CS-584 Machine Learning
Assignment 2
Name- Soham Kamble
CWID - A20517098

Problem 1
Given
X, Y, Z are binary variables
Assume Y, Z to be independent
$P(Y=1) = 0.9$
$P(X=1 \mid Y=1, Z=1) = 0.6$
$P(Z=1) = 0.8$
$P(X=1 \mid Y=1, Z=0) = 0.1$
$P(X=1 \mid Y=0) = 0.2$

1.) Compute $P(X=1)$

$$P(X=1) = P(X=1 \mid Y=1, Z=1) P(Y=1, Z=1) + P(X=1 \mid Y=1, Z=0) P(Y=1, Z=0) + P(X=1 \mid Y=0) P(Y=0)$$

$$= P(X=1 \mid Y=1, Z=1) P(Y=1) P(Z=1) + P(X=1 \mid Y=1, Z=0) P(Y=1) P(Z=0) + P(X=1 \mid Y=0) P(Y=0)$$

$$= 0.6 \times 0.9 \times 0.8 + 0.1 \times 0.9 \times (1-0.8) + 0.2 (1-0.9)$$

$$= 0.47$$

2.) Compute $E(Y)$

$E(Y) = 1 \times P(Y=1) + 0 \times P(Y=0) = P(Y=1) = 0.9$

3.) Y takes values 115 & 20

$P(Y=115) = 0.9$

Compute $E(Y)$

$E(Y) = 115 \times P(Y=115) + 20 \times P(Y=20)$

$= 115 \times 0.9 + 20(1-0.9)$

$= 105.5$

Problem 2

1. Consider X to be the event that the phone is defective

$P(X) = P(X|A) \times P(A) + P(X|B) \times P(B) + P(X|C) \times P(C)$

$= (0.02 \times 0.2) + (0.01 \times 0.3) + (0.0005 \times 0.5)$

$= 0.004 + 0.003 + 0.00025$

$= 0.00725$

2.) $P(A|X) = \frac{P(X|A) \times P(A)}{P(X)}$

$= \frac{0.02 \times 0.2}{0.00725}$

$= 0.55172$

3.) $P(B|X) = \frac{P(X|B) \times P(B)}{P(X)}$

$= \frac{0.01 \times 0.3}{0.00725}$

$= 0.4137$

4.) $P(C|X) = \frac{P(X|C) \times P(C)}{P(X)}$

$= \frac{0.0005 \times 0.5}{0.00725}$

$= 0.03448$

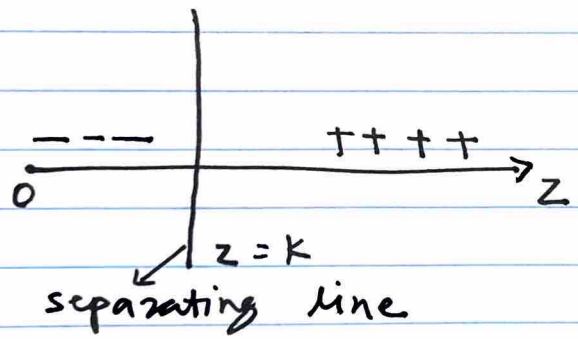
Problem 3

1.)

YES, the 1-D transformation that will make the points linearly separable is

$$z = |x_1 - c|$$

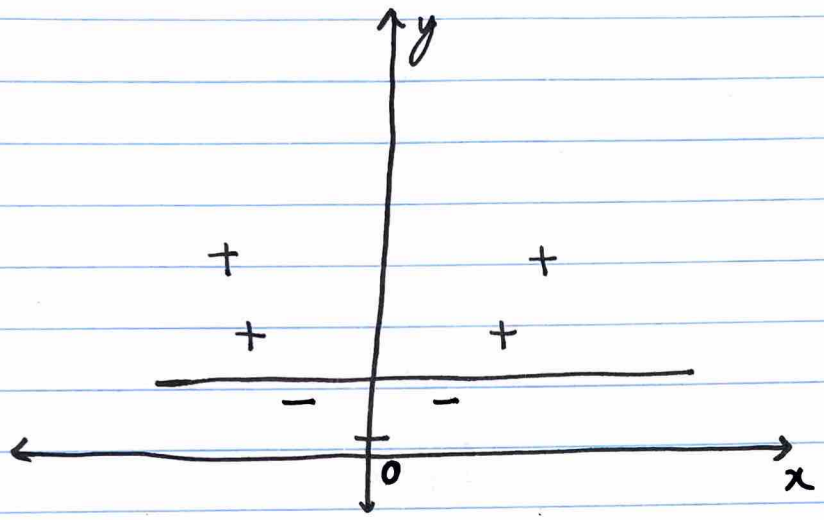
The transformed 1-D space will look like



2.) YES, the 2-D transformation that will make the points linearly separable is

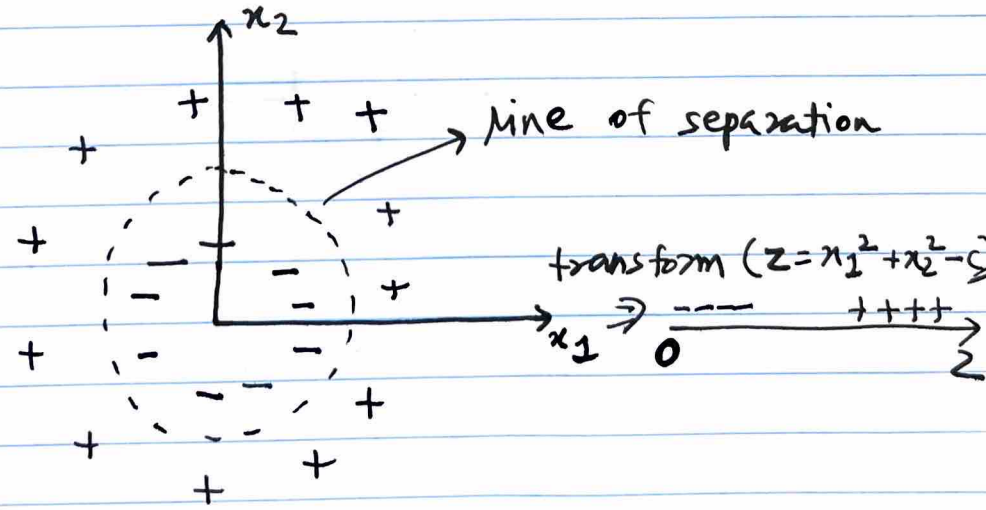
$$y = (x' - c)^2 \qquad x = x' - c$$

The transformed 2-D space will look like



$$T(x') = T[x'] = \begin{bmatrix} x' - c \\ (x' - c)^2 \end{bmatrix}$$

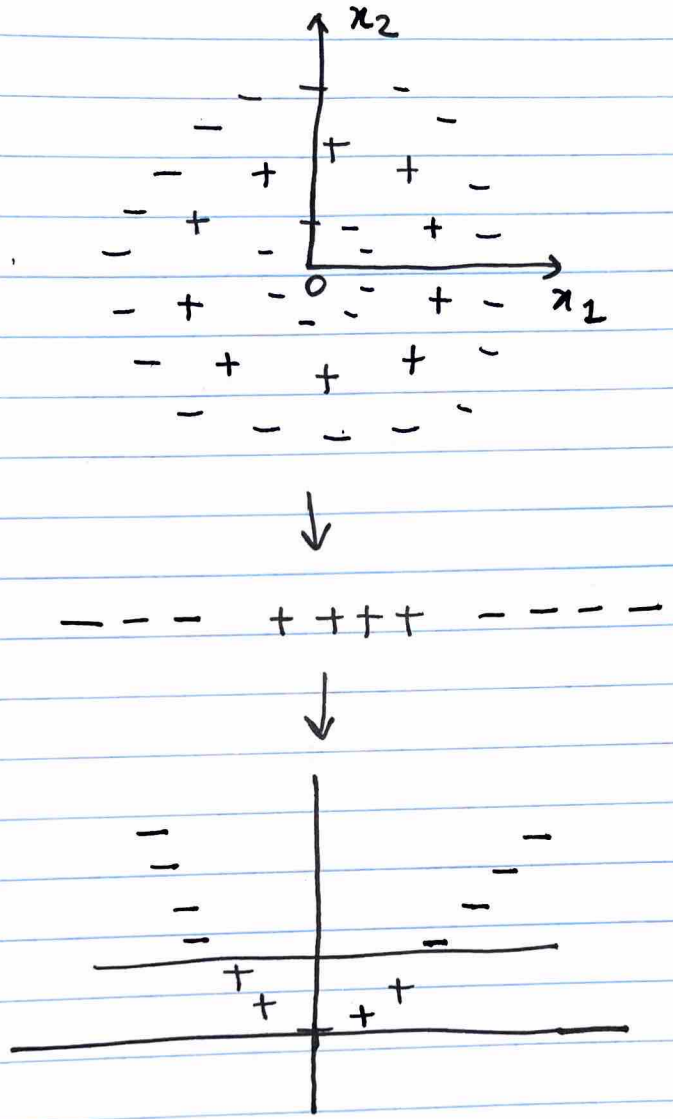
3.



YES, the transformation is given by

$$z = x_1^2 + x_2^2 - c$$

4.)



YES,

For the first transformation

$$T(x_1, x_2) = Z = x_1^2 + x_2^2$$

For the second transformation

$$T(z) = \begin{bmatrix} z - c \\ (z-c)^2 \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - c \\ (x_1^2 + x_2^2 - c)^2 \end{bmatrix}$$

Kernel of Not.

1. Lets assume x & z are variables & not vectors

$$\therefore (xz+1)^2$$
$$= x^2z^2 + 2xz + 1$$
$$= x^2z^2 + \sqrt{2}x\sqrt{2}z + 1$$
$$= \phi[x^2, \sqrt{2}x, 1]^T \phi[z^2, \sqrt{2}z, 1]$$

$\therefore (xz+1)^2$ is a valid kernel.

2. $K(x,z) = (xz-1)^3$
$$= x^3z^3 - 3x^2z^2 + 3xz - 1$$
$$= \phi[x^3, -\sqrt{3}x^2, \sqrt{3}x, -1]^T \phi[z^3, \sqrt{3}z^2, \sqrt{3}z, 1]$$
$$= \phi(x)^T \phi(z)$$

$\therefore (xz-1)^3$ is not a valid kernel because of the 2 negative terms.

Problem 4

1.) $p(y;\phi) = (1-\phi)^{y-1}\phi$

Taking log on both sides

$\ln(p(y;\phi)) = (y-1)\ln(1-\phi) + \ln(\phi)$

$\ln(p(y;\phi)) = y\ln(1-\phi) + \ln(\phi) - \ln(1-\phi)$

Taking exponent on both sides.

$p(y;\phi) = e^{(y\ln(1-\phi) + \ln(\phi) - \ln(1-\phi))}$ — (1)

We know that exponential family of distribution can be represented as

$$p(y;\eta) = b(y)\, e^{(\eta^T T(y) - a(\eta))} \quad - (2)$$

comparing (1) & (2)

→ $b(y) = 1$

$\eta^T T(y) = y\ln(1-\phi)$

→ $T(y) = y$

→ $\eta = \ln(1-\phi)$

$a(\eta) = -\ln(\phi) + \ln(1-\phi)$

we know that $\eta = \ln(1-\phi)$ & $\phi = 1 - e^{\eta}$

Therefore, replacing ϕ in $a(\eta)$

→ $a(\eta) = -\ln(1-e^{\eta}) + \eta$

$= \ln\left(\frac{e^{\eta}}{1-e^{\eta}}\right)$

2. Log likelihood

$$l(w) = \log p(y_n | x_n ; w)$$

From previous question we know that

$$p(y;\phi) = e^{y_n - \log \frac{e^n}{1-e^n}}$$

By the key assumptions of a GLM we know that the linear predictor η is related to input features and model parameters through a linear combination

$$\eta = w^T x$$

Therefore,

$$l(w) = \log \left[e^{y_i \eta_i - \log \frac{e^n}{1-e^n}} \right]$$

$$l(w) = \log \left[e^{y_i w^T x_i - \log \frac{e^{w^T x_i}}{1-e^{w^T x_i}}} \right]$$

$$= y_i w^T x_i - \log \frac{e^{w^T x_i}}{1 - e^{w^T x_i}}$$

$$= y_i w^T x_i + \log \frac{1 - e^{w^T x_i}}{e^{w^T x_i}}$$

$$= y_i w^T x_i + \log \left(\frac{1}{e^{w^T x_i}} - 1 \right)$$

$$\frac{\partial\, l(w)}{\partial w} = \frac{\partial}{\partial w}\left[y_i w^T x_i + \log\left(\frac{1}{e^{w^T x_i}} - 1\right)\right]$$

$$= y_i x_i + \frac{\partial}{\partial w}\left(\log\left(\frac{1}{e^{w^T x_i}} - 1\right)\right)$$

Using chain rule

$$= x_i y_i + \frac{1}{e^{-w^T x} - 1} \times e^{-w^T x_i} \times x_i$$

$$= x_i y_i - \frac{e^{-w^T x_i} - 1}{e^{-w^T x_i}}$$

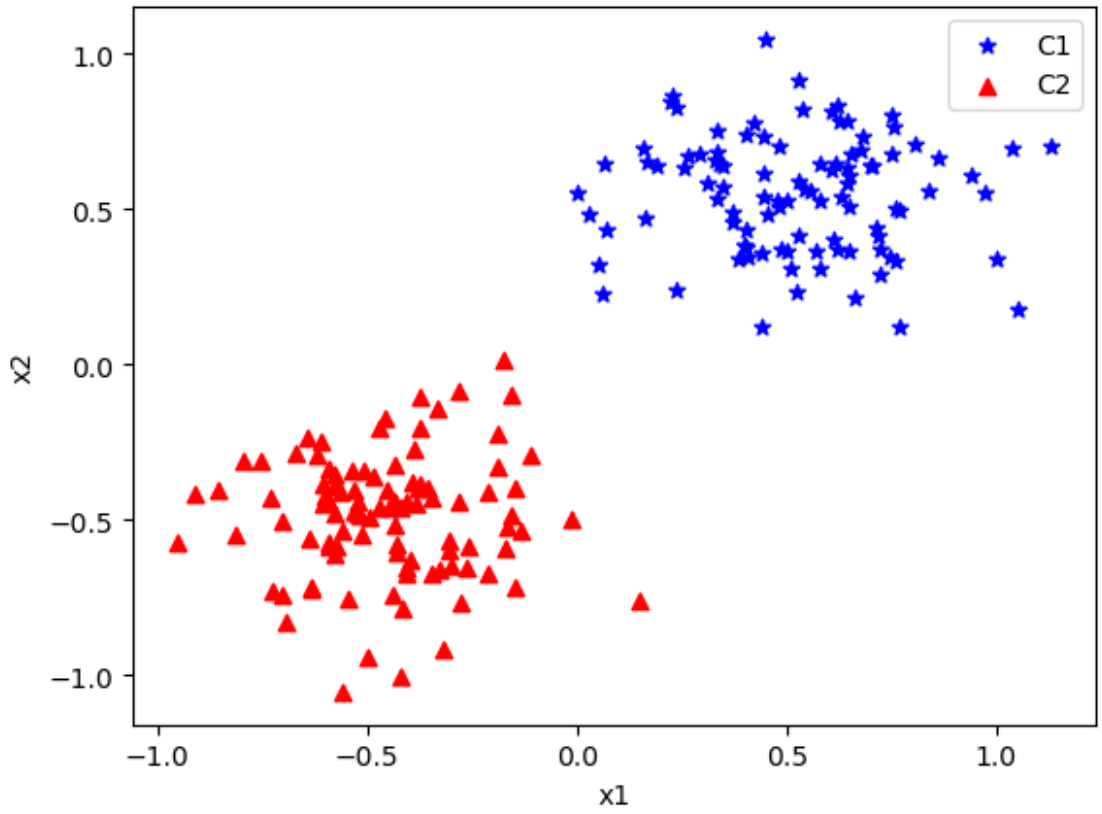
$$= x_i y_j - \frac{1}{1 - e^{-w^T x_i}} \cdot x_i$$

By stochastic gradient ascent rule.

$$w = w + \alpha \frac{\partial}{\partial w} l(w)$$

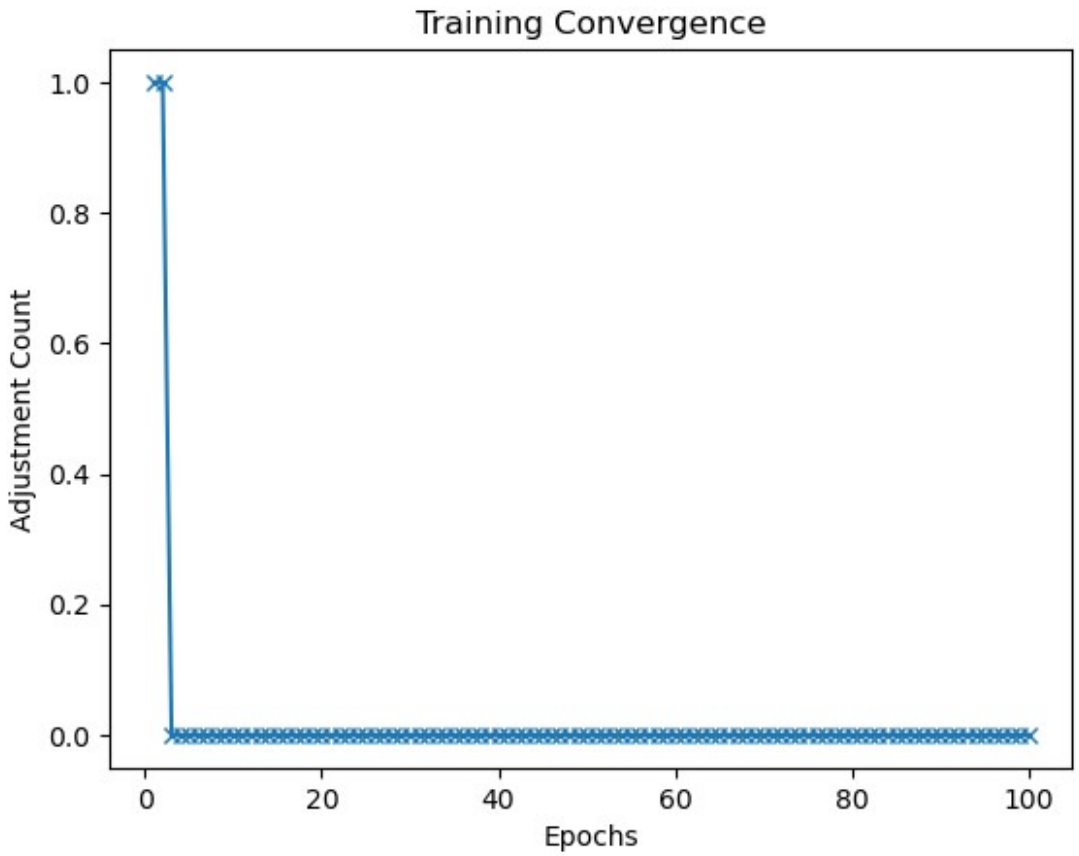
$$= w + \alpha \left(x_i y_i - \frac{1}{1 - e^{-w^T x_i}} x_i\right)$$

```
import numpy as np
import matplotlib.pyplot as plt

n_samples = 100

p = np.random.multivariate_normal([0.5, 0.5], [[0.05, 0], [0, 0.035]],
n_samples)
py = np.ones(n_samples)

q = np.random.multivariate_normal([-0.5, -0.5], [[0.05, 0], [0,
0.035]], n_samples)
qy = -np.ones(n_samples)

sp = np.vstack((p, q))
sy = np.hstack((py, qy))

datasets = [p, q]
colors = ['blue', 'red']
markers = ['*', '^']
labels = ['C1', 'C2']

for data, color, marker, label in zip(datasets, colors, markers,
labels):
    plt.scatter(data[:, 0], data[:, 1], color=color, marker=marker,
label=label)

plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(loc='best')
plt.show()
```




```
import numpy as np

class perceptron(object):
    def __init__(self, learning_rate=0.01, epochs=10):
        self.learning_rate = learning_rate
        self.epochs = epochs

    def train(self, data, labels):
        self.coefficients = np.zeros(1 + data.shape[1])
        self.adjustments = []

        for epoch in range(self.epochs):
            total_adjustments = 0
            for feature_set, desired_output in zip(data, labels):
                prediction_error = desired_output -
self.classify(feature_set)
                delta = self.learning_rate * prediction_error
                self.coefficients[1:] += delta * feature_set
                self.coefficients[0] += delta
                total_adjustments += int(delta != 0)
            self.adjustments.append(total_adjustments)
        return self

    def weighted_sum(self, data):
```

```
        return np.dot(data, self.coefficients[1:]) +
self.coefficients[0]

    def classify(self, data):
        return np.where(self.weighted_sum(data) >= 0.0, 1, -1)

neuron = perceptron(learning_rate=1, epochs=100)
neuron.train(sp,sy)

plt.plot(range(1, len(neuron.adjustments) + 1), neuron.adjustments,
marker='x')
plt.xlabel('Epochs')
plt.ylabel('Adjustment Count')
plt.title('Training Convergence')
plt.show()
```

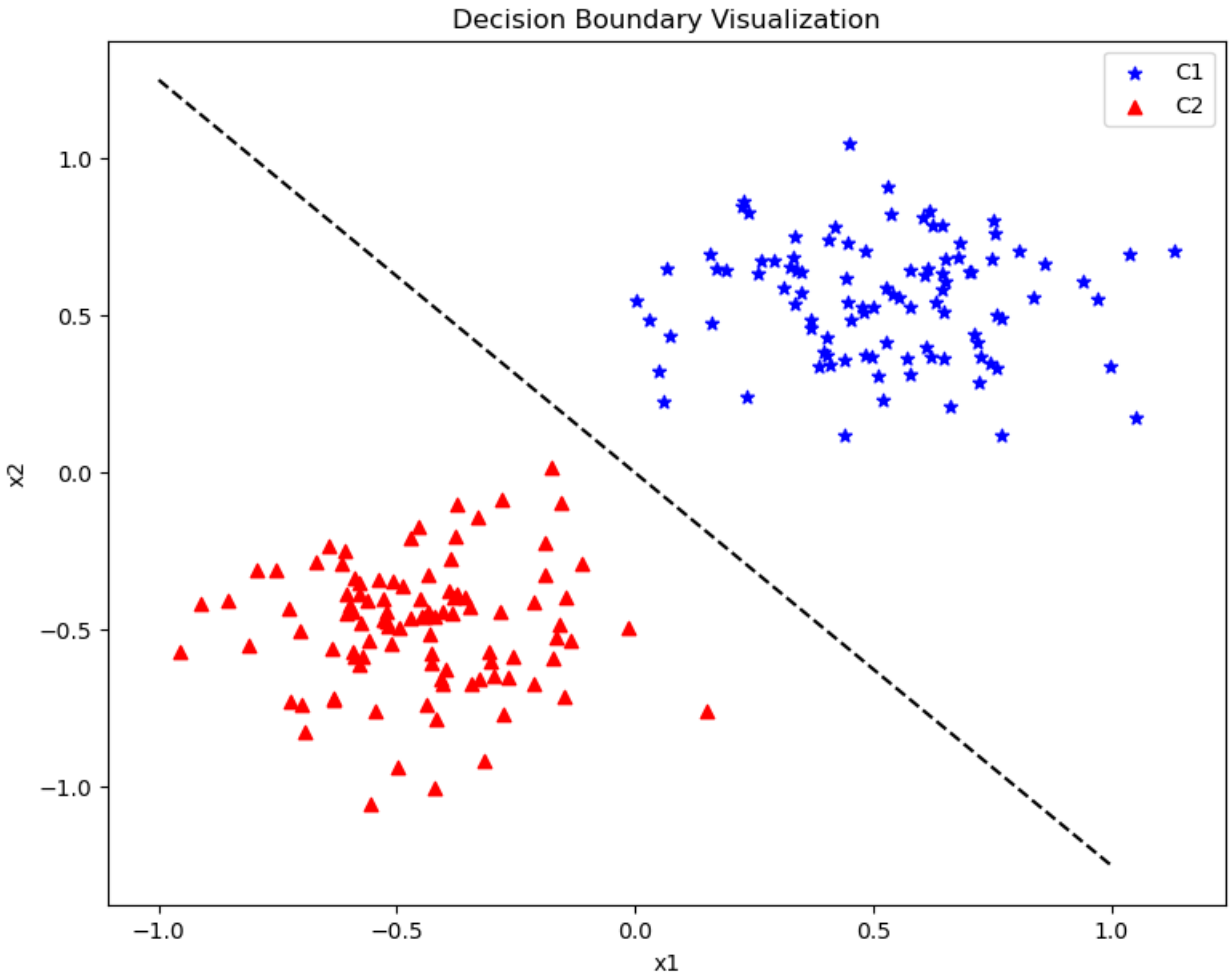



```
def convergence(E):
  for i in range(len(E)):
    if E[i] == 0:
      return i

print(convergence(neuron.adjustments)+1)
```

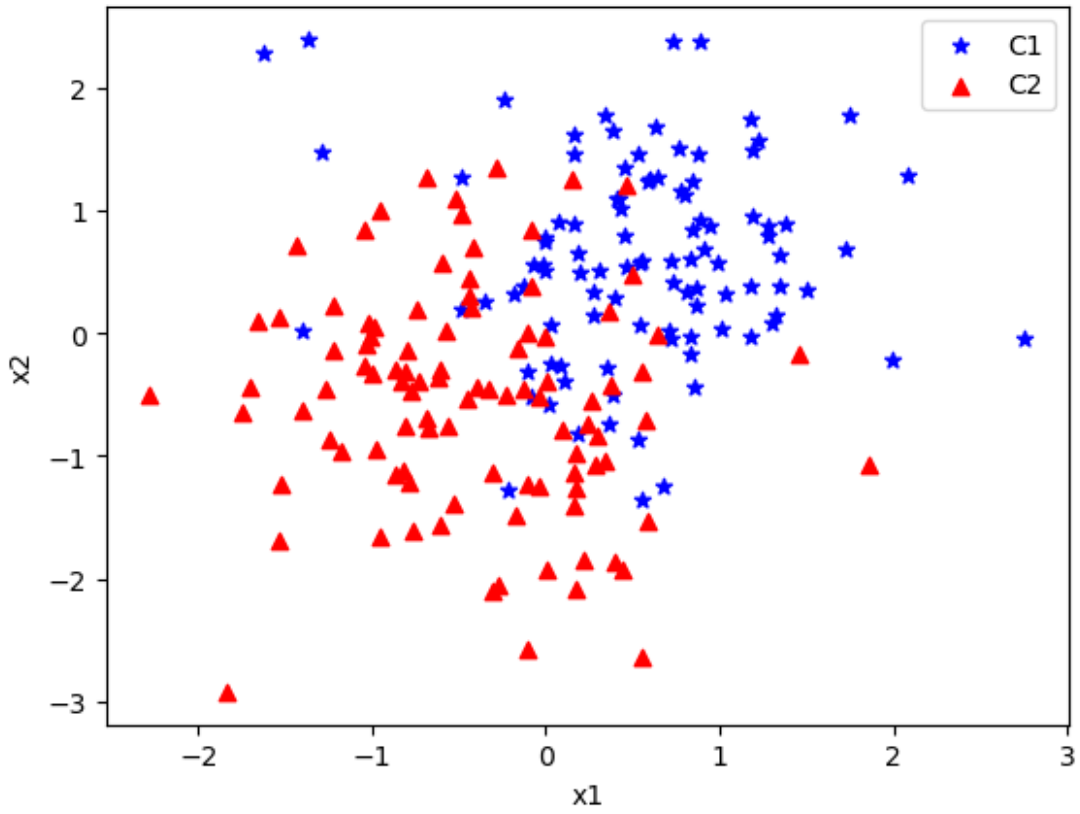
```
3
```

We can deduce from the preceding graph that the parameters converged after the 3rd iteration, indicating that a linearly separating line had been discovered.

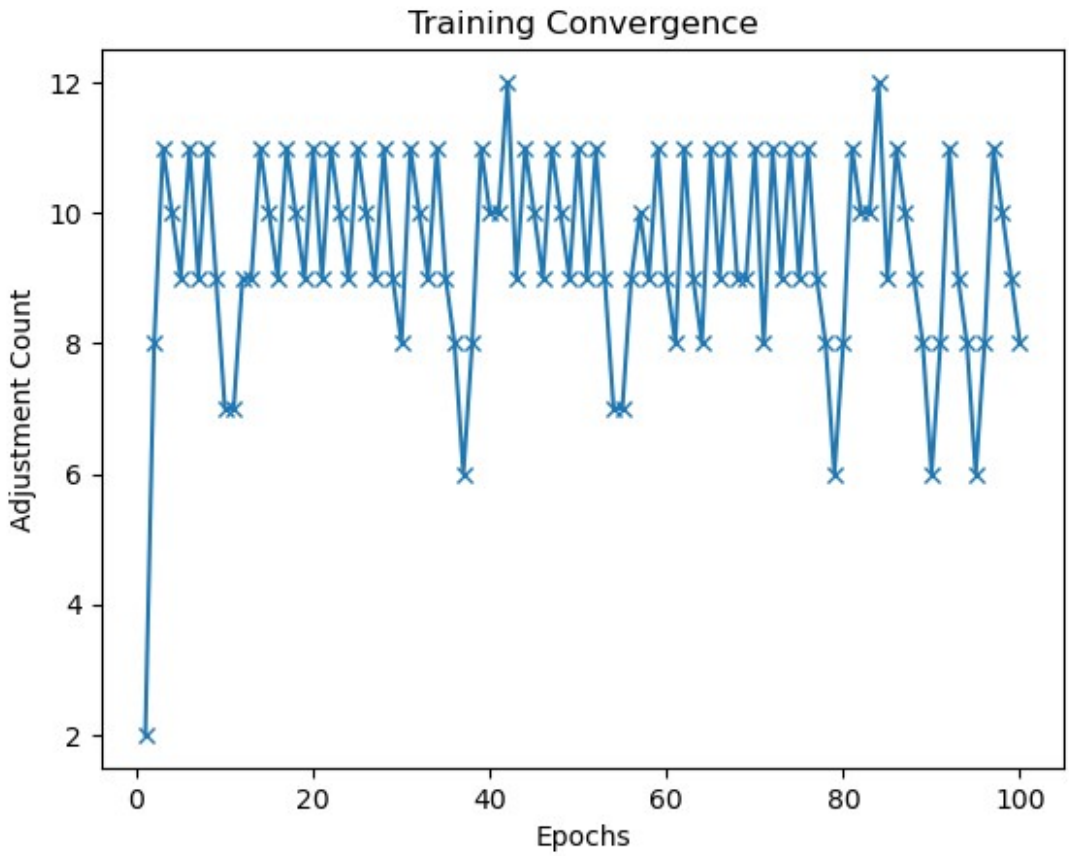
```
gradient = - neuron.coefficients[1] / neuron.coefficients[2]
y_intercept = - neuron.coefficients[0] / neuron.coefficients[2]

x_values = np.linspace(-1, 1, 10)
y_values = gradient * x_values + y_intercept

plt.figure(figsize=(9, 7))
plt.scatter(p[:, 0], p[:, 1], color='blue', marker='*', label='C1')
plt.scatter(q[:, 0], q[:, 1], color='red', marker='^', label='C2')
plt.plot(x_values, y_values, 'k--')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(loc='best')
plt.title('Decision Boundary Visualization')
plt.show()
```




```
a = np.random.multivariate_normal([0.5, 0.5], [[0.5, 0], [0, 0.75]],
n_samples)
ay = np.ones(n_samples)

b = np.random.multivariate_normal([-0.5, -0.5], [[0.5, 0], [0, 0.75]],
n_samples)
by = -np.ones(n_samples)

sa = np.vstack((a, b))
sy = np.hstack((ay, by))

plt.scatter(a[:,0],a[:,1],color='blue', marker='*', label='C1')
plt.scatter(b[:,0],b[:,1],color='red', marker='^', label='C2')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(loc='best')
plt.show()
```




```
neuron = perceptron(learning_rate=1, epochs=100)
neuron.train(sa,sy)

plt.plot(range(1, len(neuron.adjustments) + 1), neuron.adjustments,
marker='x')
plt.xlabel('Epochs')
plt.ylabel('Adjustment Count')
plt.title('Training Convergence')
plt.show()
```



As seen in the graph above, the algorithm is continuously trying to fix itself but fails, never achieving convergence. Since the data points cannot be separated linearly, the technique would still fail even if the number of epochs were increased.