ENPM 808A – Introduction to Machine Learning

Homework – 4

1.

1) Taking the same weight matrices from the example 7.1, The matrix has, for $x = 2, y = 1$

$$x^{(0)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad s^{(1)} = \begin{bmatrix} 0.7 \\ 1 \end{bmatrix}$$

Thus, $x^{(1)} = \begin{bmatrix} 1 \\ 0.7 \\ 2 \end{bmatrix}$, $s^{(2)} = [-2.1]$

$x^{(2)} = [-2.1]$, $s^{(3)} = [-3.2]$

$x^{(3)} = [-3.2]$

Applying back propogation,

$\theta'(s^{(2)}) = 1$ & computes

$\delta^{(3)} = 2(x^{(3)} - y) = -8.4$

$\delta^{(2)} = \theta'(s^{(2)}) \otimes [w^{(3)} \delta^{(3)}] = -16.8$

$\delta^{(1)} = \begin{bmatrix} -16.8 \\ 50.4 \end{bmatrix}$

$$\frac{\delta e}{\delta W^{(1)}} = x^{(0)} (\delta^{(1)})^T = \begin{bmatrix} -16.8 & 50.4 \\ -33.6 & 100.8 \end{bmatrix}$$

$$\frac{\delta e}{\delta W^{(2)}} = x^{(1)} (\delta^{(2)})^T = \begin{bmatrix} -16.8 \\ -11.76 \\ -33.6 \end{bmatrix}$$

$$\frac{\delta e}{\delta W^{(3)}} = x^{(2)} (\delta^{(3)})^T = \begin{bmatrix} -8.4 \\ -17.64 \end{bmatrix}$$

2. For code, please see the question_2 method in the hw4.py file.

Number of positive points = 12

In-Sample Error = 0.0

SVM Margin = 0.0863479230977428

Out-of-Sample Error = 0.03

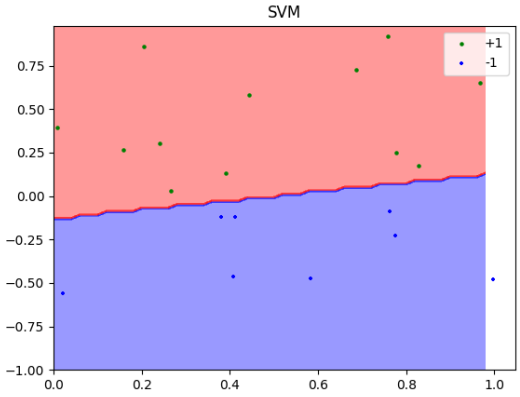
Number of positive points = 12

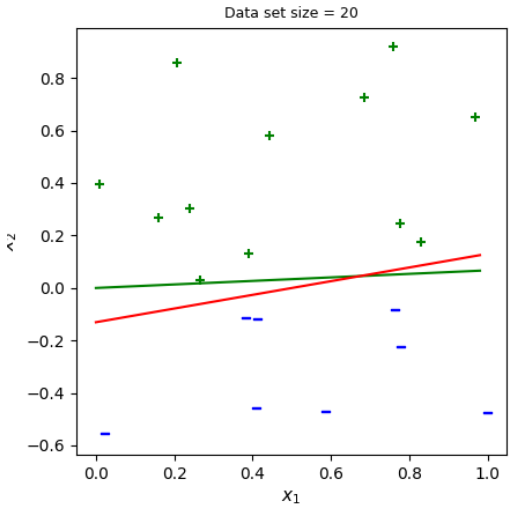
Number of negatives points = 8

Final correctness = 20 . Total iteration = 4

Final w = [0. -0.13341334 1.98459846]

Out-of-Sample Error = 0.03





Picked one positive point on the left and one negative on the right lower corner of the plot, so their perpendicular bisector separating plane is a bad separator. We need put them at the end of the points so the PLA algo will have no chance to adjust the separating line using other points. In this case, it took 4 iterations to make PLA converge. It's clear that PLA can be greatly affected by the ordering of data points, while SVM is stable with respect to the point orders. The out-of-sample error is much larger from PLA.

3.

A.3) For this data, we let $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$,

we have the object value

$$E = \frac{1}{2} w^T w = \frac{1}{2}(w_1^2 + w_2^2)$$

The constraints are

$$y_1(w_1 x_{11} + w_2 x_{12} + b) = -b \geq 1 \quad (1)$$

$$y_2(w_1 x_{21} + w_2 x_{22} + b) = w_2 - b \geq 1 \quad (2)$$

$$y_3(w_1 x_{31} + w_2 x_{32} + b) = -2w_1 + b \geq 1 \quad (3)$$

Combine the (1) & (3)
we get $w_1 \leq -1$

Combine the (1) & (2)
we get $w_2 \geq 0$

So, the objective achieves minimal at $w_1 = -1$,

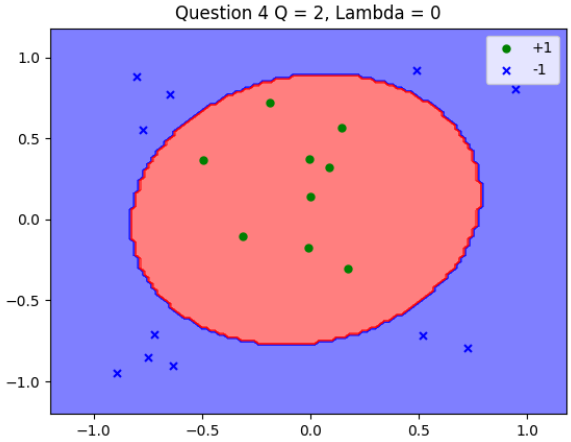
$w_2 = 0$, where $E = \frac{1}{2}$

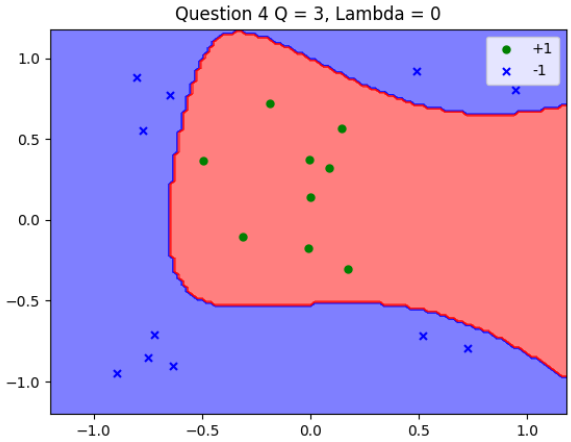
The optimal $b = -1$

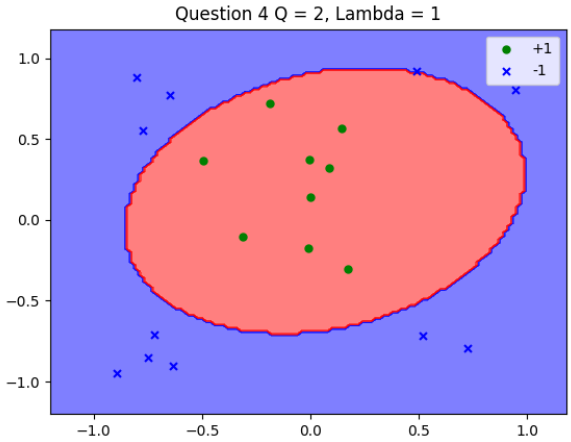
The margin is thus $\frac{1}{|w|} = \frac{1}{\sqrt{w_1^2 + w_2^2}} = 1$

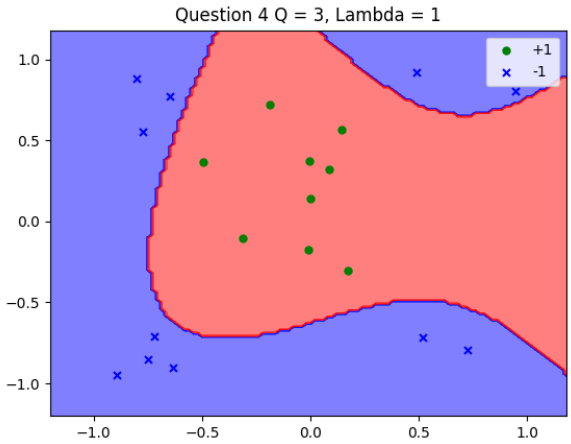
4.

a) and c) can be observed in the following diagrams with supporting question_4 method in the code.


Question 4 Q = 2, Lambda = 0
+1
-1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


Question 4 Q = 3, Lambda = 0
+1
-1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


Question 4 Q = 2, Lambda = 1
+1
-1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0



b) The fit with Q = 3 seems to have overfitted as it is complicated and seems to fit the on the far right too much.

5. Please see the code in hw4.py for more details

Enter the question number to run the program: 5

Fitting 5 folds for each of 18 candidates, totalling 90 fits

SVM:

Optimal Hyper-parameters: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}

scores: [0.83271681 0.84422524 0.86724209 0.83881579 0.87828947]

mean score: 0.8522578795941765

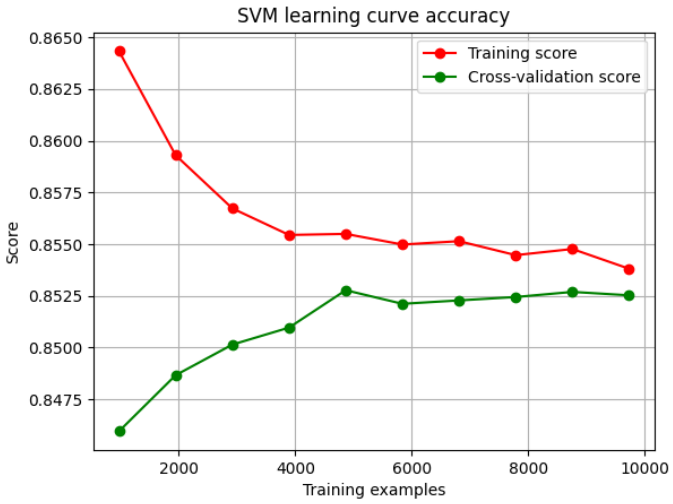
Fitting 5 folds for each of 72 candidates, totalling 360 fits

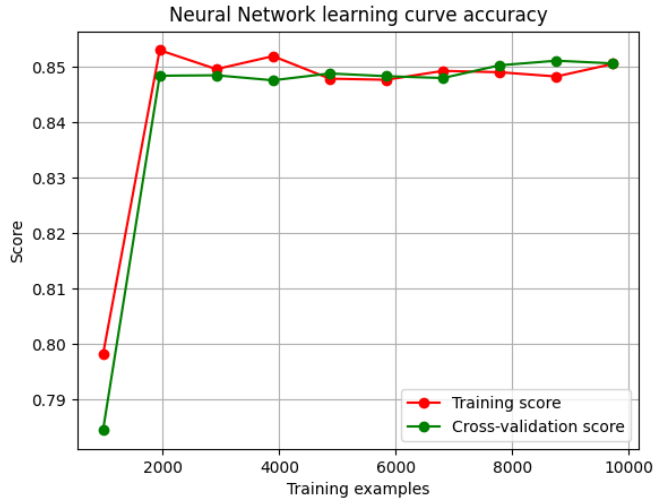
NN:

Optimal Hyper-parameters: {'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (6, 6, 6, 6), 'solver': 'adam'}

scores: [0.82449651 0.84422524 0.86642006 0.83758224 0.87129934]

mean score: 0.8488046758387954


SVM learning curve accuracy
Training score
Cross-validation score
Score
Training examples
0.8650
0.8625
0.8600
0.8575
0.8550
0.8525
0.8500
0.8475
2000
4000
6000
8000
10000


Neural Network learning curve accuracy
Training score
Cross-validation score
Score
Training examples
0.85
0.84
0.83
0.82
0.81
0.80
0.79
2000
4000
6000
8000
10000