Assignment: Perceptron Algorithm for Binary Classification

Aim

To implement the Perceptron algorithm for binary classification, train it on synthetic data, evaluate its accuracy, and visualize the decision boundary.

Lab Setup

- Software: Python 3.x (Google Colab, Jupyter Notebook, or any Python IDE)
- Required libraries:
  - scikit-learn
  - matplotlib
- Installation command (if needed):

  bash
  ```
  pip install scikit-learn matplotlib
  ```
-

Input

- Synthetic 2D dataset generated with 200 samples, 2 informative features, and 2 classes.
- Dataset is split into training (80%) and testing (20%) samples.

Expected Output

- Test accuracy of the Perceptron classifier.
- Plot showing decision boundary and classified test points.

Example output:

text
```
Test Accuracy: 0.85
```

Theory / Algorithm

The Perceptron is a simple linear classifier. It assigns weights to input features and calculates the weighted sum plus bias. The output is binary based on whether this sum surpasses a threshold. The training updates weights iteratively to minimize classification errors.

Algorithm steps:

1. Initialize weights and bias randomly or to zero.
2. For each training example, calculate the output using current weights.
3. Update weights if the prediction differs from the actual label.
4. Repeat for several iterations or until convergence.
5. Use the trained weights to classify new inputs.

Code

python
```
from sklearn.datasets import make_classification
from sklearn.linear_model import Perceptron
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Generate synthetic classification dataset
X, y = make_classification(n_samples=200, n_features=2, n_redundant=0, n_informative=2,
                  n_clusters_per_class=1, random_state=42)
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize Perceptron model
model = Perceptron(max_iter=1000, tol=1e-3, random_state=42)
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.2f}")
# Plot decision boundary
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                np.arange(y_min, y_max, 0.01))
    grid = np.c_[xx.ravel(), yy.ravel()]
    preds = model.predict(grid).reshape(xx.shape)

    plt.contourf(xx, yy, preds, alpha=0.3, cmap=plt.cm.Paired)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap=plt.cm.Paired)
    plt.title("Perceptron Decision Boundary")
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.show()
plot_decision_boundary(X_test, y_test, model)
```

Conclusion / Discussion

The Perceptron algorithm successfully classified the synthetic dataset with good accuracy on the test set. The decision boundary plotted clearly separates the two classes, showing the linear nature of the classifier. Although limited to linearly separable problems, the Perceptron remains a foundational algorithm in machine learning.