1. Neural Networks

Q1a)

(a) (10 points) Consider a neural networks for a binary classification using sigmoid function for each unit. If the network has no hidden layer, explain why the model is equivalent to logistic regression.

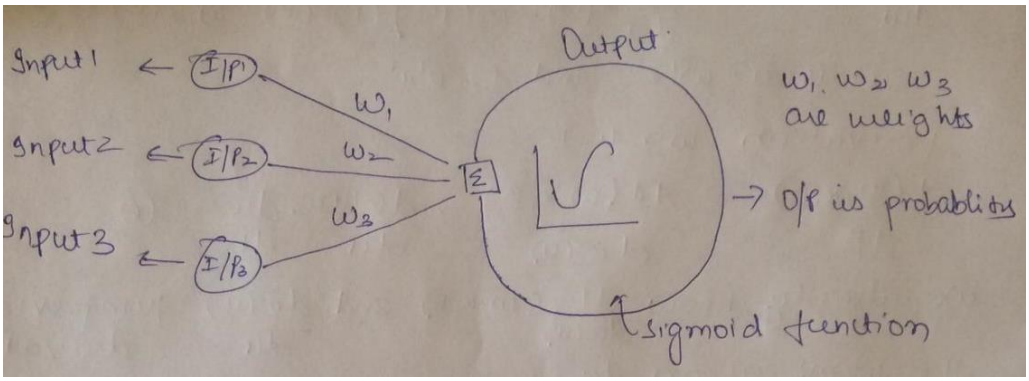
Solution 1a)

Logistic Regression is a special case of linear regression.

The logistic regression formula is derived from the standard linear equation for a straight line. The linear representation (-inf,+inf) is converted to a probability representation (0-1) using the sigmoidal curve. And hence the outcome is always binary classification of 0 or 1. Sigmoid is one function which does this. It squeezes any number generated by a function, in this case, a (w*x + b) between 0 and 1.

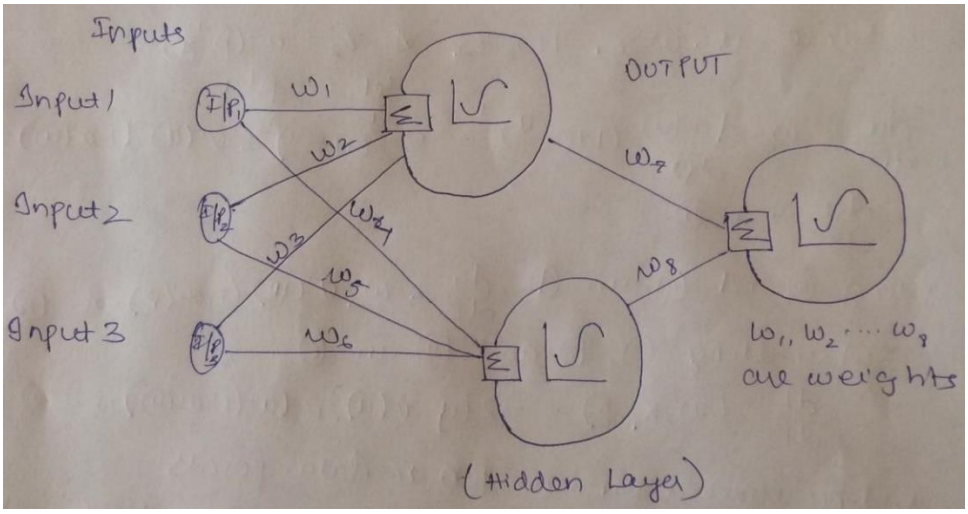
Logistic Regression for binary classification used sigmoid function to find P (Y=1 | Given X). The sigmoid unit in logistic model is a perception with inputs in form of independent variable, output as dependent variable and weights as coefficient shown below:

Logistic Regression Model:



Artificial Neural Network is a network comprising of multiple hidden layers processing input and producing output as classification. Below illustration shows neural network with n inputs, 2 hidden layers and n output class.

Neural Network Model:

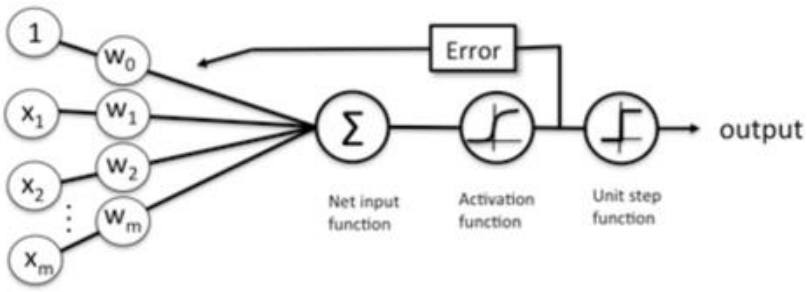


As seen from neural network, if we remove the hidden layer, model will be equivalent to logistic regression as both use sigmoid activation function and train parameters using same loss function.

In Neural Network, three things happen:

- At each layer/perceptron, every input is multiplied by weight
  $x_1 \rightarrow x_1 * w_1$, $x_2 \rightarrow x_2 * w_2$ etc.
- All weighted inputs are added and then bias (optional) is added
  $(x_1 * w_1) + (x_2 * w_2) + b$
- Activation function is then applied to linear equation derived above
  $y = f(x_1 * w_1 + x_2 * w_2 + b)$

Now, consider Neural Network with No -Hidden layer.



In this scenario, output will be just an activation function applied to linear combination of all weighted inputs.Now if we use Sigmoid function as an activation function, it will be just sigmoid function applied to above linear equation.

And Hence it will be logistic regression in other words.

Q1b)

(b) (10 points) Consider a simple two-layer network in the lecture slides. Given the cost function used to training the neural networks

$$\ell(w, \alpha, \beta) = \sum_{i=1}^{m} (y^i - \sigma(w^T z^i))^2$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function. Show the that the gradient is given by

$$\frac{\partial \ell(w, \alpha, \beta)}{\partial w} = -\sum_{i=1}^{m} 2(y^i - \sigma(u^i))\sigma(u^i)(1 - \sigma(u^i))z^i.$$

where $z_1^i = \sigma(\alpha^T x^i)$, $z_2^i = \sigma(\beta^T x^i)$. Also find the gradient of ℓ with respect to α and β.

Solution 1b)

Given below is the solution in the attached screenshots

For the sake of simplicity, let us assume $m=1$,
So for a two layered network

$$l(w,\alpha,\beta) = (y-\sigma(w^T z))^2 \quad \text{— (1)}$$

where $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \sigma(\alpha^T x) \\ \sigma(\beta^T x) \end{bmatrix}$

Let $u = w^T z$ $\qquad \sigma(u) = \sigma(w^T z)$

Making replacements in equation (1) we get.

$$l(w,\alpha,\beta) = (y-\sigma(u))^2$$

$\sigma(u) = 1/1+e^{-u}$ - given

Using chain rule and exponential derivation rule

$$\sigma(u) = (1+e^{-u})^{-1}$$

$$\frac{d\sigma(u)}{du} = -1 \times (1+e^{-u})^{-2} \times (-e^{-u})$$

$$= \frac{e^{-u}}{(1+e^{-u})^2} \quad \text{— (2)}$$

Since $1/1+e^{-u} = \sigma(u)$ and $1-\sigma(u) = \frac{e^{-u}}{1+e^{-u}}$

Combining above two in equation (2), we get.

$$\frac{d\sigma(u)}{du} = \sigma(u)(1-\sigma(u)) \quad \text{— (3)}$$

Since $u = w^T z$.

$$\frac{du}{dw} = z$$

Since $(w,\alpha,\beta) = 2(y-\sigma(u))(-1) = -2(y-\sigma(u))$ — (5)

$$\frac{dl}{dw}(w,\alpha,\beta) = \frac{dl(w,\alpha,\beta)}{d\sigma(u)} \cdot \frac{d\sigma(u)}{du} \cdot \frac{du}{dw} \quad \text{— (6)}$$

Substituting ③ ① and ⑤ in ⑥ we get:

$$\frac{dl}{dw}(w,\alpha,\beta) = -2(y-\sigma(u))\,\sigma(u)\,(1-\sigma(u))\cdot z$$

Adding summation back for m data points

$$\boxed{\frac{dl}{dw}(w,\alpha,\beta) = \sum_{i=1}^{m} 2(y^i-\sigma(u^i))\,\sigma(u^i)\,(1-\sigma(u^i))\cdot z^i}$$

Proved.

For derivation w.r.t α

$$\frac{d(w,\alpha,\beta)}{d\alpha} = \frac{dl(w,d,\beta)}{d\sigma(u)} \cdot \frac{d\sigma(u)}{d(u)} \cdot \frac{du}{d\sigma} \quad ⑨$$

Since we know $\frac{dd(w,\alpha,\beta)}{d\sigma(u)}$ and $\frac{d\sigma(u)}{d\sigma}$

with forms on $\frac{du}{d\alpha}$ $\quad u = w_1 z_1 + w_2 z_2$

$$\sigma(u) = \sigma(w^T, z) \quad z_1 = \sigma(\alpha^T x)$$

Let $s = \alpha^T x$, then from chain rule $\frac{du}{d\alpha} = w, \frac{d\sigma(s)}{ds} \cdot \frac{ds}{d\alpha}$ - ⑫

$$\frac{d\sigma(s)}{ds} = \frac{d}{ds}(1+e^{-s})^{-1} = \frac{e^{-s}}{(1+e^{-s})^2} = \sigma(s)(1-\sigma(s)) - ⑦$$

$$\frac{ds}{d\alpha} = x - ⑧$$

Substituting ⑦ and ⑧ we get

$$\frac{du}{d\alpha} = w\,\sigma(s)\,(1-\sigma(s))\,x \quad - ⑤$$

Substituting ③, ① and ⑤ in ⑨

$$\frac{dl}{d\alpha}(w\alpha,\beta) = -2(y-\sigma(u))\,\sigma(u)\,(1-\sigma(u))\cdot w_1\,\sigma(s)\,(1-\sigma(s))\,x$$

Adding the summation back for m data points

$$\frac{dl}{d\alpha}(\alpha,\omega,\beta) = -\sum_{i=1}^{m} 2\left((y^i - \sigma(u^i))\right)(1-\sigma(u^i)\cdot\omega_1\sigma(s^i)(1-\sigma(s^i))x^i$$

where $u = \omega^T z^i$ and $s = \alpha^T x^i$

Derivation w.r.t β

$$\frac{d(\omega,\alpha,\beta)}{d\beta} = \frac{dl(\omega,\alpha,\beta)}{d\sigma(u)} \cdot \frac{d\sigma(u)}{du} \cdot \frac{du}{d\beta} \quad -(c)$$

we already know $\frac{dl}{d\sigma(u)}(\omega,\alpha,\beta)$ and $\frac{d\sigma(u)}{du}$ from previous derivation the forms is on $\frac{du}{d\beta}$

Since $u = \omega_1 z_1 + \omega_2 z_2$ & $z_2 = \sigma(\beta^T x)$

let $u = \beta^T x$

$$\frac{du}{d\beta} = \omega_2 \frac{d\sigma(u)}{dv}(1+e^{-u})^{-1} = e^{-v}/(1+e^{-v})^2 = \sigma(u)(1-\sigma(u)) \quad -(10)$$

$$\frac{du}{d\beta} = x \quad (11)$$

using (10) and (11) in (9) $\frac{du}{d\beta} = \omega_2\sigma(u)(1-\sigma(v)).x$ (12)

Substituting (3), (5) and (11) in (c)

$$\frac{dl}{d\beta}(\omega,\alpha,\beta) = -2(y-\sigma(u))\sigma(u)(1-\sigma(u))\omega_2\sigma(u)(1-\sigma(v))$$

Adding summation back for m data points:

$$\frac{dl}{d\beta}(\omega,\alpha,\beta) = -\sum_{i=1}^{m} 2(y-\sigma(u))\sigma(u^i)(1-\sigma(u^i))\omega_2\sigma(u^i)(1-\sigma(u^i))x^i$$

where $u = \omega^T z^i$ and $v = \beta^T x^i$

2. Comparing SVM with Neural Network

This question is to implement and compare **SVM and simple neural networks** for the same datasets we tried for the last homework. We suggest to use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar libraries in other programming languages of your choice to perform the tasks.

You may use a neural networks function sklearn.neural_network with hidden_layer_sizes=(5, 2). Tune the step size so you have reasonable results. You may use svc and tune the penalty term C to get reasonable results.

Part One (Divorce classification/prediction). (20 points)

We will compare using the same dataset as the last homework, which is about participants who completed the personal information form and a divorce predictors scale.

The data is a modified version of the publicly available at `https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set` (by injecting noise so you will not replicate the results on uci website). There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The dataset **q3.csv**. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce").

Build two classifiers using SVM and a simple neural networks. First random shuffle the data set. Then use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train_test_split to split the dataset.

Q2a)

(a) (15 points) Report testing accuracy for each of the two classifiers. Comment on their performance: which performs better and make a guess why it performs better in this setting.

Solution 2a)

Following steps were performed initially

1) Q3.csv dataset was imported into pandas dataframe.
2) Last Variable was considered as the Response variable which tells whether there was a divorce or no-divorce and all other variables were considered as Predictors.
3) The data set was split into Training and Testing dataset with Training owning 80% of the data Testing owning 20% of the data.
4) Applied Scikit-Learn module's SVM classifier algorithm to fit the training data set and prediction was done on the testing dataset and following results were observed as shown below

C Values	Misclassified Points	Accuracy
0.001	5/34	85.29
0.1	6/34	82.35
1	6/34	82.35
100	6/34	82.35

Support Vector Machine

```
C_Values=[.001,0.1,1,100]
for i in C_Values:
    clf = svm.LinearSVC(C=i)
    y_pred = clf.fit(Xtrain, ytrain).predict(Xtest)
    print("C=",i,",Points classified incorrectly - SVM out of %d points : %d" % (Xtest.shape[0], (ytest != y_pred).sum()))
    print("Accuracy of SVM :",(ytest == y_pred).sum()*100/Xtest.shape[0])
```

```
C= 0.001 ,Points classified incorrectly - SVM out of 34 points : 5
Accuracy of SVM : 85.29411764705883
C= 0.1 ,Points classified incorrectly - SVM out of 34 points : 6
Accuracy of SVM : 82.3529411764706
C= 1 ,Points classified incorrectly - SVM out of 34 points : 6
Accuracy of SVM : 82.3529411764706
C= 100 ,Points classified incorrectly - SVM out of 34 points : 6
Accuracy of SVM : 82.3529411764706
```

5) Applied Scikit-Learn module's Neural Network algorithm to fit the training data set and prediction was done on the testing dataset and following results were observed as shown below

Learning Rate	Misclassified Points	Accuracy
0.001	7/34	79.4
0.1	3/34	91.17
1	3/34	91.17
100	3/34	91.17

Neural Network

```
stages = [0.1,0.01,0.001,0.0001]
for j in stages:
    NeuralNetwork = MLPClassifier(solver='lbfgs', alpha=1e-5,learning_rate_init=j,hidden_layer_sizes=(5, 2))
    y_pred=NN.fit(Xtrain, ytrain).predict(Xtest)
    print("Learning rate=",j,", Points classified incorrectly - Neural Network out of %d points : %d" % (Xtest.shape[0], (ytest
    print("Accuracy of Neural Network :",(ytest == y_pred).sum()*100/Xtest.shape[0])
```

```
Learning rate= 0.1 , Points classified incorrectly - Neural Network out of 34 points : 7
Accuracy of Neural Network : 79.41176470588235
Learning rate= 0.01 , Points classified incorrectly - Neural Network out of 34 points : 3
Accuracy of Neural Network : 91.17647058823529
Learning rate= 0.001 , Points classified incorrectly - Neural Network out of 34 points : 3
Accuracy of Neural Network : 91.17647058823529
Learning rate= 0.0001 , Points classified incorrectly - Neural Network out of 34 points : 3
Accuracy of Neural Network : 91.17647058823529
```

Based on the output it looks like overall Neural Networks has a slightly higher edge over SVM. Its only when C_Value/Learning rate = 0.001, SVM wins over Neural Network in terms of accuracy

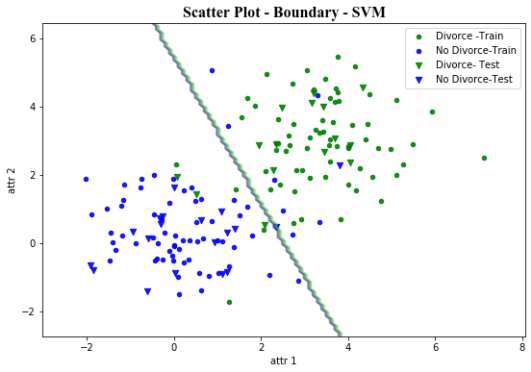
Q2b)

(b) (15 points) Use the first two features to train two new classifiers. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the two classifiers. Please clearly represent the data points with different labels using different colors.

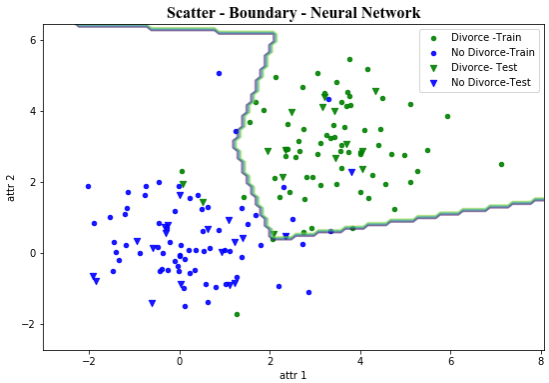
Solution 2b)

The two classifiers were plotted as shown screenshots of the plot is shown below

Plot 1: Support Vector Machine



Plot 2: Neural Networks



1) As seen in the screenshots above, for both the plots the green dots represent 'divorce' (1) and blue dots represents 'no divorce' (0).
2) After observing both the classifiers it appears that the Neural Network classifier is doing a better job of classification than the SVM classifier.
3) Due to higher accuracy for the Neural Network Classifier as compared to SVM classifier we see the Neural Network classifier has misclassified more number blue dots.

Part Two (Handwritten digits classification). (20 points) Repeat the above part (a) using the **MNIST Data** in our previous homework. Here, give "digit" 6 label $y = 1$, and give "digit" 2 label $y = 0$. All the pixels in each image will be the feature (predictor variables) for that sample (i.e., image). Our goal is to build classifiers such that given a new testing sample, we can tell it is a 2 or a 6. Using the first 80% of the samples for training and remaining 20% for testing. Report the classification accuracy on testing data, for each of the two classifiers. Comment on their performance: which performs better and make a guess why they perform better in this setting.

Solution 2c)

Following steps were performed initially

1) Data.mat and Label.mat file was imported using Scipy's loadmat library.
2) The data set was split into Training and Testing dataset with Training owning 80% of the data Testing owning 20% of the data.
3) Applied Scikit-Learn module's SVM classifier algorithm to fit the training data set and prediction was done on the testing dataset.
4) Applied Scikit-Learn module's Neural Network classifier algorithm to fit the training data set and prediction was done on the testing dataset.
5) Given below is the outcome of Step 3 and Step 4.

C Values	Misclassified Points	Accuracy
0.001	10/398	97.48
0.1	7/398	98.24
1	8/398	97.98
100	9/398	97.73

Learning Rate	Misclassified Points	Accuracy
0.001	4/398	98.99
0.1	5/398	98.74
1	8/398	97.98
100	13/398	96.73

Support Vector Machine

```
X2train, X2test, y2train, y2test = train_test_split(isomap_a.T, y, test_size=0.2, random_state=0)
for c in C_Values:
    clf1 = svm.LinearSVC(C=c)
    y2pred = clf1.fit(X2train, y2train).predict(X2test)
    print("C=",c,", Points incorrectly classified - SVM out of %d points : %d" % (X2test.shape[0], (y2test != y2pred).sum()))
    print("Accuracy SVM :",(y2test == y2pred).sum()*100/X2test.shape[0])
```

```
C= 0.001 , Points incorrectly classified - SVM out of 398 points : 10
Accuracy SVM : 97.48743718592965
C= 0.1 , Points incorrectly classified - SVM out of 398 points : 7
Accuracy SVM : 98.24120603015075
C= 1 , Points incorrectly classified - SVM out of 398 points : 8
Accuracy SVM : 97.98994974874371
C= 100 , Points incorrectly classified - SVM out of 398 points : 9
Accuracy SVM : 97.73869346733669
```

Neural Network

```
for a in stages:
    NeuralNetowrk1 =  MLPClassifier(solver='lbfgs', alpha=1e-5,learning_rate_init=a,hidden_layer_sizes=(5, 2))
    NeuralNetowrk1.fit(X2train, y2train)
    y2pred=NeuralNetowrk1.predict(X2test)
    print("Learning rate=",a,", Points incorrectly classified - Neural Network out of %d points : %d" % (X2test.shape[0], (y2tes
    print("Accuracy Neural Network :",(y2test == y2pred).sum()*100/X2test.shape[0])
```

```
Learning rate= 0.1 , Points incorrectly classified - Neural Network out of 398 points : 4
Accuracy Neural Network : 98.99497487437186
Learning rate= 0.01 , Points incorrectly classified - Neural Network out of 398 points : 5
Accuracy Neural Network : 98.74371859296483
Learning rate= 0.001 , Points incorrectly classified - Neural Network out of 398 points : 8
Accuracy Neural Network : 97.98994974874371
Learning rate= 0.0001 , Points incorrectly classified - Neural Network out of 398 points : 13
Accuracy Neural Network : 96.73366834170854
```

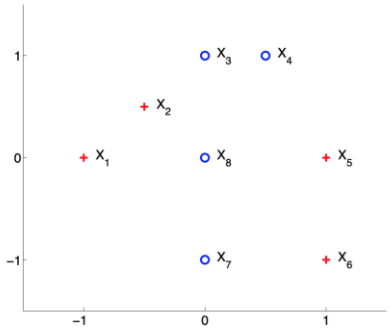
On the MNIST Data – on average SVM performs slightly better than Neural Network. In the case of not linearly separable data, SVMs have the powerful "kernel trick", which allows to map the data to a very high dimension space in which the data can be separable by a hyperplane, almost at zero cost.

3. AdaBoost

Consider the following dataset, plotting in Figure 1. The first two coordinates represent the value of two features, and the last coordinate is the binary label of the data.

$$X_1 = (-1, 0, +), X_2 = (-0.5, 0.5, +), X_3 = (0, 1, -), X_4 = (0.5, 1, -),$$
$$X_5 = (1, 0, +), X_6 = (1, -1, +), X_7 = (0, -1, -), X_8 = (0, 0, -).$$

In this problem, you will run through $T = 3$ iterations of AdaBoost with decision stumps (axis-aligned half planes) as weak learners.



Q3a)

(a) (20 points) For each iteration $t = 1, 2, 3$, compute ϵ_t, α_t, Z_t, D_t by hand (i.e., show all the calculation steps) and draw the decision stumps on Figure 1. Recall that Z_t is the normalization factor to ensure that the weights D_t sum to one. (*Hint: At each iteration, you may specify any reasonable decision rule h_t as you would like.*)

Solution 3a)

Given below is the data for which Adaboost algorithm is performed

Co-ordinate	Y1	Y2	Label	Actual Label
X1	-1	0	TRUE	1
X2	-0.5	0.5	TRUE	1
X3	0	1	FALSE	-1
X4	0.5	1	FALSE	-1
X5	1	0	TRUE	1
X6	1	-1	TRUE	1
X7	0	-1	FALSE	-1
X8	0	0	FALSE	-1

Round 1

1) There are 8 data points
   So, **m = 8**
2) Since this is the first round, so **t = 1**
3) Initially, we distribute weights in uniform distribution. I set weights of all instances to 1/m where n is the total number of instances.
   Weight = 1/8 = 0.125
4) Given below is the formula to derive Weighted Actual values.
   Weighted Actual = Weight * Actual Label
5) Given below is my decision rule I have chosen for round 1
   Decision Stump Prediction: **IF** *$Y_1 > 0.75$* **THEN 1 ELSE -1**
6) Based on decision rule, given below will be the formula for loss
   Loss: **IF** *Prediction = Actual Label* **THEN** 1 **ELSE** 0

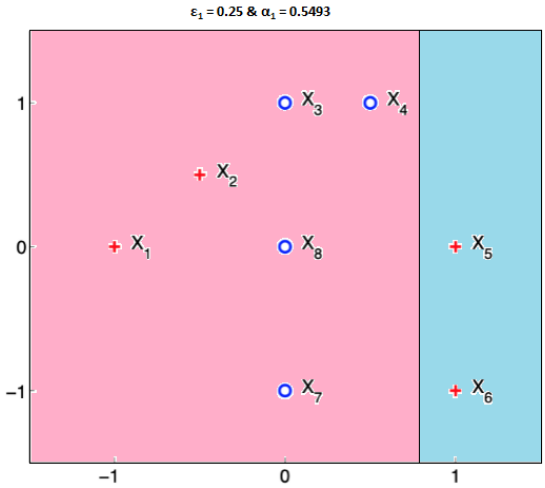
7) Given below is the table where total error ε_1, weight of classifier α_1, normalization constant Z_1 and normalized weight D_1(norm) is calculated. The upper table shows the calculation and the lower table shows the formula for calculation of

each column.

Co-ordinate	Y1	Y2	Actual Label	Weight (D_i)	Weighted_Actual	Prediction	Loss	Weight*Loss	D_1	Z_1	D_1 (Norm)
X1	-1	0	1	0.125	0.125	-1	1	0.125	0.216506351		0.25
X2	-0.5	0.5	1	0.125	0.125	-1	1	0.125	0.216506351		0.25
X3	0	1	-1	0.125	-0.125	-1	0	0	0.072168784		0.08
X4	0.5	1	-1	0.125	-0.125	-1	0	0	0.072168784		0.08
X5	1	0	1	0.125	0.125	1	0	0	0.072168784	0.866025404	0.08
X6	1	-1	1	0.125	0.125	1	0	0	0.072168784		0.08
X7	0	-1	-1	0.125	-0.125	-1	0	0	0.072168784		0.08
X8	0	0	-1	0.125	-0.125	-1	0	0	0.072168784		0.08

ε_1 = (0.125+0.125)	0.25
$\alpha_1 = \ln[(1-\varepsilon_1)/\varepsilon_1] / 2 = \ln[(1-0.75)/0.75] / 2$	0.549306144
$D_{i+1} = D_1 * e^{(-\alpha 1 * \text{Actual Label} * \text{Prediction})}$	
Z_1 = sum(Di+1)	0.866025404
$D_{i+1}(\text{Norm}) = D_i/Z_1 * \exp^{(-\alpha 1 * \text{Actual Label} * \text{Prediction})}$	

Decision Stump Plot



Analysis: Based on the output of the Round 1 and the decision stump plot given below are the analysis

- All the points except X_1 and X_2 are correctly identified. X_1 and X_2 are actually +1, but the round 1 decision stump identified it as -1. Therefore, the two points X_1 and X_2 results in loss as they are not predicted correctly
- Since two points X_1 and X_2 are incorrectly identified, the total error ε_1 = sum of the weight of wrongly identified points
  ε_1 = 0.125+0.125= 0.25
- The weight of the classifier $\alpha_1 = \ln[(1-\varepsilon_1)/\varepsilon_1] / 2$
  $\alpha_1 = \ln [(1 - 0.75)/0.75] / 2 = 0.5493$
- The weights D_1 was initially calculated without the aim of normalizing it. In order to normalize it so that the weights sum up to 1, we had to calculate the Normalization Constant Z_1 = Sum of all weights
  Z_1 = 0.866
- Finally, the weights D1 were normalized using the normalization constant Z_1 such that sum of all weights equals to 1.

Round 2

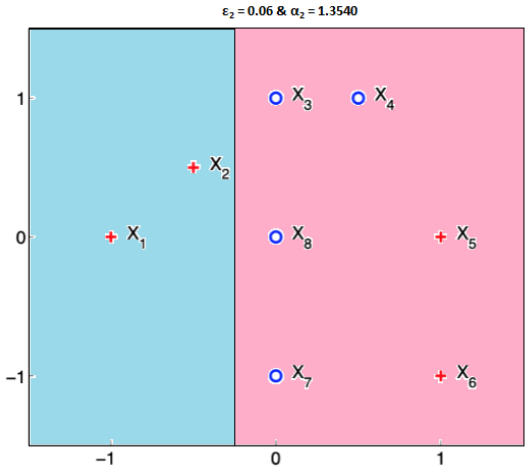
1) There are 8 data points
   So, **m = 8**
2) Since this is the second round, so **t = 2**
3) We transfer weights from Round 1 to this Round 2. The points which were correctly classified in Round 1 are given lower weights and points which are wrongly classified in Round 1 are given higher weights based on formula below
   Weight New = $D_1/2*(\varepsilon_1)$ for wrongly classified points
   Weight New = $D_1/2*(1-\varepsilon_1)$ for correctly classified points
4) Given below is my decision rule I have chosen for round 2
   Decision Stump Prediction: **IF** *$Y_1 < -0.25$* **THEN 1 ELSE -1**
5) Based on decision rule, given below will be the formula for loss
   Loss: **IF** *Prediction = Actual Label* **THEN** 1 **ELSE** 0

6) Given below is the table where total error ε_2, weight of classifier α_2, normalization constant Z_2 and normalized weight D_2(norm) is calculated. The upper table shows the calculation and the lower table shows the formula for calculation of each column.

Co-ordinate	Y1	Y2	Actual Label	Weight (D1)Formula	Weight (D_1)	Weighted_Actual	Prediction	Loss	Weight*Loss	D_2	Z_2	D_2 (Norm)
X1	-1	0	1	$D_1/2*(\varepsilon_1)$	0.03125	0.03125	1	0	0	0.008068715		0.027777778
X2	-0.5	0.5	1	$D_1/2*(\varepsilon_1)$	0.03125	0.03125	1	0	0	0.008068715		0.027777778
X3	0	1	-1	$D_1/2*(1-\varepsilon_1)$	0.03125	-0.03125	-1	0	0	0.008068715		0.027777778
X4	0.5	1	-1	$D_1/2*(1-\varepsilon_1)$	0.03125	-0.03125	-1	0	0	0.008068715		0.027777778
X5	1	0	1	$D_1/2*(1-\varepsilon_1)$	0.03125	0.03125	-1	1	0.03125	0.12103073	0.290473751	0.416666667
X6	1	-1	1	$D_1/2*(1-\varepsilon_1)$	0.03125	0.03125	-1	1	0.03125	0.12103073		0.416666667
X7	0	-1	-1	$D_1/2*(1-\varepsilon_1)$	0.03125	-0.03125	-1	0	0	0.008068715		0.027777778
X8	0	0	-1	$D_1/2*(1-\varepsilon_1)$	0.03125	-0.03125	-1	0	0	0.008068715		0.027777778

ε_2 = 0.03125 + 0.03125	0.06
$\alpha_2 = \ln[(1-\varepsilon_2)/\varepsilon_2] / 2 = \ln[(1-0.11)/0.11] / 2$	1.354025101
Weight New = $D_1/2*(\varepsilon_1)$ for wrongly classified points	
Weight New = $D_1/2*(1-\varepsilon_1)$ for correctly classified points	
$D_{i+1} = D_2 * \exp^{(-\alpha 2 * \text{Actual Label} * \text{Prediction})}$	
Z_2 = sum(Di+1)	0.290473751
D_{i+1}(Norm) = $D_i/Z_2 * \exp^{(-\alpha 1 * \text{Actual Label} * \text{Prediction})}$	

Decision Stump Plot



Analysis: Based on the output of the Round 2 and the decision stump plot given below are the analysis

- All the points except X_5 and X_6 are correctly identified. X_5 and X_6 are actually +1, but the round 1 decision stump predicted it as -1. Therefore, the two points X_5 and X_6 results in loss as they are not predicted correctly
- Since two points X_5 and X_6 are incorrectly identified, the total error ε_2 = sum of the weight of wrongly identified points
  ε_2= 0.03125 + 0.03125= 0.06
- The weight of the classifier $\alpha_2 = \ln[(1-\varepsilon_2)/\varepsilon_2] / 2$
  α_2= ln [(1 – 0.11)/0.11] / 2 = 1.3540
- The weights D_2 was initially calculated without the aim of normalizing it. In order to normalize it so that the weights sum up to 1, we had to calculate the Normalization Constant Z_2 = Sum of all weights
  Z_2 = 0.2904
- Finally, the weights D_2 were normalized using the normalization constant Z_2 such that sum of all weights equals to 1.

Round 3

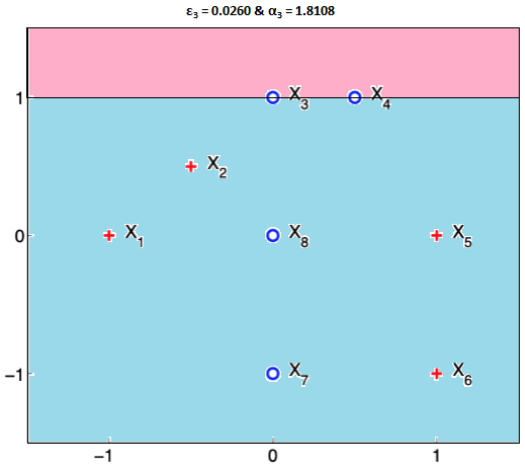
1) There are 8 data points
   So, **m = 8**
2) Since this is the third round, so **t = 3**
3) We transfer weights from Round 2 to Round 3. The points which were correctly classified in Round 2 are given lower weights and points which are wrongly classified in Round 2 are given higher weights based on formula below
   Weight New = $D_2/2*(\varepsilon_2)$ for wrongly classified points
   Weight New = $D_2/2*(1-\varepsilon_2)$ for correctly classified points
4) Given below is my decision rule I have chosen for round 2
   Decision Stump Prediction: **IF** *$Y_2 < 1$* **THEN 1 ELSE -1**
5) Based on decision rule, given below will be the formula for loss
   Loss: **IF** *Prediction = Actual Label* **THEN** 1 **ELSE** 0

6) Given below is the table where total error ε_3, weight of classifier α_3, normalization constant Z_3 and normalized weight D_3(norm) is calculated. The upper table shows the calculation and the lower table shows the formula for calculation of each column.

Co-ordinate	Y1	Y2	Actual Label	Weight (D2)Formula	Weight (D_2)	Weighted_Actual	Prediction	Loss	Weight*Loss	D_3	Z_3	D_3 (Norm)
X1	-1	0	1	$D_2/2*(1-\varepsilon_2)$	0.013020833	0.013020833	1	0	0	0.002129134	0.172034021	0.012376238
X2	-0.5	0.5	1	$D_2/2*(1-\varepsilon_2)$	0.013020833	0.013020833	1	0	0	0.002129134		0.012376238
X3	0	1	-1	$D_2/2*(1-\varepsilon_2)$	0.013020833	-0.013020833	-1	0	0	0.002129134		0.012376238
X4	0.5	1	-1	$D_2/2*(1-\varepsilon_2)$	0.013020833	-0.013020833	-1	0	0	0.002129134		0.012376238
X5	1	0	1	$D_2/2*(\varepsilon_2)$	0.013020833	0.013020833	1	0	0	0.002129134		0.012376238
X6	1	-1	1	$D_2/2*(\varepsilon_2)$	0.013020833	0.013020833	1	0	0	0.002129134		0.012376238
X7	0	-1	-1	$D_2/2*(1-\varepsilon_2)$	0.013020833	-0.013020833	1	1	0.013020833	0.079629609		0.462871287
X8	0	0	-1	$D_2/2*(1-\varepsilon_2)$	0.013020833	-0.013020833	1	1	0.013020833	0.079629609		0.462871287

ε_3 = 0.0130208333333333 + 0.0130208333333333	0.026041667
$\alpha_3 = \ln[(1-\varepsilon_3)/\varepsilon_3] / 2 = \ln[(1-0.081)/0.081] / 2$	1.810835352
Weight New = $D_2/2*(\varepsilon_2)$ for wrongly classified points	
Weight New = $D_2/2*(1-\varepsilon_2)$ for correctly classified points	
$D_{i+1} = D_2 * \exp(-\alpha_2 * \text{Actual Label} * \text{Prediction})$	
Z_3 = sum(Di+1)	0.172034021
D_{i+1}(Norm) = $D_i/Z_3 * \exp(-\alpha_1 * \text{Actual Label} * \text{Prediction})$	

Decision Stump Plot



Analysis: Based on the output of the Round 3 and the decision stump plot given below are the analysis

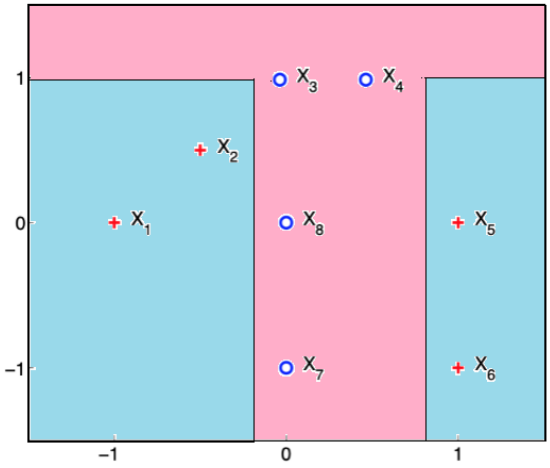
- All the points except X_7 and X_8 are correctly identified. X_7 and X_8 are actually +1, but the round 1 decision stump predicted it as -1. Therefore, the two points X_7 and X_8 results in loss as they are not predicted correctly
- Since two points X_7 and X_8 are incorrectly identified, the total error ε_3 = sum of the weight of wrongly identified points
  ε_3= 0.0130208333333333 + 0.0130208333333333= 0.02604
- The weight of the classifier $\alpha_3 = \ln[(1-\varepsilon_3)/\varepsilon_3] / 2$
  α_3= ln [(1 – 0.081)/0.081] / 2 = 1.8108
- The weights D_3 was initially calculated without the aim of normalizing it. In order to normalize it so that the weights sum up to 1, we had to calculate the Normalization Constant Z_3 = Sum of all weights
  Z_3 = 0.17203
- Finally, the weights D_3 were normalized using the normalization constant Z_3 such that sum of all weights equals to 1.

Summary

t	ε_t	α_t	Z_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$
1	0.25	0.549306144	1.443368578	0.25	0.25	0.08	0.08	0.08	0.08	0.08	0.08
2	0.0625	1.3540	1.270138004	0.027777778	0.027777778	0.027777778	0.027777778	0.416666667	0.416666667	0.027777778	0.027777778
3	0.026041667	1.810835352	0.172034021	0.012376238	0.012376238	0.012376238	0.012376238	0.012376238	0.012376238	0.462871287	0.462871287

Co-ordinate	α_1	h_1	α_2	h_2	α_3	h_3	$\sum(\alpha_t h_t)$	H_{final} = Sign(Sum($\alpha_t h_t$))
X1	0.549306144	-1	1.354025101	1	1.810835352	1	2.615554308	1
X2		-1		1		1	2.615554308	1
X3		-1		-1		-1	-3.714166597	-1
X4		-1		-1		-1	-3.714166597	-1
X5		1		-1		1	1.006116396	1
X6		1		-1		1	1.006116396	1
X7		-1		-1		1	-0.092495893	-1
X8		-1		-1		1	-0.092495893	-1

Final Classifier



Finally, we have the values of $\alpha_1, h_1, \alpha_2, h_2$ and α_3, h_3. Now we calculate the value of H_{Final} using the formula below

$H_{Final} = \alpha_1 * h_1 + \alpha_2 * h_2 + \alpha_3 * h_3$

Looking at the summary table above the AdaBoost algorithm implemented above in 3 rounds has predicted all the points correctly without any misclassification.

Q3b)

(b) (20 points) What is the training error of AdaBoost? Give a one-sentence reason for why AdaBoost outperforms a single decision stump.

Solution 3b)

Given below is the solution for calculating the Training Error of the AdaBoost algorithm implemented in Q3a.
Comparing Actual Labels with AdaBoost estimates

Training Error Calculation

Co-ordinate	Actual Labels	H_{final}	Misclassified ?
X1	1	1	No
X2	1	1	No
X3	-1	-1	No
X4	-1	-1	No
X5	1	1	No
X6	1	1	No
X7	-1	-1	No
X8	-1	-1	No

Total Misclassified Points: 0
Training Error = (0/8) *100 = 0%

As seen above, none all the points were correctly classified i.e. not a single point was misclassified. So as a result the training error is 0/8* 100 = 0%

AdaBoost outperforms single decision stump because, in AdaBoost algorithm because incorrectly classified values in the weak learners are weighted higher (boosted) in every subsequent iteration till all the points are correctly classified.