x^{11}} = [0 \quad 1]^{T}$

$\mathbf{x^2} = [0.5 \quad 0]^{T}$　　$\mathbf{x^7} = [0.5 \quad -0.5]^{T}$　　$\mathbf{x^{12}} = [-1 \quad 0]^{T}$

$\mathbf{x^3} = [0 \quad 0.5]^{T}$　　$\mathbf{x^8} = [-0.5 \quad 0.5]^{T}$　　$\mathbf{x^{13}} = [0 \quad -1]^{T}$

$\mathbf{x^4} = [-0.5 \quad 0]^{T}$　　$\mathbf{x^9} = [-0.5 \quad -0.5]^{T}$

$\mathbf{x^5} = [0 \quad -0.5]^{T}$　　$\mathbf{x^{10}} = [1 \quad 0]^{T}$

$\mathrm{b} = 1.0000$

接著，我們用隨機產生的亂數當作測試集資料，並作圖可得以下圖二：

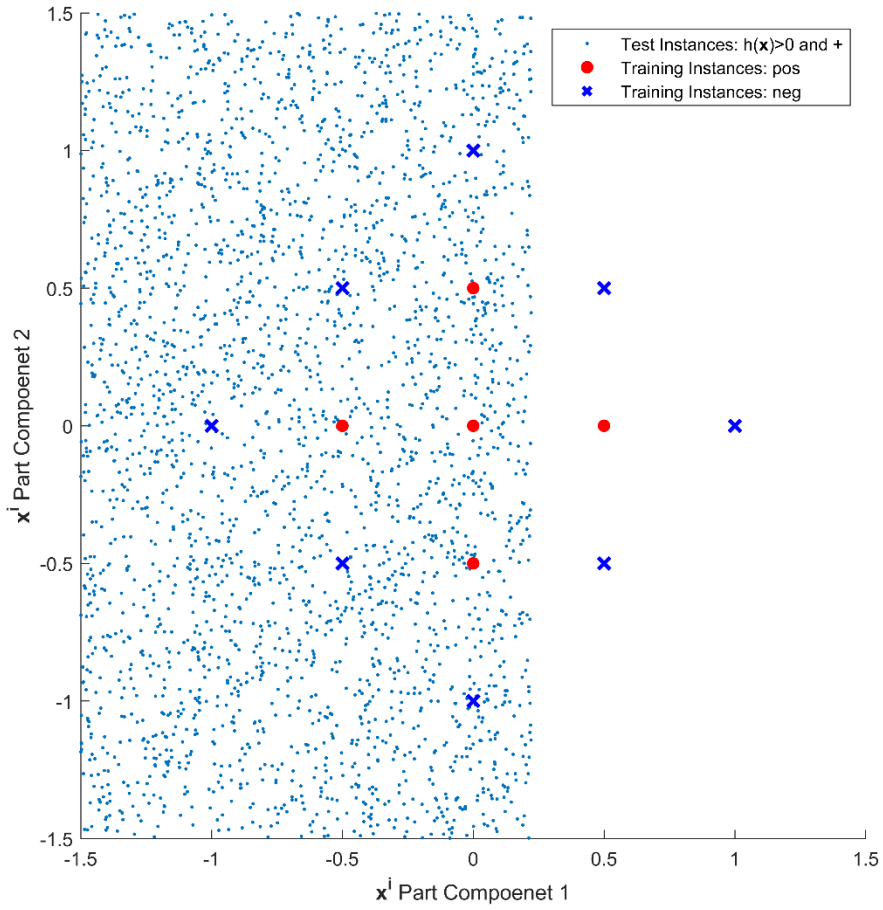

Figure 2 Scatter Plot of Question 4 (b)

上圖當中，較大顆的紅色、藍色點為訓練集資料；較小的淺藍色點為測試集資料且被 Perceptron algorithm (dual form)認為 positive 且實際上也是 positive 之資料點。

我們可以發現：因為訓練集不是 linearly separable，所以 Perceptron 找出切開平面的水平線只好犧牲一個 positive instance 被歸類成 negative。

(c) 仿造(a)、(b)小題的步驟使用「Q4_c.m」，我們得到以下結果：

hypothesis $f(\mathbf{x}) = \sum_{i=1}^{8} \alpha_i y_i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$，其中

$\boldsymbol{\alpha} = [5 \quad 2 \quad 0 \quad 0 \quad 3 \quad 3 \quad 0 \quad 0]^T$

$\mathbf{y} = [1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1]^T$

$\mathbf{x^1} = [0.5 \quad 0]^T$ $\mathbf{x^4} = [0 \quad -0.5]^T$ $\mathbf{x^7} = [-0.5 \quad 0.5]^T$

$\mathbf{x^2} = [0 \quad 0.5]^T$ $\mathbf{x^5} = [0.5 \quad 0.5]^T$ $\mathbf{x^8} = [-0.5 \quad -0.5]^T$

$\mathbf{x^3} = [-0.5 \quad 0]^T$ $\mathbf{x^6} = [0.5 \quad -0.5]^T$

$b = 0.2500$

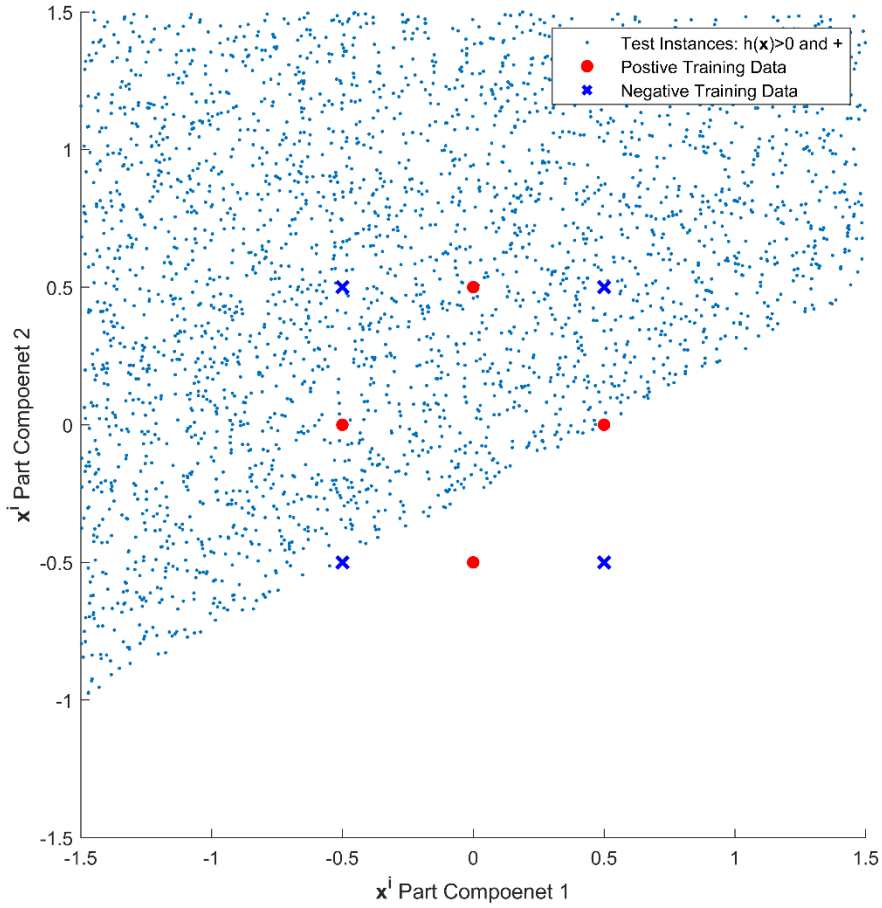

Figure 3 Scatter Plot of Question 4 (c)

(d) 請參考(e)小題內容。

(e) 利用我撰寫的 MATLAB code「Q4_e.m」，可以得到：

hypothesis $f(\phi(\mathbf{x})) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$，其中

$\mathbf{w} = [0 \quad -0.3500 \quad 0 \quad -0.3889]^T$、$b \approx 0.1556$

接著，我們用隨機產生的亂數當作測試集資料，並作圖可得以下圖四：

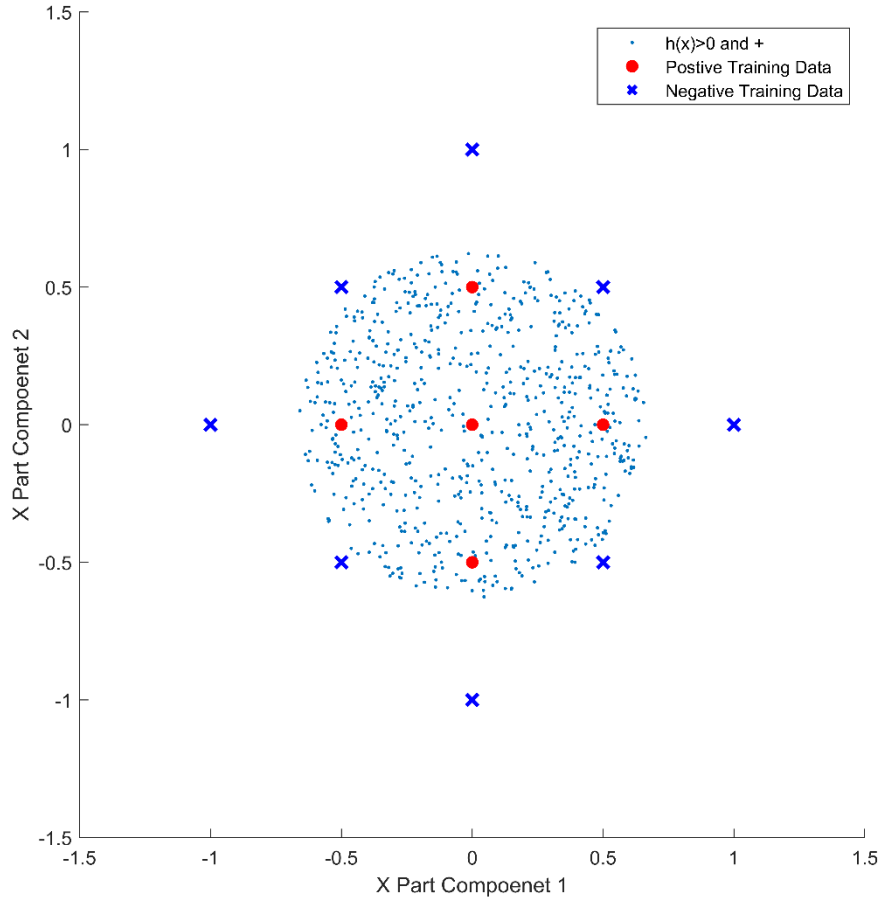

Figure 4 Scatter Plot of Question 4 (e)

上圖當中，較大顆的紅色、藍色點為訓練集資料；較小的淺藍色點為測試集資料且被 Perceptron algorithm (primal form)認為 positive 且實際上也是 positive 之資料點。

我們可以發現：應用了映射到高維度的技巧，原本不是 linearly separable 的訓練集，也能被 Perceptron 找出完美的分割面，另測試集資料以合理的方式來分類。