**E9 253: Neural Network and Learning Systems** **Semester 2020-2021**

# Home Work 1: Solution

*Course Coordinator: Prof. Shayan S. Garani* *Scribes: Ankit Kumar Gupta,Navneet Kaur, Sajin S*

**Note**: *LaTeX template courtesy of UC Berkeley EECS dept.*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Course Coordinator.*

## Problem 1

Sketch the following activation functions (a) sigmoid (b) logistic (c) ReLu and (d) leaky ReLu. Assume any scaling constants in the above functions to be unity. How does the derivative of the above activation functions behave near zero?

**Sigmoid and Logistic function:** Sigmoid and Logistic functions are mathematical functions having a characteristics "S" shaped curve. Sigmoid (function like tanh) takes values from -1 to 1 where logistic function (functions like $\frac{1}{1+e^{-z}}$) takes values from 0 to 1. Both functions are differentiable at zero.

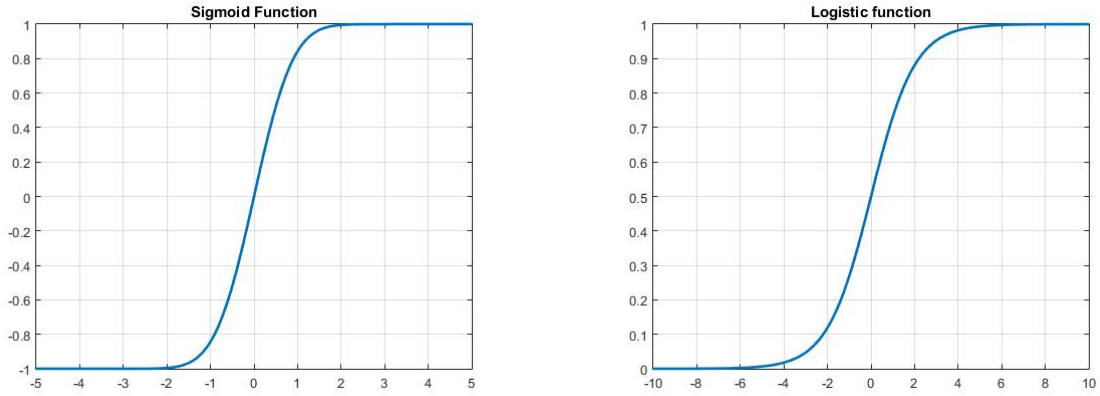


Figure 1.1: Sigmoid and logistic activation functions

Derivative of sigmoid/logistic function exist at origin and continuous in nature. But it can see that for large positive and negative value of inputs, gradient become very small (Vanishing gradient) and this can be a problem while learning.

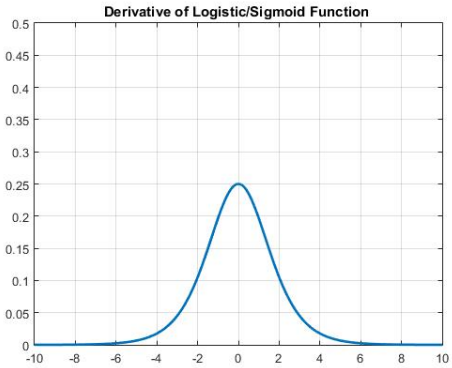


Figure 1.2: Shape of derivative of Sigmoid/Logistic function

**ReLU:** Relu activation function is defined as $\max(0, x)$. Thus, it takes 0 value for all negative inputs. Derivative of Relu activation function takes value 1 and 0 for $x > 0$ and $x < 0$, respectively. But derivative does not exist at $x = 0$.

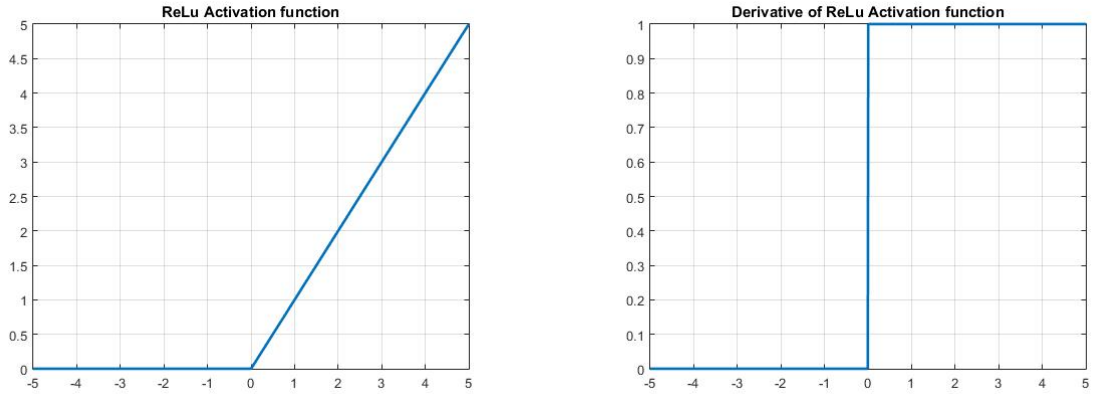


Figure 1.3: ReLU Activation function and derivative

**Leaky ReLU:** Leaky ReLu is a modified version ReLu function which is defined $\max(\alpha x, x)$ where $\alpha$ is usually a small value (Normally 0.01). Leaky ReLu is also not differentiable at zero as seen from the plot. But it solves the issue of vanishing gradient seen in sigmoid activation functions.

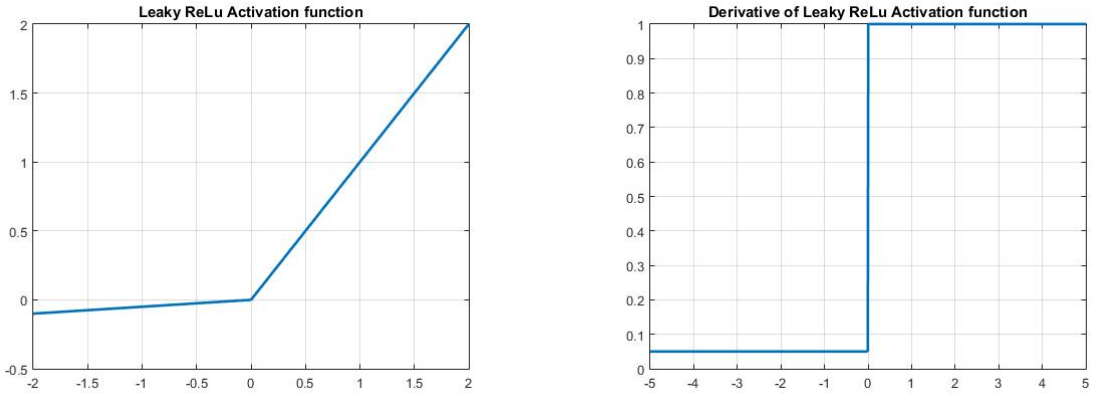


Figure 1.4: Leaky ReLU Activation function($\alpha$=0.05) and derivative

# Problem 2:

Solve problems 1.1 and 1.2 from Haykins book (3rd edition).

## Solution for Textbook Problem: 1.1

In Perceptron algorithm, if x(n) is correctly classified, we have d(n)=y(n). From equation 1.21, if $x(n) \in C_1$ and $d(n) = +1$, we need y(n)=d(n)=1, i.e. $\text{sgn}(w^T(n)x(n)) = 1$ from equation 1.22 or $w^T(n)x(n) > 0$ from equation 1.19. Similarly, if $x(n) \in C_1$, we can say that we need $w^T(n)x(n) < 0$. Thus, when x(n) is correctly classified the update rule is the following:

$$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n) \tag{1.1}$$

$$w(n+1) = \begin{cases} w(n) + \eta.0.x(n) & \text{if } x(n) \in C_1, w^T(n)x(n) > 0 \\ w(n) + \eta.0.x(n) & \text{if } x(n) \in C_2, w^T(n)x(n) < 0 \end{cases} \tag{1.2}$$

$$w(n+1) = \begin{cases} w(n) & \text{if } x(n) \in C_1, w^T(n)x(n) > 0 \\ w(n) & \text{if } x(n) \in C_2, w^T(n)x(n) < 0 \end{cases} \tag{1.3}$$

which is same as equation 1.5 of Rosenblatt's elementary perceptron.

In second case, let x(n) is not correctly classified. Again from 1.19-1.21 we have,

$$x(n) \in C_1 \implies d(n) = +1 \implies y(n) = -1 \tag{1.4}$$

or equivalently

$$\text{sgn}(w^T(n)x(n)) = -1 \implies w^T(n)x(n) < 0 \tag{1.5}$$

Similarly,

$$x(n) \in C_2 \implies d(n) = -1 \implies y(n) = +1 \tag{1.6}$$

or equivalently

$$\text{sgn}(w^T(n)x(n)) = +1 \implies w^T(n)x(n) > 0 \tag{1.7}$$

Thus the update formula for x(n) mis-classified case is the following:

$$w(n+1) = \begin{cases} w(n) + \eta.[(+1) - (-1)].x(n) & \text{if } x(n) \in C_1, w^T(n)x(n) < 0 \\ w(n) + \eta.[(-1) - (+1)].x(n) & \text{if } x(n) \in C_2, w^T(n)x(n) > 0 \end{cases} \tag{1.8}$$

$$w(n+1) = \begin{cases} w(n) + 2\eta x(n) & \text{if } x(n) \in C_1, dw^T(n)x(n) < 0 \\ w(n) - 2\eta x(n) & \text{if } x(n) \in C_2, w^T(n)x(n) > 0 \end{cases} \tag{1.9}$$

which is same as equation 1.6 of Rosenblatt's perceptron update rule with $\eta(n)$ as a fixed constant, i.e. $\eta(n) = 2\eta$ which is positive. Thus, we may infer that (Lipmann, 1987) Perceptron convergence algorithm is consistent with Rosenblatt's elementary perceptron weight update rule.

## Solution for Textbook Problem: 1.2

Let $w = [w_1 w_2 ... w_m]^T$ be the weight vector for given neural network. From the given perceptron classification decision, we can say that there exists a weight vector $w_o$ such that the output $y > \zeta$ if $x \in C_1$ and $y \leq \zeta$ if $x \in C_2$. In other words, $y = \zeta$ corresponds to $w = w_o$ which forms the decision boundary. From the signal flow graph, we have the following output:

$$y = \tanh(v/2) = \frac{\exp(v/2) - \exp(v/2)}{\exp(v/2) + \exp(v/2)} \tag{1.10}$$

or

$$y = \frac{\exp v - 1}{\exp v + 1} \tag{1.11}$$

where the induced field

$$v = \sum_{i=1}^{m} w_i x_i + b = w^T x + b \tag{1.12}$$

At decision boundary, we have $v = w_o^T x + b$ and $y = \zeta$. Solving both equations simultaneously we get,

$$\frac{\exp(w_o^T x + b) - 1}{\exp(w_o^T x + b) + 1} = \zeta \tag{1.13}$$

$$\exp(w_o^T x + b) = \frac{1 + \zeta}{1 - \zeta} \tag{1.14}$$

Taking natural logarithm on both sides, we get decision boundary as:

$$w_o^T x = \ln \frac{1+\zeta}{1-\zeta} - b = \text{constant} \tag{1.15}$$

The above form of $w_o$ proves that decision boundary is indeed a hyperplane.

# Problem 3

Consider a 3-class classification problem, comprising labels $\omega_i$, i = 1; 2; 3 corresponding to data points which are uniformly distributed over [-1; 1], [-2; 2] and [-4; 4] respectively. The corresponding apriori probabilities for the classes are $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{6}$. Are the points linearly separable? Determine the optimum thresholds and provide a Bayes decision rule to decide the label for a point randomly sampled from the interval [-4; 4]. Compute the probability of misclassification error.

Let $\omega_1, \omega_2$ and $\omega_3$ are three classes. Prior probabilities of each class is given:
$P(\omega_1) = \frac{1}{2}, P(\omega_2) = \frac{1}{3}, P(\omega_3) = \frac{1}{6}$

Also $P(\, x \mid \omega_1 \,) = \begin{cases} \frac{1}{2}, & \text{if } -1 \le x \le 1. \\ 0, & \text{otherwise.} \end{cases}$ $P(\, x \mid \omega_2 \,) = \begin{cases} \frac{1}{4}, & \text{if } -2 \le x \le 2. \\ 0, & \text{otherwise.} \end{cases}$ $P(\, x \mid \omega_3 \,) = \begin{cases} \frac{1}{8}, & \text{if } -4 \le x \le 4. \\ 0, & \text{otherwise.} \end{cases}$

Since classes are having common intersection, these points are not linearly separable.

**In region $\mathbf{x \in [-1, 1]}$:** $P(\, \omega_i \mid x \,)$ for each class label $i = \{1, 2, 3\}$ can be calculated using Bayes rule.

$$P(\, x \in \omega_i \mid x \in [-1, 1] \,) = \frac{P(\, x \in [-1, 1] \mid x \in \omega_i \,) P(\omega_i)}{P(x \in [-1, 1])} \tag{1.16}$$

Denominator term is common for all the classes, so it is enough to calculate the numerator term for identifying which class is having probability

$P(\, x \in [-1, 1] \mid x \in \omega_1 \,) = \frac{1}{2} \times 2 = 1$
$P(\, x \in [-1, 1] \mid x \in \omega_2 \,) = \frac{1}{4} \times 2 = \frac{1}{2}$
$P(\, x \in [-1, 1] \mid x \in \omega_3 \,) = \frac{1}{8} \times 2 = \frac{1}{4}$

So numerator term in Bayes rule for each class will be
$P(\, x \in [-1, 1] \mid x \in \omega_1 \,) P(\omega_1) = \frac{1}{2}$
$P(\, x \in [-1, 1] \mid x \in \omega_2 \,) P(\omega_2) = \frac{1}{6}$
$P(\, x \in [-1, 1] \mid x \in \omega_3 \,) P(\omega_3) = \frac{1}{24}$

Thus $P(\, x \in \omega_1 \mid x \in [-1, 1] \,) > P(\, x \in \omega_2 \mid x \in [-1, 1] \,) > P(\, x \in \omega_3 \mid x \in [-1, 1] \,)$ and we will predict label as class $\omega_1$ for $x \in [-1, 1]$

**In region $\mathbf{x \in R = [-2, -1] \cup [1, 2]}$:** Similar way as above,
$P(\, x \in R \mid x \in \omega_1 \,) = 0$
$P(\, x \in R \mid x \in \omega_2 \,) = \frac{1}{4} \times 2 = \frac{1}{2}$
$P(\, x \in R \mid x \in \omega_3 \,) = \frac{1}{8} \times 2 = \frac{1}{4}$
So numerator term in Bayes rule for each class will be

$P(\, x \in R \mid x \in \omega_1\,)P(\omega_1) = 0$
$P(\, x \in R \mid x \in \omega_2\,)P(\omega_2) = \frac{1}{6}$
$P(\, x \in R] \mid x \in \omega_3\,)P(\omega_3) = \frac{1}{24}$
Thus $P(\, x \in \omega_2 \mid x \in R\,) > P(\, x \in \omega_3 \mid x \in R\,) > P(\, x \in \omega_1 \mid x \in [-1, 1]\,)$ and we will predict label as class $\omega_2$ for $x \in R = [-2, -1] \cup, 2]$

**In region $\mathbf{x \in R = [-4, -2] \cup [2, 4]}$:** Repeating the same method as above,
$P(\, x \in R \mid x \in \omega_1\,) = 0$
$P(\, x \in R \mid x \in \omega_2\,) = 0$
$P(\, x \in R \mid x \in \omega_3\,) = \frac{1}{8} \times 2 = \frac{1}{2}$
Numerator term in Bayes rule for each class will be
$P(\, x \in R \mid x \in \omega_1\,)P(\omega_1) = 0$
$P(\, x \in R \mid x \in \omega_2\,)P(\omega_2) = 0$
$P(\, x \in R] \mid x \in \omega_3\,)P(\omega_3) = \frac{1}{12}$
Thus in region $R = [-4, -2] \cup [2, 4]$ we will predict label as class $\omega_3$.

**Probability of misclassification** :

$P(\text{Error}) = P(x \in [-1, 1] \mid x \in \omega_2)P(\omega_2) + P(x \in [-1, 1] \mid x \in \omega_3)P(\omega_3) + P(x \in [-2, -1] \cup [1, 2] \mid x \in \omega_3)P(\omega_3)$

$P(\text{Error}) = \frac{1}{4} \times 2 \times \frac{1}{3} + \frac{1}{8} \times 2 \times \frac{1}{6} + \frac{1}{8} \times 2 \times \frac{1}{6} = \frac{1}{4} = 25\%$

## Problem 4a:

Scatter plot of three classes is given below:

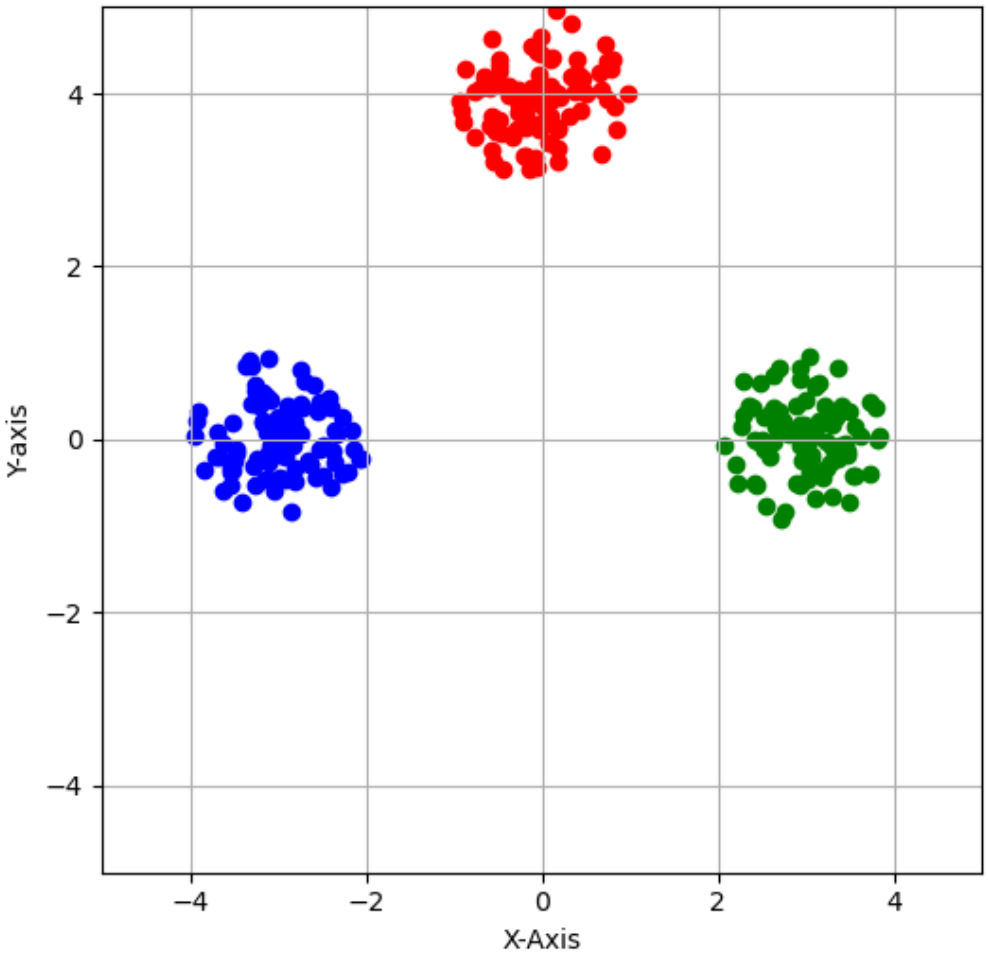


Figure 1.5: Scatter Plot for three classes

## Problem 4b:

Procedure to construct a perceptron based algorithm to classify the points:
**i**-Create a training data set such that it contains information to which class it belongs to(+1) and does not belongs to(-1).

| one | X-axis | Y-axis | c1 | c2 | c3 |
|---|---|---|---|---|---|
| 1 | -0.83372 | 4.229096 | 1 | -1 | -1 |
| 1 | -3.23591 | -0.0371 | -1 | 1 | -1 |
| 1 | 2.959617 | 0.020573 | -1 | -1 | 1 |

Figure 1.6: Sample data

**ii**-Implement one vs all algorithm, which invloves training a single classifier per class,with the samples of that class as positive and other samples as negative.Here, I have used matrices to compute values for each class, instead of using three different classifier models.
**iii**-After training the model with perceptron algorithm,it might happen that the model classifies a point into more than one class, so we assign confidence to each classification made given by $\max(\mathbf{w}^T\mathbf{x})$).[Note:Weight matrix used by me is of 3x3 size,where each row gives eqn of one hyperplane]

## Problem 4c:

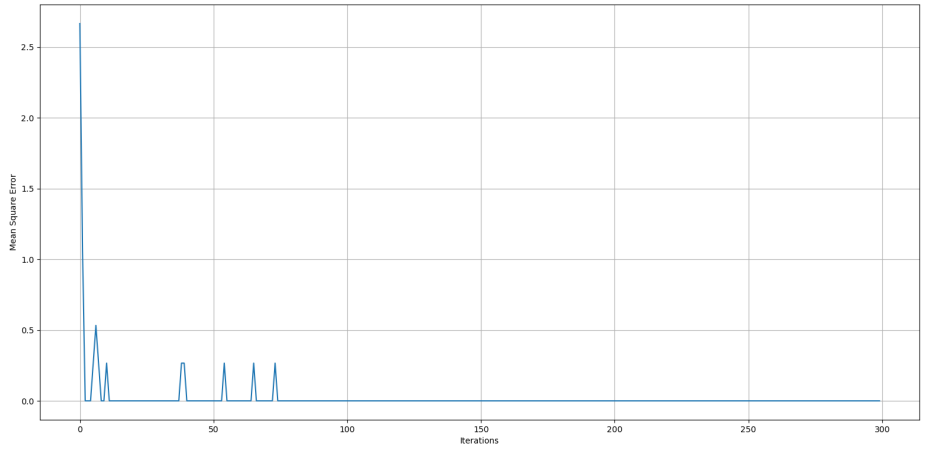


Figure 1.7: Error Trajectory for batch mode on train data

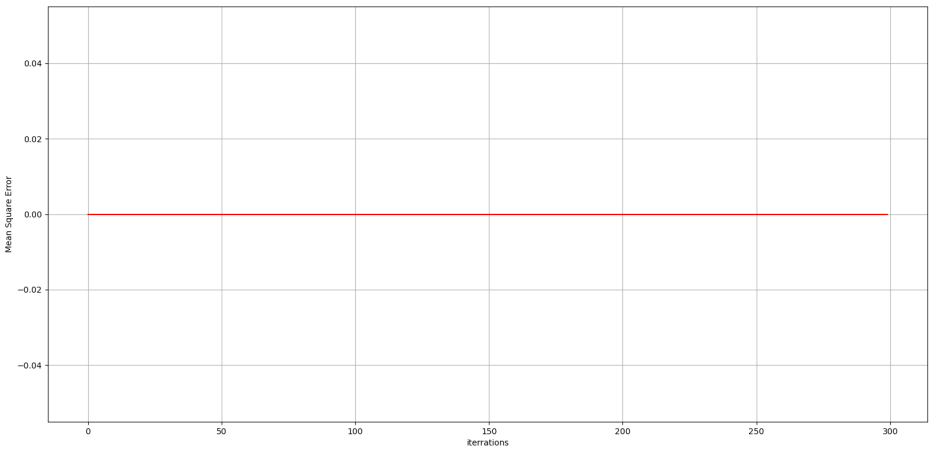


Figure 1.9: Error Trajectory for a particular test data

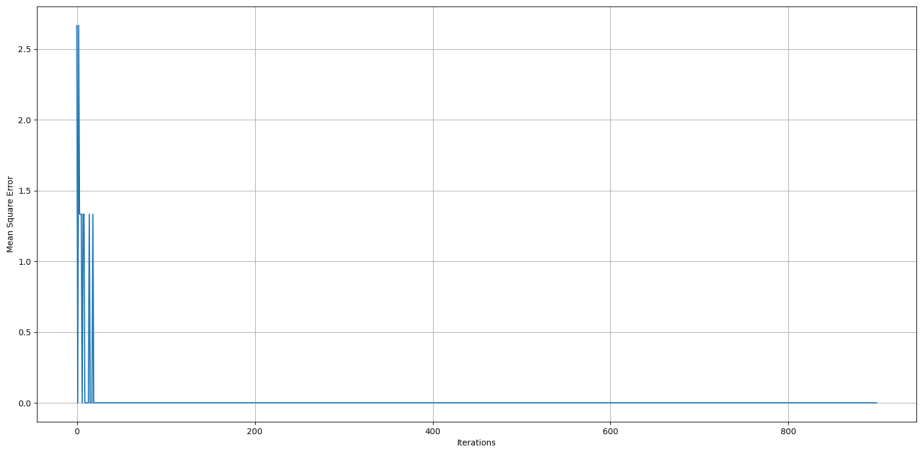


Figure 1.8: Error Trajectory for online mode on train data

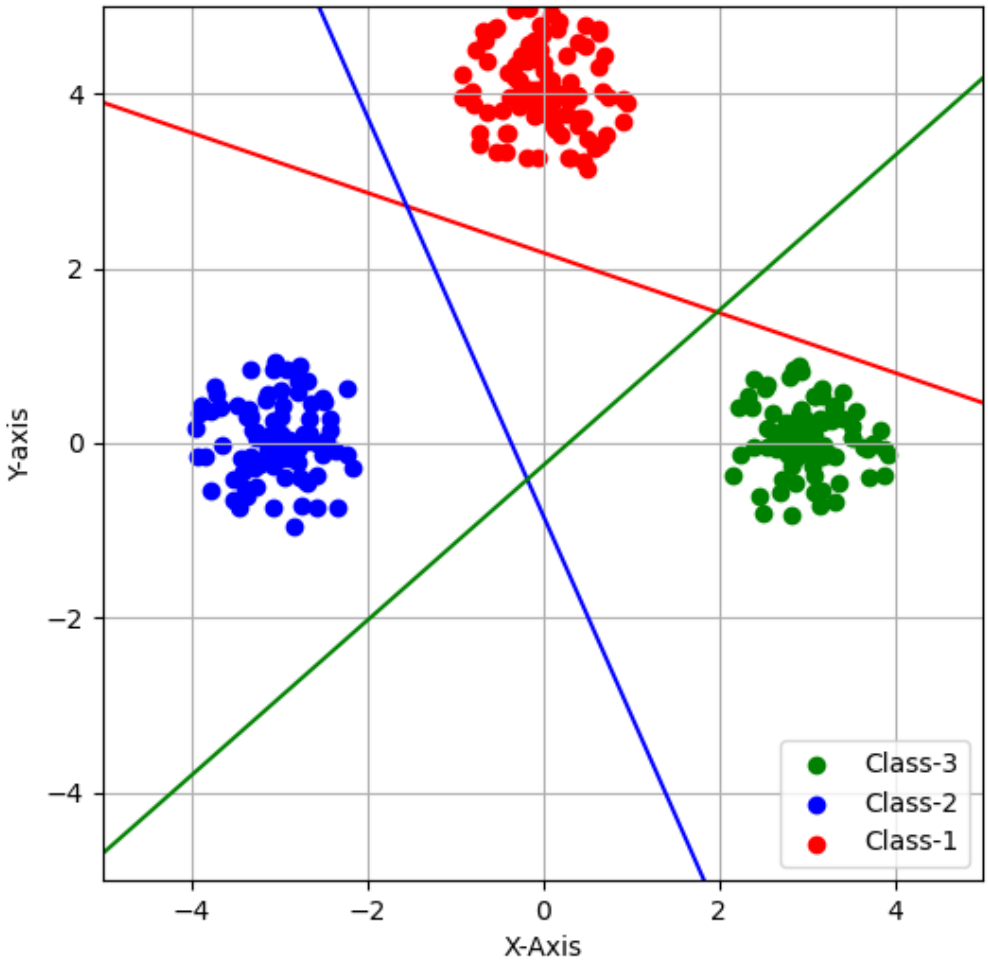


Figure 1.10: Final Decision boundaries after convergence

Final Decision Boundary video link- Click here for video.

The three decision boundaries cannot be collinear, as the three clusters are spatially distributed approximately at the vertices of an isoscles triangle.If the data points had been present one above the another or side by side,then we could atleast get parallel decision boundaries.

Even after reshuffling the data with different learning rates after each epochs and random weight initializations, the decision boundary still converges given proper training time.However,the time after which it converges varies accordingly.
Random Weight and learning rate initialisation video link- Click here for video.

## Problem 4d:

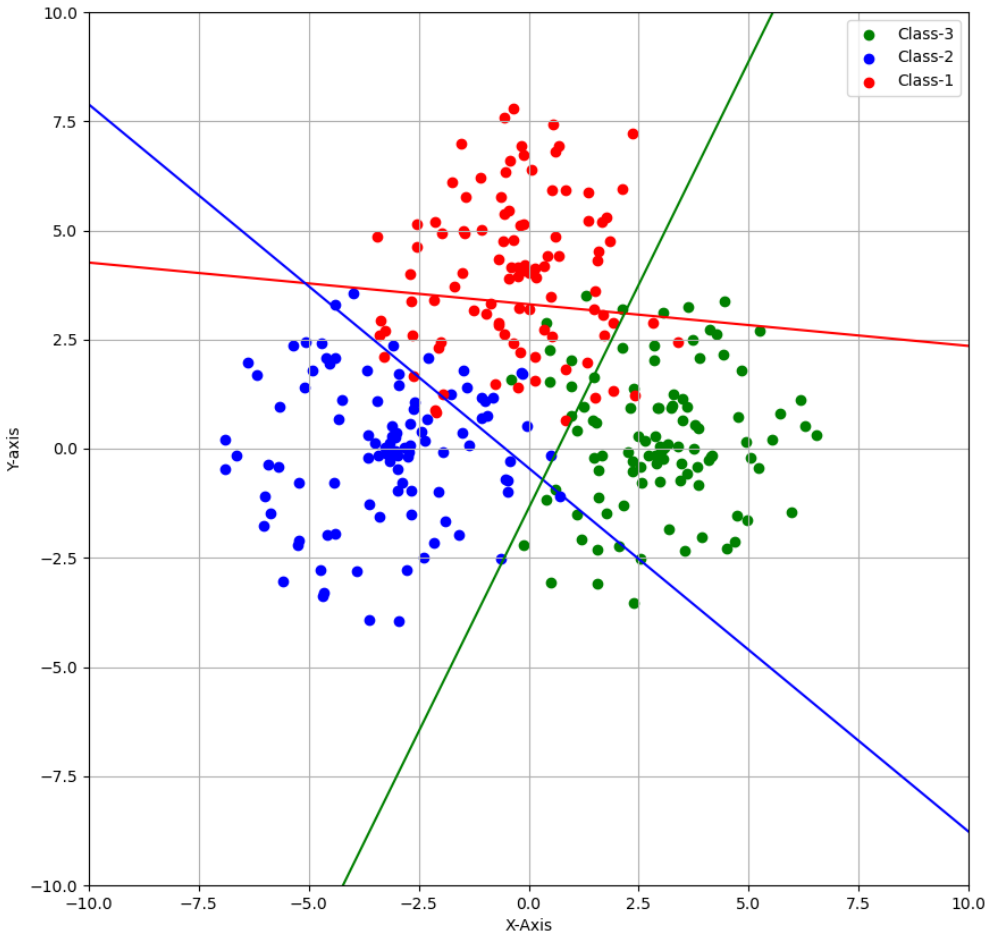


Figure 1.11: Decision boundary with circle radius as 4 units

As the clusters are overlapping,proper hyperplanes does not exist.The decision boundaries keep changing and never settle down.Hence perceptron model will not be able to assign proper classes.
link-
Click here for video.