Machine Learning
Assignment 11
Aakanksha Darekar
202200733
A1 09

Implement backpropagation algorithm on Iris dataset.

Introduction

This project demonstrates the implementation of a **feedforward neural network with backpropagation** for classifying iris flowers into their respective species based on sepal and petal measurements. The Iris dataset is a classic benchmark in machine learning, consisting of three species of iris flowers (*Iris-setosa*, *Iris-versicolor*, and *Iris-virginica*), each described by four features:

- **SepalLengthCm** (Length of the sepal in cm)
- **SepalWidthCm** (Width of the sepal in cm)
- **PetalLengthCm** (Length of the petal in cm)
- **PetalWidthCm** (Width of the petal in cm)

Key Objectives

1. **Implement a Neural Network from Scratch**
   - Design a **two-layer neural network** (input → hidden → output) using NumPy.
   - Use **sigmoid activation** for non-linearity.
   - Train the model using **backpropagation** and **gradient descent**.
2. **Data Preprocessing**
   - Load and preprocess the dataset.
   - Standardize features using **StandardScaler**.
   - Encode target labels (*Species*) into numerical form using **LabelEncoder**.
   - Convert labels into **one-hot encoded vectors** for multi-class classification.
3. **Model Training & Evaluation**
   - Split data into **training (80%) and testing (20%) sets**.
   - Train the neural network over **10,000 epochs** with a learning rate of **0.1**.

- Monitor **training loss** to ensure convergence.
- Evaluate performance using:
  - **Accuracy**
  - **Precision, Recall, F1-Score** (per class)
  - **Confusion Matrix**

Why This Implementation?

- **Educational Value**: Helps understand how neural networks work under the hood.
- **Hands-on Learning**: Covers key concepts like forward/backward propagation, activation functions, and gradient descent.
- **Practical Application**: Demonstrates how to preprocess data, train a model, and evaluate its performance.

This code is designed to be **easy to follow** while providing **detailed insights** into neural network training. By the end, you'll have a working classifier that can predict iris species with high accuracy.

Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


class NeuralNetwork:

    def __init__(self, input_size, hidden_size, output_size):

        self.W1 = np.random.randn(input_size, hidden_size) * 0.01
```

```
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.randn(hidden_size, output_size) * 0.01
        self.b2 = np.zeros((1, output_size))

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def sigmoid_derivative(self, x):
        return x * (1 - x)

    def forward(self, X):
        self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = self.sigmoid(self.z1)
        self.z2 = np.dot(self.a1, self.W2) + self.b2
        self.a2 = self.sigmoid(self.z2)
        return self.a2

    def backward(self, X, y, output, learning_rate):
        m = X.shape[0]
        dZ2 = output - y
        dW2 = (1/m) * np.dot(self.a1.T, dZ2)
        db2 = (1/m) * np.sum(dZ2, axis=0, keepdims=True)

        dZ1 = np.dot(dZ2, self.W2.T) * self.sigmoid_derivative(self.a1)
        dW1 = (1/m) * np.dot(X.T, dZ1)
        db1 = (1/m) * np.sum(dZ1, axis=0, keepdims=True)
```

```
        self.W2 -= learning_rate * dW2
        self.b2 -= learning_rate * db2
        self.W1 -= learning_rate * dW1
        self.b1 -= learning_rate * db1

    def train(self, X, y, epochs, learning_rate):
        loss_history = []
        for i in range(epochs):
            output = self.forward(X)
            self.backward(X, y, output, learning_rate)
            loss = np.mean(np.square(y - output))
            loss_history.append(loss)
            if i % 1000 == 0:
                print(f"Epoch {i}, Loss: {loss:.4f}")
        return loss_history

    def predict(self, X):
        output = self.forward(X)
        return np.argmax(output, axis=1)


def load_and_preprocess_data(file_path):
    # Load data with specified column names
    df = pd.read_csv(file_path)

    # Verify columns
    print("Columns in dataset:", df.columns.tolist())
```

```
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
OneHotEncoder

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target.reshape(-1, 1)

# Preprocessing
scaler = StandardScaler()
X = scaler.fit_transform(X)

encoder = OneHotEncoder(sparse_output=False)  # Fixed
parameter issue
y = encoder.fit_transform(y)

# Split data (training + backtesting set)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Build neural network
model = tf.keras.Sequential([
    tf.keras.layers.Dense(8, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dense(3, activation='softmax')
])

# Compile model (Backpropagation works through optimizer)
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])

# Train model
model.fit(X_train, y_train, epochs=50, batch_size=8, verbose=0)

# Backtesting (Evaluating performance)
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {test_acc:.4f}")
```

```
    # Encode labels
    le = LabelEncoder()
    df['target'] = le.fit_transform(df['Species'])

    # Select feature columns (excluding Id and Species)
    feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
    X = df[feature_cols].values
    y = df['target'].values

    # Convert y to one-hot encoding
    y_onehot = np.zeros((y.size, y.max()+1))
    y_onehot[np.arange(y.size), y] = 1

    # Split into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=0.2,
random_state=42)

    # Standardize features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test, le, feature_cols

if __name__ == "__main__":
    # Update this path to your Iris dataset
    file_path = "C:\\Users\\sanke\\Documents\\6th Sem\\MLL\\iris.csv"
```

```
try:
    # Load and preprocess data
    X_train, X_test, y_train, y_test, le, feature_cols = load_and_preprocess_data(file_path)

    print("\nFeature columns used:", feature_cols)
    print("Class labels:", le.classes_)
    print("Number of training samples:", X_train.shape[0])
    print("Number of test samples:", X_test.shape[0])

    # Initialize network
    input_size = X_train.shape[1]
    hidden_size = 5
    output_size = y_train.shape[1]
    nn = NeuralNetwork(input_size, hidden_size, output_size)

    # Train network
    epochs = 10000
    learning_rate = 0.1
    print("\nTraining network...")
    loss_history = nn.train(X_train, y_train, epochs, learning_rate)

    # Plot training loss
    plt.figure(figsize=(10, 6))
    plt.plot(loss_history)
    plt.title('Training Loss Over Epochs')
    plt.xlabel('Epoch')
```

```
    plt.ylabel('Loss')
    plt.show()

    # Evaluate on test set
    y_pred = nn.predict(X_test)
    y_test_labels = np.argmax(y_test, axis=1)

    # Calculate metrics
    accuracy = accuracy_score(y_test_labels, y_pred)
    print(f"\nTest Accuracy: {accuracy:.4f}")

    # Classification report
    print("\nClassification Report:")
    print(classification_report(y_test_labels, y_pred, target_names=le.classes_))

    # Confusion matrix
    cm = confusion_matrix(y_test_labels, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=le.classes_, yticklabels=le.classes_)
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

except Exception as e:
    print(f"\nError: {str(e)}")
```

```
print("\nPlease verify:")
print(f"1. The file exists at: {file_path}")
print("2. The CSV file has exactly these columns:")
print("   Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species")

# Try to show file contents for debugging
try:
    df_sample = pd.read_csv(file_path, nrows=5)
    print("\nFirst few rows of your file:")
    print(df_sample)
except:
    print("\nCould not read the file to show sample data")
```

Output:

```
Test Accuracy: 1.0000

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

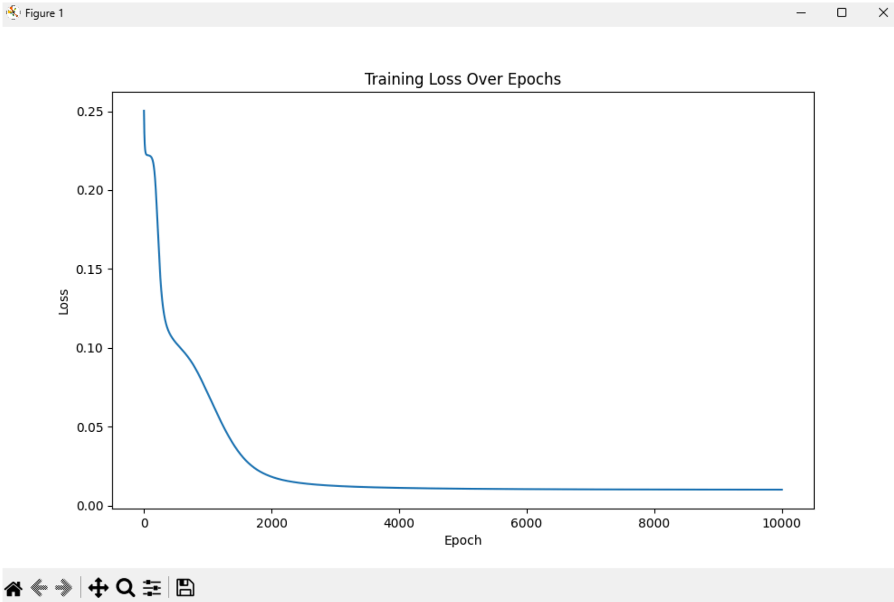
Figure 1
Training Loss Over Epochs
0.25
0.20
0.15
0.10
0.05
0.00
Loss
0
2000
4000
6000
8000
10000
Epoch

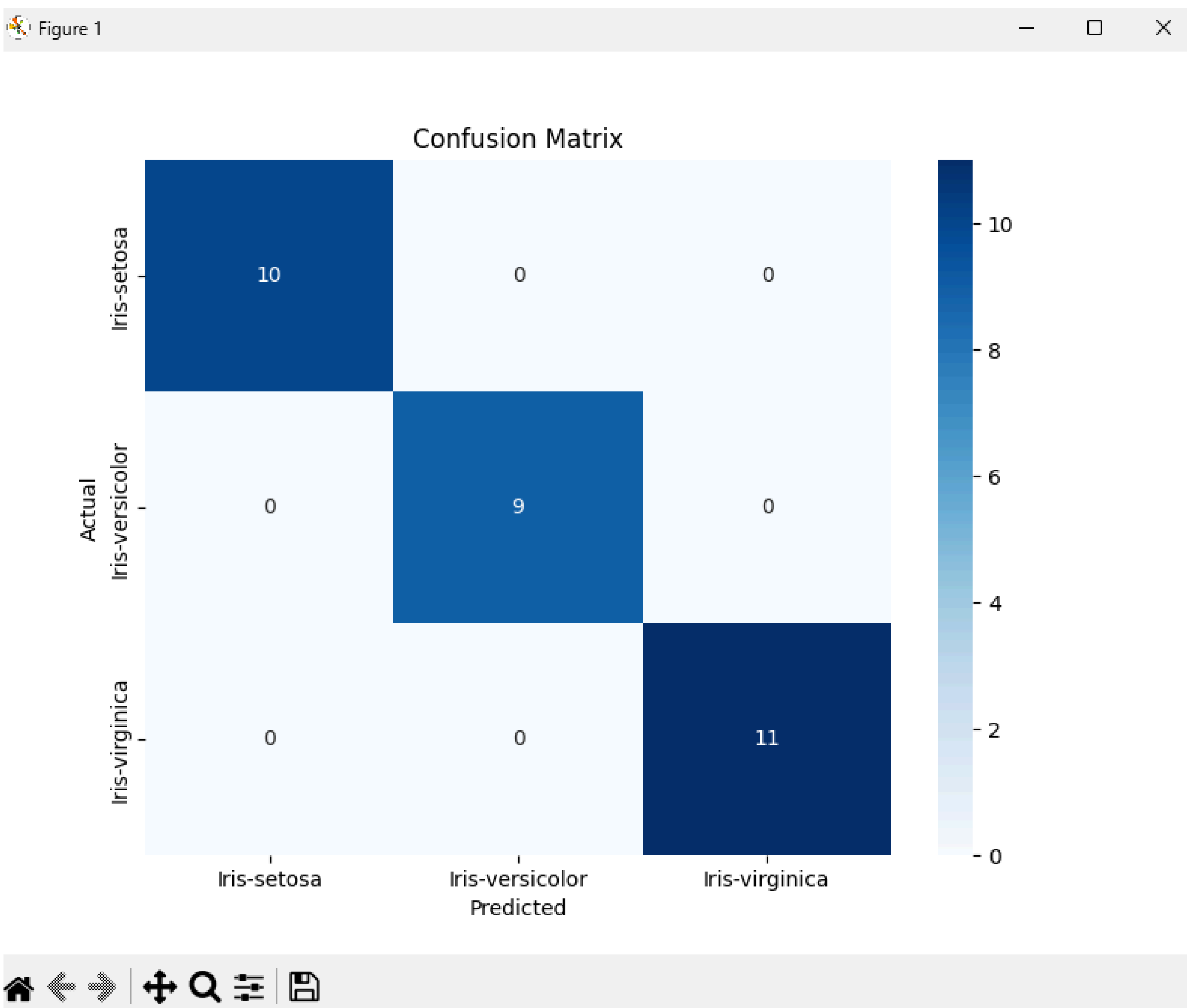
Figure 1
Confusion Matrix
Actual
Iris-setosa
Iris-versicolor
Iris-virginica
10
0
0
0
9
0
0
0
11
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted
10
8
6
4
2
0