CS613 Project -Francis Kurian

Perceptron Background:

This project implements the Perceptron algorithm, a fundamental concept in machine learning and neural network history. The Perceptron is a type of artificial neuron or binary linear classifier, inspired by the functioning of a biological neuron. Developed by Frank Rosenblatt in the late 1950s, the Perceptron is the building block of many modern machine learning algorithms and artificial neural networks.

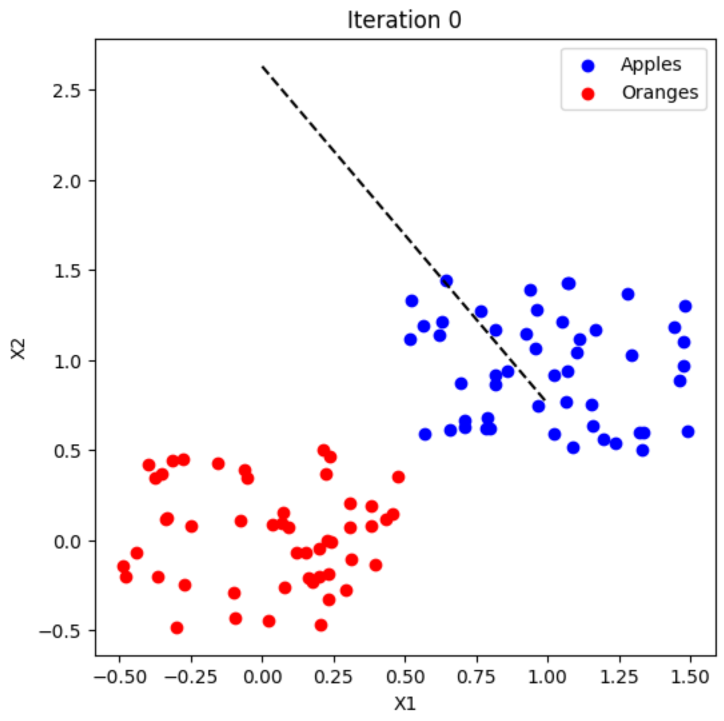
The Perceptron operates by taking a set of input features, applying weights to those features, and producing an output. It's a simple yet powerful algorithm that can be used for binary classification tasks. In this code, we use the Perceptron to learn and separate two classes in a linearly separable dataset.

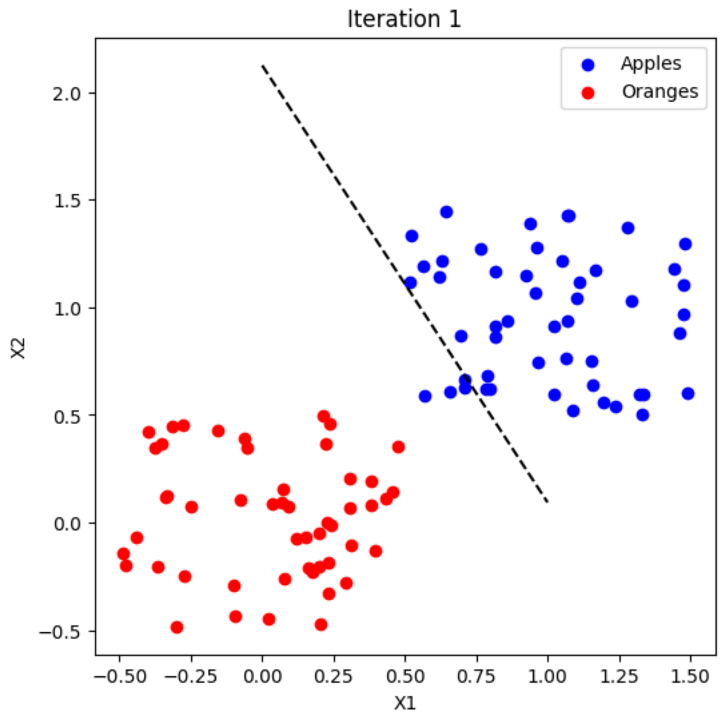
Objectives:

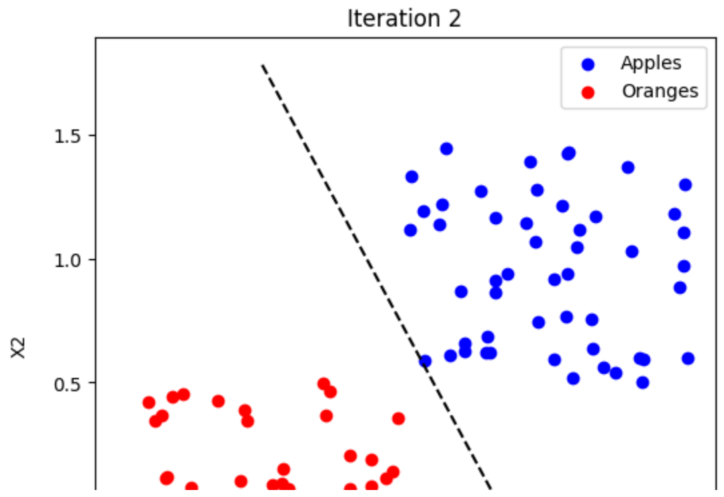
The objective of this project is to demonstrate the Perceptron algorithm in action. Specifically, the code aims to:

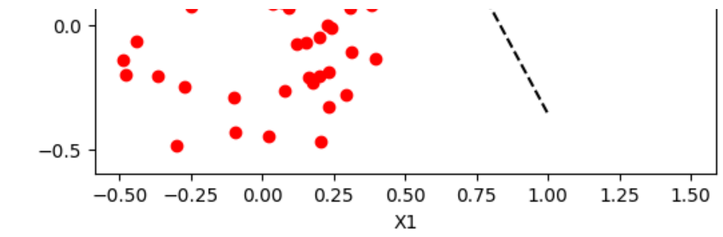
- Generate a synthetic linearly separable dataset with two classes (Apples and Oranges).
- Apply the Perceptron algorithm to learn a decision boundary that separates these two classes.
- Visualize the learning process by showing the decision boundary at each iteration (epoch).
- Present the final decision boundary that correctly classifies the dataset into Apples and Oranges.

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

# Generate a linearly separable dataset
np.random.seed(0)
X = np.random.rand(100, 2)
X[:50] += 0.5  # Class A
X[50:] -= 0.5  # Class B
y = np.array([1] * 50 + [-1] * 50)

# Perceptron algorithm
eta = 0.5  # Learning rate
epochs = 10

omega = np.random.rand(3)  # Initialize weights
X_bias = np.c_[np.ones(X.shape[0]), X]

for epoch in range(epochs):
    for i, x in enumerate(X_bias):
        if y[i] * np.dot(x, omega) <= 0:
            omega += eta * y[i] * x

    if epoch % 1 == 0:
        plt.figure(figsize=(6, 6))
        plt.scatter(X[:50, 0], X[:50, 1], color='b', label='Apples')
        plt.scatter(X[50:, 0], X[50:, 1], color='r', label='Oranges')
        x_values = np.array([0, 1])
        y_values = (-omega[0] - omega[1] * x_values) / omega[2]
        plt.plot(x_values, y_values, color='k', linestyle='--')
        plt.title(f'Iteration {epoch}')
        plt.xlabel('X1')
        plt.ylabel('X2')
        plt.legend()
        plt.show()

# Final decision boundary
plt.figure(figsize=(6, 6))
plt.scatter(X[:50, 0], X[:50, 1], color='b', label='Apples')
plt.scatter(X[50:, 0], X[50:, 1], color='r', label='Oranges')
x_values = np.array([0, 1])
y_values = (-omega[0] - omega[1] * x_values) / omega[2]
plt.plot(x_values, y_values, color='k', linestyle='--')
plt.title('Final Decision Boundary')
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend()
plt.show()

```

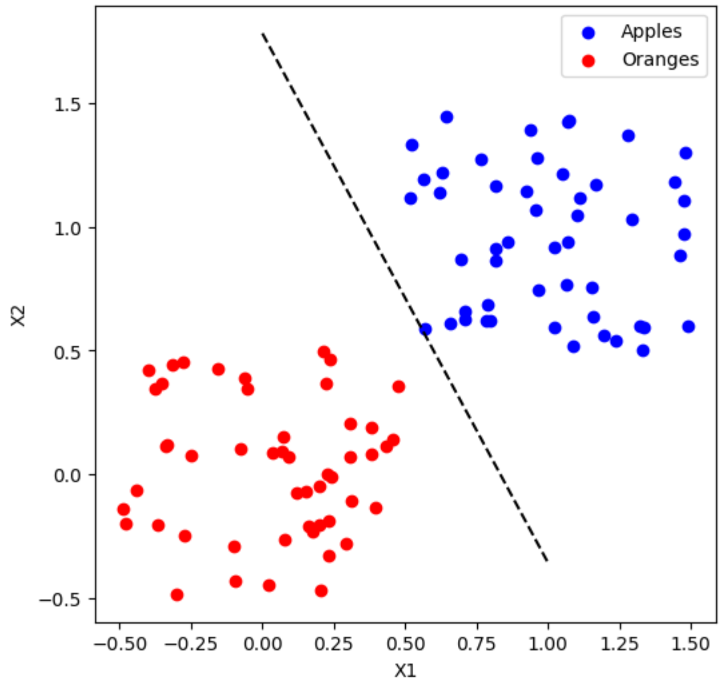

Iteration 0
Apples
Oranges
X2
X1
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


Iteration 1
Apples
Oranges
X2
X1
2.0
1.5
1.0
0.5
0.0
−0.5
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

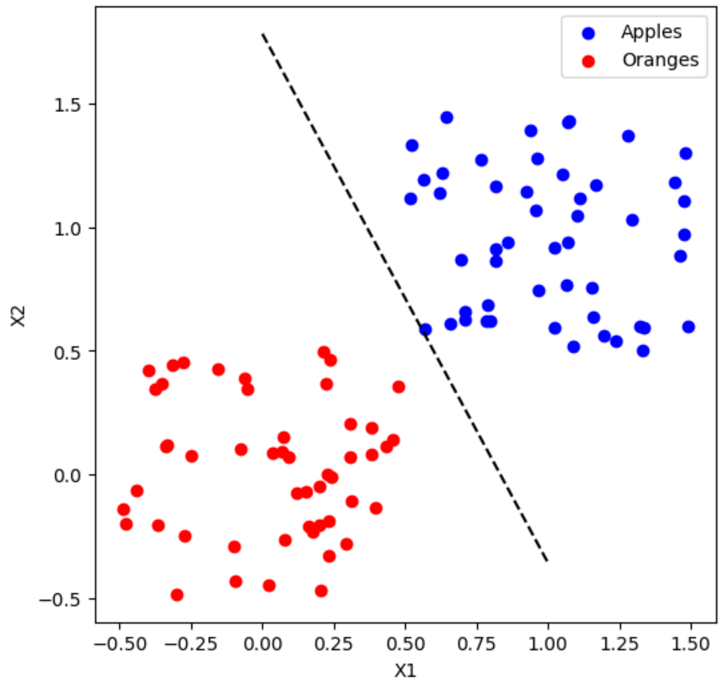

Iteration 2
Apples
Oranges
X2
1.5
1.0
0.5



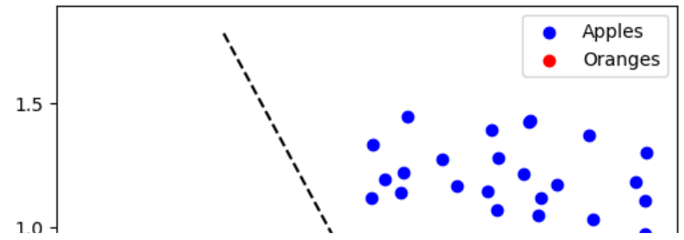
Iteration 3

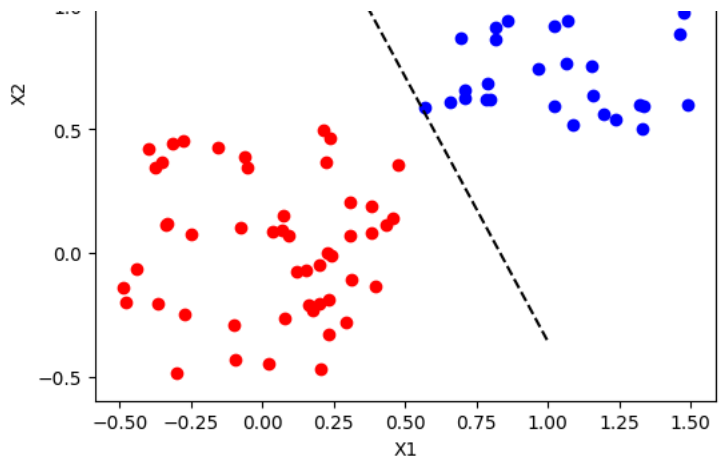


Iteration 4

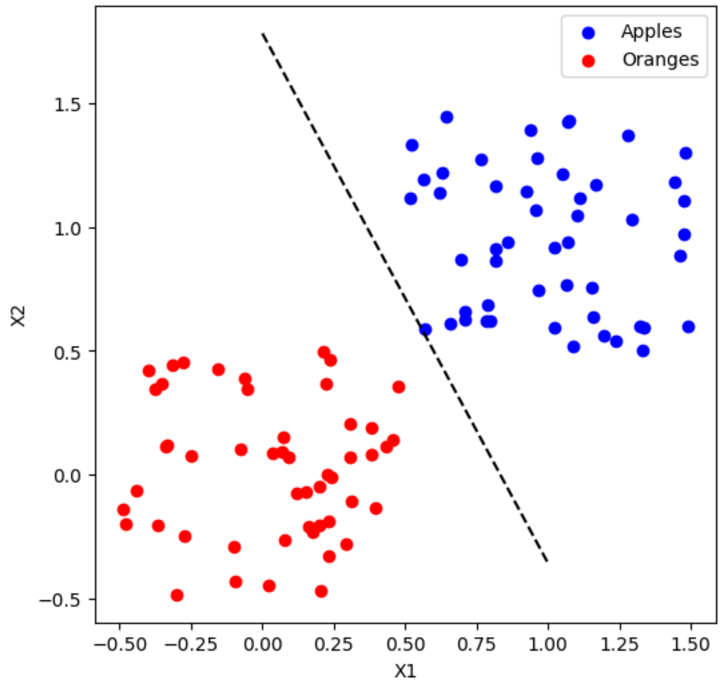


Iteration 5

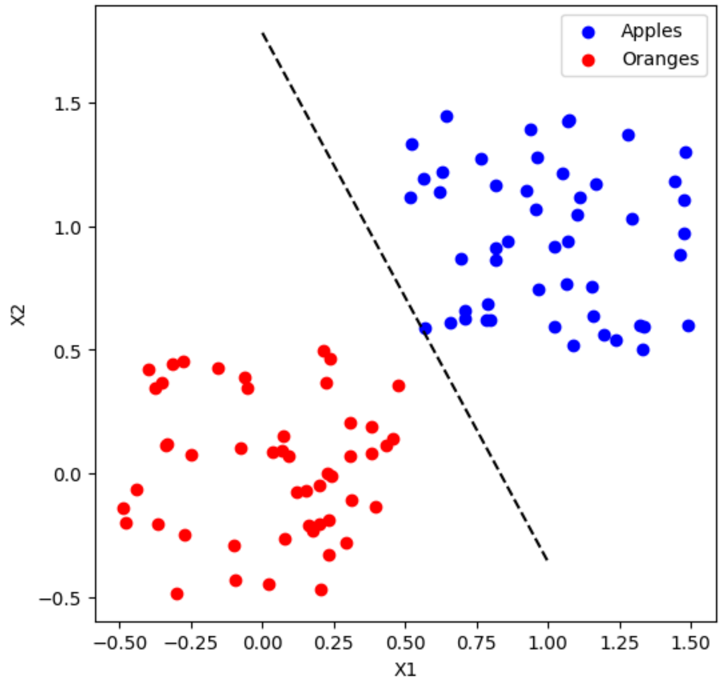




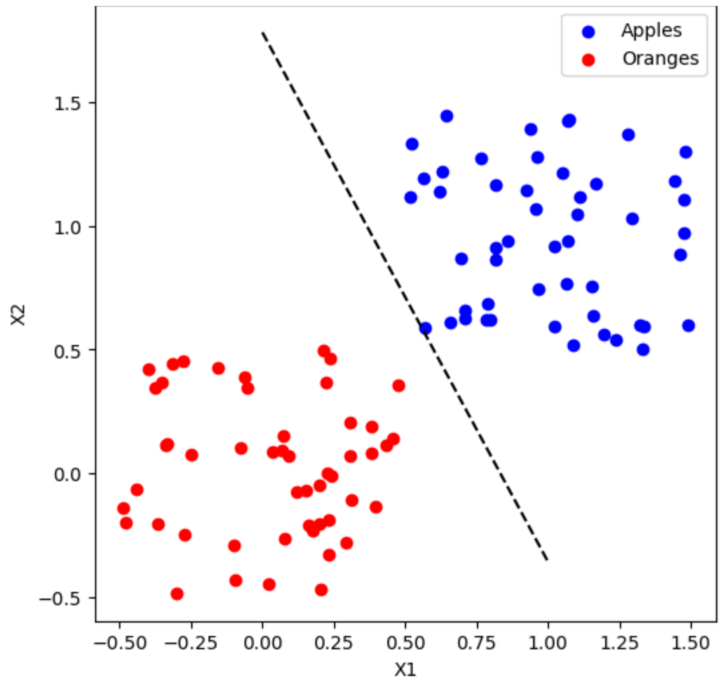
Iteration 6



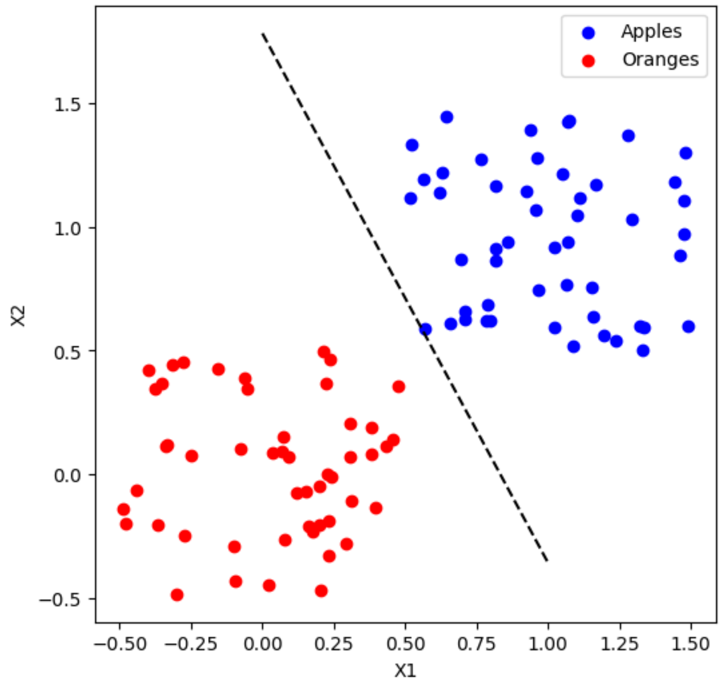
Iteration 7



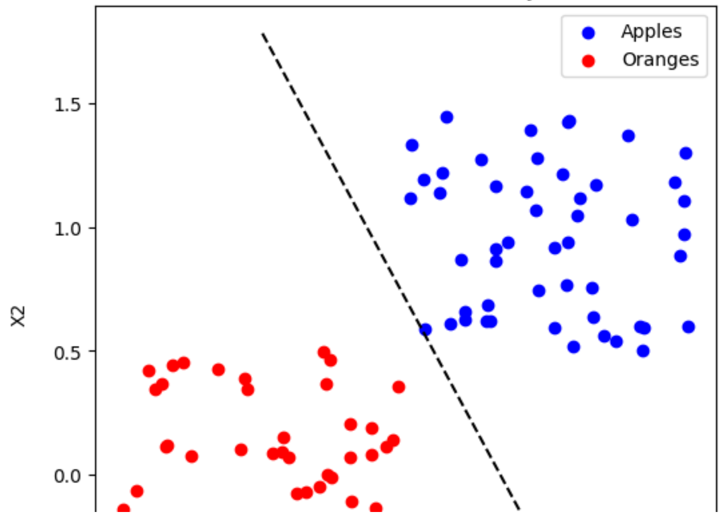
Iteration 8

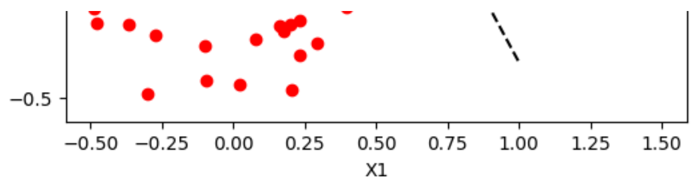


Iteration 9



Final Decision Boundary





Summary:

In this project, we first generate a linearly separable dataset with two classes, Apples and Oranges. The Perceptron algorithm is then applied to this dataset. The Perceptron iteratively updates its weights based on the misclassifications until it correctly classifies all data points or until a specified number of epochs is reached.

The code demonstrates the learning process by visualizing the decision boundary at each iteration (epoch). As the algorithm progresses, the decision boundary adapts to separate the two classes. The code concludes with the final decision boundary that accurately distinguishes between Apples and Oranges, showcasing the Perceptron's ability to perform binary classification on linearly separable data. This code serves as a simple but illustrative example of the Perceptron algorithm's functionality and its significance in machine learning.